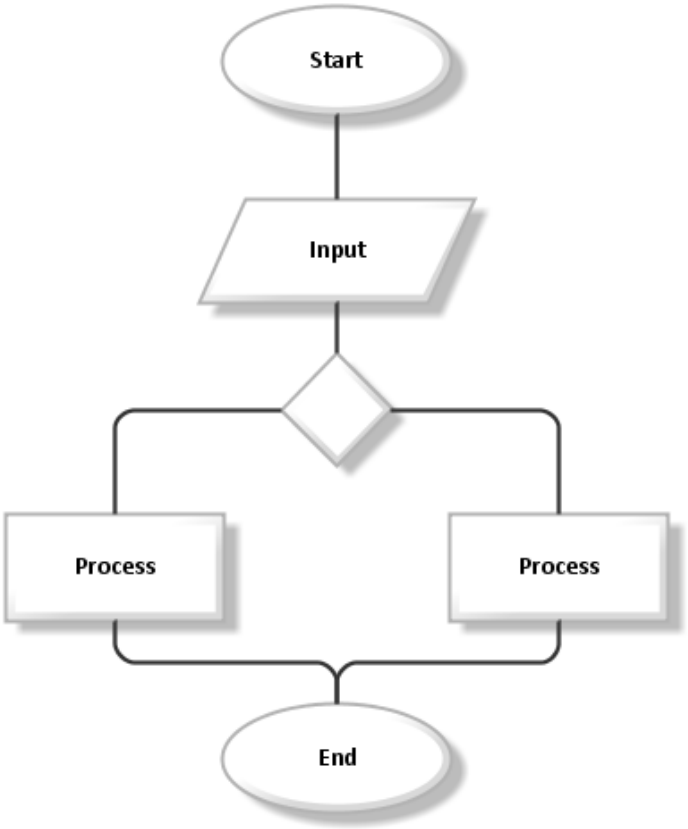
Αλεξάνδρειο ΤΕΙ Θεσσαλονίκης
Τμήμα Μηχανικών Πληροφορικής Τ.Ε.

Αλγοριθμική & Προγραμματισμός με Java

Διδακτικές Σημειώσεις για το Μάθημα
Αλγοριθμική και Προγραμματισμός



Γουλιάνας Κώστας
Επίκουρος Καθηγητής

Θεσσαλονίκη 2014

ΠΡΟΛΟΓΟΣ

Οι Σημειώσεις που ακολουθούν δε φιλοδοξούν να αποτελέσουν ένα ολοκληρωμένο σύγγραμμα. Αποτελούν μια πιο αναλυτική καταγραφή των παραδόσεων στα πλαίσια του μαθήματος του Α' Εξαμήνου, "Αλγοριθμική και Προγραμματισμός", του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. του Αλεξάνδρειου ΤΕΙ Θεσσαλονίκης. Στόχος των παραδόσεων και των σημειώσεων που ακολουθουν μέσα από πολλά παραδείγματα είναι να μάθουν οι πρωτοετείς φοιτητές τι σημαίνει προγραμματισμός, τι είναι και πώς σχεδιάζεται ένας αλγόριθμος ή ένα λογικό διάγραμμα, τι σημαίνουν και πώς χρησιμοποιούνται οι δομές-εντολές ελέγχου και επανάληψης, τι είναι και πώς χρησιμοποιούνται οι μέθοδοι, τι είναι παράμετροι και πώς χρησιμοποιούνται. Η `Java` χρησιμοποιείται σαν εργαλείο κατανόησης των βασικών προγραμματιστικών δομών στα πρώτα κεφάλαια, ενώ μετά το $7^{ο}$ κεφάλαιο γίνεται μια εισαγωγή στα περισσότερα χαρακτηριστικά της. Στόχος μας είναι οι φοιτητές, μετά το τέλος του εξαμήνου, να είναι σε θέση να σχεδιάζουν μια ολοκληρωμένη Κλάση Αντικειμένων, με τα απαραίτητα πεδία, τις μεθόδους κατασκευής (Δομητές), τις μεθόδους επεξεργασίας δεδομένων της κλάσης και τη μέθοδο `toString()`. Να μπορούν να χρησιμοποιούν τα προσδιοριστικά πρόσβασης `public` και `private` καθώς και τις αντίστοιχες μεθόδους `get` και `set`. Να κατανοήσουν την έννοια της καθολικής σταθεράς `final` και της καθολικής μεταβλητής `static` και πώς να χρησιμοποιούν κλάσεις με `static` μεθόδους επεξεργασίας δεδομένων πολλών αντικειμένων. Να μάθουν επίσης πώς να δημιουργούν Πίνακες Αντικειμένων, πώς να χειρίζονται τους Πίνακες και τις Συμβολοσειρές – `Strings`, πώς να χειρίζονται τις Εξαιρέσεις- `Exceptions` και ότι αφορά την είσοδο - έξοδο δεδομένων με τη χρήση των `Byte Streams`, `Character Streams` και της κλάσης `Scanner`.

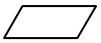
Πίνακας Περιεχομένων

1 ΕΙΣΑΓΩΓΗ – ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Αλγόριθμος

Μια σειρά από σαφή και καθορισμένα βήματα, τα οποία οδηγούν στη λύση ενός προβλήματος, περιγραφή του κάθε βήματος με λόγια και λέξεις-κλειδιά, π.χ. **διάβασε**, **υπολόγισε**, **εμφάνισε**, **αν**, **διαφορετικά**, **για όσο**, **για**.

Διάγραμμα Ροής (Λογικό Διάγραμμα)

Μια σειρά από σαφή και καθορισμένα βήματα, τα οποία οδηγούν στη λύση ενός προβλήματος με ειδικά σχήματα για την κάθε ενέργεια π.χ. **ορθογώνιο**  για την ανάθεση τιμής, **ρόμβος**  για έλεγχο ή επανάληψη, **παραλληλόγραμμο**  για εισαγωγή δεδομένων, ταινία  για εμφάνιση αποτελεσμάτων κ.λ.π..

Πρόγραμμα

Μετατροπή των παραπάνω βημάτων σε **εντολές** που μπορούν να μεταφραστούν από ένα πρόγραμμα στον Ηλεκτρονικό Υπολογιστή (Μεταγλωττιστής ή Διερμηνέας μιας Γλώσσας Προγραμματισμού), με λέξεις-κλειδιά, π.χ. **read**, **print**, **if**, **else**, **for**, **while**.

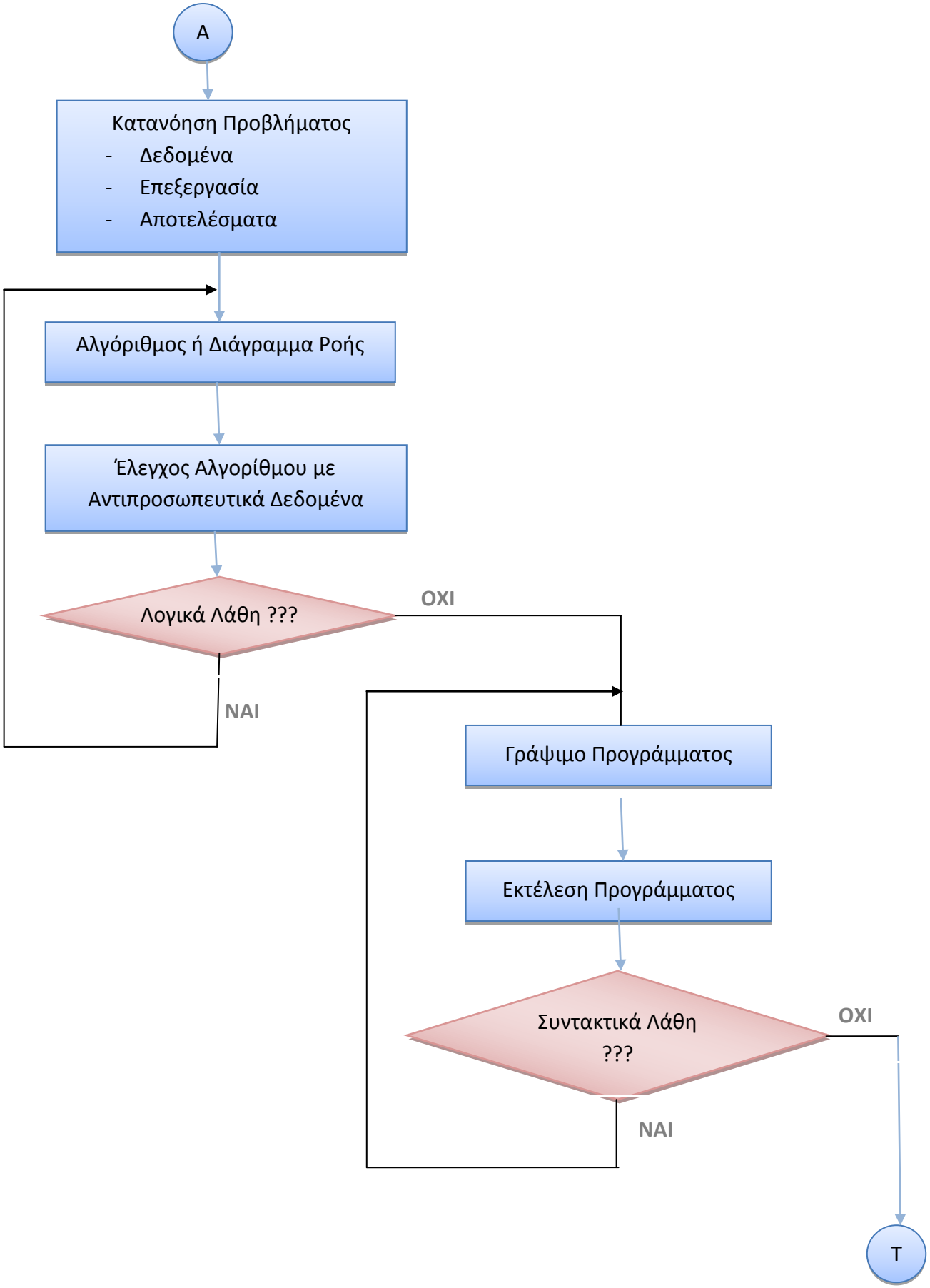
Γλώσσες Προγραμματισμού

Γλώσσα Μηχανής → Συμβολική Γλώσσα (ASSEMBLY) → Γλώσσες Υψηλού Επιπέδου (BASIC, FORTRAN, COBOL, PASCAL, C, Dephi, Visual Basic, C++, Java).

Μεταβλητές

Ονόματα στα αγγλικά, τα οποία αντιπροσωπεύουν θέσεις μνήμης, στις οποίες θα αποθηκεύονται δεδομένα, αριθμοί, χαρακτήρες κ.λ.π.. Συνήθως τα ονόματα που επιλέγονται έχουν σχέση με την ποσότητα που θα αποθηκεύσουν, π.χ. **num** για κάποιον αριθμό, **sum** για άθροισμα, **mo** για το Μέσο Όρο.

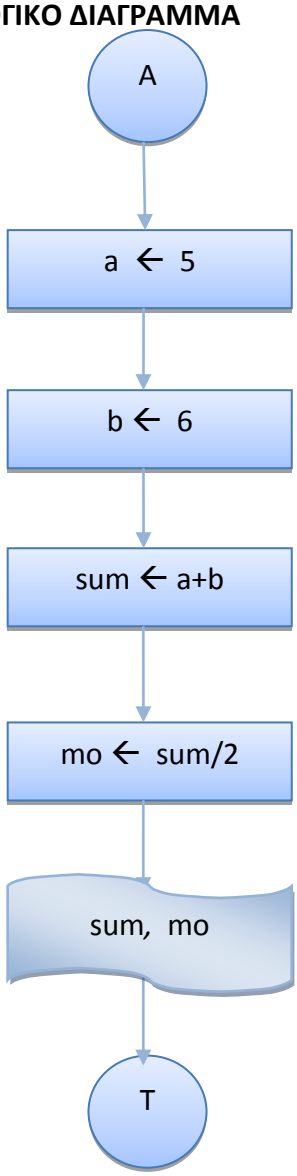
ΔΙΑΓΡΑΜΜΑ ΕΡΓΑΣΙΩΝ ΓΙΑ ΜΕΘΟΔΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ



1. 1 Ένα Απλό Πρόγραμμα

Να γραφεί πρόγραμμα, το οποίο θα δίνει τις τιμές 5 και 6 σε δύο μεταβλητές **num1** και **num2** και θα υπολογίζει και θα εμφανίζει το άθροισμά τους **sum** και το μέσο όρο **mo**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δίνω την τιμή 5 στο a (**a ← 5**)
2. Δίνω την τιμή 6 στο b (**b ← 6**)
3. Βρίσκω το άθροισμα (**sum ← a + b**)
4. Βρίσκω τον μέσο όρο (**mo ← sum/2**)
5. Εμφανίζω τις τιμές των **sum, mo**

ΠΡΟΓΡΑΜΜΑ

```
public class SumMO1 {
/*
Πρόγραμμα που δίνει τις τιμές 5 και 6 σε 2 ακέραιες μεταβλητές και βρίσκει
και εμφανίζει το άθροισμά τους και το Μέσο Όρο, ο οποίος είναι μεταβλητή
τύπου double

*/
    public static void main(String[] args) {

        // Δήλωση των ακέραιων μεταβλητών num1, num2
        int num1, num2;

        // Δήλωση της ακέραιας μεταβλητής sum για το άθροισμα
        int sum;

        // Δήλωση της πραγματικής μεταβλητής mo για το Μέσο Όρο
        double mo;

        // Ανάθεση των τιμών 5 και 6 στις ακέραιες μεταβλητές num1, num2
        num1 = 5;
        num2 = 6;

        // Υπολογισμός του αθροίσματος sum
        sum = num1 + num2;

        // Υπολογισμός του Μέσου Όρου mo
        mo = sum/2;

        // Εμφάνιση  του αθροίσματος sum και του Μέσου Όρου mo
        System.out.println("Άθροισμα = " + sum + " Μέσος Όρος = " + mo);
        }
}
```

Έξοδος Προγράμματος

```
Άθροισμα = 11 Μέσος Όρος = 5.0
```

- Μέσος Όρος = 5, γιατί γίνεται ακέραια διαίρεση του 11/2 και αποκόπτεται το δεκαδικό μέρος.

Λέξεις – κλειδιά στον ορισμό του προγράμματος :

`class SumMO1` : Ορισμός της κλάσης `Sum_mo_1`. Ο ορισμός περιλαμβάνει μόνο μία μέθοδο, τη `main()`.

`public` : Η λέξη **`public`** δηλώνει ότι η μέθοδος είναι προσπελάσιμη από παντού.

`static` : Η λέξη **`static`** δηλώνει ότι η μέθοδος είναι προσπελάσιμη ακόμη και αν δεν έχουν δημιουργηθεί αντικείμενα της κλάσης.

`void` : Σημαίνει ότι η μέθοδος `main()` δεν επιστρέφει καμιά τιμή.

`main()` : Η βασική μέθοδος.

ΠΑΡΑΤΗΡΗΣΕΙΣ

1. Οι αγκύλες **`{}`** πηγαίνουν ανά ζεύγη και περικλείουν αυτόνομα κομμάτια κώδικα.
2. Όλες οι εντολές τελειώνουν με το ελληνικό ερωτηματικό "**`;`**".
3. Τα Σχόλια πολλών γραμμών αρχίζουν με το σύμβολο "**`/*`**" και τελειώνουν με το ύμβολο "**`*/`**".
4. Τα Σχόλια μιας γραμμής ή μετά από μια εντολή αρχίζουν με το σύμβολο "**`//`**".
5. Οι μεταβλητές πρέπει να δηλώνουν κάποιο τύπο ανάλογα με την ποσότητα που θα αποθηκεύσουν, π.χ. `int num1, num2` για τους αριθμούς, `double mo` για το Μέσο Όρο.
6. Τα ονόματα των μεταβλητών μπορούν να αρχίζουν από οποιοδήποτε γράμμα ή τα σύμβολα "**`_`**" "**`$`**", αλλά <u>όχι</u> από ψηφίο. Π.χ. `num1` **<u>και όχι</u>** `1num`.
7. Η δήλωση μιας μεταβλητής περιλαμβάνει τον τύπο των δεδομένων που θα αποθηκεύσει και το όνομά της.

   Παράδειγμα

   ```
   double mo; // Δήλωση της πραγματικής μεταβλητής mo για το Μέσο Όρο
   ```

8. Μια δήλωση μπορεί να περιλαμβάνει περισσότερες από μια μεταβλητές.

   Παράδειγμα

   ```
   int num1, num2; // Δήλωση των ακέραιων μεταβλητών num1, num2
   ```

9. Για την εμφάνιση των αποτελεσμάτων χρησιμοποιούμε την εντολή **`System.out.println(<μηνύματα και ονόματα μεταβλητών>);`**

Η εντολή `System.out.println()` μπορεί **μέσα στις παρενθέσεις** να περιλαμβάνει :

- **Ονόματα** μεταβλητών, οπότε εμφανίζει απλώς τις τιμές τους.

Παράδειγμα :

```
System.out.println(mo);     // Θα εμφανίσει ΜΟΝΟ την τιμή της μεταβλητής mo
```

- **Ονόματα** μεταβλητών και **μηνύματα**, οπότε εμφανίζει και μηνύματα και τις τιμές των μεταβλητών.

Παράδειγμα :

```
System.out.println("mo =  " + mo);    // Θα εμφανίσει το μήνυμα mo  = και την τιμή της μεταβλητής mo
```

- **Τίποτα**, οπότε γίνεται απλώς αλλαγή γραμμής.

```
System.out.println();          // αλλαγή γραμμής
```

1.2 ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τα δεδομένα που μπορούν να αποθηκευθούν σε μια μεταβλητή μπορεί να είναι ακέραιοι αριθμοί (π.χ. ο αριθμός `5`), πραγματικοί αριθμοί ή αριθμοί κινητής υποδιαστολής (π.χ. ο αριθμός `5.5`), χαρακτήρες (π.χ. το γράμμα `X`), συμβολοσειρές (π.χ. το όνομα `Nikos`).

Οι **ακέραιες** μεταβλητές ανάλογα με τα `bits` που περιλαμβάνουν και το μέγιστο αριθμό που μπορούν να αποθηκεύσουν φαίνονται στον επόμενο πίνακα :

Τύπος	**bits**	**Ελάχιστη - Μέγιστη Τιμή**	
byte	8	-128	127
short	16	-32628	32627
int	32	$-2*10^{9}$	$2*10^{9}$
long	64	$-9*10^{18}$	$9*10^{18}$

Οι **πραγματικές** (**Κινητής Υποδιαστολής**) μεταβλητές ανάλογα με τα `bits` που περιλαμβάνουν συνολικά, τα `bits` της `mantissa` και το μέγιστο αριθμό που μπορούν να αποθηκεύσουν φαίνονται στον επόμενο πίνακα :

Τύπος	**Συνολικά bits**	**bits mantissa**	**Ελάχιστη -Μέγιστη Τιμή**
float	32	23	$-3.4*10^{38}$ $3.4*10^{38}$
double	64	52	$-1.7*10^{308}$ $1.7*10^{308}$

❖ Όλες οι μέθοδοι δέχονται `double` παραμέτρους.

Το σύνολο **χαρακτήρων** της `Java` είναι το `Unicode` και όχι το `ASCII`. Χρησιμοποιεί `16` bits και όχι `8`, όπως το σύνολο των χαρακτήρων `ASCII`, με τη δυνατότητα αποθήκευσης χαρακτήρων από όλες τις γλώσσες (από `0-65536`, ενώ οι χαρακτήρες από `0-127` είναι όπως και στις άλλες γλώσσες προγραμματισμού, σύνολο `ASCII`).

```
char ch;     // Δήλωση μεταβλητής που θα αποθηκεύσει χαρακτήρες

ch = 'X';    // Ανάθεση τιμής σε μεταβλητή που αποθηκεύει το χαρακτήρα X

ch = 88;    // Ανάθεση τιμής σε μεταβλητή που αποθηκεύει το αριθμητικό
              ισοδύναμο του χαρακτήρα X
```

Οι λογικές (**boolean**) μεταβλητές αποθηκεύουν τις τιμές **true-false**, σαν το αποτέλεσμα κάποιας σύγκρισης. Χρησιμοποιούνται σε εντολές ελέγχου και επανάληψης.

```
boolean b; // Δήλωση της λογικής μεταβλητής b
b = true; // Ανάθεση της τιμής true  στη λογική μεταβλητή b
System.out.println(10 > 9); // Εμφανίζει true
```

ΚΥΡΙΟΛΕΚΤΙΚΕΣ ΣΤΑΘΕΡΕΣ

Πολλές φορές χρειάζεται να χρησιμοποιήσουμε συγκεκριμένους αριθμούς, χαρακτήρες ή ονόματα σε εντολές εκχώρησης, σε εκφράσεις ή σε συγκρίσεις, όπως στο προηγούμενο παράδειγμα που δώσαμε στη μεταβλητή `num1` την τιμή `5`, τα οποία χαρακτηρίζονται σαν κυριολεκτικές σταθερές και μπορεί να είναι :

- Σταθερές ακεραίων, π.χ. `10, -100.` Είναι εξ ορισμού τύπου `int`. Αν θέλουμε να ορίσουμε σταθερά τύπου `long`, βάζουμε ένα `L` ή `l` στο τέλος, π.χ. `long a = 10L`.
- Σταθερές κινητής υποδιαστολής, π.χ. `2.35`. Είναι εξ ορισμού τύπου `double`. Για να ορίσουμε σταθερά τύπου `float` , βάζουμε ένα `F` ή `f` στο τέλος, π.χ. `float a=2.35F`.
- Σταθερές συμολοσειρές (`Strings`). Είναι αλυσίδες χαρακτήρων π.χ. “`kostas`” και χρησιμοποιούνται συνήθως σε εντολές `println`.

ΧΑΡΑΚΤΗΡΕΣ ΔΙΑΦΥΓΗΣ

Χρησιμοποιούνται σε εντολές `println` και είναι το “`\t`” για tab και το “`\n`” για αλλαγή γραμμής.

Παράδειγμα

Η εντολή `System.out.println(“a\tb\nc\t+d\n”);` θα έχει σαν αποτέλεσμα την παρακάτω εμφάνιση :

```
a  b
c  d
```

Οι χαρακτήρες απόστροφος και διπλά εισαγωγικά για να χρησιμοποιθούν ή να εκτυπωθούν πρέπει να προηγείται μια ανάποδη παύλα.

Παράδειγμα

```
ch1 =  ‘\’’;
ch2 = ’\”’;
```

ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ

Μαζί με τη δήλωση κάποιας μεταβλητής μπορούμε να εκχωρήσουμε και κάποια αρχική τιμή.

Παράδειγμα

```
int a;   // Δήλωση Μεταβλητής a
a = 10; // Ανάθεση του 10 σαν αρχική τιμή στη μεταβλητή a
```

Το ίδιο αποτέλεσμα έχουμε με την εντολή

```
int a = 10;   // Δήλωση - Ανάθεση του 10 σαν αρχική τιμή στη μεταβλητή a
```

❖ Αρχικοποίηση μεταβλητών μπορεί να γίνει σε περισσότερες από 1 μεταβλητές.

Παράδειγμα

```
int a = 5, b = 6, sum; // Αρχικοποίηση a, b, δήλωση sum
```

ΔΥΝΑΜΙΚΗ ΑΡΧΙΚΟΠΟΙΗΣΗ

Μπορεί να γίνει δήλωση μιας μεταβλητής στο σημείο του προγράμματος που θέλουμε να **υπολογίσουμε** και να **αποθηκεύσουμε** σε μια μεταβλητή το αποτέλεσμα μιας αριθμητικής έκφρασης.

Παράδειγμα

```
int sum = a + b;
double mo = sum/2;
```

❖ Προσοχή, η μεταβλητή αν υπάρχει σε block (if, while, for) είναι τοπική, έχει εμβέλεια μόνο στο block.

ΑΝΑΘΕΣΗ-ΕΚΧΩΡΗΣΗ ΤΙΜΗΣ

Γενικός Τύπος : <όνομα_μεταβλητής> = <έκφραση>;

Παράδειγμα

```
a = 10; // Ανάθεση του 10 σαν αρχική τιμή στη με ταβλητή a
```

❖ Υπάρχει η δυνατότητα στη Java να έχουμε αλυσίδα εκχωρήσεων.

Παράδειγμα

```
x = y = z = 10; // Το z παίρνει την τιμή 10, την οποια παίρνει το y και μετά το x
```

ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Οι τελεστές πράξεων που χρησιμοποιούνται στις αριθμητικές εκφράσεις είναι οι παρακάτω :

`+, -, *` : Τελεστές για πρόσθεση, αφαίρεση και πολλαπλασιασμό, όπως και στα μαθηματικά.

`/` : Τελεστής για Διαίρεση

- Με ακέραιους γίνεται ακέραια διαίρεση

Παράδειγμα

```
int a = 10;
int b = a/3;        // Αποτέλεσμα της διαίρεσης → b = 3

float a = 10.0f;
float b = a/3;      // Αποτέλεσμα της διαίρεσης → b = 3.333….
```

`%` : Τελεστής για το υπόλοιπο της διαίρεσης 2 αριθμών, ακέραιων ή κινητής υποδιαστολής.

Παράδειγμα

```
int a = 10;
int b = a%3;   //  Αποθηκεύεται στη μεταβλητή b το υπόλοιπο της διαίρεσης a/3 = 1

float a = 10.5f;
float b = a%3; //Αποθηκεύεται στη μεταβλητή b το υπόλοιπο της διαίρεσης a/3 =  1.5
```

1. 3 Πρώτη Τροποποίηση Προγράμματος 1.1

Να τροποποιηθεί το πρόγραμμα 1.1, το οποίο θα δίνει τις τιμές 5 και 6 σε δύο μεταβλητές **num1** και **num2** και θα υπολογίζει και θα εμφανίζει το άθροισμά τους **sum** και το μέσο όρο **mo**, ώστε να μπορεί να εμφανίζει το σωστό αποτέλεσμα (η μεταβλητή `sum` να δηλωθεί τύπου `double`, ώστε να ΜΗ γίνει ακέραια διαίρεση στο μέσο όρο).

ΠΡΟΓΡΑΜΜΑ

```
public class SumMO2 {
/*
Πρόγραμμα που δίνει τις τιμές 5 και 6 σε 2 ακέραιες μεταβλητές και βρίσκει
και εμφανίζει το άθροισμά τους και το Μέσο Όρο. To άθροισμα και ο Μέσος Όρος
είναι μεταβλητές τύπου double
*/
    public static void main(String[] args) {

        // Δήλωση των ακέραιων μεταβλητών num1, num2
        int num1, num2;

        // Δήλωση της πραγματικής μεταβλητής sum για το άθροισμα
        double sum;

        // Δήλωση της πραγματικής μεταβλητής mo για το Μέσο Όρο
        double mo;

        // Ανάθεση των τιμών 5 και 6 στις ακέραιες μεταβλητές num1, num2
        num1 = 5;
        num2 = 6;

        // Υπολογισμός του αθροίσματος sum
        sum = num1 + num2;

        // Υπολογισμός του Μέσου Όρου mo
        mo = sum/2;

        // Εμφάνιση  του αθροίσματος sum και του Μέσου Όρου mo
        System.out.println("Άθροισμα = " + sum + " Μέσος Όρος = " + mo);
        }
}
```

Έξοδος Προγράμματος

```
Άθροισμα = 11.0 Μέσος Όρος = 5.5
```

- Το Άθροισμα είναι `11.0` και ο Μέσος Όρος είναι `5.5`

ΤΕΛΕΣΤΕΣ ΣΥΝΤΟΜΩΝ ΕΚΧΩΡΗΣΕΩΝ

Υπάρχει η δυνατότητα στη Java να έχουμε σύντομες εκχωρήσεις .

Παράδειγμα

```
x = x + y;     Συντομευμένος τελεστής    x += y;
x = x – y;     Συντομευμένος τελεστής    x –= y;
x = x * y;     Συντομευμένος τελεστής    x *= y;
x = x/y;       Συντομευμένος τελεστής    x /= y;
x = x%y        Συντομευμένος τελεστής    x %= y;
```

ΤΕΛΕΣΤΕΣ ΠΡΟΣΑΥΞΗΣΗΣ-ΠΡΟΜΕΙΩΣΗΣ ++, --

Η Java διαθέτει τους παρακάτω τελεστές προσαύξησης ή προμείωσης :

```
x = x + 1;    τελεστές προσαύξησης     x++; ++x;
x = x – 1;    τελεστές προμείωσης      x--; --x;
```

- Αν το ++, -- βρίσκεται **πριν** τη μεταβλητή, **πρώτα** ενημερώνεται η τιμή της μεταβλητής και **μετά** χρησιμοποιείται.

Παράδειγμα

```
x = 10;
y = x++;    // Το y παίρνει την τιμή του x που ήταν 10, το x αυξάνεται ΜΕΤΑ και γίνεται
              11 ( x = 11, y = 10 )

x = 10;
y = ++x;    // ΠΡΩΤΑ αυξάνεται το x και γίνεται 11 και ΜΕΤΑ το y παίρνει την τιμή του x
              που έγινε 11 ( x = 11, y = 11 )
```

ΜΕΤΑΤΡΟΠΗ ΤΥΠΩΝ ΣΕ ΕΚΧΩΡΗΣΕΙΣ

Όταν σε μια αριθμητική έκφραση έχουμε μεταβλητές διαφορετικού τύπου γίνεται **διευρυμένη μετατροπή**, όπου ο τύπος της μεταβλητής στο δεξί μέρος της έκφρασης μετατρέπεται στον τύπο της μεταβλητής στο αριστερό μέρος της έκφρασης. Η μετατροπή γίνεται από το μικρότερο τύπο στο μεγαλύτερο με την παρακάτω σειρά:

byte → short → int → long → float → double

Για να γίνει διευρυμένη μετατροπή θα πρέπει ο τύπος προορισμού (αριστερά) να είναι μεγαλύτερος από τους τύπους των μεταβλητών στο δεξί μέρος της έκφρασης. Η διευρυμένη μετατροπή δεν ισχύει για τους τύπους boolean και char.

Εξαίρεση αποτελεί ο τύπος `char`, όπου μπορεί να αποθηκευτεί ένας ακέραιος `int` σε μεταβλητή τύπου `char`.

Παράδειγμα

```
char ch = 88; // Είναι το ίδιο με την εντολή char ch = 'X'
```

Στην περίπτωση που δεν ισχύουν τα παραπάνω ή αν θέλουμε να αποθηκεύσουμε το περιεχόμενο της μεταβλητής κάποιου τύπου σε μεταβλητή διαφορετικού τύπου χρησιμοποιούμε τη **διανομή – casting,** όπου πριν τη **μεταβλητή** ή την **έκφραση** (η οποία πρέπει να είναι μέσα σε **παρενθέσεις**, αν θέλουμε η διανομή να ισχύσει για το αποτέλεσμα της έκφρασης) βάζουμε μέσα σε **παρενθέσεις** τον τύπο στον οποίο θέλουμε να μετατραπεί η τιμή της μεταβλητής ή της έκφρασης. Σ' αυτές τις περιπτώσεις χρειάζεται προσοχή, γιατί μπορεί να μην αποθηκευθεί σωστά η τιμή μιας μεταβλητής σε κάποια άλλη ή να μή χωράει.

Παράδειγμα

```
double d = 10.5;
int i1 = (int)d;          // Αποθηκεύεται στη μεταβλητή i1 το 10, χάνεται το 0.5
int i2 = (int)(d/3);      // Αποθηκεύεται στη μεταβλητή i2 το 3, χάνεται το 0.5
double d1 = (double)i1;   // Αποθηκεύεται στη μεταβλητή d1 το 10.0

byte b1 = (byte)i2;       // Αποθηκεύεται στη μεταβλητή b1 το 3

byte b2 = (byte)(i2+200)//Αποθηκεύεται στη μεταβλητή b2 το 1, γιατί η τιμή
                          i2+200=203 είναι μεγαλύτερη από τη μέγιστη τιμή 127
                          που μπορεί να αποθηκεύσει μια μεταβλητή τύπου byte

byte b = 88;
char ch1 = (char) b;  // Αποθηκεύεται στη μεταβλητή ch1 ο χαρακτήρας 'X'

int i = 88;
char ch2 = (char) i;  // Αποθηκεύεται στη μεταβλητή ch2 ο χαρακτήρας 'X'
```

1. 4 Δεύτερη Τροποποίηση του Προγράμματος 1.1

Να τροποποιηθεί το πρόγραμμα 1.1, το οποίο θα δίνει τις τιμές 5 και 6 σε δύο μεταβλητές **num1** και **num2** και θα υπολογίζει και θα εμφανίζει το άθροισμά τους **sum** και το μέσο όρο **mo**, ώστε να μπορεί να εμφανίζει το σωστό αποτέλεσμα (**η μεταβλητή sum να δηλωθεί τύπου int, αλλά να γίνει μετατροπή - casting στον υπολογισμό του μέσο όρου, ώστε να ΜΗ γίνει ακέραια διαίρεση**).

ΠΡΟΓΡΑΜΜΑ

```
public class SumMO3 {
/*
Πρόγραμμα που δίνει τις τιμές 5 και 6 σε 2 ακέραιες μεταβλητές και βρίσκει
και εμφανίζει το άθροισμά τους και το Μέσο Όρο, ο οποίος είναι μεταβλητή
τύπου double. Η μεταβλητή sum θα δηλωθεί τύπου int, αλλά θα γίνει μετατροπή -
casting στον υπολογισμό του μέσο όρου, ώστε να ΜΗ γίνει ακέραια διαίρεση
*/
    public static void main(String[] args) {

        // Δήλωση των ακέραιων μεταβλητών num1, num2
        int num1, num2;

        // Δήλωση της ακέραιας μεταβλητής sum για το άθροισμα
        int sum;

        // Δήλωση της πραγματικής μεταβλητής mo για το Μέσο Όρο
        double mo;

        // Ανάθεση των τιμών 5 και 6 στις ακέραιες μεταβλητές num1, num2
        num1 = 5;
        num2 = 6;

        // Υπολογισμός του αθροίσματος sum
        sum = num1 + num2;

        // Υπολογισμός  Μέσου Όρου mo με μετατροπή-casting του sum σε double
        mo = (double)sum/2;

        // Εμφάνιση  του αθροίσματος sum και του Μέσου Όρου mo
        System.out.println("Άθροισμα = " + sum + " Μέσος Όρος = " + mo);
        }
}
```

Έξοδος Προγράμματος

```
Άθροισμα = 11 Μέσος Όρος = 5.5
```

- Παρόλο που το άθροισμα είναι ακέραιος (= 11) με τη μετατροπή - casting του αθροίσματος στον υπολογισμό του μέσο όρου (**mo = (double)sum/2**) ΔΕ γίνεται ακέραια διαίρεση του 11/2 και ΔΕΝ αποκόπτεται το δεκαδικό μέρος, οπότε ο Μέσος Όρος = 5.5.

ΥΠΟΛΟΓΙΣΜΟΣ ΕΚΦΡΑΣΕΩΝ

Στον υπολογισμό των εκφράσεων, οι τιμές των μεταβλητών προάγονται σε αντίστοιχες τιμές των μεταβλητών μεγαλύτερων τύπων με την παρακάτω σειρά :

char, byte, short → `int` → `long` → `float` → `double`

- **Προσοχή σε πράξεις με μεταβλητές τύπου `char`, `byte`, `short` σε εκφράσεις στο δεύτερο μέλος της εντολής ανάθεσης τιμής χρειάζεται διανομή, γιατί οι μεταβλητές των παραπάνω τύπων προάγονται αυτόματα σε `int`.**

Παράδειγμα

```
byte b = 10;
int i = b * b;      // i = 100, byte → μετατροπή σε int

b = b * b;          // Πρόβλημα, b*b → μετατροπή σε int
b = (byte)(b*b);    // i = 100, byte → μετατροπή σε int → (byte)
                       μετατροπή σε byte
short s = 10;
s = s + 1;             // Πρόβλημα, s+1 → μετατροπή σε int
s = (short)(s + 1);    // s+1 → μετατροπή σε int → (short)
                       μετατροπή σε short

char ch = 'a';
ch = ch+1;             // Πρόβλημα, ch+1 → μετατροπή σε int
ch =(char)(ch+1);      // ch+1 → μετατροπή σε int → (char) μετατροπή
                       σε char
```

- Η ακέραια διαίρεση, αν είναι ατελής, χρειάζεται διανομή.

Παράδειγμα

```
int i = 10/3;                  // Αποτέλεσμα i = 3
double d = (double)(10/3);     // Αποτέλεσμα i = 3.0, η διανομή γίνεται στο
                                  αποτέλεσμα της ακέραιας διαίρεσης 10/3 = 3
d = (double)10/3;              // Αποτέλεσμα i = 3.333, η διανομή γίνεται
                                  στο 10 → μετατροπή σε 10.0/3 = 3.333
```

2 ΕΝΤΟΛΕΣ ΕΛΕΓΧΟΥ

Στα πιο πολλά προγράμματα απαιτούνται να γίνονται κάποιοι έλεγχοι γαι το αν μπορεί να γίνει μια πράξη (π.χ. αν ο διαιρέτης δεν είναι μηδέν), αν ένας αριθμός ή όνομα υπάρχει σε μια λίστα, αν ένας βαθμός που θα εισαχθεί σε ένα πρόγραμμα είναι μια αποδεκτή τιμή (από 1 μέχρι 10) κ.λ.π. πριν προβούμε στην εκτέλεση κάποιας ή κάποιων εντολών. Για να γίνει ο έλεγχος θα πρέπει να γίνει κάποια σύγκριση, π.χ. ο διαιρέτης δεν είναι ίσος με το μηδέν, ο βαθμός είναι μεταξύ του 0 και του 10 κ.λ.π..

Η πιο απλή μορφή σύγκρισης – εντολής ελέγχου έχει τη μορφή :

```
if (<συνθήκη>)
   εντολή;

if(<συνθήκη>){
   block εντολών;
   }
```

όπου ελέγχεται η συνθήκη αν είναι αληθής. **Αν** ισχύει η συνθήκη, εκτελείται η εντολή ή οι εντολές (οι οποίες θα πρέπει να περικλείονται σε άγγιστρα, αν είναι περισσότερες από μια) μετά το `if`. Αν ΔΕΝ ισχύει η συνθήκη, δε γίνεται τίποτα.

Η συνθήκη μπορεί να περιλαμβάνει μεταβλητές ή αριθμητικές εκφράσεις, ενώ οι συγκρίσεις γίνονται με τους παρακάτω Σχεσιακούς Τελεστές ή Τελεστές Σύγκρισης :

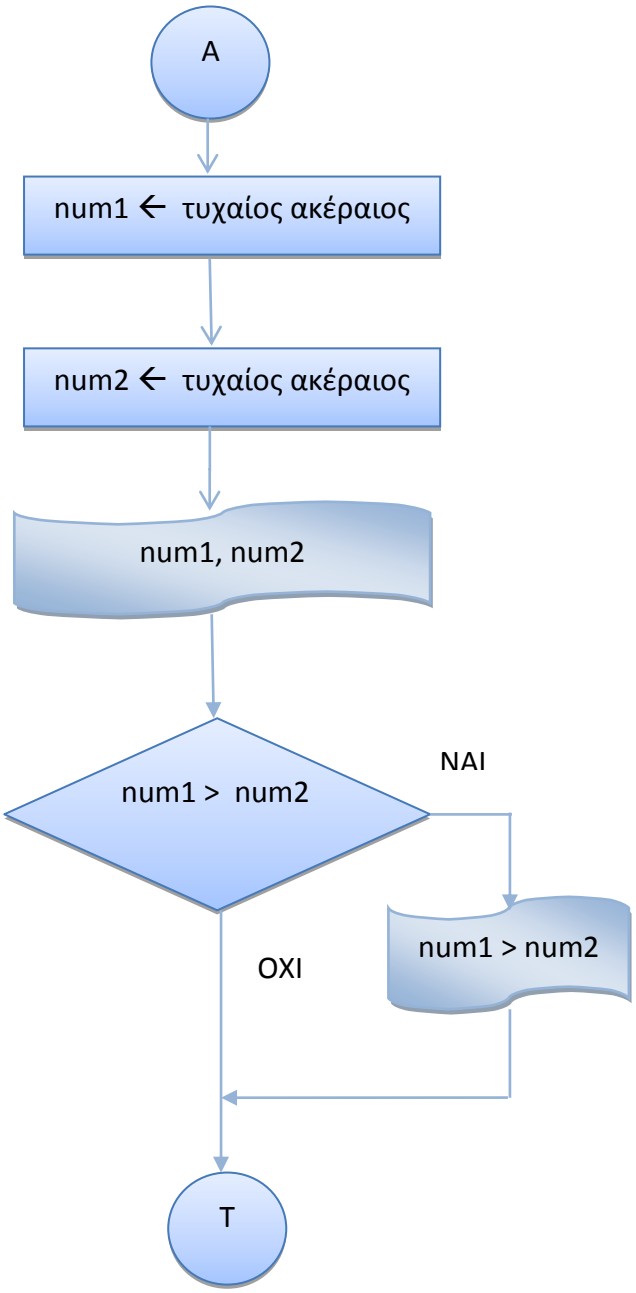
Σύγκριση	**Σύμβολο**	**Σχεσιακός Τελεστής**
Μικρότερο	<	<
Μικρότερο ή Ίσο	≤	<=
Ίσο	=	==
Μεγαλύτερο ή Ίσο	≥	>=
Μεγαλύτερο	>	>
Διάφορο	≠	!=

- Αν η συνθήκη περιέχει **`boolean`** μεταβλητή **ΔΕΝ ΧΡΕΙΑΖΕΤΑΙ** να ελεγχθεί αν η τιμή της είναι **`true`** ή **`false`**. Αντί **`του if (b == true)`** μπορούμε να γράψουμε **`if(b)`.**

2. 1 Ένα Απλό Πρόγραμμα με την Εντολή Ελέγχου if

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα δημιουργεί 2 τυχαίους ακέραιους αριθμούς **num1** και **num2** μεταξύ του 1 και 10. Θα εμφανίζει τις τιμές τους και θα ελέγχει αν ο αριθμός **num1** είναι μεγαλύτερος του **num2,** οπότε σ' αυτή την περίπτωση θα εμφανίζει τη σχέση τους, **num1 > num2 .**

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ 2 τυχαίους ακέραιους αριθμούς **num1** και **num2**

2. Εμφανίζω τις τιμές των **num1** και **num2**

3. **Αν** το num1 > num2 **τότε**

   Εμφανίζω το **num1 > num2**

ΠΡΟΓΡΑΜΜΑ

```
public class If1 {
/*
Δίνονται 2 τυχαίοι ακέραιοι αριθμοί num1, num2 μεταξύ του 1 και 10. Να
ελεγχθεί αν ο αριθμός num1 είναι μεγαλύτερος του num2, οπότε σ' αυτή την
περίπτωση θα εμφανίζει τη σχέση τους, num1  > num2
*/
public static void main(String[] args) {
        // Δήλωση των ακέρασιων μεταβλητών num1, num2
        int num1, num2;

        // Δημιουργία 2 τυχαίων ακεραίων num1, num2 με τιμές στο 0 - 10
        num1 = (int)(Math.random()*10) + 1;
        num2 = (int)(Math.random()*10) + 1;

        // Εμφάνιση των τιμών του num1 και num2
        System.out.println("num1 = " + num1 + "\nnum2 = " + num2);

        // Έλεγχος αν ο num1 είναι μεγαλύτερος του num2, εμφάνιση της σχέσης
        if (num1 > num2) System.out.println( num1 + " > " + num2);
    }
}
```

Έξοδος Προγράμματος

```
num1 = 3
num2 = 1
3 > 1

num1 = 5
num2 = 9
```

- Η μέθοδος `random()` δημιουργεί έναν τυχαίο **πραγματικό** αριθμό μεταξύ του `0` και του `1` και ανήκει στην κλάση `Math`, η οποία περιέχει μεθόδους με τις οποίες μπορούμε να κάνουμε βασικές πράξεις με εκθετικά, λογαρίθμους, τετραγωνικές ρίζες και τριγωνομετρικές συναρτήσεις, όπως π.χ. τη `sin(x)` για τον υπολογισμό του ημιτόνου του `x`, την `cos(x)` για τον υπολογισμό του συνημιτόνου του `x`, την `sqrt(x)` για τον υπολογισμό της τετραγωνικής ρίζας του `x`, την `exp(x)` για τον υπολογισμό του του e^x, και την `pow(x,y)` για τον υπολογισμό του x^y.

Μια άλλη μορφή σύγκρισης – εντολής ελέγχου έχει τη μορφή :

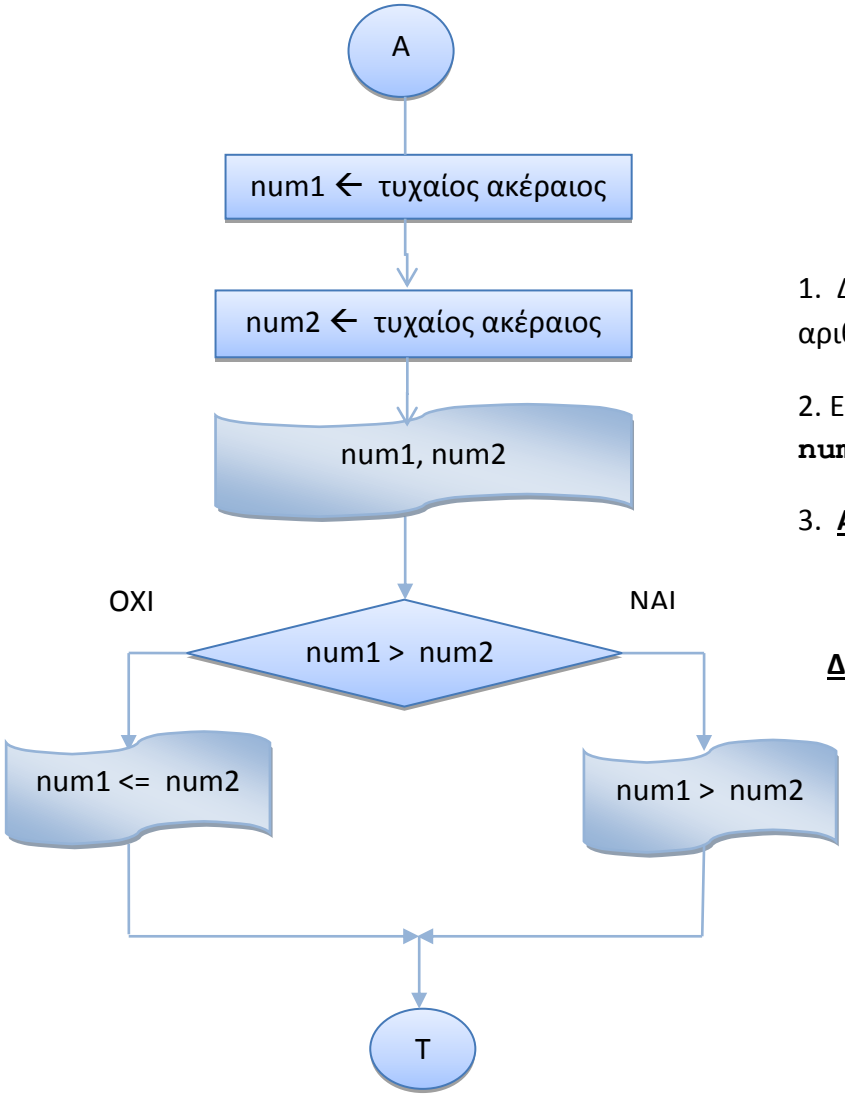
```
if (<συνθήκη>)
  εντολή-1; ή >){ block εντολών-1; }
else
  εντολή-2; ή >){ block εντολών-2; }
```

όπου ελέγχεται η συνθήκη αν είναι αληθής. **Αν ισχύει** η συνθήκη, εκτελείται η εντολή-1 ή οι εντολές-1, μετά το `if`. Αν **ΔΕΝ** ισχύει η συνθήκη, εκτελείται η εντολή-2 ή οι εντολές-2, μετά το `else`.

2. 2 Τροποποίηση Προγράμματος 2.1 με την Εντολή Ελέγχου if - else

Να τροποποιηθεί το Πρόγραμμα 2.1, ώστε να δημιουργεί 2 τυχαίους ακέραιους αριθμούς **num1** και **num2** μεταξύ του `1` και `10`. Θα εμφανίζει τις τιμές τους και θα ελέγχει αν ο αριθμός **num1** είναι μεγαλύτερος του **num2,** οπότε σ' αυτή την περίπτωση θα εμφανίζει τη σχέση τους, **num1 > num2**. Αν δεν ισχύει η συνθήκη, θα εμφανίζει **num1 <= num2 .**

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ 2 τυχαίους ακέραιους αριθμούς **num1** και **num2**

2. Εμφανίζω τις τιμές των **num1** και **num2**

3. **Αν** το num1 > num2 **τότε**

   Εμφανίζω το **num1 > num2**

Διαφορετικά

   Εμφανίζω το **num1 <= num2**

ΠΡΟΓΡΑΜΜΑ

```
public class IfElse {
/*
Δίνονται 2 τυχαίοι ακέραιοι αριθμοί num1, num2 μεταξύ του 1 και 10.
Να ελεγχθεί αν ο αριθμός num1 είναι μεγαλύτερος του num2, οπότε σ' αυτή
την
περίπτωση θα εμφανίζει τη σχέση τους, num1  > num2. Αν δεν ισχύει η
συνθήκη,
θα εμφανίζει num1  <= num2.
*/
public static void main(String[] args) {
        // Δήλωση των ακέραιων μεταβλητών num1, num2
        int num1, num2;

        // Δημιουργία 2 τυχαίων ακεραίων num1, num2 με τιμές στο 0 - 10
        num1 = (int)(Math.random()*10) + 1;
        num2 = (int)(Math.random()*10) + 1;

        // Εμφάνιση των τιμών του num1 και num2
        System.out.println("num1 = " + num1 + "\nnum2 = " + num2);

        // Έλεγχος αν ο num1 είναι μεγαλύτερος του num2, εμφάνιση της
σχέσης
        if (num1 > num2)
            System.out.println( num1 + " > " + num2);
        else
            System.out.println( num1 + " <= " + num2);
    }
}
```

Έξοδος Προγράμματος

```
num1 = 3
num2 = 1
3 > 1

num1 = 2
num2 = 5
2 <= 5
```

Όταν σε έναν έλεγχο υπάρχουν **περισσότερα από δύο ενδεχόμενα** μπορούμε να χρησιμοποιήσουμε τη σκάλα `if – else – if`. Σ' αυτήν την περίπτωση υπάρχουν περισσότερες συνθήκες να ελεγχθούν και εκτελούνται οι αντίστοιχες εντολές.

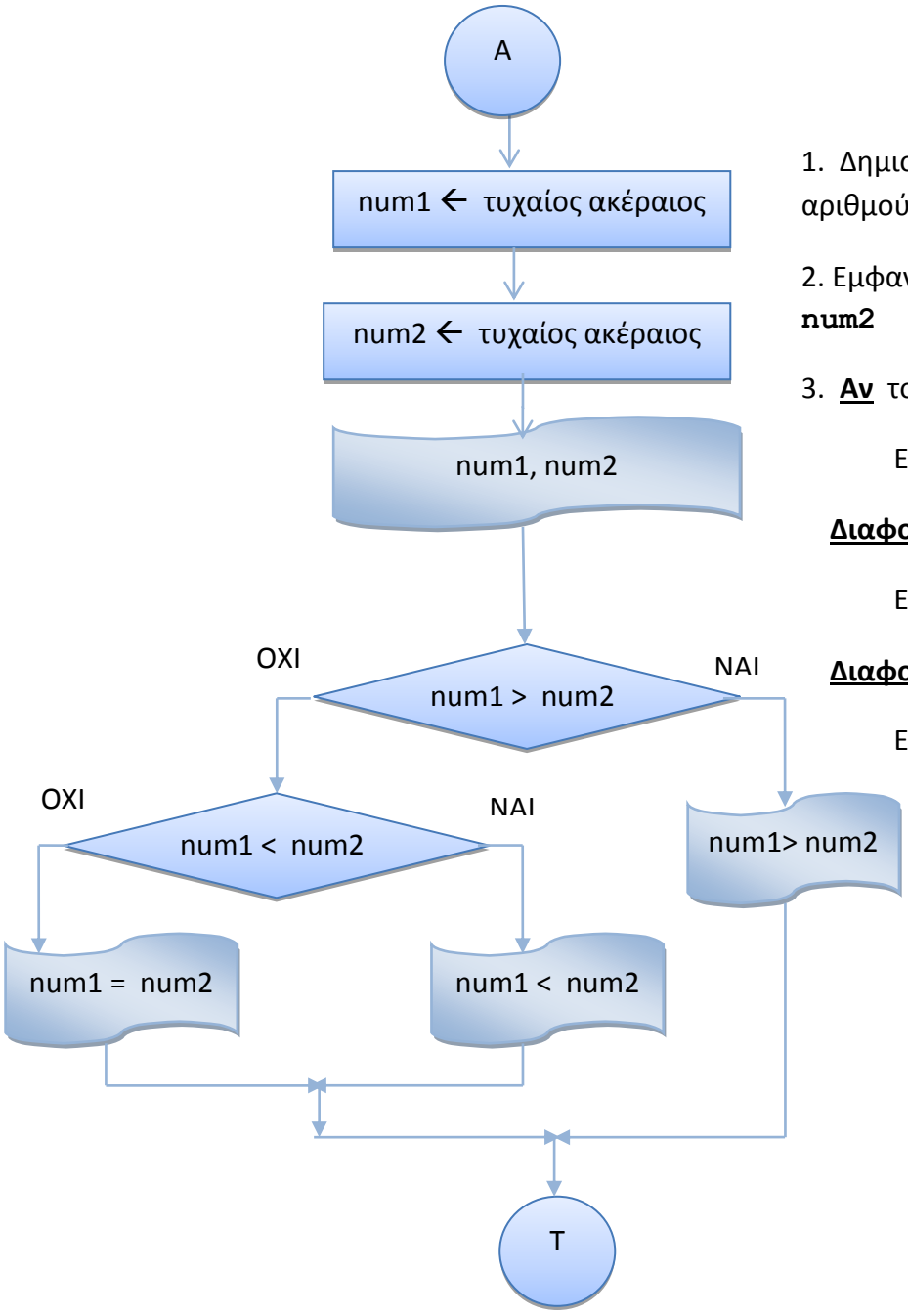
```
if (<συνθήκη-1>)
  εντολή-1; ή >){ block εντολών-1; }
else if (<συνθήκη-2>)
  εντολή-2; ή >){ block εντολών-2; }
else
  εντολή-3; ή >){ block εντολών-3; }
```

όπου ελέγχεται η συνθήκη-1. **Αν** ισχύει η συνθήκη-1, εκτελείται η εντολή-1 ή οι εντολές-1, μετά το `if`. **Αν** ΔΕΝ ισχύει η συνθήκη-1, ελέγχεται η συνθήκη-2. Αν ισχύει, εκτελείται η εντολή-2 ή οι εντολές-2, μετά το `else if`. **Αν** ΔΕΝ ισχύει η συνθήκη-2, εκτελείται η εντολή-3 ή οι εντολές-3, μετά το `else`.

2.3 Τροποποίηση Προγράμματος 2.1 με την Εντολή Ελέγχου if – else -if

Να τροποποιηθεί το Πρόγραμμα 2.1, ώστε να δημιουργεί 2 τυχαίους ακέραιους αριθμούς **num1** και **num2** μεταξύ του 1 και 10. Θα εμφανίζει τις τιμές τους και θα ελέγχει αν ο αριθμός **num1** είναι μεγαλύτερος του **num2,** οπότε σ' αυτή την περίπτωση θα εμφανίζει τη σχέση τους, **num1 > num2**. Διαφορετικά, θα ελέγχει αν ο αριθμός **num1** είναι μικρότερος του **num2**, οπότε θα εμφανίζει **num1 < num2 .** Αν δεν ισχύει ούτε αυτό, θα εμφανίζει **num1 = num2 .**

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ 2 τυχαίους ακέραιους αριθμούς **num1** και **num2**

2. Εμφανίζω τις τιμές των **num1** και **num2**

3. **Αν** το num1 > num2 **τότε**

   Εμφανίζω το **num1 > num2**

Διαφορετικά Αν το num1 < num2

   Εμφανίζω το **num1 < num2**

Διαφορετικά

   Εμφανίζω το **num1 = num2**

ΠΡΟΓΡΑΜΜΑ

```
public class IfElseIf {
/*
Δίνονται 2 τυχαίοι ακέραιοι αριθμοί num1, num2 μεταξύ του 1 και 10.
Να ελεγχθεί αν ο αριθμός num1 είναι μεγαλύτερος του num2, οπότε σ' αυτή την
περίπτωση θα εμφανίζει τη σχέση τους, num1  > num2. Αν δεν ισχύει η συνθήκη,
θα ελέγχει αν ο αριθμός num1 είναι μικρότερος του num2, οπότε θα εμφανίζει
num1  <  num2. Αν δεν ισχύει ούτε αυτό, θα εμφανίζει num1  =  num2.
*/
    public static void main(String[] args) {
     // Δήλωση των ακέραιων μεταβλητών num1, num2
        int num1, num2;

        // Δημιουργία 2 τυχαίων ακεραίων num1, num2 με τιμές στο 0 - 10
        num1 = (int)(Math.random()*10) + 1;
        num2 = (int)(Math.random()*10) + 1;

        // Εμφάνιση των τιμών του num1 και num2
        System.out.println("num1 = " + num1 + "\nnum2 = " + num2);

        // Έλεγχος αν ο num1 είναι μεγαλύτερος του num2, εμφάνιση της σχέσης
        if (num1 > num2)
            System.out.println( num1 + " > " + num2);
        else if (num1 < num2)
            System.out.println( num1 + " < " + num2);
        else
            System.out.println( num1 + " = " + num2);
     }
}
```

Έξοδος Προγράμματος

```
num1 = 4
num2 = 1
4 > 1

num1 = 4
num2 = 6
4 < 6

num1 = 4
num2 = 4
4 = 4
```

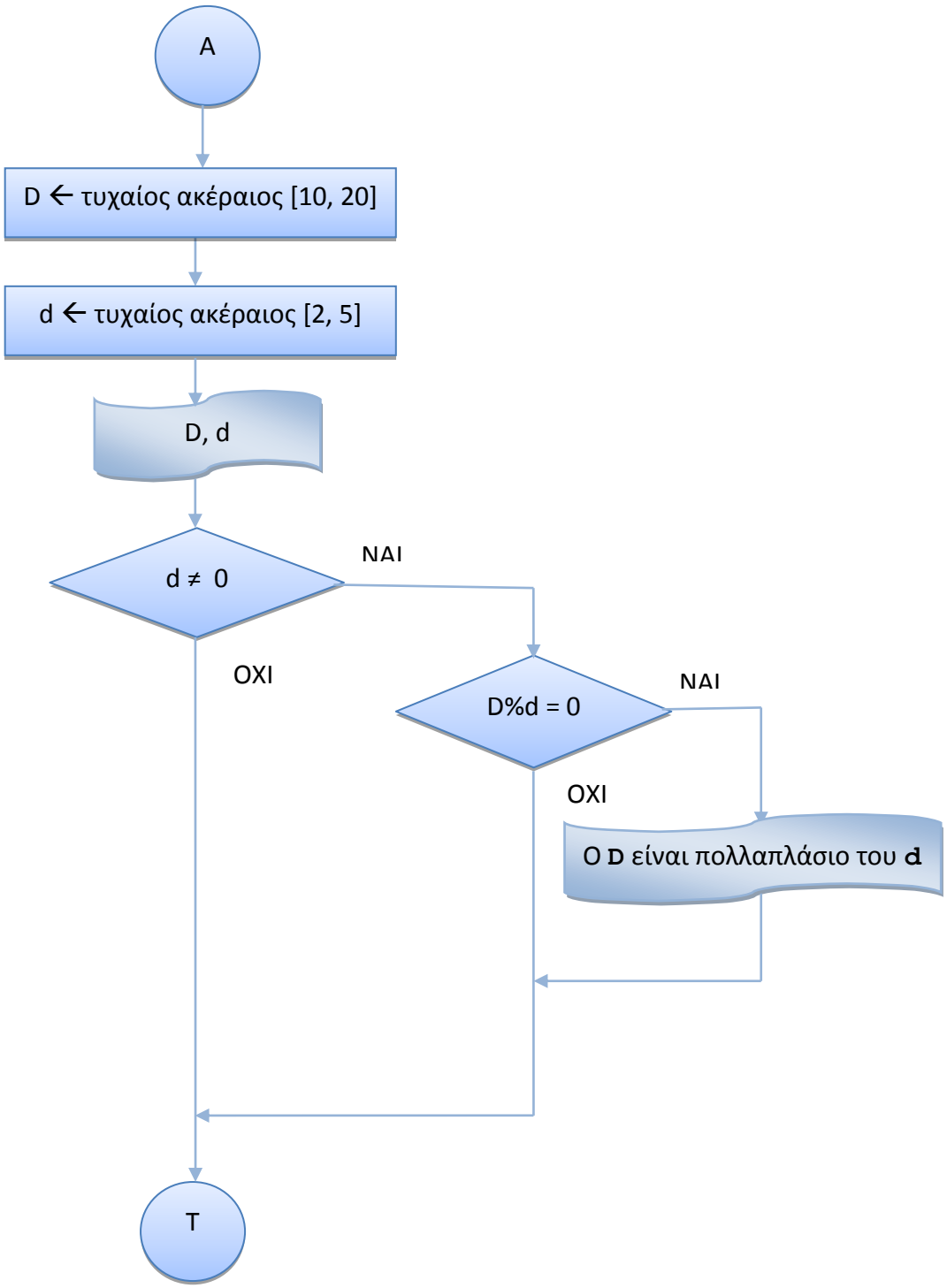
Παρατήρηση

- Μπορεί στο τμήμα του `if` να υπάρχει και δεύτερο `if`, δηλαδή να ελεγχθεί και κάποια άλλη συνθήκη, όπως φαίνεται στο επόμενο παράδειγμα :

2.4 Πρόγραμμα με την Εντολή Ελέγχου if – if

Να γραφεί Αλγόριθμος/Πρόγραμμα, το οποίο θα δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το **D** μεταξύ του 10 και 20 και το **d** μεταξύ του 0 και 5 και θα εμφανίζει τις τιμές τους. Αν ο αριθμός **d** είναι διάφορος του μηδενός, θα ελέγχει αν ο αριθμός **d** διαιρεί ΑΚΡΙΒΩΣ τον αριθμό **D**, οπότε και θα εμφανίζει το μήνυμα "Ο **D** είναι πολλαπλάσιο του **d** ".

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο D μεταξύ του 10 και 20
2. Δημιουργώ έναν τυχαίο ακέραιο d μεταξύ του 0 και 5
3. Εμφανίζω την τιμή των D και d
4. **Αν** ο αριθμός **d** είναι διάφορος του μηδενός (d ≠ 0)
   Αν ο αριθμός **d** διαιρεί ΑΚΡΙΒΩΣ τον αριθμό **D**
   Εμφανίζω το μήνυμα "Ο **D** είναι πολλαπλάσιο του **d** "

ΠΡΟΓΡΑΜΜΑ

```
public class IfIfThenFactor {
/*
Πρόγραμμα, το οποίο  δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το D  μεταξύ
του 10 και 20 και το  d  μεταξύ του 0 και 5 και  εμφανίζει τις τιμές τους. Αν
ο αριθμός d  είναι διάφορος του μηδενός, θα ελέγχει αν ο αριθμός d  διαιρεί
ΑΚΡΙΒΩΣ τον αριθμό D, οπότε και θα εμφανίζει το μήνυμα "Ο D είναι πολλαπλάσιο
του d ".
*/
    public static void main(String[] args) {
        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 10 και 20
        int D = (int)(Math.random()*11) + 10;

        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 0 και5
        int d = (int)(Math.random()*5);

        // Εμφάνιση της τιμής του D και d
        System.out.println("D = " + D + " d =  " + d);

        // Έλεγχος αν ο d είναι διάφορος του 0
        if (d != 0)
          // Έλεγχος αν ο αριθμός d διαιρεί ΑΚΡΙΒΩΣ τον αριθμό D
          if (D % d == 0)
           System.out.println("Ο " + D + " είναι πολλαπλάσιο του " + d );
   }
}
```

Έξοδος Προγράμματος

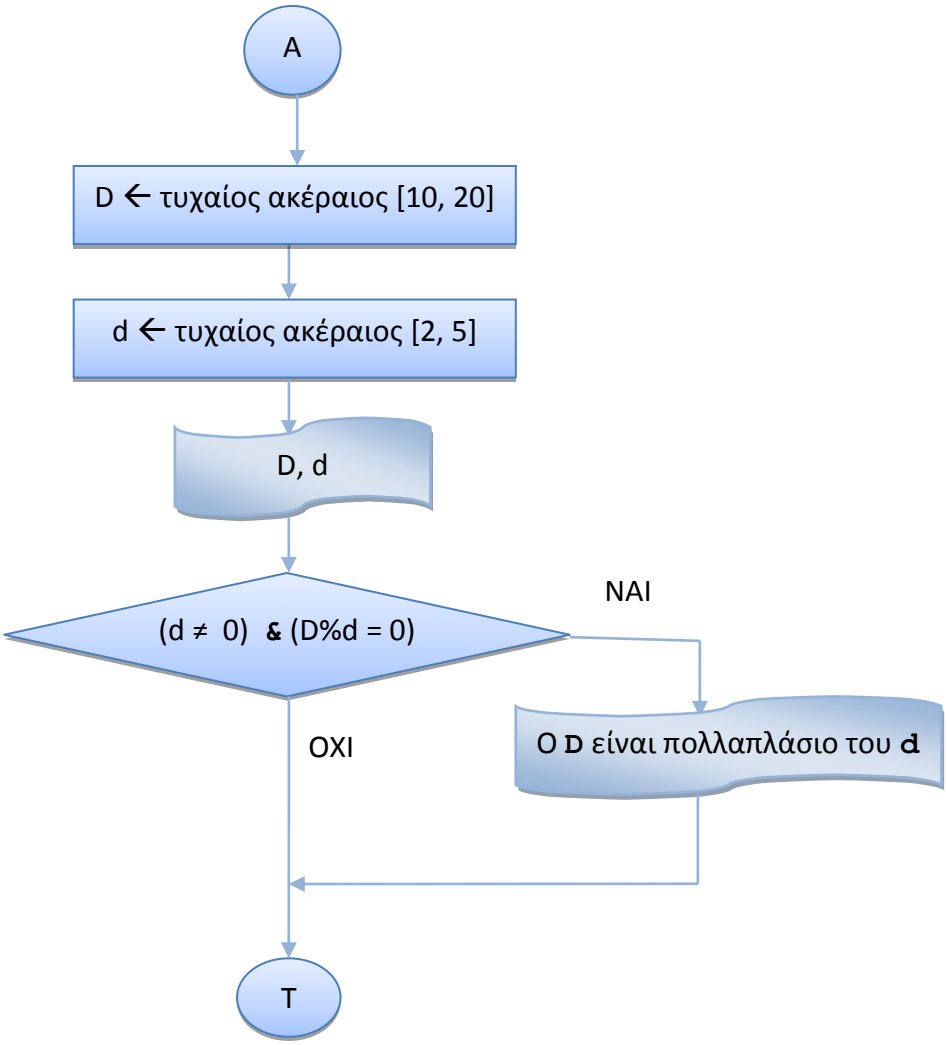
```
D = 18 d =  6
Ο 18 είναι πολλαπλάσιο του 6

D = 15 d =  4

D = 12 d =  0
```

❖ Όταν έχουμε ένα **if** χωρίς **else** αμέσως μετά από ένα άλλο **if**, μπορούμε αντί των δύο **if** να χρησιμοποιήσουμε ένα, αλλά με **διπλή συνθήκη**. Οι δύο συνθήκες μπορούν να συνδεθούν με το **&**, ώστε η συνολική συνθήκη να είναι αληθής, αν είναι και οι δύο επί μέρους συνθήκες αληθείς. Στην προκείμενη περίπτωση, θα μπορούσαμε να ελέγξουμε με μια εντολή **if** αν ισχύουν και οι δύο συνθήκες, δηλαδή ο αριθμός **d** είναι διάφορος του μηδενός και ο αριθμός **d** διαιρεί ΑΚΡΙΒΩΣ τον αριθμό **D**.

2.4.1 Το Πρόγραμμα 2.4 με Διπλή Συνθήκη στην Εντολή Ελέγχου if

Να τροποποιηθεί το Πρόγραμμα 2.4, ώστε να δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το **D** μεταξύ του 10 και 20 και το **d** μεταξύ του 0 και 5 και να εμφανίζει τις τιμές τους. Αν (ο αριθμός **d** είναι διάφορος του μηδενός) και (ο αριθμός **d** διαιρεί ΑΚΡΙΒΩΣ τον αριθμό **D**), θα εμφανίζει το μήνυμα "Ο **D** είναι πολλαπλάσιο του **d** ".

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο `D` μεταξύ του `10` και `20`
2. Δημιουργώ έναν τυχαίο ακέραιο `d` μεταξύ του `0` και `5`
3. Εμφανίζω την τιμή του D και d
4. **Αν** (ο **d** είναι διάφορος του μηδενός - d ≠ 0) **&** (ο **d** διαιρεί ΑΚΡΙΒΩΣ τον **D**)
   Εμφανίζω το μήνυμα "Ο **D** είναι πολλαπλάσιο του **d** "

ΠΡΟΓΡΑΜΜΑ

```
public class IfFactorLogic {
/*
Πρόγραμμα, το οποίο  δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το D  μεταξύ
του 10 και 20 και το  d  μεταξύ του 0 και 5 και  εμφανίζει τις τιμές τους. Αν
ο αριθμός d  είναι διάφορος του μηδενός και ο αριθμός d  διαιρεί ΑΚΡΙΒΩΣ τον
αριθμό D, θα εμφανίζει το μήνυμα "Ο D είναι πολλαπλάσιο του d ".
*/
    public static void main(String[] args) {
        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 10 και 20
        int D = (int)(Math.random()*11) + 10;

        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 0 και5
        int d = (int)(Math.random()*5);

        // Εμφάνιση της τιμής του D και d
        System.out.println("D = " + D + " d =  " + d);

        // Έλεγχος αν ο d είναι διάφορος του 0 & ο  d διαιρεί ΑΚΡΙΒΩΣ τον D
        if (d != 0 & D % d == 0)
           System.out.println("O " + D + " είναι πολλαπλάσιο του " + d );
    }
}
```

Έξοδος Προγράμματος

```
D = 10 d =  3

D = 14 d =  2
O 14 είναι πολλαπλάσιο του 2

D = 17 d =  0
Exception in thread "main" java.lang.ArithmeticException: / by zero
                at if_factor_logic.If_factor_logic.main(If_factor_logic.java:21)
Java Result: 1
```

- Στο προηγούμενο πρόγραμμα χρησιμοποιήσαμε το Λογικό Τελεστή **&** για να συνδέσουμε τις δύο συνθήκες, ώστε να ισχύουν ταυτόχρονα και οι δύο, αλλά δημιούργησε ένα **σφάλμα** (Εξαίρεση - διαίρεση με το μηδέν) για την τιμή **d = 0**, γιατί ελέγχονται και οι δύο συνθήκες `(d != 0 )` και `( D % d == 0)`, οπότε κάνει τη διαίρεση `D / d` για να βρεί το υπόλοιπο. Για να αποφύγουμε τέτοια προβλήματα, μπορούμε να χρησιμοποιήσουμε το Βραχυκυκλωμένο Λογικό Τελεστή **&&**, ο οποίος σε αντίθεση με τον Τελεστή **&** **ΔΕΝ ελέγχει τη δεύτερη συνθήκη**, **αν η πρώτη συνθήκη είναι ψευδής** (ο έλεγχος της δεύτερης συνθήκης πραγματικά δε χρειάζεται αν η πρώτη συνθήκη είναι ψευδής, αφού θα πρέπει να είναι αληθείς ΚΑΙ ΟΙ ΔΥΟ συνθήκες).

2.4.2 Το Πρόγραμμα 2.4 με Διπλή Συνθήκη στην Εντολή Ελέγχου if και το Βραχυκυκλωμένο Λογικό Τελεστή &&

Να τροποποιηθεί το Πρόγραμμα 2.7, ώστε να δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το **D** μεταξύ του 10 και 20 και το **d** μεταξύ του 0 και 5 και να εμφανίζει τις τιμές τους. Αν (ο αριθμός **d** είναι διάφορος του μηδενός) και (ο αριθμός **d** διαιρεί ΑΚΡΙΒΩΣ τον αριθμό **D**), θα εμφανίζει το μήνυμα "Ο **D** είναι πολλαπλάσιο του **d** ". Να χρησιμοποιηθεί στη διπλή συνθήκη ο Βραχυκυκλωμένος Λογικός Τελεστής **&&**.

ΠΡΟΓΡΑΜΜΑ

```
public class IfFactorLogicShort {
/*
Πρόγραμμα, το οποίο  δημιουργεί δύο τυχαίους ακέραιους αριθμούς, το D  μεταξύ
του 10 και 20 και το  d  μεταξύ του 0 και 5 και  εμφανίζει τις τιμές τους. Αν
ο αριθμός d  είναι διάφορος του μηδενός και ο αριθμός d  διαιρεί ΑΚΡΙΒΩΣ τον
αριθμό D, θα εμφανίζει το μήνυμα "Ο D είναι πολλαπλάσιο του d ".
Να χρησιμοποιηθεί στη διπλή συνθήκη ο Βραχυκυκλωμένος Λογικός Τελεστής &&.
*/
    public static void main(String[] args) {
        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 10 και 20
        int D = (int)(Math.random()*10) + 10;

        // Δημιουργία τυχαίου ακέραιου αριθμού D μεταξύ του 0 και5
        int d = (int)(Math.random()*5 );

        // Εμφάνιση της τιμής του D και d
        System.out.println("D = " + D + " d =  " + d);

        // Έλεγχος αν ο d είναι διάφορος του 0 & ο  d διαιρεί ΑΚΡΙΒΩΣ τον D
        if (d != 0 && D % d == 0)
           System.out.println("Ο " + D + " είναι πολλαπλάσιο του " + d );
    }
}
```

Έξοδος Προγράμματος

```
D = 17 d =  2

D = 12 d =  0

D = 18 d =  3
Ο 18 είναι πολλαπλάσιο του 3
```

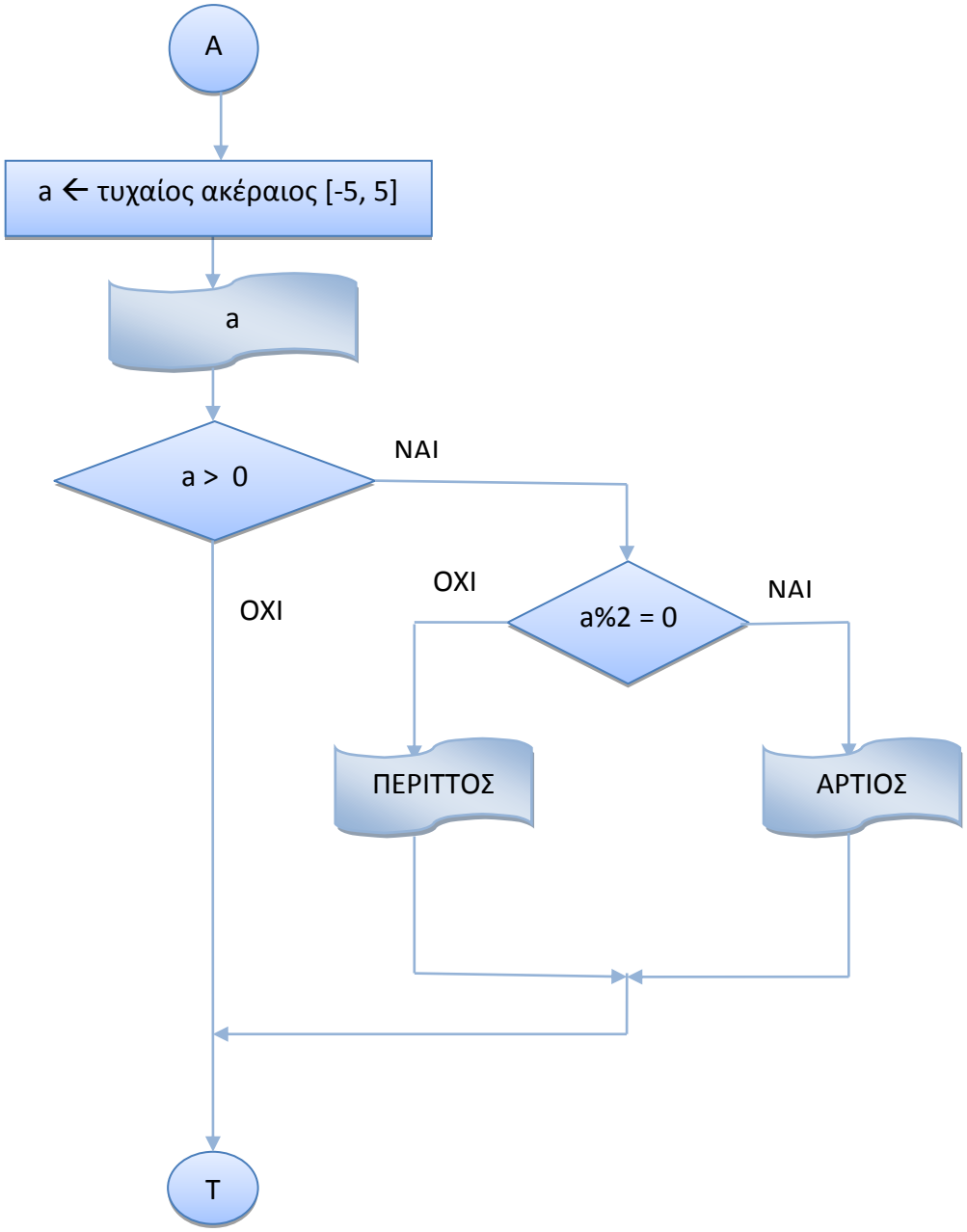
Παρατήρηση

Μπορεί στο τμήμα ενός `if` να υπάρχει ένα πλήρες `if`, όπως φαίνεται στο επόμενο παράδειγμα :

2.4.3 Πρόγραμμα με την Εντολή Ελέγχου if – if – else

Να γραφεί Αλγόριθμος/Πρόγραμμα, το οποίο θα **δημιουργεί** έναν τυχαίο ακέραιο αριθμό **a** μεταξύ του `-5` και `5`. Αν ο αριθμός είναι **θετικός**, θα εμφανίζει την τιμή του και θα ελέγχει αν ο αριθμός **a** είναι **άρτιος** ή **περιττός** και θα εμφανίζει την τιμή του με αντίστοιχο μήνυμα.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο **a** μεταξύ του **−5** και **5**
2. Εμφανίζω την τιμή του **a**
3. **Αν** ο αριθμός **a** είναι θετικός (**a > 0**)
   Αν Το υπόλοιπο της διαίρεσης του **a** δια του **2** είναι 0 (**a % 2 = 0**)
   Εμφανίζω το μήνυμα **a** ΑΡΤΙΟΣ
   Διαφορετικά
   Εμφανίζω το μήνυμα **a** ΠΕΡΙΤΤΟΣ

ΠΡΟΓΡΑΜΜΑ

```
public class IfIfArtios {
 /*
 Το πρόγραμμα δημιουργεί έναν τυχαίο ακέραιο αριθμό μεταξύ του −5 και 5.
 Αν ο αριθμός είναι θετικός,  ελέγχεται αν είναι άρτιος ή περιττός και
 εμφανίζεται το αντίστοιχο μήνυμα
 */
    public static void main(String[] args) {

     // Δημιουργία τυχαίου ακέραιου αριθμού a μεταξύ του −5 και5
     int a = (int)(Math.random()*11) - 5;

     // Εμφάνιση της τιμής του a
     System.out.println("Ο αριθμός a είναι το " + a);

     // Έλεγχος αν ο a είναι μεγαλύτερος του 0
     if (a > 0)
       // Έλεγχος αν ο a είναι άρτιος ή περιττός, εμφάνιση μηνύματος
       if (a %2 == 0)
           System.out.println(a + " = αρτιος " );
       else
          System.out.println(a + " = περιττός " );
    }
}
```

Έξοδος Προγράμματος

```
Ο αριθμός a είναι το −4

Ο αριθμός a είναι το 0

Ο αριθμός a είναι το 2
2 = αρτιος

Ο αριθμός a είναι το 1
1 = περιττός
```

Άσκηση 2.1 : Να τροποποιηθεί ο Αλγόριθμος 2.4.3, ώστε να βρίσκει αν ένας θετικός ακέραιος αριθμός είναι άρτιος ή περιττός με τη **χρήση διπλής συνθήκης**.

Λογικοί Τελεστές

Οι Τελεστές που συνδέουν δύο ή περισσότερες συνθήκες μεταξύ τους είναι οι παρακάτω :

Σύνδεση	**Λογικός Τελεστής**
ΚΑΙ (AND)	&
Ή (OR)	|
Αποκλειστικό Ή (XOR)	^
ΟΧΙ	!
Βραχυκυκλωμένο ΚΑΙ (Short Cirquit AND)	&&
Βραχυκυκλωμένο Ή (Short Cirquit OR)	||

Η διαφορά ανάμεσα στο βραχυκυκλωμένο τελεστή **&&** και τον απλό **&** είναι ότι ο βραχυκυκλωμένος τελεστής **&&** σε αντίθεση με τον Τελεστή **&** **ΔΕΝ** ελέγχει τη δεύτερη συνθήκη, **αν** η πρώτη συνθήκη είναι **ψευδής**, ενώ η διαφορά ανάμεσα στο βραχυκυκλωμένο τελεστή **||** και τον απλό **|** είναι ότι ο βραχυκυκλωμένος τελεστής **||** σε αντίθεση με τον Τελεστή **|** **ΔΕΝ** ελέγχει τη δεύτερη συνθήκη, αν η πρώτη συνθήκη είναι **αληθής**.

Ο Τριαδικός Τελεστής ?

Μπορούμε, αντί να χρησιμοποιήσουμε την εντολή με **if**, **else**, να χρησιμοποιήσουμε τον τριαδικό τελεστή **?**, ο οποίος έχει το ίδιο αποτέλεσμα. Ας δούμε ένα παράδειγμα υπολογισμού της απόλυτης τιμής ενός ακέραιου αριθμού :

Παράδειγμα

```
int n = (int)(Math.random()*10 - 10);
if ( n < 0 )
  absN = -n;
else
  absN = n;
```

Οι παραπάνω εντολές θα μπορούσαν να γραφούν :

```
int n = (int)(Math.random()*11) - 10;
absN = n < 0 ? -n:n;
```

όπου ελέγχεται η συνθήκη `n < 0` και αν είναι αληθής, το `absN = -n,` διαφορετικά, το `absN = n`, δηλαδή ότι κάνει και το `if-else`.

2.5 Η Σκάλα if – else - if

Για να ελέγξουμε την περίπτωση που έχουμε περισσότερα από τρία ενδεχόμενα, χρησιμοποιούμε τη σκάλα `if else if`, όπου κάθε `else` αντιστοιχεί στο κοντινότερο `if`:

```
if (<συνθήκη-1>)
  εντολή-1; ή >){ block εντολών-1; }
else if (<συνθήκη-2>)
  εντολή-2; ή >){ block εντολών-2; }
else if (<συνθήκη-3>)
  εντολή-3; ή >){ block εντολών-3; }
…
else
  εντολή-n; ή >){ block εντολών-n; }
```

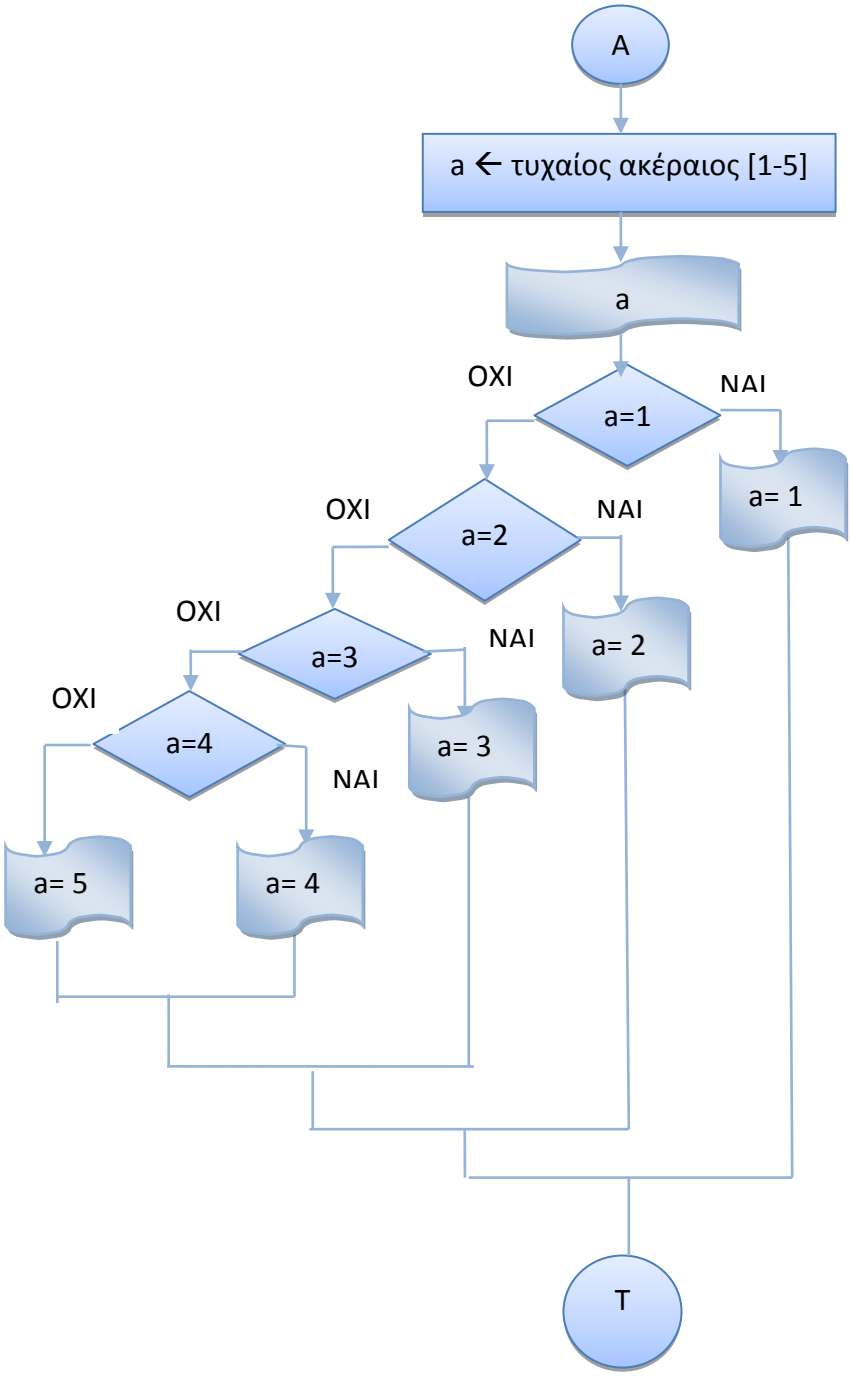
2.5.1 Πρόγραμμα με την Εντολή Ελέγχου if – else if –else if … else

Να γραφεί Αλγόριθμος/Πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο αριθμό **a** μεταξύ του 1 και 5, θα εμφανίζει την τιμή του και θα ελέγχει αν ο αριθμός **a** είναι το 1, το 2, το 3, το 4 ή το 5. Σε κάθε περίπτωση θα εμφανίζει την τιμή του.

ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο μεταξύ 1-5
2. **Αν** (`a = 1`)
   Εμφάνισε “`a = 1`”
   Διαφορετικά, **Αν** (`a = 2`)
   Εμφάνισε “`a = 2`”
   Διαφορετικά **Αν** (`a = 3`)
   Εμφάνισε “`a = 3`”
   Διαφορετικά **Αν** (`a = 4`)
   Εμφάνισε “`a = 4`”
   Διαφορετικά
   Εμφάνισε “`a = 5`”

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΠΡΟΓΡΑΜΜΑ

```
public class IfElseIfElseIfElse {

/* Πρόγραμμα, το οποίο  δημιουργεί 1 τυχαίο ακέραιο αριθμό a μεταξύ του 1 και
5,  ελέγχει αν ο αριθμός a είναι το 1, το 2, το 3, το 4 ή το 5 και σε κάθε
περίπτωση θα εμφανίζει την τιμή του.
*/
    public static void main(String[] args) {
     // Δημιουργία τυχαίου ακέραιου αριθμού a μεταξύ του 1 και5
     int a = (int)(Math.random()*5) + 1;
```

```
      // Εμφάνιση της τιμής του a
      System.out.println("Ο αριθμός a είναι το " + a);

      // Έλεγχος, εμφάνιση της τιμής του a
      if (a == 1)
        System.out.println("a = 1");
      else if (a == 2)
        System.out.println("a = 2");
      else if (a == 3)
        System.out.println("a = 3");
      else if (a == 4)
        System.out.println("a = 4");
      else
        System.out.println("a = 5");
      }
}
```

Έξοδος Προγράμματος

```
Ο αριθμός a είναι το 2
a = 2
```

2.6 Η Εντολή Ελέγχου switch

Στο παράδειγμα 2.5.1 χρειάστηκε να χρησιμοποιήσουμε **4** εντολές **if** για να ελέγξουμε αν η τιμή του **a** είναι το **1**, το **2**, το **3**, το **4** ή το **5**. Σ' αυτές τις περιπτώσεις, μπορούμε να χρησιμοποιήσουμε την εντολή **switch**, η οποία επιλέγει ανάμεσα σε διάφορες περιπτώσεις. Η γενική της μορφή είναι :

```
switch ( <έκφραση> )
      {
         case <σταθερά_1> :  εντολές_1;
                             break;
         case <σταθερά_2> :  εντολές_2;
                             break;
         ...                 ...
         ...                 ...
         case <σταθερά_n> :  εντολές_n;
                             break;
         default :           εντολές_n+1
      }
```

όπου η <έκφραση> μπορεί να είναι μεταβλητή ή αριθμητική έκφραση τύπου **char**, **byte**, **short**, **int**, ενώ οι <σταθερά_1>, <σταθερά_2>, ..., <σταθερά_n> είναι κυριολεκτικές σταθερές του τύπου της έκφρασης.

- Η επιλογή default είναι προαιρετική και θα εκτελεστούν οι εντολές της, αν η έκφραση <έκφραση> δεν πάρει καμιά απ' τις τιμές που εμφανίζονται στις γραμμές case.

- Ανάλογα με την τιμή που θα έχει η <έκφραση> θα εκτελεστούν οι αντίστοιχες εντολές της κάθε περίπτωσης.

- Η εντολή break είναι προαιρετική, αλλά, αν δεν υπάρχει, μετά την εκτέλεση των εντολών κάποιας περίπτωσης, θα εκτελεστούν και οι εντολές των υπόλοιπων περιπτώσεων.

- Μπορεί να υπάρχει εντολή switch σε κάποια περίπτωση (case) μιας άλλης εντολής switch και οι περιπτώσεις της να παίρνουν ίδιες τιμές με τις περιπτώσεις της εξωτερικής εντολής switch.

- Μπορεί να υπάρχουν κενές περιπτώσεις χωρίς τιμές. Αυτές εντάσσονται στην πρώτη επόμενη περίπτωση που έχει εντολές. Π.χ. αν θέλω να κάνω κάτι για τις περιπτώσεις 1, 2, 3, αφήνω κενές τις περιπτώσεις 1,2 και γράφω τις εντολές στην περίπτωση 3.

2.6.1 Τροποποίηση του Προγράμματος 2.5.1 με την Εντολή Ελέγχου switch

Να γραφεί Πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο αριθμό **a** μεταξύ του **1** και **5**, θα εμφανίζει την τιμή του και θα ελέγχει με τη χρήση της εντολής **switch** αν ο αριθμός **a** είναι το **1**, το **2**, το **3**, το **4** ή το **5**. Σε κάθε περίπτωση θα εμφανίζει την τιμή του.

ΠΡΟΓΡΑΜΜΑ

```
public class SwitchInt {

    /*
     Πρόγραμμα, το οποίο  δημιουργεί έναν τυχαίο ακέραιο αριθμό a  μεταξύ του
     * 1 και 5,  εμφανίζει την τιμή του και  ελέγχει με τη χρήση της εντολής
     * switch αν ο αριθμός a είναι το 1, το 2, το 3, το 4 ή το 5.
     * Σε κάθε περίπτωση  εμφανίζει την τιμή του.
     */
    public static void main(String[] args) {
    // Δημιουργία τυχαίου ακέραιο αριθμού a μεταξύ του 1 και 5
    int num = (int)(Math.random()*5) + 1;
    switch ( num )
     {
        case 1 :  System.out.println("num = 1");
                  break;
        case 2 :  System.out.println("num = 2");
                  break;
        case 3 :  System.out.println("num = 3");
                  break;
        case 4 :  System.out.println("num = 4");
                  break;
        case 5 :  System.out.println("num = 5");
                  break;
        default :  System.out.println("num not in 1:5");
                   break;
     }
    }
}
```

Έξοδος Προγράμματος

```
num = 3

num = 5

num = 4

num = 2

num = 1

num not in 1:5
```

2.6.2 Τροποποίηση του Προγράμματος 2.5.1, Διάβασμα Χαρακτήρα

Να γραφεί Πρόγραμμα, το οποίο θα διαβάζει απ' το πληκτρολόγιο ένα χαρακτήρα **ch**, θα εμφανίζει την τιμή του και θα ελέγχει με τη χρήση της εντολής **switch** αν ο χαρακτήρας **ch** είναι το **'a'**, το **'b'**, το **'c'**, το **'d'** ή το **'e'**. Σε κάθε περίπτωση θα εμφανίζει την τιμή του.

ΠΡΟΓΡΑΜΜΑ

```
public class SwitchChar {
/*
Πρόγραμμα, το οποίο διαβάζει απ' το πληκτρολόγιο ένα χαρακτήρα ch, εμφανίζει
την τιμή του και θα ελέγχει με τη χρήση της εντολής switch αν ο χαρακτήρας
ch είναι το 'a', το 'b', το 'c', το 'd' ή το 'e'. Σε κάθε περίπτωση εμφανίζει
την τιμή του.
*/
    public static void main(String[] args)
                        throws java.io.IOException {
     // Διάβασμα απ' το πληκτρολόγιο ενός χαρακτήρα ch
     char ch;
     System.out.print("Give a character a - e : ");
     ch = (char)(System.in.read());

     // Έλεγχος, εμφάνιση της τιμής του ch
      switch (ch){
        case 'a' :  System.out.println("ch = a");
                    break;
        case 'b' :  System.out.println("ch = b");
                    break;
        case 'c' :  System.out.println("ch = c");
                    break;
        case 'd' : System.out.println("ch = d");
                    break;
        case 'e' :  System.out.println("ch = e");
                    break;
        default :  System.out.println("ch = " + ch + " not in a:e");
      }
     }
}
```

Έξοδος Προγράμματος

```
Give a character a - e : a
ch = a

Give a character a - e : f
ch = f not in a:e
```

- Το **`throws java.io.IOException`** είναι απαραίτητο στη δήλωση της `main()`, γιατί μπορεί να προκύψει σφάλμα (εξαίρεση – `exception`) στο διάβασμα του χαρακτήρα.

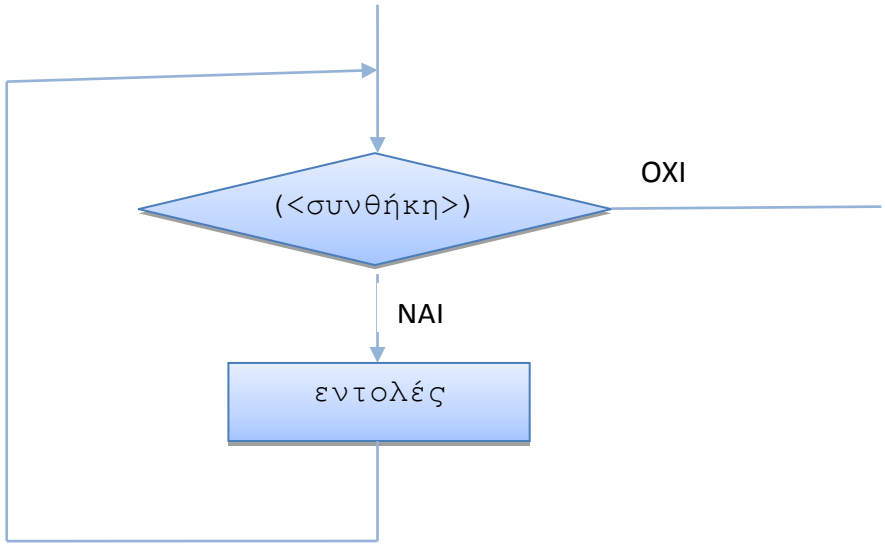
- Ο χαρακτήρας **`ch`** διαβάζεται με μια εντολή παρόμοια της `System.out.println()`, την εντολή :

```
ch = (char)(System.in.read());
```

αλλά απαιτείται διανομή τύπου **`char`**, γιατί ο χαρακτήρας που διαβάζεται μετατρέπεται σε ακέραια μορφή.

3 ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ (while, do…while)

Στα πιο πολλά προγράμματα απαιτείται κάποια ή κάποιες εντολές να εκτελούνται πολλές φορές για όσο ισχύει κάποια **συνθήκη**. Ο αριθμός των επαναλήψεων μπορεί να είναι άγνωστος ή γνωστός. Στα λογικά διαγράμματα η επανάληψη αυτή των εντολών συμβολίζεται με μια **ανακύκλωση** του ελέγχου ροής του προγράμματος. Όπως και στις εντολές ελέγχου, θα πρέπει να γίνει κάποια **σύγκριση**. Η **εντολή επανάληψης** (σε μορφή διαγράμματος ροής, αλγορίθμου και `Java`) στην περίπτωση που ο αριθμός των επαναλήψεων είναι **άγνωστος**, έχει τη μορφή :




```
Για όσο (<ισχύει κάποια συνθήκη>)
   εκτέλεση εντολής ή block εντολών

ή

while (<συνθήκη>)
   εντολή; ή { block εντολών; }
```

όπου ελέγχεται η συνθήκη, αν είναι αληθής. **Αν** ισχύει η συνθήκη, εκτελείται η εντολή ή οι εντολές (οι οποίες θα πρέπει να περικλείονται σε άγγιστρα, αν είναι περισσότερες από μια) μετά το `while`. Η εκτέλεση των εντολών **τερματίζεται**, όταν πάψει να ισχύει η συνθήκη. Για να γίνει αυτό απαιτείται να υπάρχει μέσα στο `block` εντολών του `while` κάποια εντολή ή εντολές που **αλλάζουν** την τιμή κάποιας ή κάποιων μεταβλητών που περιλαμβάνονται στη συνθήκη.

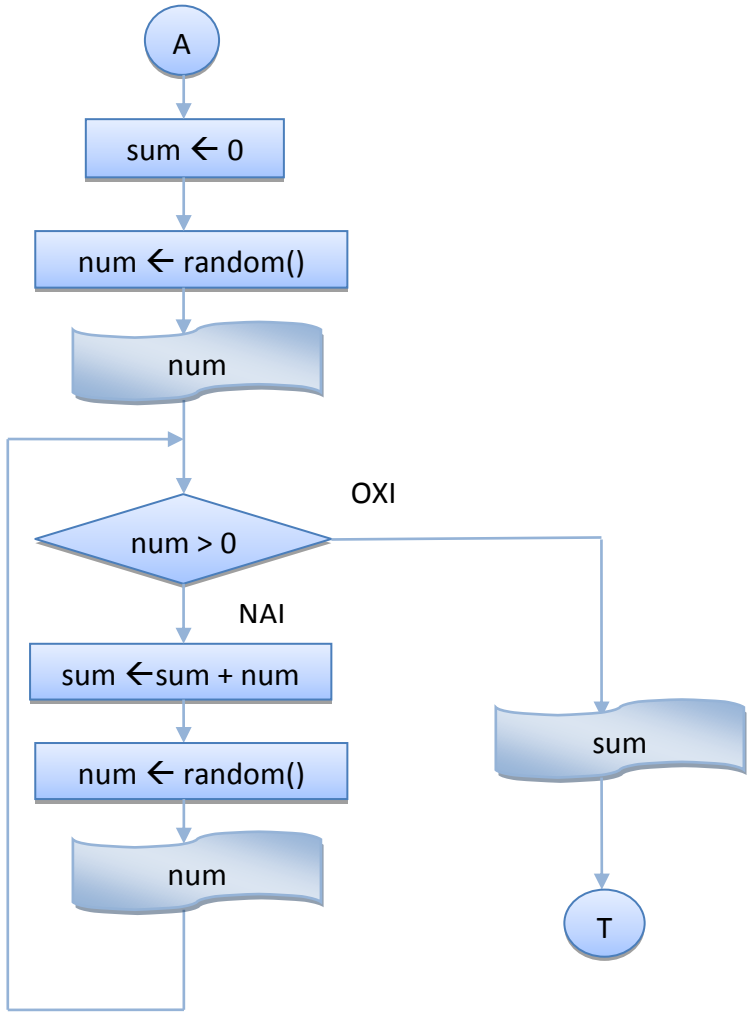
Παρατηρήσεις

- Ανάλογα με τις τιμές που έχουν οι μεταβλητές που περιλαμβάνονται στη συνθήκη πριν το `while`, οι εντολές μπορεί να μην εκτελεστούν **καμιά** φορά, αν η **συνθήκη** είναι εξ αρχής **ψευδής**.

3. 1 Ένα Απλό Πρόγραμμα με την Εντολή Επανάληψης while

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα αποθηκεύονται στη μεταβλητή **num** μεταξύ του 0 και 10, εμφανίζει τις τιμές τους και τους προσθέτει σε έναν αθροιστή **sum**, για όσο οι αριθμοί είναι θετικοί. Όταν δημιουργηθεί κάποιος μη θετικός αριθμός, εμφανίζει την τιμή του αθροιστή **sum** και τερματίζει.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δίνω αρχική τιμή το μηδέν στον αθροιστή `sum` (`sum` = 0)
2. Δημιουργώ έναν τυχαίο αριθμό `num`
3. Εμφανίζω την τιμή του `num`
4. **Για όσο** ο αριθμός `num` είναι > 0 (`num > 0`)
   a. Προσθέτω το `num` στο `sum` (`sum` ← `sum + num`)
   b. Δημιουργώ έναν νέο τυχαίο αριθμό `num`
   c. Εμφανίζω την τιμή του `num`
5. Εμφανίζω την τιμή του αθροιστή `sum`

ΠΡΟΓΡΑΜΜΑ

```
public class WhileSum {

/* Πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι
θα αποθηκεύονται στη μεταβλητή num μεταξύ του 0 και 10, εμφανίζει τις τιμές
τους και τους προσθέτει σε έναν αθροιστή sum, για όσο οι αριθμοί είναι
θετικοί.  Όταν δημιουργηθεί κάποιος μη θετικός αριθμός, εμφανίζει την τιμή
του αθροιστή sum και τερματίζει.
*/
    public static void main(String[] args) {
        int num;
        int sum = 0;

        // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο 0-10
        num = (int)(Math.random()*10);
        System.out.println("Ο αριθμός είναι : " + num);

        // Για όσο ο αριθμός είναι θετικός
        while ( num > 0)
        {
            // Πρόσθεση του αριθμού στον αθροιστή sum
            sum = sum + num;

            // Δημιουργία - Εμφάνιση νέου τυχαίου ακέραιου αριθμού στο 0-10
            num = (int)(Math.random()*10);
            System.out.println("Ο αριθμός είναι : " + num);
        }
        // Εμφάνιση αθροίσματος sum
        System.out.println("\nΤο άθροισμά τους είναι :" + sum + "\n");
    }
}
```

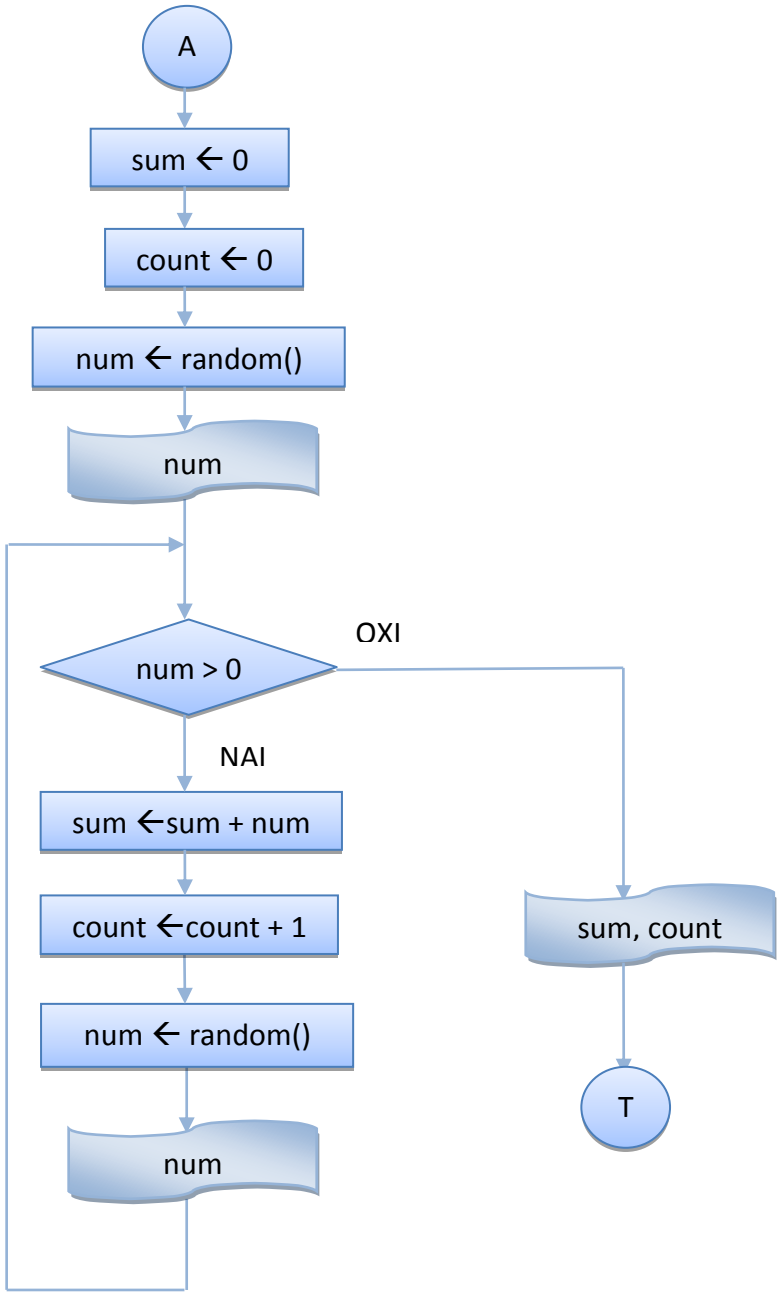
Έξοδος Προγράμματος

```
run:
Ο αριθμός είναι : 9
Ο αριθμός είναι : 6
Ο αριθμός είναι : 3
Ο αριθμός είναι : 0

Το άθροισμά τους είναι : 18
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. 2 Τροποποίηση Προγράμματος 3.1 για τον Υπολογισμό και του Πλήθους των Αριθμών με την Εντολή Επανάληψης while

Να τροποποιηθεί το Πρόγραμμα 3.1, ώστε να δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα αποθηκεύονται στη μεταβλητή **num** μεταξύ του 0 και 10, θα εμφανίζει τις τιμές τους και θα τους προσθέτει σε έναν αθροιστή **sum**, για όσο οι αριθμοί είναι θετικοί **και θα τους μετράει**. Όταν δημιουργηθεί κάποιος μη θετικός αριθμός, θα εμφανίζει **την τιμή του μετρητή** και του αθροιστή **sum** και θα τερματίζει.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δίνω αρχική τιμή το μηδέν στον αθροιστή `sum` (`sum` = 0)
2. **Δίνω αρχική τιμή το μηδέν στο μετρητή `count` (`count` = 0)**
3. Δημιουργώ έναν τυχαίο αριθμό `num`
4. Εμφανίζω την τιμή του `num`
5. **Για όσο** ο αριθμός `num` είναι > 0 (`num > 0`)
   a. Προσθέτω το `num` στο `sum` (`sum ← sum + num`)
   b. Αυξάνω την τιμή του μετρητή κατά 1 (`count ← count + 1`)
   c. Δημιουργώ έναν νέο τυχαίο αριθμό `num`
   d. Εμφανίζω την τιμή του `num`
6. Εμφανίζω την τιμή του αθροιστή `sum` **και του μετρητή `count`**

ΠΡΟΓΡΑΜΜΑ

```
public class WhileSumMo {
/* Πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι
θα αποθηκεύονται στη μεταβλητή num μεταξύ του 0 και 10, εμφανίζει τις
τιμές τους, τους προσθέτει σε έναν αθροιστή sum και τους μετράει, για όσο οι
αριθμοί είναι θετικοί.  Όταν δημιουργηθεί κάποιος μη θετικός αριθμός,
εμφανίζει την τιμή του αθροιστή sum και του μετρητή και τερματίζει.
*/
    public static void main(String[] args) {
        int num;
        int sum = 0;
        int count = 0;

        // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο [0,10]
        num = (int)(Math.random()*10);
        System.out.println("Ο αριθμός είναι : " + num);

        // Για όσο ο αριθμός είναι θετικός
        while ( num > 0)
        {
            // Πρόσθεση του αριθμού στον αθροιστή sum
            sum = sum + num;

            // Αύξηση του μετρητή count κατά 1
            count++;

           // Δημιουργία - Εμφάνιση νέου τυχαίου ακέραιου αριθμού στο [0,10]
            num = (int)(Math.random()*10);
            System.out.println("Ο αριθμός είναι : " + num);
        }
        // Εμφάνιση αθροίσματος sum και μετρητή count
       System.out.println("\nΔημιουργήθηκαν " + count + " μη μηδενικοί " +
                          " αριθμοί και το άθροισμά τους είναι " +
                          sum + "\n");
    }
}
```

Έξοδος Προγράμματος

```
run:
Ο αριθμός είναι : 7
Ο αριθμός είναι : 7
Ο αριθμός είναι : 5
Ο αριθμός είναι : 2
Ο αριθμός είναι : 0

Δημιουργήθηκαν 4 μη μηδενικοί  αριθμοί και το άθροισμά τους είναι 21
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
Ο αριθμός είναι : 0

Δημιουργήθηκαν 0 μη μηδενικοί  αριθμοί και το άθροισμά τους είναι 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Άσκηση 3.1

Να τροποποποιηθεί ο αλγόριθμος 3.2, ώστε να βρίσκει - σε κάθε περίπτωση - και να εμφανίζει και το **Μέσο Όρο**.

Παρατηρήσεις

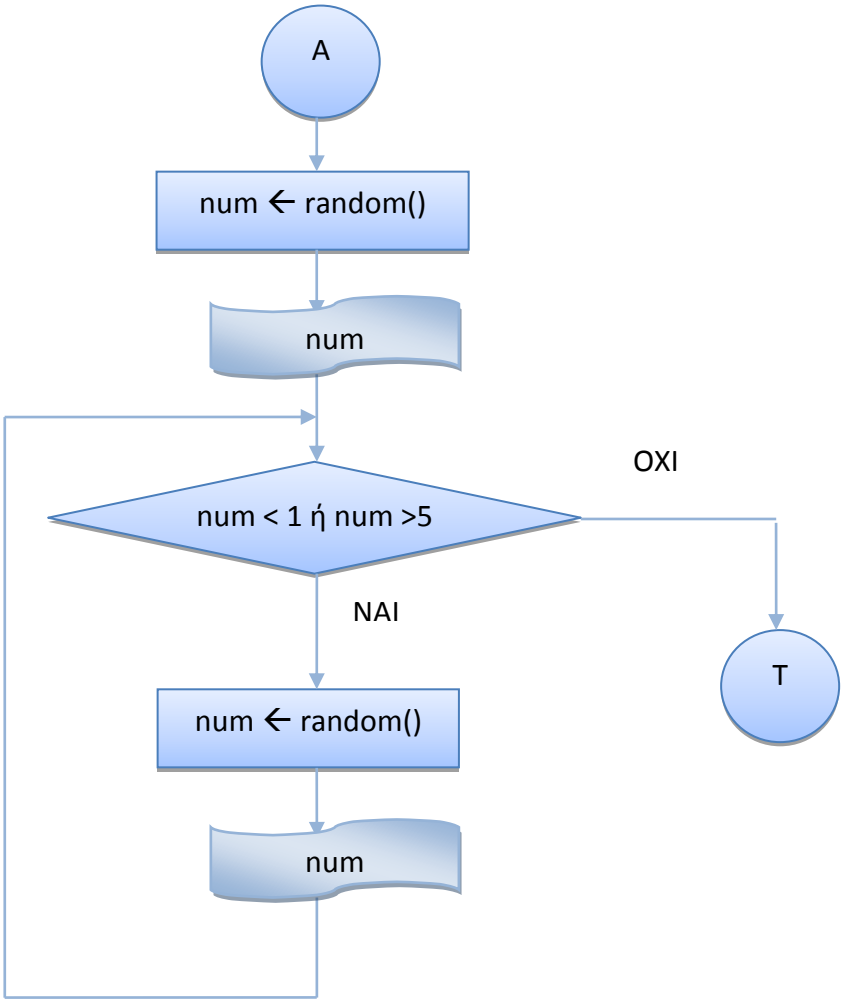
- Η δημιουργία του τυχαίου αριθμού και η εμφάνιση της τιμής του επαναλαβάνονται δύο φορές. **Πριν** την εντολή `while`, γιατί η μεταβλητή `num` που κάνει τον έλεγχο στη συνθήκη πρέπει να έχει κάποια τιμή για να γίνει η πρώτη σύγκριση, αλλά και **μέσα** στο σώμα του `while`, γιατί μας χρειάζεται ο νέος αριθμός. Στην περίπτωση που ο πρώτος τυχαίος αριθμός είναι το μηδέν, η συνθήκη του `while` είναι εξ αρχής ψευδής, οπότε δεν εκτελούνται ποτέ οι εντολές που υπάρχουν στο σώμα του `while`.

- Σε πολλές εφαρμογές θα χρειαστεί να εμφανίσουμε ένα `menu` επιλογών και να διαβάσουμε κάποια τιμή για την επιλογή, ώστε να εκτελέσουμε και τις αντίστοιχες εντολές. Στο επόμενο παράδειγμα υλοποιούμε κάτι αντίστοιχο, δημιουργώντας έναν τυχαίο αριθμό μεταξύ του `1` και `5`.

3. 3 Ένα Απλό Πρόγραμμα για Δημιουργία Επιλογής με while

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα αποθηκεύονται στη μεταβλητή **num** μεταξύ του `0` και `10` και εμφανίζει τις τιμές τους. Για όσο οι αριθμοί είναι μικρότεροι του `1` ή μεγαλύτεροι του `5`, το πρόγραμμα δημιουργεί νέο αριθμό. Το πρόγραμμα τερματίζει, όταν ο αριθμός που θα δημιουργηθεί είναι μεταξύ του `1` και του `5`.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο αριθμό num στο [0,10]
2. Εμφανίζω την τιμή του num
3. **Για όσο** ο αριθμός num είναι < 1 ή > 5 (num < 1 ή num > 5)
   a. Δημιουργώ έναν νέο τυχαίο αριθμό num
   b. Εμφανίζω την τιμή του num

ΠΡΟΓΡΑΜΜΑ

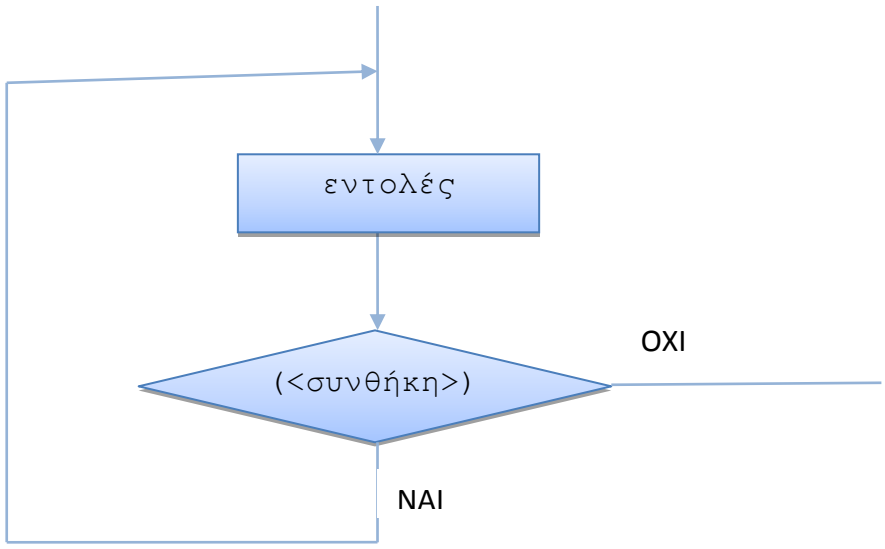
```
public class WhileEpiloghRandom {
/*
Πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα
αποθηκεύονται στη μεταβλητή num μεταξύ του 0 και 10 και εμφανίζει τις τιμές
τους. Για όσο οι αριθμοί είναι μικρότεροι του 1 και μεγαλύτεροι του 5, το
πρόγραμμα δημιουργεί νέο αριθμό.  Το πρόγραμμα τερματίζει, όταν ο αριθμός που
θα δημιουργηθεί είναι μεταξύ του 1 και του 5.
*/
    public static void main(String[] args) {
         int num;

         // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο [0, 10]
         num = (int) ( Math.random()*10);
         System.out.println("num = " + num );

         // Για όσο ο αριθμός είναι εκτός των ορίων ( 1-5) που θέλουμε
         while (( num < 1) || ( num > 5))
        {
           // Δημιουργία-Εμφάνιση νέου τυχαίου ακέραιου αριθμού στο [0, 10]
            num = (int) ( Math.random()*10);
            System.out.println("num = " + num );
        }

        System.out.println("\n\n");
    }
}
```

Έξοδος Προγράμματος

```
run:
Ο αριθμός είναι : -3
Ο αριθμός είναι : 8
Ο αριθμός είναι : -4
Ο αριθμός είναι : 0
Ο αριθμός είναι : 2
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Ο αριθμός είναι : 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

❖ Στο προηγούμενο πρόγραμμα χρειαζόμαστε **οπωσδήποτε** έναν αριθμό μεταξύ του 1 και 5. Σ' αυτή την περίπτωση, θα μπορούσαμε να αποφύγουμε την επανάληψη των εντολών Δημιουργίας και Εμφάνισης της τιμής του τυχαίου αριθμού `num`, χρησιμοποιώντας μια άλλη μορφή της εντολής `while`, τη `do while`, στην οποία ο έλεγχος τη συνθήκης γίνεται **ΜΕΤΑ** την εκτέλεση των εντολών και όχι **ΠΡΙΝ**, όπως γίνεται με την εντολή `while`.

Η **σύνταξη** (σε μορφή διαγράμματος ροής, αλγορίθμου και `Java`) της εντολής `do while` είναι :




```
Κάνε τα παρακάτω
   εντολές;
Για όσο (<η συνθήκη ισχύει>
```

ή

```
do {
        εντολή; ή εντολές;
   }
while ( <συνθήκη>);
```

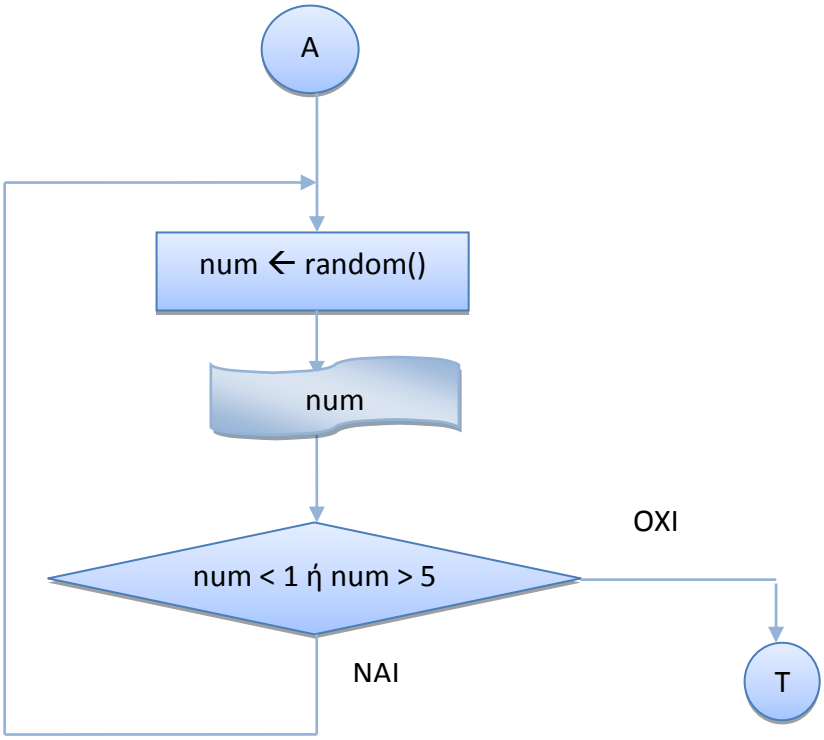
Παρατήρηση

Επειδή με το `while` τελειώνει η εντολή, χρειάζεται στο τέλος ερωτηματικό.

3. 4 Τροποποίηση του Προγράμματος 3.3 για τη Δημιουργία Επιλογής με την εντολή do while

Να τροποποιηθεί το πρόγραμμα 3.3, ώστε να δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα αποθηκεύονται στη μεταβλητή **num** μεταξύ του `0` και `10` και να εμφανίζει τις τιμές τους. Για όσο οι αριθμοί είναι μικρότεροι του `1` ή μεγαλύτεροι του `5`, το πρόγραμμα θα δημιουργεί νέο αριθμό. Το πρόγραμμα θα τερματίζει, όταν ο αριθμός που θα δημιουργηθεί είναι μεταξύ του `1` και του `5`. **Η υλοποίηση να γίνει με την εντολή `do while`.**

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

Κάνε τα παρακάτω
a. Δημιουργώ έναν νέο τυχαίο αριθμό `num`
b. Εμφανίζω την τιμή του `num`

Για όσο ο αριθμός `num` είναι < 1 ή > 5 (`num < 1 ή num > 5`)

ΠΡΟΓΡΑΜΜΑ

```
public class DoWhileEpiloghRandom {
/*
Πρόγραμμα, το οποίο δημιουργεί τυχαίους ακέραιους αριθμούς, οι οποίοι θα
αποθηκεύονται στη μεταβλητή num μεταξύ του 0 και 10 και εμφανίζει τις τιμές
τους. Για όσο οι αριθμοί είναι μικρότεροι του 1 και μεγαλύτεροι του 5, το
πρόγραμμα ζητάει νέο αριθμό.  Το πρόγραμμα τερματίζει, όταν ο αριθμός που θα
δημιουργηθεί είναι μεταξύ του 1 και του 5. Η υλοποίηση θα γίνει με την εντολή
do while.
*/
    public static void main(String[] args) {
         int num;

         do { // Για όσο ο αριθμός είναι εκτός των ορίων ( 1-5)
         // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο [0, 10]
         num = (int) ( Math.random()*10);
         System.out.println("Ο αριθμός είναι : " + num );
         }
         while (( num < 1) || ( num > 5));
         System.out.println("\n\n");
    }
}
```

Έξοδος Προγράμματος

```
run:
Ο αριθμός είναι : 4
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Ο αριθμός είναι : -3
Ο αριθμός είναι : 8
Ο αριθμός είναι : -4
Ο αριθμός είναι : 0
Ο αριθμός είναι : 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

- Οι εντολές μετά το `do` στην εντολή `do while` εκτελούνται **ΤΟΥΛΑΧΙΣΤΟΝ** μία φορά.

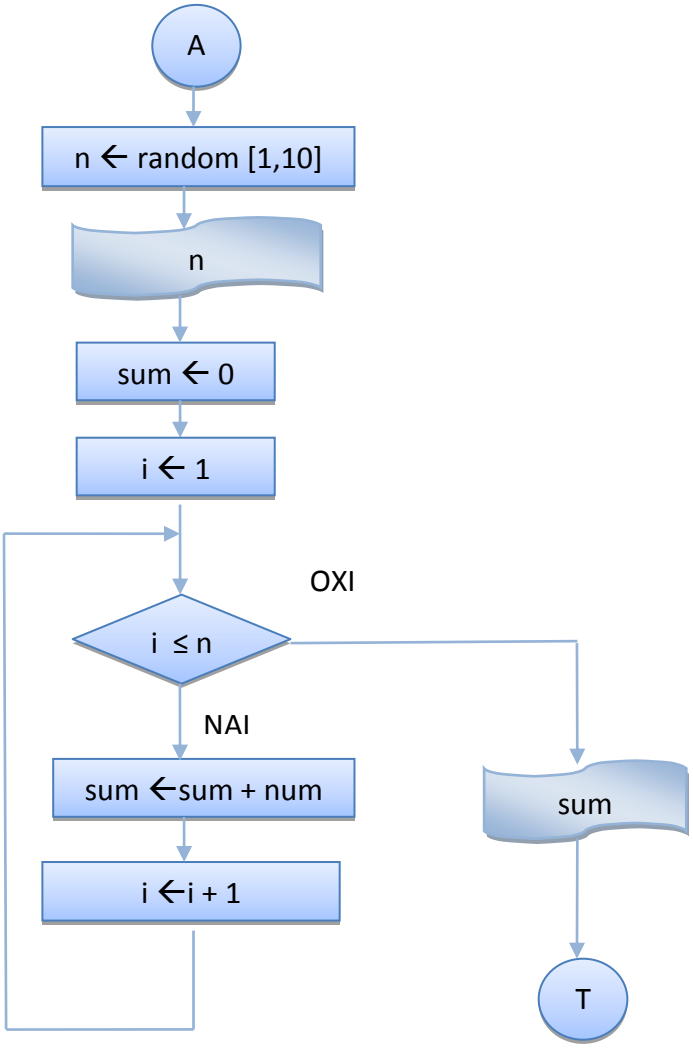
4 ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ - for

Υπάρχουν προβλήματα, στα οποία ο αριθμός των επαναλήψεων κάποιων εντολών είναι **γνωστός** εκ των προτέρων, όπως στο επόμενο παράδειγμα :

4. 1 Πρόγραμμα για τον Υπολογισμό του Αθροίσματος 1+2+… + n με την **Εντολή Επανάληψης while**

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο αριθμό μεταξύ του `1` και `10`, ο οποίος θα αποθηκεύεται στη μεταβλητή **n**, θα εμφανίζει την τιμή του και θα υπολογίζει και θα εμφανίζει το άθροισμα 1 + 2 + ... + **n**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο αριθμό n στο [1, 10]
2. Τον εμφανίζω
3. Δίνω αρχική τιμή το μηδέν στον αθροιστή sum (sum ← 0)
4. Δίνω αρχική τιμή το 1 στον αριθμό i (i ← 1)
5. **Για όσο** ο αριθμός i είναι ≤ n (i ≤ n)
   a. Προσθέτω το i στο sum (sum ← sum + i)
   b. Αυξάνω την τιμή του αριθμού κατά 1 (i ← i + 1)
6. Εμφανίζω την τιμή του αθροιστή sum

ΠΡΟΓΡΑΜΜΑ

```
public class WhileFor12n {
/*Πρόγραμμα, το οποίο  δημιουργεί έναν τυχαίο ακέραιο αριθμό μεταξύ του  1
και 10, ο οποίος θα αποθηκεύεται στη μεταβλητή n,  εμφανίζει την τιμή     του
και  υπολογίζει και  εμφανίζει το άθροισμα 1 + 2 + ... + n.*/
    public static void main(String[] args) {
       int n, sum, i;

       // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο 1 - 10
        n = (int) ( Math.random()*10 + 1);
        System.out.println("n = " + n );

        // Αρχικές τιμές στο άθροισμα και τον αριθμό
        sum = 0;
        i = 1;

        // Για όσο ο αριθμός δεν ξεπέρασε το n
        while (i <= n)
        {
          // Πρόσθεση του αριθμού στον αθροιστή sum
          sum += i;

          // Αύξηση του αριθμού i κατά 1
          i++;
        }

       // Εμφάνιση αθροίσματος sum
       System.out.println("Το άθροισμά 1+2+...+" + n + " είναι " + sum +
"\n\n");
    }
}
```

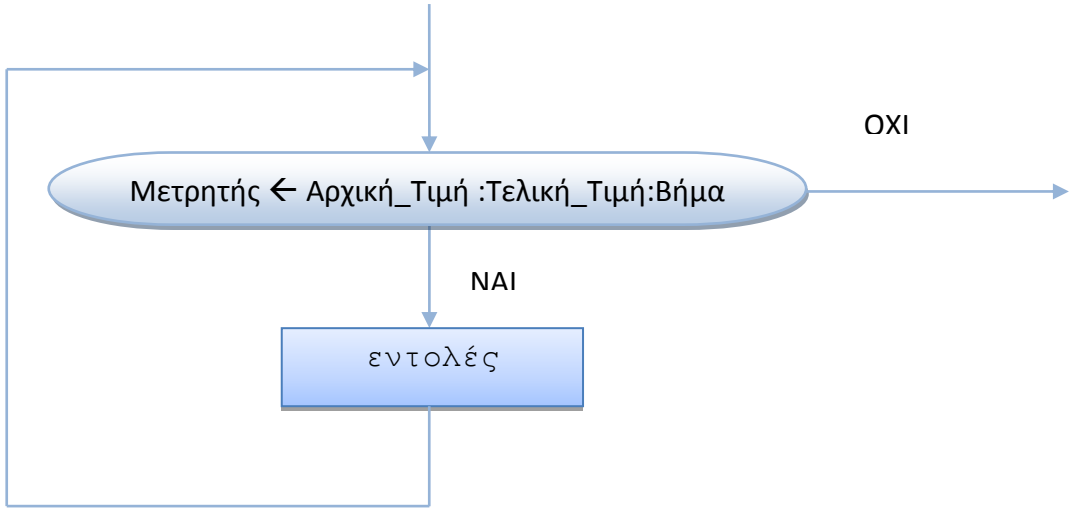
Έξοδος Προγράμματος

```
run:
n = 6
Το άθροισμα 1+2+...+6 είναι 21
BUILD SUCCESSFUL (total time: 0 seconds)
```

ΑΣΚΗΣΗ 4.1 : Να τροποποιηθεί ο προηγούμενος αλγόριθμος ώστε να υπολογίζει/ εμφανίζει και το **Μέσο Όρο** των αριθμών 1 + 2 + ... + n με **do**…**while**.

4. 2 Η Εντολή Επανάληψης for

Στα προβλήματα με γνωστό αριθμό επαναλήψεων, οι εντολές ανάθεσης **αρχικής τιμής** στο μετρητή, ελέγχου της **συνθήκης**, αν **ο μετρητής ξεπέρασε την τελική τιμή** και **ενημέρωσης της τιμής του μετρητή** μέσα στο σώμα της επανάληψης μπορούν να συμπεριληφθούν σε μια εντολή, την εντολή `for`, της οποίας η σύνταξη (σε μορφή διαγράμματος ροής, αλγορίθμου και `Java`) είναι :



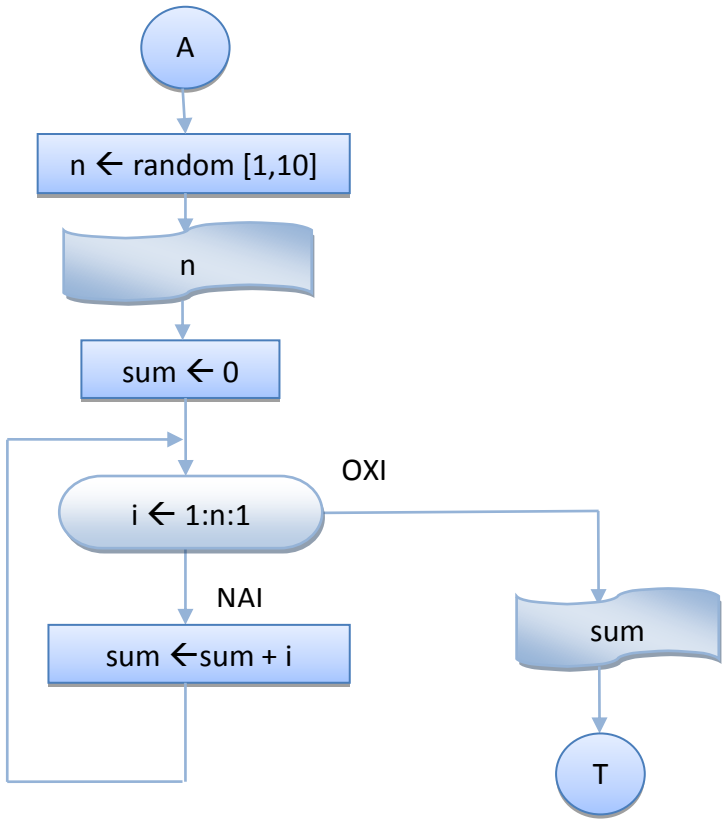
Για <μετρητής>=<αρχική τιμή>:<τελική τιμή>:<βήμα>
    εντολές;

```
for (<αρχική τιμή μετρητή>; <ο μετρητής ξεπέρασε την τελική τιμή?>; <ενημέρωση τιμής μετρητή>)
    {
        εντολές;
    }
```

4.2.1 Πρόγραμμα για τον Υπολογισμό του Αθροίσματος 1+2+…+ n με την Εντολή Επανάληψης for

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο αριθμό μεταξύ του `1` και `10`, ο οποίος θα αποθηκεύεται στη μεταβλητή **n**, θα εμφανίζει την τιμή του και θα υπολογίζει και θα εμφανίζει το άθροισμα `1 + 2 + ... + n` **με την εντολή επανάληψης for**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο ακέραιο αριθμό `n` στο `[1, 10]`
2. Τον εμφανίζω
3. Δίνω αρχική τιμή το μηδέν στον αθροιστή `sum` (`sum` ← 0)
4. **<u>Για</u>** τις τιμές του μετρητή `i` από το `1` μέχρι και το `n` με βήμα `1`
   Προσθέτω το `i` στο `sum` (`sum` ← `sum + i`)
5. Εμφανίζω την τιμή του αθροιστή `sum`

ΠΡΟΓΡΑΜΜΑ

```
public class ForSum {
/*Πρόγραμμα, το οποίο  δημιουργεί έναν τυχαίο ακέραιο αριθμό μεταξύ του     1
και 10, ο οποίος αποθηκεύεται στη μεταβλητή n,  εμφανίζει την τιμή      του
και υπολογίζει και εμφανίζει το άθροισμα 1 + 2 + ... + n με την      εντολή
for.
*/
    public static void main(String[] args) {

        int n, sum, i;

       // Δημιουργία - Εμφάνιση τυχαίου ακέραιου αριθμού στο 1 - 10
        n = (int) ( Math.random()*10 + 1);
        System.out.println("n = " + n );

        // Αρχική τιμή 0 στο άθροισμα
        sum = 0;

        // Για την τιμή του μετρητή i από το 1 μέχρι και το n
        for ( i = 1;i <= n;i++ )
          // Πρόσθεση του αριθμού στον αθροιστή sum
          sum += i;

        // Εμφάνιση αθροίσματος sum
        System.out.println("Το άθροισμα 1+2+...+" + n + " είναι " + sum
               + "\n");
    }
}
```

Έξοδος Προγράμματος

```
run:
n = 6
Το άθροισμα 1+2+...+6 είναι 21
BUILD SUCCESSFUL (total time: 0 seconds)
```

ΑΣΚΗΣΗ 4.2 : Να τροποποιηθεί ο προηγούμενος αλγόριθμος ώστε να υπολογίζει και να εμφανίζει και το **Μέσο Όρο** των αριθμών `1 + 2 + ... + n`.

ΑΣΚΗΣΗ 4.3 : Να τροποποιηθεί ο προηγούμενος αλγόριθμος ώστε να υπολογίζει και να εμφανίζει το **Άθροισμα** και το **Μέσο Όρο** των αριθμών `n+(n-1)+…+2+1` (ο μετρητής θα αρχίσει από το `n` και θα **μειώνεται** κατά `1`, μέχρι και το `1`).

Παρατηρήσεις

Η εντολή `for` δεν είναι απαραίτητο να περιέχει και την <αρχική τιμή στο μετρητή> τον έλεγχο αν <ο μετρητής ξεπέρασε την τελική τιμή> και την <ενημέρωση της τιμής του μετρητή>.

Παράδειγμα

Από την προηγούμενη πλήρη εντολή επανάληψης `for`

```
for (i = 1;i <= n;i++)
  sum += i;
```

μπορεί να λείπουν κάποια απ' αυτά ή όλα, αρκεί να υπάρχουν τα αντίστοιχα ερωτηματικά.

Π.χ. θα μπορούσε να λείπει :

- Η <αρχική τιμή στο μετρητή>, οπότε θα έχουμε :

```
i = 1;
for (;i <= n;i++)
  sum += i;
```

- Η <ενημέρωση της τιμής του μετρητή>, οπότε θα έχουμε:

```
for (i = 1;i <= n;){
  sum += i;
  i++;
}
```

- Η <αρχική τιμή στο μετρητή> και η <ενημέρωση της τιμής του μετρητή>, οπότε θα έχουμε :

```
i = 1;
for (;i <= n;){
  sum += i;
  i++;
}
```

- Το σώμα της εντολής επανάληψης . Π.χ.

```
for (i = 1;i <= n; sum += i++);
```

- Όλα τα παραπάνω και ο έλεγχος αν <ο μετρητής ξεπέρασε την τελική τιμή> (ατέρμονος βρόχος), οπότε απαιτείται η χρήση της εντολής `break`.

Παράδειγμα

Με τις επόμενες εντολές διαβάζουμε χαρακτήρες μέχρι να δώσουμε το χαρακτήρα `q` (quit).

```
char ch;
for (;;){
   ch = (char) System.in.read(); // Εισαγωγή χαρακτήρα
   if ( ch = 'q' ) break; // Τερματισμός βρόχου με q = quit
}
```

❖ Στο προηγούμενο παράδειγμα, απαιτείται και η χρήση του `throws java.io.IOException` στη δήλωση της μεθόδου που χρησιμοποιεί το προηγούμενο τμήμα του κώδικα.

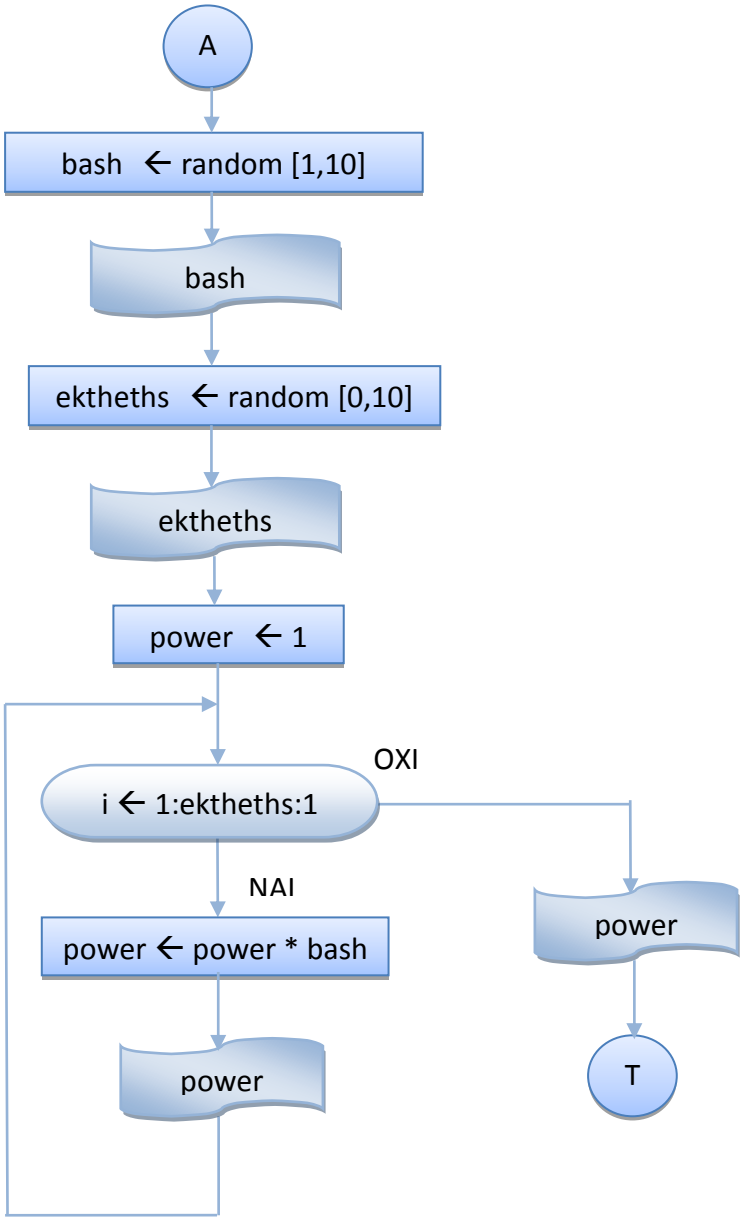
4.2.2 Πρόγραμμα για τον Υπολογισμό του x^y με την Εντολή Επανάληψης for

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα προσομοιώνει τη μέθοδο `Math.pow( Βάση, Εκθέτης )` με **Βάση** έναν **ακέραιο** αριθμό και **εκθέτη** έναν **ακέραιο** αριθμό. Δημιουργεί έναν τυχαίο **ακέραιο** αριθμό στο `1-10` για τη Βάση και έναν τυχαίο ακέραιο αριθμό στο `0-10` για τον Εκθέτη και εμφανίζει τις τιμές τους. Για να υπολογίσει τη δύναμη `Βάση^Εκθέτη`, δίνει την τιμή `1` σαν αρχική τιμή στη δύναμη και με την εντολή `for` πολλαπλασιάζει τη δύναμη με τη Βάση, όσες φορές είναι η ακέραια τιμή του εκθέτη και εμφανίζει κάθε φορά την τιμή της δύναμης.

ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο **ακέραιο** αριθμό `bash` στο `[1, 10]`
2. Τον εμφανίζω
3. Δημιουργώ έναν τυχαίο **ακέραιο** αριθμό `ektheths` στο `[0, 10]`
4. Τον εμφανίζω
5. Δίνω αρχική τιμή στη `Δύναμη power = 1`
6. **Για** τις τιμές του μετρητή `i` από το `1` μέχρι και τον εκθέτη `ektheths`
   Πολλαπλασιάζω τη Δύναμη με τη Βάση (`power ← power * bash`)
   Εμφανίζω την τιμή της Δύναμης `power`
7. Εμφανίζω την τελική τιμή της Δύναμης `power`

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΠΡΟΓΡΑΜΜΑ

```
public class PowerFor {
/*
Πρόγραμμα, το οποίο προσομοιώνει τη μέθοδο pow( Βάση, Εκθέτης ) με Βάση έναν
ακέραιο αριθμό και εκθέτη έναν ακέραιο αριθμό. Δημιουργεί έναν τυχαίο ακέραιο
αριθμό στο 1-10 για τη Βάση και έναν τυχαίο ακέραιο αριθμό στο 0-10 για τον
Εκθέτη και εμφανίζει τις τιμές τους. Για να υπολογίσει τη δύναμη Βάση^Εκθέτη,
δίνει την τιμή 1 σαν αρχική τιμή στη δύναμη και με την εντολή for
πολλαπλασιάζει τη δύναμη με τη Βάση, όσες φορές είναι η ακέραια τιμή του
εκθέτη και εμφανίζει κάθε φορά την τιμή της δύναμης.
*/


public static void main(String[] args) {
       int ektheths, i;
       int power;

       // Δημιουργία τυχαίου ακέραιου αριθμού στο 1 - 10 για τη Βάση
       int bash = (int)(Math.random()*10) + 1;

       // Εμφάνιση της Τιμής της Βάσης
       System.out.println("Η Βάση είναι : " + bash );

       // Δημιουργία τυχαίου ακέραιου αριθμού στο 0 - 10 για τον εκθέτη
       ektheths = (int) ( Math.random()*10);

       // Εμφάνιση της Τιμής του Εκθέτη
       System.out.println("Ο Εκθέτης είναι : " + ektheths );


       // Αρχική Τιμή στη Δύναμη = 1
       power = 1;

       // Για τόσες φορές όσες η τιμή του εκθέτη
       for ( i = 1; i <= ektheths; i++ ) {
           // Υπολογισμός Επόμενης Τιμής της Δύναμης
           power = power * bash;

           // Εμφάνιση της νέας Τιμής της Δύναμης
           System.out.println("Η " + i + "-η Δύναμη είναι : " + power );
       }
       // Εμφάνιση της Τελικής Τιμής της Δύναμης
          System.out.println("\nΤελική Τιμή για τη Δύναμη = " + power );
       }
}
```

Έξοδος Προγράμματος

```
run:
Η Βάση είναι : 6
Ο Εκθέτης είναι : 0

Τελική Τιμή για τη Δύναμη = 1
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Η Βάση είναι : 6
Ο Εκθέτης είναι : 1
Η 1-η Δύναμη είναι : 6

Τελική Τιμή για τη Δύναμη = 6
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Η Βάση είναι : 3
Ο Εκθέτης είναι : 4
Η 1-η Δύναμη είναι : 3
Η 2-η Δύναμη είναι : 9
Η 3-η Δύναμη είναι : 27
Η 4-η Δύναμη είναι : 81

Τελική Τιμή για τη Δύναμη = 81
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Η Βάση είναι : 1
Ο Εκθέτης είναι : 1
Η 1-η Δύναμη είναι : 1

Τελική Τιμή για τη Δύναμη = 1
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Η Βάση είναι : 1
Ο Εκθέτης είναι : 6
Η 1-η Δύναμη είναι : 1
Η 2-η Δύναμη είναι : 1
Η 3-η Δύναμη είναι : 1
Η 4-η Δύναμη είναι : 1
Η 5-η Δύναμη είναι : 1
Η 6-η Δύναμη είναι : 1

Τελική Τιμή για τη Δύναμη = 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

ΑΣΚΗΣΗ 4.4 : Να τροποποιηθεί ο προηγούμενος αλγόριθμος ώστε να ελέγχει και την περίπτωση που **η βάση είναι 0 ή 1** και να εμφανίζει την τιμή της, **χωρίς** να χρειαστεί να κάνει καμιά επανάληψη.

4. 3 Εμφωλευμένες Εντολές Επανάληψης for-while, break, continue

Στα επόμενα παραδείγματα εξετάζεται η χρήση της εντολής επανάληψης `for` μέσα στο σώμα μιας εντολής επανάληψης `while`, η χρήση της εντολής επανάληψης `for` μέσα στο σώμα μιας άλλης εντολής επανάληψης `for`, και η χρήση των εντολών `break` και `continue`.

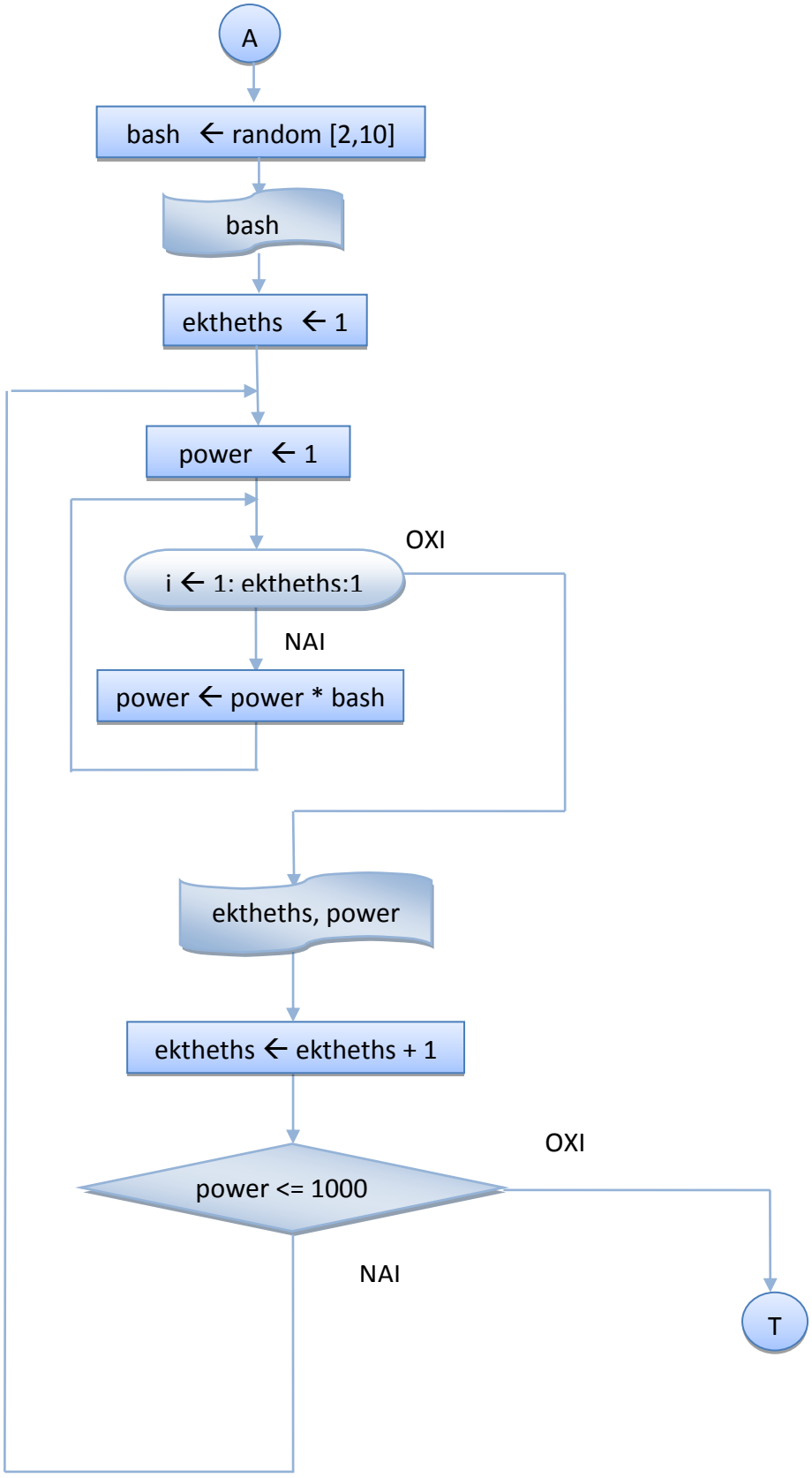
4.3.1 Εμφωλευμένοι Βρόχοι (for – while)

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα υπολογίζει και θα εμφανίζει όλες τις **Δυνάμεις** ενός τυχαίου ακέραιου αριθμού στο [2,10] μέχρι που η Δύναμη να ξεπεράσει το **1000** χρησιμοποιώντας τις Εντολές Επανάληψης **`for`** για τον υπολογισμό της κάθε δύναμης και **`do while`** για τον έλεγχο του τερματισμού.

ΑΛΓΟΡΙΘΜΟΣ

1. Δημιουργώ έναν τυχαίο **ακέραιο** αριθμό `bash` στο `[2, 10]`
2. Τον εμφανίζω
3. Αρχική Τιμή στον Εκθέτη `ektheths` ← `1`
4. **Κάνε** τα παρακάτω
   Δίνω αρχική τιμή στη `Δύναμη power` ← `1`
   Για τις τιμές του μετρητή `i` από το `1` μέχρι και τον εκθέτη `ektheths`
   Πολλαπλασιάζω τη Δύναμη με τη Βάση (`power` ← `power * bash`)
   Εμφανίζω την τιμή της Δύναμης και του εκθέτη
   Αυξάνω την τιμή του εκθέτη κατά 1
   Για όσο η Δύναμη είναι μικρότερη ή ίση του `1000`

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΠΡΟΓΡΑΜΜΑ

```
public class DoWhileFor {
/*
Πρόγραμμα, το οποίο Υπολογίζει και εμφανίζει όλες τις Δυνάμεις ενός τυχαίου
ακέραιου αριθμού στο [2,10] μέχρι που η Δύναμη να ξεπεράσει το 1000
χρησιμοποιώντας τις Εντολές Επανάληψης for για τον υπολογισμό της κάθε
δύναμης και do while για τον έλεγχο του τερματισμού.
*/
public static void main(String[] args) {
        int ektheths, i, power;

       // Δημιουργία τυχαίου ακέραιου αριθμού στο 2 - 10 για τη Βάση
       int bash = (int)(Math.random()*9) + 2;

       // Εμφάνιση της Τιμής της Βάσης
       System.out.println("Η Βάση είναι : " + bash );

       // Αρχική Τιμή στον Εκθέτη = 1
       ektheths = 1;

       do {
         // Αρχική Τιμή στη Δύναμη = 1
         power = 1;

         // Για τόσες φορές όσες η τιμή του εκθέτη
         for ( i = 1; i <= ektheths; i++) {
           // Υπολογισμός Επόμενης Τιμής της Δύναμης
           power = power * bash;
         }
         // Εμφάνιση του εκθέτη και της νέας Τιμής της Δύναμης
                        System.out.println("Η Δύναμη του αριθμού " +
                          bash + "^" + ektheths + " είναι : " + power );

         // Αϋξηση του Εκθέτη κατά 1
         ektheths += 1;
       }
       while (power <= 1000);
}
}
```

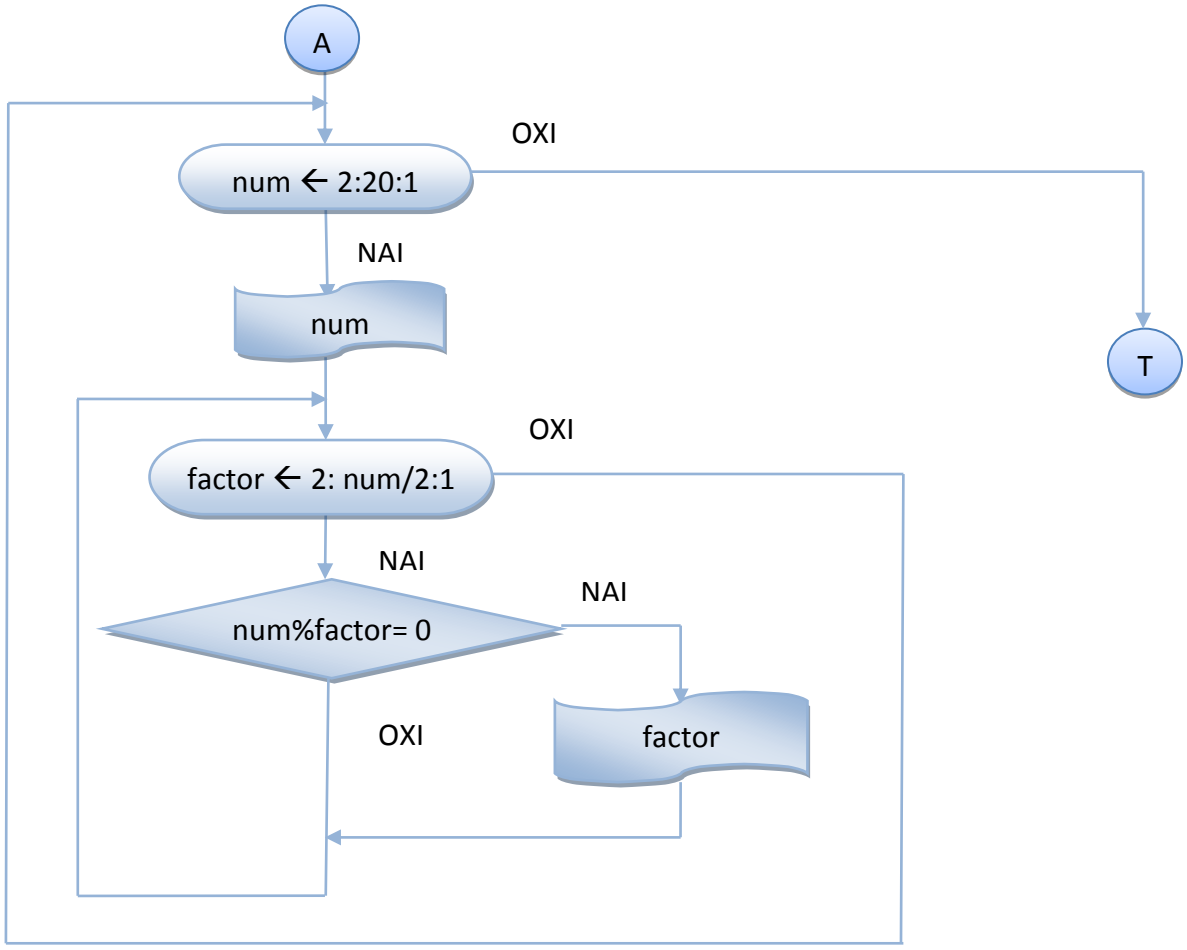
Έξοδος Προγράμματος

```
run:
Η Βάση είναι : 2
Η Δύναμη του αριθμού 2^1 είναι : 2
Η Δύναμη του αριθμού 2^2 είναι : 4
Η Δύναμη του αριθμού 2^3 είναι : 8
Η Δύναμη του αριθμού 2^4 είναι : 16
Η Δύναμη του αριθμού 2^5 είναι : 32
Η Δύναμη του αριθμού 2^6 είναι : 64
Η Δύναμη του αριθμού 2^7 είναι : 128
Η Δύναμη του αριθμού 2^8 είναι : 256
Η Δύναμη του αριθμού 2^9 είναι : 512
Η Δύναμη του αριθμού 2^10 είναι : 1024
BUILD SUCCESSFUL (total time: 0 seconds)
```

4.3.2 Εμφωλευμένοι Βρόχοι (for – for)

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα υπολογίζει όλους τους **παράγοντες** των αριθμών από το `2` μέχρι το `20` (για τον κάθε αριθμό θα βρίσκει και θα εμφανίζει τους **διαιρέτες** του **εκτός** της μονάδας και του ίδιου του αριθμού) χρησιμοποιώντας **δύο** Εντολές Επανάληψης **`for`**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

1. **Για** τις τιμές του μετρητή `num` από το `2` μέχρι και το `20`
   Εμφάνιση της τιμής του μετρητή `num`
   Για τις τιμές του `factor` από το `2` μέχρι και το `num/2`
   Αν ο αριθμός `num` διαιρείται ακριβώς με το `factor`
   Εμφανίζω την τιμή του παράγοντα `factor`

ΠΡΟΓΡΑΜΜΑ

```
public class ForFor {
/*Πρόγραμμα, το οποίο υπολογίζει όλους τους παράγοντες των αριθμών από το 2
μέχρι το 20 ( για τον κάθε αριθμό θα βρίσκει και θα εμφανίζει  τους διαιρέτες
του εκτός της μονάδας και του ίδιου του αριθμού ) χρησιμοποιώντας δύο Εντολές
Επανάληψης for.*/
    public static void main(String[] args) {
        int num, factor;

        // Για τις τιμές του μετρητή num από το 2 μέχρι και το 20
        for ( num = 2;num <= 20;num++ ) {

            // Εμφάνιση της τιμής του μετρητή num
            System.out.print("Ο αριθμός " + num + " έχει παράγοντες : " );

            // Για τις τιμές του factor από το 2 μέχρι και το num/2
            for ( factor = 2;factor <= num/2;factor++ )

                // Αν ο αριθμός num διαιρείται ακριβώς με το factor
                if ( num % factor == 0)

                    // Εμφάνιση της τιμής του παράγοντα factor
                    System.out.print(" " + factor + " " );

            System.out.println();
        }
     }
}
```

Έξοδος Προγράμματος

```
run:
Ο αριθμός 2 έχει παράγοντες :
Ο αριθμός 3 έχει παράγοντες :
Ο αριθμός 4 έχει παράγοντες :  2
Ο αριθμός 5 έχει παράγοντες :
Ο αριθμός 6 έχει παράγοντες :  2  3
Ο αριθμός 7 έχει παράγοντες :
Ο αριθμός 8 έχει παράγοντες :  2  4
Ο αριθμός 9 έχει παράγοντες :  3
Ο αριθμός 10 έχει παράγοντες :  2  5
Ο αριθμός 11 έχει παράγοντες :
Ο αριθμός 12 έχει παράγοντες :  2  3  4  6
Ο αριθμός 13 έχει παράγοντες :
Ο αριθμός 14 έχει παράγοντες :  2  7
Ο αριθμός 15 έχει παράγοντες :  3  5
Ο αριθμός 16 έχει παράγοντες :  2  4  8
Ο αριθμός 17 έχει παράγοντες :
Ο αριθμός 18 έχει παράγοντες :  2  3  6  9
Ο αριθμός 19 έχει παράγοντες :
Ο αριθμός 20 έχει παράγοντες :  2  4  5  10
BUILD SUCCESSFUL (total time: 0 seconds)
```

Άσκηση 4.5 : Να τροποποιηθεί το προηγούμενο πρόγραμμα, ώστε να εμφανίζει **μόνο** τους αριθμούς που έχουν παράγοντες, **ΟΧΙ** και τους πρώτους.

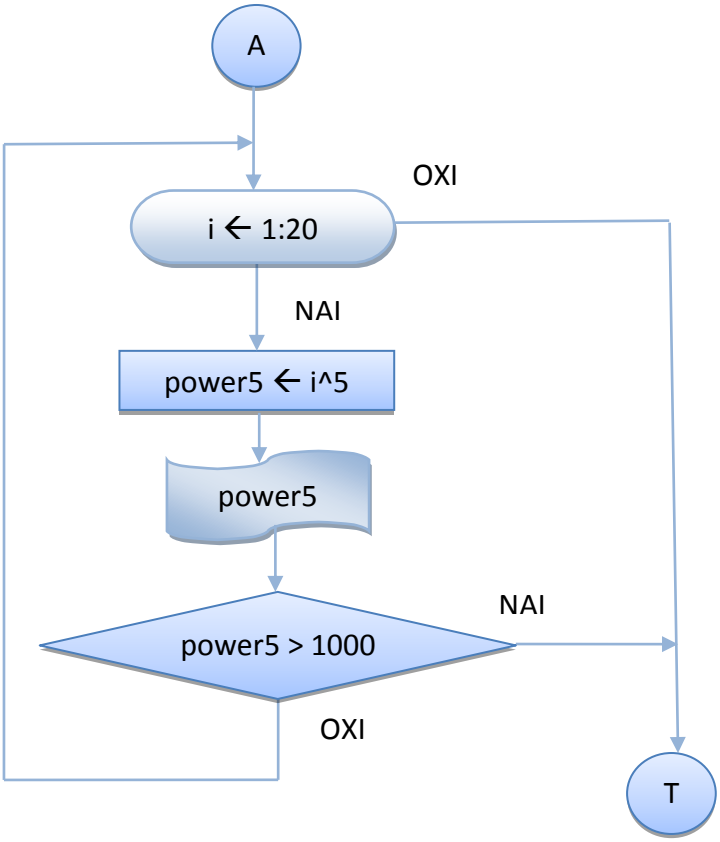
4.3.3 Η Εντολή break

Η εντολή `break` χρησιμοποιείται για τη διακοπή οποιουδήποτε βρόχου επανάληψης, ακόμη κι αν εξακολουθεί να ισχύει η συνθήκη. Εκτός από την έξοδο από κάθε περίπτωση (`case`) της εντολής επιλογής (`switch`) και την έξοδο από τον ατέρμονο βρόχο επανάληψης `for (;;)` μπορεί να χρησιμοποιηθεί για τη διακοπή οποιουδήποτε βρόχου επανάληψης, όπως φαίνεται στο επόμενο παράδειγμα :

Παράδειγμα

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα υπολογίζει με τη χρήση της μεθόδου `Math.pow()` την **πέμπτη** δύναμη των ακέραιων αριθμών από το `1` μέχρι και το `20` και θα την εμφανίζει. Το πρόγραμμα θα τερματίζει με την εντολή `break`, αν η τιμή της πέμπτης δύναμης κάποιου από τους αριθμούς `1-20` είναι μεγαλύτερη του `1000`.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

Για τους αριθμούς `i = 1:20`
    Υπολόγισε την Πέμπτη δύναμη `power5` του αριθμού `i`
    Εμφάνισε την Πέμπτη δύναμη `power5` του αριθμού `i`
    Αν η `power5` είναι μεγαλύτερη του `1000`
            `Διακοπή Βρόχου Επανάληψης`

ΠΡΟΓΡΑΜΜΑ

```
public class Power5 {
    /*
    Πρόγραμμα το οποίο υπολογίζει με τη χρήση της μεθόδου Math.pow() την
    πέμπτη δύναμη των ακέραιων αριθμών από το 1 μέχρι και το 20 και  την
    εμφανίζει. Το πρόγραμμα  τερματίζει με την εντολή break, αν  η τιμή
    της πέμπτης δύναμης κάποιου από τους αριθμούς 1-20 είναι μεγαλύτερη
    του 1000
     */
    public static void main(String[] args) {
        int i;
        int power5;
        // Για τους αριθμούς από το 1 μέχρι και το 20
        for ( i=1;i<=20;i++ ) {
          // Υπολογισμός i^5
          power5 = (int)Math.pow(i,5);
          // Εμφάνιση i^5
          System.out.println(i + "^5 = " + power5 );
          // Έλεγχος - Διακοπή, αν η δύναμη ξεπέρασε το 1000
          if (power5 > 1000 ) break;
        }
    }
}
```

Έξοδος Προγράμματος

```
run:
1^5 = 1
2^5 = 32
3^5 = 243
4^5 = 1024
BUILD SUCCESSFUL (total time: 0 seconds)
```

Άσκηση 4.6 : Να τροποποιηθεί ο προηγούμενος Αλγόριθμος/πρόγραμμα ώστε να ΜΗΝ εμφανίζει την τιμή του `power5`, αν είναι μεγαλύτερη του `1000`.

Άσκηση 4.7 : Να τροποποιηθεί ο προηγούμενος Αλγόριθμος/πρόγραμμα ώστε να κάνει το ίδιο **ΧΩΡΙΣ** τη χρήση των εντολών `for` και `break`.

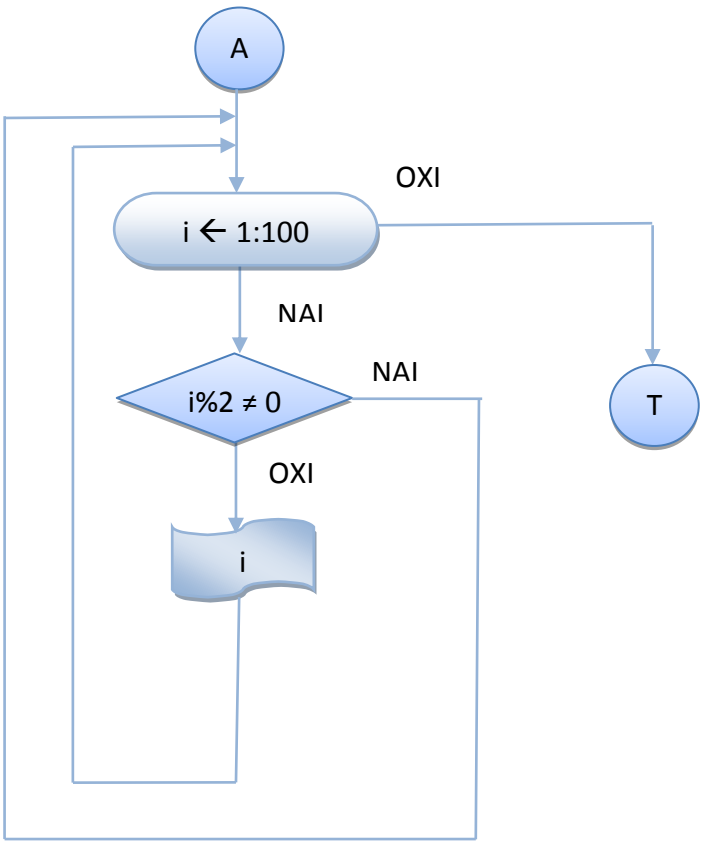
4.3.4 Η Εντολή continue

Η εντολή `continue` είναι το συμπλήρωμα της εντολής `break` και χρησιμοποιείται για να στείλει τον έλεγχο στη συνθήκη του οποιουδήποτε βρόχου επανάληψης, όπως φαίνεται στο επόμενο παράδειγμα :

Παράδειγμα

Να γραφεί Αλγόριθμος/πρόγραμμα, το οποίο θα βρίσκει και θα εμφανίζει όλους τους άρτιους ακέραιους αριθμούς από το 1 ως το 20 με τη χρήση της εντολής `continue`.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

Για τους αριθμούς `i = 1:20`
    Αν ο αριθμός `i` **δεν** διαιρείται ακριβώς με το `2`
        Συνέχισε με τον επόμενο αριθμό
    Εμφάνισε τον αριθμό `i`

ΠΡΟΓΡΑΜΜΑ

```
public class ForContinue {
    /*
     * Πρόγραμμα, το οποίο βρίσκει και  εμφανίζει όλους τους άρτιους
     * ακέραιους αριθμούς από το 1 ως το 20 με τη χρήση της εντολής
     * continue.
     */
    public static void main(String[] args) {
        int i;
        // Για τους αριθμούς από το 1 μέχρι και το 200
        for ( i=1;i<=20;i++ ) {
          // Έλεγχος αν ο αριθμός είναι άρτιος
          if (i%2 != 0) continue;
          // Εμφάνιση i
          System.out.println(i  );
        }
    }
}
```

Έξοδος Προγράμματος

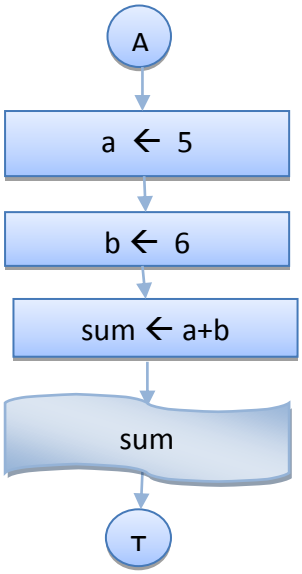
```
run:
2
4
6
8
10
12
14
16
18
20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Άσκηση 4.8 : Να τροποποιηθεί ο προηγούμενος Αλγόριθμος/πρόγραμμα ώστε να κάνει το ίδιο **ΧΩΡΙΣ** τη χρήση της εντολής `continue`.

5 ΜΕΘΟΔΟΙ - ΠΑΡΑΜΕΤΡΟΙ

Να γραφεί πρόγραμμα, το οποίο θα δίνει τις τιμές 5 και 6 σε δύο μεταβλητές **a** και **b** και θα υπολογίζει και θα εμφανίζει το άθροισμά τους **sum**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



ΑΛΓΟΡΙΘΜΟΣ

6. Δίνω την τιμή 5 στο a (**a ← 5**)
7. Δίνω την τιμή 6 στο b (**b ← 6**)
8. Βρίσκω το άθροισμα (**sum ← a+b**)
9. Εμφανίζω την τιμή του **sum**

ΠΡΟΓΡΑΜΜΑ

```
public class SumAB {
/* Πρόγραμμα που δίνει τις τιμές 5 και 6 σε 2 ακέραιες μεταβλητές και
βρίσκει και εμφανίζει το άθροισμά τους*/
    public static void main(String[] args) {

    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
        int a = 5, b = 6;

        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Εμφάνιση  του αθροίσματος sum
        System.out.println("Άθροισμα = " + sum);
        }
}
```

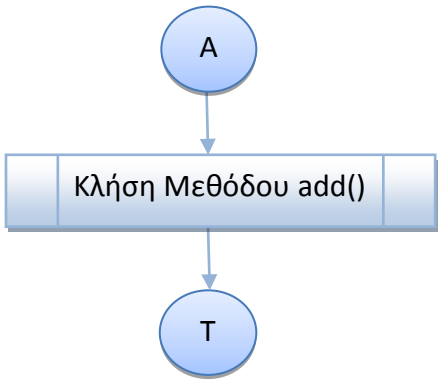
Έξοδος Προγράμματος

```
Άθροισμα = 11
```

5. 1 ΜΕΘΟΔΟΙ

Θα μπορούσαμε να καλέσουμε μια μέθοδο `add()`, η οποία να δίνει τις τιμές στις μεταβλητές **`a, b`** και να υπολογίζει και να εμφανίζει το άθροισμα **`sum`** και η μέθοδος `main()` απλώς να την καλεί. Σ' αυτή την περίπτωση το λογικό διάγραμμα και ο αλγόριθμος της `main()` και της μεθόδου `add()` θα γινόταν :

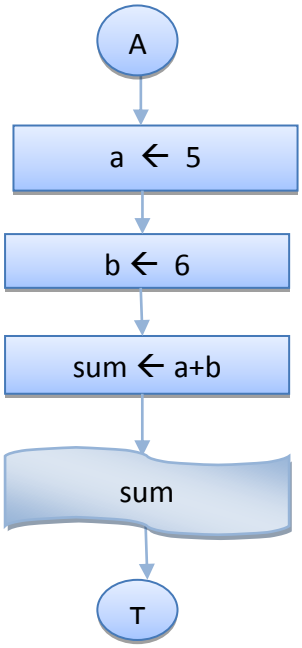
ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ main()



ΑΛΓΟΡΙΘΜΟΣ main()

1. Κλήση Μεθόδου `add()`

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕΘΟΔΟΥ `add()`



ΑΛΓΟΡΙΘΜΟΣ ΜΕΘΟΔΟΥ `add()`

1. Δίνω την τιμή `5` στο `a` (**`a ← 5`**)
2. Δίνω την τιμή `6` στο `b` (**`b ← 6`**)
3. Βρίσκω το άθροισμα **`sum`** (**`sum ← a+b`**)
4. Εμφανίζω την τιμή του **`sum`**

Οι **Μέθοδοι** είναι αυτοτελή τμήματα κώδικα που εκτελούν κάποιες εργασίες και περιέχουν ότι και η μέθοδος `main()` **δηλώσεις** μεταβλητών και **εντολές**. Μια μορφή μεθόδου έχει την παρακάτω σύνταξη :

```
static void <όνομα_μεθόδου>(){
     εντολές;
}
```

ΠΡΟΓΡΑΜΜΑ

```
public class MethodsAdd {
/* Πρόγραμμα που καλεί τη μέθοδο add(), η οποία δίνει τις τιμές 5 και 6
σε 2 ακέραιες μεταβλητές και βρίσκει και εμφανίζει το άθροισμά τους
*/
    static void add() {

    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
        int a = 5, b = 6;

        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Εμφάνιση  του αθροίσματος sum
        System.out.println("Άθροισμα με κλήση της add() = " + sum);
   }


    public static void main(String[] args) {

    // Κλήση Μεθόδου add()
    add();
    }
}
```

Έξοδος Προγράμματος

```
Άθροισμα με κλήση της add() = 11
```

Παρατηρήσεις

- Η μέθοδος `add()` δεν παίρνει **καμία** πληροφορία από την `main()`.
- Δεν στέλνει **καμιά** πληροφορία στην `main()`, γι' αυτό και δηλώνεται τύπου `void`.
- Το `static` χρειάζεται, γιατί δεν είναι μέθοδος κλάσης αντικειμένων.
- Ο κώδικας της μεθόδου `add()` βρίσκεται στην ίδια κλάση με τη μέθοδο `main()` και η κλήση της γίνεται χρησιμοποιώντας απλώς το όνομά της.
- Οι μεταβλητών της μεθόδου λέγονται **τοπικές μεταβλητές**, δημιουργούνται όταν καλείται η μέθοδος, **δεν είναι ορατές στην καλούσα μέθοδο** και παύουν να υπάρχουν, όταν ολοκληρωθεί η εκτέλεση των εντολών της μεθόδου.

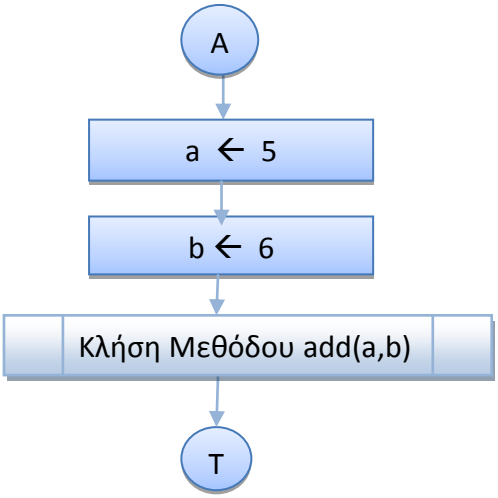
5.1.1 Μέθοδος που Δέχεται Παραμέτρους από τη main()

Η προηγούμενη μέθοδος κάνει μια πολύ συγκεκριμμένη δουλειά. Υπολογίζει το άθροισμα των μεταβλητών **a** και **b** για τις συγκεκριμμένες τιμές 5 και 6. Αν θέλουμε να υπολογίζει και να εμφανίζει το άθροισμα οποιονδήποτε 2 αριθμών **a** και **b** που θα παίρνουν τιμές στην `main()` θα πρέπει να περάσουμε τις τιμές των μεταβλητών **a** και **b** σαν **παραμέτρους** στη μέθοδο. Αυτό γίνεται στη δήλωση της μεθόδου, οπότε η γενική της σύνταξη θα είναι :

```
static void <όνομα_μεθόδου>(<λίστα_παραμέτρων>){
     εντολές;
}
```

όπου η `<λίστα_παραμέτρων>` περιλαμβάνει δηλώσεις μεταβλητών χωρισμένες με κόμμα, μέσω των οποίων **περνάμε** στη μέθοδο τις **τιμές** της μεθόδου που την καλεί.

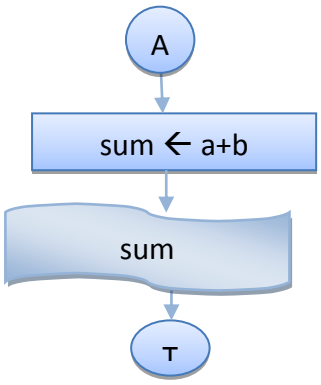
ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ main()



ΑΛΓΟΡΙΘΜΟΣ main()

1. Δίνω την τιμή 5 στο a (**a ← 5**)
2. Δίνω την τιμή 6 στο b (**b ← 6**)
3. Κλήση Μεθόδου `add(a, b)`

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕΘΟΔΟΥ `add(a, b)`

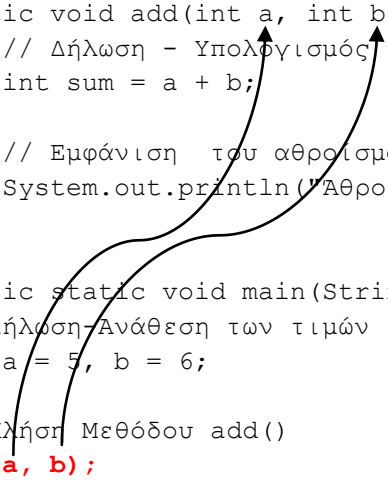


ΑΛΓΟΡΙΘΜΟΣ ΜΕΘΟΔΟΥ `add(a,b)`

1. Βρίσκω το άθροισμα (**sum ← a+b**)
2. Εμφανίζω την τιμή του **sum**

ΠΡΟΓΡΑΜΜΑ

```
public class MethodsAddAB {
/* Πρόγραμμα που καλεί τη μέθοδο add(), η οποία δίνει τις τιμές 5 και 6
σε 2 ακέραιες μεταβλητές και βρίσκει και εμφανίζει το άθροισμά τους
*/
    static void add(int a, int b) {
        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Εμφάνιση  του αθροίσματος sum
        System.out.println("Άθροισμα με κλήση της add(a, b) = " + sum);
    }

    public static void main(String[] args) {
    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
    int a = 5, b = 6;

    // Κλήση Μεθόδου add()
    add(a, b);
    }
}
```



Έξοδος Προγράμματος

```
Άθροισμα με κλήση της add(a, b) = 11
```

Παρατηρήσεις

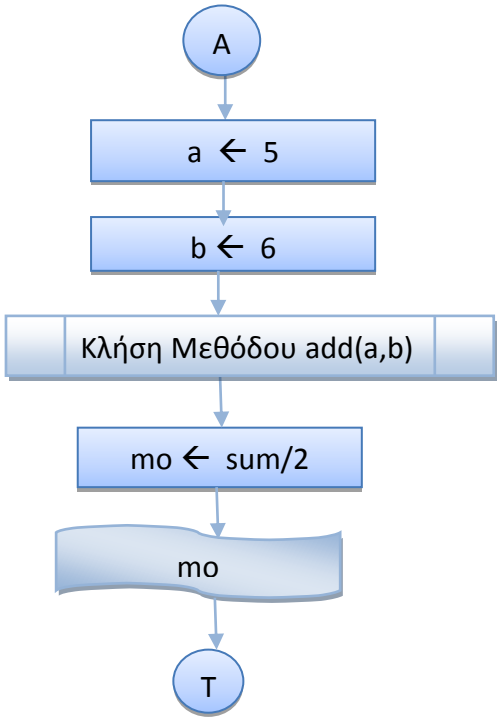
- Οι πληροφορίες περνάνε στη μέθοδο μέσω των μεταβλητών `a` και `b` στη δήλωση.
- Οι μεταβλητές `a` και `b` στη δήλωση της μεθόδου λέγονται **παράμετροι** και μπορούν να έχουν **διαφορετικά** ονόματα από τις μεταβλητές της `main()`).
- Μπορεί να χρησιμοποιηθεί **οποιοδήποτε όνομα** για τις παραμέτρους, αρκεί να χρησιμοποιούνται τα ίδια ονόματα και μέσα στη μέθοδο.
- Πρέπει να συμφωνεί ο **αριθμός**, η **σειρά** και ο **τύπος** των **ορισμάτων** (μεταβλητών της καλούσας μεθόδου) και των **παραμέτρων** της μεθόδου.
- Στην κλήση της μεθόδου **δε χρειάζονται οι τύποι** των μεταβλητών- ορισμάτων.
- Στην κλήση της μεθόδου τα ονόματα των παραμέτρων αντικαθίστανται από τα ονόματα των μεταβλητών της καλούσας μεθόδου. Οι τιμές των μεταβλητών (ορίσματα) περνάνε στην μέθοδο.
- Η μέθοδος **τελειώνει** όταν εκτελεστούν όλες οι εντολές ή αν υπάρχει η εντολή `return`.
- Μπορεί να μη χρησιμοποιηθεί η τοπική μεταβλητή `sum`. Σ' αυτή την περίπτωση, ο κώδικας της μεθόδου θα γίνει :

```
static void add(int a, int b) {
    // Υπολογισμός - Εμφάνιση  του αθροίσματος (a + b)
    System.out.println("Άθροισμα = " + (a + b);
    }
```

5.1.2 Μέθοδος που Επιστρέφει και το Άθροισμα στη `main()`

Να γραφεί πρόγραμμα, το οποίο θα δίνει τις τιμές `5` και `6` σε δύο μεταβλητές `a` και `b` και θα καλεί τη μέθοδο `add()`, η οποία υπολογίζει και **επιστρέφει** στη `main()` την τιμή του αθροίσματος **`sum`**, την οποία και θα εμφανίζει. Μετά το πρόγραμμα θα υπολογίζει και θα εμφανίζει και το μέσο όρο **`mo`**.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ `main()`



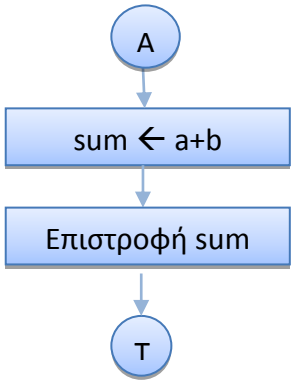
ΑΛΓΟΡΙΘΜΟΣ `main()`

1. Δίνω την τιμή `5` στο `a` (**`a ← 5`**)
2. Δίνω την τιμή `6` στο `b` (**`b ← 6`**)
3. Κλήση Μεθόδου `add(a, b)`
4. Υπολογισμός Μέσου Όρου (**`mo ← sum/2`**)
5. Εμφάνιση Μέσου Όρου **`mo`**

ΑΛΓΟΡΙΘΜΟΣ ΜΕΘΟΔΟΥ
`add(a,b)`

1. Βρίσκω το άθροισμα (**sum ← a+b**)
2. **Επιστρέφω** το άθροισμα **sum**

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕΘΟΔΟΥ `add(a, b)`



Για να μπορέσει μια μέθοδος να **επιστρέψει** μια τιμή στην καλούσα μέθοδο, θα πρέπει :

- Να δηλωθεί στην **επικεφαλίδα** της μεθόδου ο **τύπος** της τιμής που θα επιστραφεί.
- Να υπάρχει **μέσα** στον κώδικα της μεθόδου η εντολή **return** με την οποία θα επιστραφεί αυτή η τιμή.

Η σύνταξη της δήλωσης της μεθόδου θα έχει τη μορφή :

```
static <τύπος_επιστροφής> <όνομα_μεθόδου> (<λίστα_παραμέτρων>){
    εντολές;
    return <έκφραση_τύπου_επιστροφής>;
    }
```

ΠΡΟΓΡΑΜΜΑ

```
public class MethodsIntAddAB {
/*
Πρόγραμμα το οποίο δίνει τις τιμές 5 και 6 σε δύο μεταβλητές a και b και
καλεί τη μέθοδο add(), η οποία υπολογίζει και επιστρέφει στη main()την τιμή
του αθροίσματος sum, την οποία και εμφανίζει. Μετά το πρόγραμμα υπολογίζει
και εμφανίζει και το μέσο όρο mo.
*/
    static int add(int a, int b) {

        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Επιστροφή Τιμής Αθροίσματος sum
        return sum;
   }

    public static void main(String[] args) {

    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
    int a = 5, b = 6;

    // Κλήση Μεθόδου add()
    int sum = add(a, b);

    // Δήλωση - Υπολογισμός του Μέσου Όρου mo
    double mo = (double) sum/2;

    // Εμφάνιση  του αθροίσματος sum και του Μέσου Όρου mo
    System.out.println("Άθροισμα = " + sum + " Μέσος Όρος = " + mo );     }
}
```

Έξοδος Προγράμματος

```
run:
Άθροισμα = 11 Μέσος Όρος = 5.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατηρήσεις

- Όταν επιστρέφει η τιμή μιας μεταβλητής από κάποια μέθοδο πρέπει να δηλώνεται ο **τύπος** της.
- Ο **τύπος της μεθόδου** πρέπει να είναι ίδιος με τον **τύπο** της **μεταβλητής** που επιστρέφει.
- Η **κλήση** της μεθόδου χρησιμοποιείται στην καλούσα μέθοδο σαν μια οποιαδήποτε μεταβλητή σε εντολές εκχώρησης, ελέγχου, εμφάνισης κ.λ.π..
- Η μέθοδος επιστρέφει **μόνο μια τιμή** για **μεταβλητές**, εκφράσεις **απλού τύπου**.
- Μπορεί να υπάρχουν στη μέθοδο **περισσότερες** από μία εντολές **`return`**.
- Μπορεί να μη χρησιμοποιηθεί η τοπική μεταβλητή `sum` και να επιστρέψει η έκφραση υπολογισμού του. Σ' αυτή την περίπτωση ο κώδικας της μεθόδου θα είναι :

```
static int add(int a, int b) {
    // Υπολογισμός - Επιστροφή Τιμής Αθροίσματος (a + b)
    return (a + b);
}
```

- Μπορεί να μην χρησιμοποιηθεί η μεταβλητή `sum` στη `main()` και να χρησιμοποιηθεί το όνομα της μεθόδου με τις παραμέτρους. Αντί των εντολών :

```
// Κλήση Μεθόδου add()
int sum = add(a, b);
// Δήλωση - Υπολογισμός του Μέσου Όρου mo
double mo = (double) sum/2;
```

θα μπορούσαμε να γράψουμε :

```
// Δήλωση - Υπολογισμός του Μέσου Όρου mo - Κλήση Μεθόδου add()
double mo = (double) (add(a, b))/2;
```

- Ο τύπος επιστροφής μιας μεθόδου μπορεί να είναι οποιοσδήποτε από τους απλούς τύπους, ακέραιος (`byte, short, int, long`), κινητής υποδιαστολής (`float, double`), χαρακτήρας (`char`), ή λογικός (`boolean`) που επιστρέφει τις τιμές `true` ή `false`, όπως στο επόμενο παράδειγμα :

5.1.2 Μέθοδος που Επιστρέφει Αποτέλεσμα Τύπου boolean

Να γραφεί πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο **a** στο [1, 10], θα καλεί μια μέθοδο, η οποία θα βρίσκει αν ο αριθμός είναι ΑΡΤΙΟΣ ή ΠΕΡΙΤΤΟΣ και θα εμφανίζει το κατάλληλο μήνυμα.

ΠΡΟΓΡΑΜΜΑ

```
public class BooleanArtios {
/*
Πρόγραμμα, το οποίο θα δημιουργεί έναν τυχαίο ακέραιο a στο [1, 10], καλεί
μια μέθοδο, η οποία  βρίσκει αν ο αριθμός είναι ΑΡΤΙΟΣ ή ΠΕΡΙΤΤΟΣ      και
εμφανίζει το κατάλληλο μήνυμα
*/
    static boolean isArtios(int a){
        boolean artios = false;  // Αρχική τιμή false
        if ( a%2 == 0) // Ο a διαιρείται ακριβώς με το 2
          artios = true; // Είναι ΑΡΤΙΟΣ
        return artios; // Επιστροφή Boolean τιμής
    }
    public static void main(String[] args) {
        int a = (int)(Math.random()*10) + 1; // Τυχαίος Ακέραιος
        if (isArtios(a)) // Είναι άρτιος ???
           System.out.println("Ο αριθμός " + a + " είναι ΑΡΤΙΟΣ");
        else
           System.out.println("Ο αριθμός " + a + " είναι ΠΕΡΙΤΤΟΣ");
    }
}
```

Έξοδος Προγράμματος

```
run:
Ο αριθμός 6 είναι ΑΡΤΙΟΣ
BUILD SUCCESSFUL (total time: 0 seconds)

run:
Ο αριθμός 9 είναι ΠΕΡΙΤΤΟΣ
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

Δε χρειάζεται να ελέγξουμε αν η τιμή που επιστρέφει η μέθοδος **`isArtios(a)`** είναι `true` ή `false`, δηλαδή η εντολή **`if (isArtios(a))`** να γίνει :

`if (isArtios(a)) == true)`

5.3 Πέρασμα Παραμέτρων σε Μεθόδους

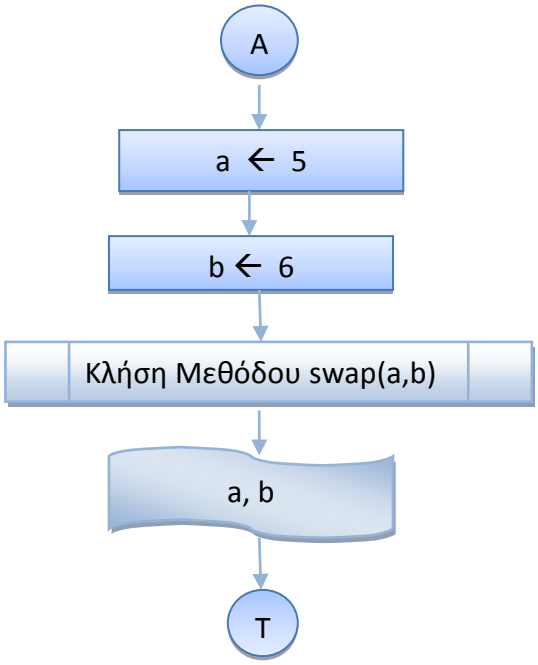
Οι παράμετροι των βασικών τύπων περνάνε στην μέθοδο με **τιμή**, δηλαδή στη μέθοδο είναι διαθέσιμο ένα **αντίγραφο** της τιμής της μεταβλητής και **ΟΧΙ** η **διεύθυνσή** της, οπότε **δεν αλλάζει το περιεχόμενο** που είχαν πριν, ακόμα και αν αλλάζει η τιμή τους μέσα στη μέθοδο. Αυτό μπορεί να γίνει κατανοητό με το επόμενο παράδειγμα :

5.3.1 Παράμετροι με Τιμή – Ανταλλαγή των Τιμών Δυο Μεταβλητών

Να γραφεί πρόγραμμα που να καλεί μια μέθοδο η οποία **ανταλλάσει** τις τιμές 2 ακέραιων αριθμών.

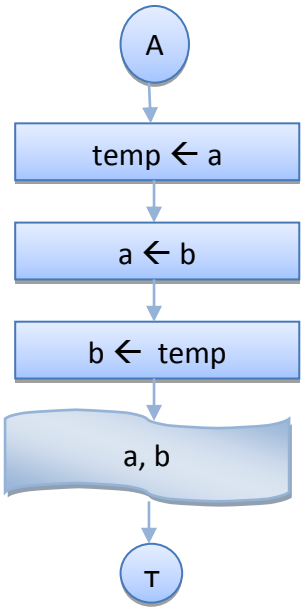
ΑΛΓΟΡΙΘΜΟΣ main()

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ main()



1. Δίνω την τιμή `5` στο `a` (**a ← 5**)
2. Δίνω την τιμή `6` στο `b` (**b ← 6**)
3. Κλήση Μεθόδου `swap(a, b)`
4. Εμφάνιση `a, b`

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ ΜΕΘΟΔΟΥ `swap(a, b)`



ΑΛΓΟΡΙΘΜΟΣ ΜΕΘΟΔΟΥ `swap(a,b)`

1. Αποθηκεύω το **a** στο **temp** (**temp←a**)
2. Αποθηκεύω το **b** στο **a** (**a ← b**)
3. Αποθηκεύω το **temp** στο **b** (**temp ← b**)
4. Εμφάνιση `a, b`

ΠΡΟΓΡΑΜΜΑ

```
public class SwapAB {
/* Πρόγραμμα το οποίο δίνει τις τιμές 5 και 6 σε δύο μεταβλητές a και b και
καλεί τη μέθοδο swap(), η οποία ανταλάσσει τις τιμές των a και b, τις οποίες
και εμφανίζει.
*/
static void swap(int a, int b){
    // Μέθοδος ανταλλαγής των τιμών 2 ακέραιων μεταβλητών
    int temp = a;
    a = b;
    b = temp;
    System.out.print("Τιμές a, b μέσα στη Μέθοδο : ");
    System.out.println("a = " + a + ", b = " + b);
    }

   public static void main(String[] args) {
    int a = 5, b = 6;
    swap( a,  b);
    System.out.print("Τιμές a, b μετά την κλήση της μεθόδου : ");
    System.out.println("a = " + a + ", b = " + b);
    }
}
```

Έξοδος Προγράμματος

```
run:
Τιμές a, b μέσα στη Μέθοδο : a = 6, b = 5
Τιμές a, b μετά την κλήση της μεθόδου : a = 5, b = 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατηρήσεις

- Τα ορίσματα (οι μεταβλητές `a, b` της `main()`) περνάνε στην μέθοδο **με τιμή**, δηλαδή ένα αντίγραφο του περιεχομένου των μεταβλητών `a, b` **και <u>όχι</u> οι διευθύνσεις** των μεταβλητών.
- Με τις αλλαγές στην μέθοδο το περιεχόμενο των μεταβλητών `a, b` της `main()` **δεν επηρεάζεται.**
- Μετά την κλήση της μεθόδου οι μεταβλητές `a, b` περιέχουν **τις ίδιες τιμές** που είχαν και **πριν** την κλήση της μεθόδου.

5.4 Υπερφόρτωση Μεθόδων

Η `Java` έχει το πλεονέκτημα σε σχέση με άλλες γλώσσες προγραμματισμού να μη χρειάζεται διαφορετικά ονόματα μεθόδων στην περίπτωση που είναι διαφορετικός ο τύπος των

παραμέτρων. Π.χ. για τον υπολογισμό της απόλυτης τιμής ενός αριθμού, η C διαθέτει στη Μαθηματική της Βιβλιοθήκη τις μεθόδους abs και fabs για τον υπολογισμό της απόλυτης τιμής ενός ακεραίου ή πραγματικού αριθμού αντίστοιχα. Στη Java μπορεί να χρησιμοποιηθεί το **ίδιο όνομα** της μεθόδου (**υπερφόρτωση** μεθόδων), αρκεί να διαφέρει η **υπογραφή** της μεθόδου, δηλαδή ο τύπος επιστροφής ή και ο τύπος των παραμέτρων. Θα μπορούσαν στην ίδια κλάση να υπάρχουν όλες οι εκδόσεις της μεθόδου add() που χρησιμοποιήθηκαν στο Κεφάλαιο 5 – **δεν μπορούν να συνυπάρχουν οι μέθοδοι void add(int a, int b)** και **int add(int a, int b) γιατί ενώ έχουν παραμέτρους του ιδίου τύπου διαφέρουν ΜΟΝΟ στον τύπο επιστροφής** - και η αντίστοιχη μέθοδος που δέχεται και επιστρέφει τιμές τύπου **double** :

```
static double add(double a, double b) {
    // Υπολογισμός – Επιστροφή Τιμής Αθροίσματος (a + b)
    return (a + b);
}
```

οπότε, ανάλογα με τους **τύπους** των ορισμάτων που περνάμε στη main() θα εκτελείται και ο κώδικας της **αντίστοιχης** μεθόδου.

5.4.1 Εφαρμογή Υπερφόρτωσης Μεθόδων

Να γραφεί πρόγραμμα που να καλεί τις μεθόδους void add(),int add(int a, int b) και double add(double a, double b) και να εμφανίζει το άθροισμα και το Μέσο Όρο.

ΠΡΟΓΡΑΜΜΑ

```
public class YperfortoshAdd {
/*
Πρόγραμμα που  καλεί τις μεθόδους void add(), int add(int a, int b) και
double add(double a, double b) και εμφανίζει το άθροισμα και το Μέσο Όρο.
*/
    static void add() {

    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
        int a = 5, b = 6;
        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Εμφάνιση  του αθροίσματος sum
        System.out.println("Κλήση void add() - Άθροισμα = " + sum);
        }


   static int add(int a, int b) {

        // Δήλωση - Υπολογισμός του αθροίσματος sum
        int sum = a + b;

        // Επιστροφή Τιμής Αθροίσματος sum
        return sum;
        }
```

```
    static double add(double a, double b) {

        // Υπολογισμός - Επιστροφή Τιμής Αθροίσματος (a + b)
        return (a + b);
    }

    public static void main(String[] args) {

    // Κλήση Μεθόδου add()
    add();


    // Δήλωση-Ανάθεση των τιμών 5, 6 στις ακέραιες μεταβλητές a, b
    int a = 5, b = 6;

    // Κλήση Μεθόδου int add(a, b)
    int isum = add(a, b);

    // Δήλωση - Υπολογισμός του Μέσου Όρου mo
    double mo = (double) isum/2;

    // Εμφάνιση  του αθροίσματος isum και του Μέσου Όρου mo
    System.out.println("Κλήση int add() - Άθροισμα = " + isum
                       + " Μέσος Όρος = " + mo );

    // Κλήση Μεθόδου double add(a, b)
    double dsum = add(a, b);

    // Δήλωση - Υπολογισμός του Μέσου Όρου mo
    mo = dsum/2;

    // Εμφάνιση  του αθροίσματος dsum και του Μέσου Όρου mo
    System.out.println("Κλήση double add() - Άθροισμα = " + dsum
                       + " Μέσος Όρος = " + mo );

    }
}
```

Έξοδος Προγράμματος

```
run:
Κλήση void add() - Άθροισμα = 11
Κλήση int add() - Άθροισμα = 11 Μέσος Όρος = 5.5
Κλήση double add() - Άθροισμα = 11 Μέσος Όρος = 5.5
BUILD SUCCESSFUL (total time: 0 seconds)
```

6 ΠΙΝΑΚΕΣ

Πίνακας είναι μια διάταξη στοιχείων που το καθένα τους έχει κάποια συγκεκριμένη **θέση**. Καταλαμβάνει μια περιοχή μνήμης με το λογικό όνομα του πίνακα, ενώ κάθε στοιχείο του ξεχωρίζει με κάποιο **δείκτη**. Στη Java η αρίθμηση των δεικτών ξεκινάει από το 0 μέχρι το μέγεθος του πίνακα-1.

Παράδειγμα ενός πίνακα 5 θέσεων με περιεχόμενα τους αριθμούς 1-5.

1	2	3	4	5
Θέση 0	Θέση 1	Θέση 2	Θέση 3	Θέση 4

Δήλωση Πίνακα

Η Δήλωση ενός Πίνακα στη Java γίνεται με τη δήλωση του **τύπου** των στοιχείων που θα αποθηκεύσει, το **όνομά** του, το **μέγεθός** του και τον τελεστή **new**, επειδή οι πίνακες στη Java είναι αντικείμενα.

Παράδειγμα

```
int pin1[] = new int[5];
```

όπου δηλώνουμε τον πίνακα pin1, ο οποίος θα αποθηκεύσει 5 ακέραιους αριθμούς.

Το ίδιο αποτέλεσμα έχουμε με την εντολή

```
int[] pin1 = new int[5];
```

αρκεί να υπάρχουν οι αγκύλες στον τύπο ή στο όνομα του πίνακα.

Το μέγεθος του πίνακα μπορεί να είναι η τιμή μιας μεταβλητής. Οι επόμενες εντολές έχουν το ίδιο αποτέλεσμα :

```
int n = 5;
int pin1[] = new int[n];
```

Ένας πίνακας μπορεί να δημιουργηθεί με **δυναμική αρχικοποίηση**, όπου οι τιμές του περικλείονται σε άγκιστρα και χωρίζονται με κόμμα. Σ' αυτή την περίπτωση **δε** χρειάζεται ο τελεστής **new**.

Παράδειγμα

```
int pin2[] = {1, 2, 3, 4, 5};
```

όπου δηλώνουμε τον πίνακα pin2, ο οποίος θα αποθηκεύσει τους ακέραιους αριθμούς από το 1 μέχρι το 5. **Σαν στοιχεία μέσα στις αγκύλες, εκτός από σταθερούς αριθμούς, μπορεί να είναι ονόματα μεταβλητών, εκφράσεις ή κλήσεις μεθόδων.**

Η πρόσβαση σε κάποιο στοιχείο του πίνακα γίνεται με το όνομά του και τη θέση του (δείκτης) μέσα σε αγκύλες. Π.χ. το `pin1[3]` αναφέρεται στο 4^{ο} στοιχείο του `pin1`.

6.1 Παράδειγμα Δημιουργίας – Γεμίσματος 2 πινάκων με random τιμές και Δυναμική Αρχικοποίηση

- Να γραφεί Αλγόριθμος/Πρόγραμμμα το οποίο γεμίζει έναν πίνακα με **δυναμική αρχικοποίηση** και δίνει τις τιμές από το `1` μέχρι το `5`, δηλώνει έναν άλλον πίνακα `5` θέσεων, τον οποίο γεμίζει με τυχαίες τιμές και εμφανίζει τα περιεχόμενα των `2` πινάκων.

Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin1`
2. Δήλωση - Γέμισμα πίνακα `pin2` με τυχαίες τιμές
   Για κάθε θέση `i = 0` μέχρι (`n – 1`)
   Εκχωρούμε μια τυχαία τιμή μεταξύ `0` και `10` στο στοιχείο `pin2[i]`
3. Εμφάνιση στοιχείων πίνακα `pin1`
   Για κάθε θέση `i = 0` μέχρι 4
   Εμφανίζουμε το στοιχείο `pin1[i]`
4. Εμφάνιση στοιχείων πίνακα `pin2`
   Για κάθε θέση `i = 0` μέχρι (`n – 1`)
   Εμφανίζουμε το στοιχείο `pin2[i]`

Πρόγραμμα

```
public class Pin1 {
/*
Το πρόγραμμα δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση και τον
γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ το δεύτερο τον  γεμίζει με
τυχαίες τιμές από 0 μέχρι 10 και εμφανίζει τα στοιχεία των 2 πινάκων
*/
    public static void main(String[] args) {
        int i;

        // Εκχώρηση ακέραιας τιμής στο μέγεθος του πίνακα 2
        int n = 5;

        // Δήλωση - Αρχικοποίηση πίνακα 1
        int pin1[] = {1,2,3,4,5};

       // Δήλωση πίνακα 2
        double pin2[] = new double[n];
        // Γέμισμα πίνακα 2 με τυχαίες ακέραιες τιμές
        for (i = 0;i <= n-1;i++)
        {
         pin2[i] = Math.random()*10;
        }

        // Εμφάνιση στοιχείων πίνακα 1
        System.out.print("Πίνακας 1 = ");
        for (i = 0;i <= 4;i++)
```

```
        {
         System.out.print(pin1[i] + " " );
        }
        System.out.println();

        // Εμφάνιση στοιχείων πίνακα 2
        System.out.println("\nΠίνακας 2\n");
        for (i = 0;i <= n-1;i++)
        {
         System.out.println(pin2[i] + " " );
        }
        System.out.println();
    }

}
```

Έξοδος Προγράμματος :

```
run:

Πίνακας 1 = 1 2 3 4 5

Πίνακας 2

4.577221658514528
1.2033315418144874
6.063131490340102
9.267092004007694
1.1890233835178776
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.3 Οι Πίνακες σαν Παράμετροι σε Μεθόδους

Σε αντίθεση με της μεταβλητές απλού τύπου, οι πίνακες περνάνε **με αναφορά** σαν παράμετροι στις μεθόδους, οπότε μπορεί να αλλάξει το περιεχόμενό τους.

Η μέθοδος μπορεί επίσης να **επιστρέφει πίνακα**, πράγμα που πρέπει να δηλωθεί στην υπογραφή της μεθόδου.

Παράδειγμα

Η παρακάτω μέθοδος

```
static double[] fillPin2a(int n ){
// Μέθοδος δημιουργίας στοιχείων πίνακα 2, επιστρεφόμενος τύπος = Πίνακας
        double pin2[] = new double[n];
        for (int i = 0;i <= n-1;i++)
        {
         pin2[i] = Math.random()*10;
        }
        return pin2;
    }
```

επιστρέφει πίνακα με την εντολή **`return pin2 - double pin2[] = new double[n],`** ο οποίος περιέχεται στη δήλωσή της μεθόδου (**`double[]`** `fillPin2a(int n ){`).

6.3.1 Παράδειγμα Δημιουργίας – Γεμίσματος 2 πινάκων με random τιμές και Δυναμική Αρχικοποίηση – Χρήση Μεθόδων για Γέμισμα - Εμφάνιση

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο γεμίζει έναν πίνακα με **δυναμική αρχικοποίηση** και δίνει τις τιμές από το 1 μέχρι το 5, δηλώνει έναν άλλον πίνακα 5 θέσεων, τον οποίο γεμίζει με **τυχαίες τιμές** καλώντας τη μέθοδο **`fillPin2()`** και τη μέθοδο **`fillPin2a()`**, η οποία επιστρέφει πίνακα και εμφανίζει τα περιεχόμενα των 2 πινάκων καλώντας τις μεθόδους **`showPin1()`** και **`showPin2()`**. Η μέθοδος `showPin2()` καλείται επίσης με παράμετρο την κλήση της μεθόδου `fillPin2a()` για να εμφανίσει τα στοιχεία του πίνακα που επιστρέφει η μέθοδος `fillPin2a()`.

Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα 1
2. Δήλωση – Κλήση μεθόδου **`fillPin2()`** για το γέμισμα του πίνακα 2 με τυχαίες τιμές
3. Εμφάνιση στοιχείων πίνακα 1 - Κλήση μεθόδου **`showPin1()`**
4. Εμφάνιση στοιχείων πίνακα 2 - Κλήση μεθόδου **`showPin2()`**
5. Γέμισμα πίνακα 2 με τυχαίες τιμές - Επιστροφή πίνακα - Κλήση μεθόδου `fillPin2a()`
6. Εμφάνιση στοιχείων πίνακα 2 - Κλήση μεθόδου **`showPin2()`**
7. Γέμισμα πίνακα 2 με τυχαίες τιμές - Επιστροφή πίνακα - Κλήση μεθόδου `fillPin2a()` – παράμετρος στην εντολή `System.out.println`
8. Εμφάνιση στοιχείων πίνακα 2 - Κλήση μεθόδου **`showPin2()`**

Πρόγραμμα

```
public class PinMethods {
/*
    Το πρόγραμμα δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση,
    και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ το δεύτερο τον
    γεμίζει με τυχαίες τιμές από 0 μέχρι 10 και εμφανίζει τα στοιχεία των 2
    πινάκων. Για το γέμισμα του πίνακα 2 και την εμφάνιση των στοιχείων των
    2 πινάκων χρησιμοποιεί τις static μεθόδους showPin1(), showPin2(),
    fillPin2() και fillPin2a() η οποία επιστρέφει πίνακα.
    */
    static void showPin1(int pin1[] ){
    // Μέθοδος Εμφάνισης στοιχείων πίνακα 1
        System.out.print("Πίνακας 1 = ");
        for (int i = 0;i <= 4;i++)
        {
         System.out.print(pin1[i] + " " );
        }
        System.out.println();
     }

    static void fillPin2(int n, double pin2[] ){
```

```
    // Μέθοδος δημιουργίας στοιχείων πίνακα1, πέρασμα μεταβλητής pin2 με
αναφορά
        for (int i = 0;i <= n-1;i++)
        {
         pin2[i] = Math.random()*10;
        }
    }

    static double[] fillPin2a(int n ){
    // Μέθοδος δημιουργίας στοιχείων πίνακα 2, επιστρεφόμενος τύπος = Πίνακας
        double pin2[] = new double[n];
        for (int i = 0;i <= n-1;i++)
        {
         pin2[i] = Math.random()*10;
        }
        return pin2;
     }

    static void showPin2(int n, double pin2[] ){
    // Μέθοδος Εμφάνισης στοιχείων πίνακα 2
       for (int i = 0;i <= n-1;i++)
        {
         System.out.println(pin2[i] + " " );
        }
        System.out.println();
    }

    public static void main(String[] args) {
        int i;

        // Εκχώρηση ακέραιας τιμής στο μέγεθος του πίνακα 2
        int n = 5;

        // Δήλωση - Αρχικοποίηση πίνακα 1
        int pin1[] = {1,2,3,4,5};

        // Δήλωση πίνακα 2
        double pin2[] = new double[n];

        // Γέμισμα πίνακα 2 με τυχαίες  τιμές
        fillPin2( n,  pin2);

        // Εμφάνιση στοιχείων πίνακα 1
        showPin1(pin1);
        // Εμφάνιση στοιχείων πίνακα 2
        System.out.println("\nΠίνακας 2\n ");
        showPin2(n, pin2);

        // Γέμισμα πίνακα 2 με τυχαίες τιμές - Επιστροφή πίνακα
        pin2 = fillPin2a( n );

        // Εμφάνιση στοιχείων πίνακα 2 - Επιστροφή πίνακα
        System.out.println("Πίνακας 2 - Επιστροφή πίνακα\n ");
        showPin2(n, pin2);

        // Εμφάνιση στοιχείων πίνακα 2 - Γέμισμα πίνακα 2 με τυχαίες τιμές -
        // Επιστροφή πίνακα
       System.out.println("Πίνακας 2 - Επιστροφή πίνακα - κλήση μεθόδου στην"
                          + " εντολή System.out.println\n ");
        showPin2(n, fillPin2a( n ));
    }
}
```

Έξοδος Προγράμματος :

```
run:

Πίνακας 1 = 1 2 3 4 5

Πίνακας 2

3.6518718113025295
9.028672546645574
9.803978963288898
5.312458213990624
7.522460445626495

Πίνακας 2 - Επιστροφή πίνακα

5.432465106350786
1.4239093219143784
4.4804895751466685
5.617742872661365
2.666166716016672

Πίνακας 2 - Επιστροφή πίνακα - κλήση μεθόδου στην εντολή System.out.println

0.5326041961274375
9.610193943893067
7.385668259217066
2.919175861736183
6.606580446346813
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.4 Το μέλος length

Το `length` είναι **μέλος** του αντικειμένου τύπου πίνακας και παίρνει την τιμή του μεγέθους του πίνακα που δηλώνουμε με `new` ή με δυναμική αρχικοποίηση.

Παράδειγμα 1

```
int pin1[] = {1, 2, 3, 4,5};          // pin1.length = 5
int pin2[] = new int[5];              // pin2.length = 5
```

Παράδειγμα 2

Ο αριθμός 4 (η θέση του τελευταίου στοιχείου του πίνακα `pin1`) στη μέθοδο `showPin1(int pin1[])` θα μπορούσε να αντικατασταθεί από το `pin1.length-1`, οπότε η μέθοδος `showPin1` θα μπορούσε να γραφεί :

```
static void showPin1(int pin1[] ){
// Μέθοδος Εμφάνισης στοιχείων πίνακα 1
    System.out.print("Πίνακας 1 = ");
```

```
        for (int i = 0;i <= pin1.length-1;i++)
        {
         System.out.print(pin1[i] + " " );
        }
        System.out.println();

    }
```

Παράδειγμα 3

Το `n-1` (η θέση του τελευταίου στοιχείου του πίνακα `pin2`) στη μέθοδο `showPin2(int n, double pin2[])` θα μπορούσε να αντικατασταθεί από το `pin2.length-1`, οπότε η μέθοδος `showPin2` θα μπορούσε να γραφεί :

```
static void showPin2( double pin2[] ){
// Μέθοδος Εμφάνισης στοιχείων πίνακα 2
   for (int i = 0;i <= pin2.length -1;i++)
    {
     System.out.println(pin2[i] + " " );
    }
    System.out.println();
}
```

Παράδειγμα 4

Το ίδιο μπορεί να γίνει και για τη μέθόδο `fillPin2(int n, double pin2[])`

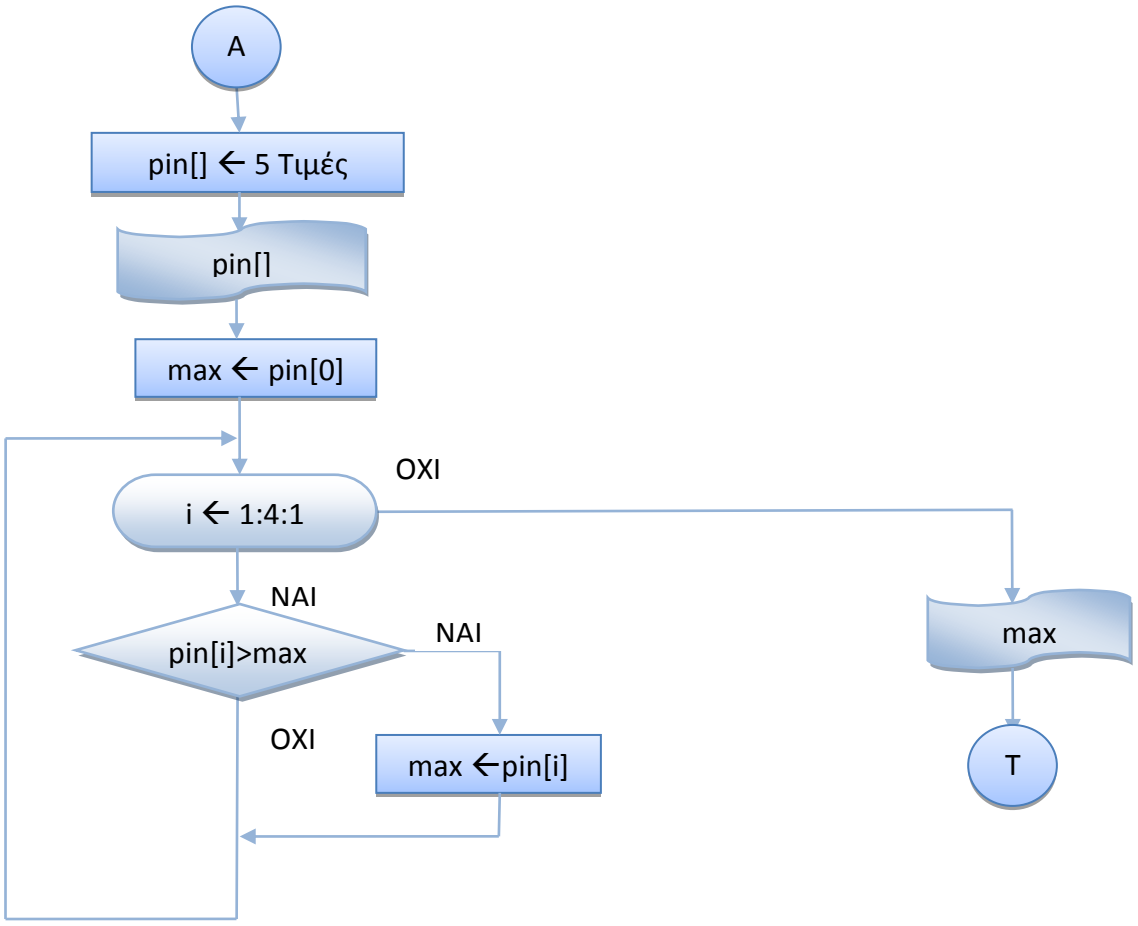
6.5 Εύρεση στοιχείου με τη Μέγιστη ή Ελάχιστη Τιμή σε έναν Πίνακα

Σε πολλά προβλήματα, χρειάζεται να βρούμε το στοιχείο με τη μεγαλύτερη ή μικρότερη τιμή σε έναν πίνακα. Ο πιο συνηθισμένος τρόπος είναι να ελέγξουμε **όλα** τα στοιχεία του πίνακα και να κρατάμε κάθε φορά το μεγαλύτερο, όπως φαίνεται στο επόμενο παράδειγμα :

6.5.1 Εύρεση στοιχείου με τη Μέγιστη Τιμή σε έναν Πίνακα

- Να γραφεί Πρόγραμμα που να δημιουργεί και να γεμίζει έναν πίνακα ακεραίων με δυναμική αρχικοποίηση, να εμφανίζει τα στοιχεία του, να βρίσκει το στοιχείο με τη μεγαλύτερη τιμή και να το εμφανίζει.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin[]` με `5` θέσεις
2. Εμφάνιση στοιχείων πίνακα `pin`
3. Θέτουμε σαν μέγιστο στοιχείο `max` το πρώτο στοιχείο `pin[0]` (αφού δεν υπάρχει άλλο μέχρι στιγμής να συγκριθεί)
4. **Για** τα υπόλοιπα στοιχεία `pin[i]`, για `i = 1` μέχρι `4` :
   Αν το στοιχείο `pin[i]` είναι μεγαλύτερο από το `max`
   Θέτουμε σαν μέγιστο στοιχείο `max` το `pin[i]`
5. Εμφάνιση της τιμής του μεγίστου στοιχείου `max`

Πρόγραμμα

```
public class PinMax {
/* Πρόγραμμα που δημιουργεί και γεμίζει έναν πίνακα ακεραίων με δυναμική
αρχικοποίηση, εμφανίζει τα στοιχεία του, Βρίσκει το στοιχείο με τη
μεγαλύτερη τιμή και το Εμφανίζει
*/
    public static void main(String[] args) {
        int i, max;
```

```
        // Δήλωση - Αρχικοποίηση πίνακα
        int pin[] = {10,2,13,14,5};
        // Εμφάνιση στοιχείων πίνακα
        System.out.print("Πίνακας = ");
        for (i = 0;i <= pin.length-1;i++)
        {
         System.out.print(pin[i] + " " );
        }
        System.out.println();

        // Εύρεση μεγίστου στοιχείου του πίνακα
        max = pin[0];
        for (i = 1;i <= pin.length-1;i++)
        {
         if (pin[i] > max ) {
             max = pin[i];
         }
        }

        // Εμφάνιση μεγίστου στοιχείου του πίνακα
        System.out.println("Μέγιστο στοιχείο = " + max);
       }
}
```

Έξοδος Προγράμματος :

```
run:
Πίνακας = 10 2 13 14 5
Μέγιστο στοιχείο = 14
BUILD SUCCESSFUL (total time: 0 seconds)
```

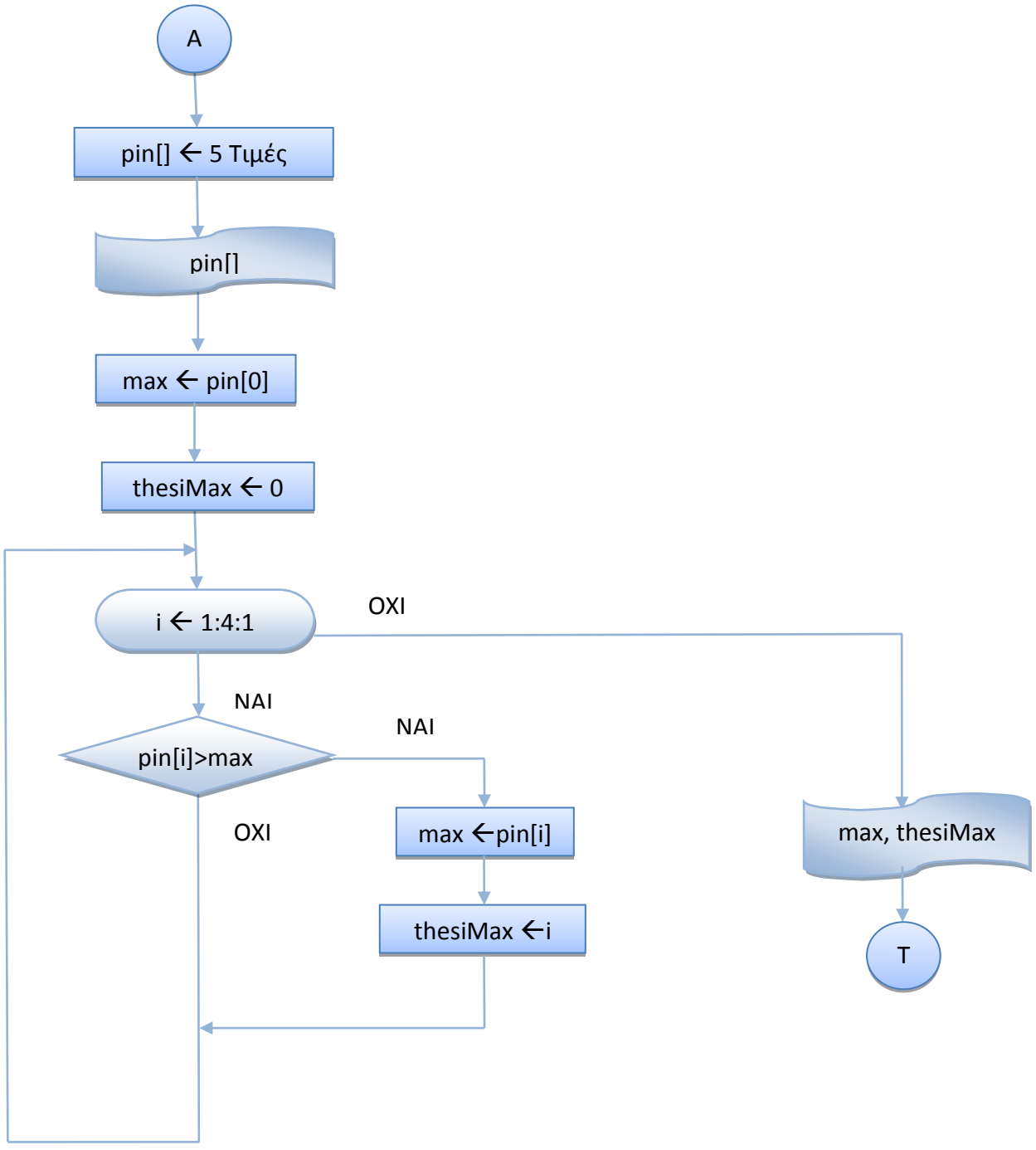
Παρατήρηση :

Πολλές φορές μας ενδιαφέρει να βρούμε όχι μόνο το στοιχείο με τη μεγαλύτερη τιμή, αλλά και τη **θέση** αυτού του στοιχείου στον πίνακα. Έχοντας τη θέση, μπορούμε να βρούμε και το αντίστοιχο στοιχείο. Αυτό φαίνεται στο επόμενο παράδειγμα :

6.5.2 Εύρεση στοιχείου με τη Μέγιστη Τιμή σε έναν Πίνακα και της θέσης του στον πίνακα

- Να γραφεί Πρόγραμμα που να δημιουργεί και να γεμίζει έναν πίνακα ακεραίων με δυναμική αρχικοποίηση, να εμφανίζει τα στοιχεία του, να βρίσκει το στοιχείο με τη μεγαλύτερη τιμή και τη θέση του στον πίνακα και να εμφανίζει το στοιχείο με τη μεγαλύτερη τιμή και **τη θέση του** στον πίνακα

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin[]` με 5 θέσεις
2. Εμφάνιση στοιχείων πίνακα `pin`
3. Θέτουμε σαν μέγιστο στοιχείο `max` το πρώτο στοιχείο `pin[0]` (αφού δεν υπάρχει άλλο μέχρι στιγμής να συγκριθεί)
4. **Θέτουμε σαν θέση μεγίστου στοιχείου `thesiMax` το 0**
5. **Για** τα υπόλοιπα στοιχεία `pin[i]`, για `i = 1` μέχρι 4 :
   Αν το στοιχείο `pin[i]` είναι μεγαλύτερο από το `max`
   Θέτουμε σαν μέγιστο στοιχείο `max` την τιμή του `pin[i]`
   Θέτουμε την τιμή του `i` στο `thesiMax`
6. Εμφάνιση της τιμής του μεγίστου στοιχείου `max` και **της θέσης του `thesiMax`**

Πρόγραμμα

```
public class PinMaxThesi {
/* Πρόγραμμα που δημιουργεί και γεμίζει έναν πίνακα ακεραίων με δυναμική
αρχικοποίηση, εμφανίζει τα στοιχεία του, Βρίσκει το στοιχείο με τη μεγαλύτερη
τιμή και τη θέση του στον πίνακα και Εμφανίζει το στοιχείο μεγαλύτερη τιμή και
τη θέση του στον πίνακα
*/
    public static void main(String[] args) {
        int i, max;

        // Δήλωση - Αρχικοποίηση πίνακα
        int pin[] = {10,2,13,14,5};

        // Εμφάνιση στοιχείων πίνακα
        System.out.print("Πίνακας = ");
        for (i = 0;i <= pin.length-1;i++)
        {
         System.out.print(pin[i] + " " );
        }
        System.out.println();

        // Εύρεση μεγίστου στοιχείου του πίνακα και της θέσης του στον πίνακα
        max = pin[0];
        int thesiMax = 0;
        for (i = 1;i <= pin.length-1;i++)
        {
         if (pin[i] > max ) {
             max = pin[i];
             thesiMax = i;
        }
        }
        // Εμφάνιση μεγίστου στοιχείου και της θέσης του στον πίνακα
        System.out.println("Μέγιστο στοιχείο = " + max + " στη θέση " +
thesiMax );
    }
}
```

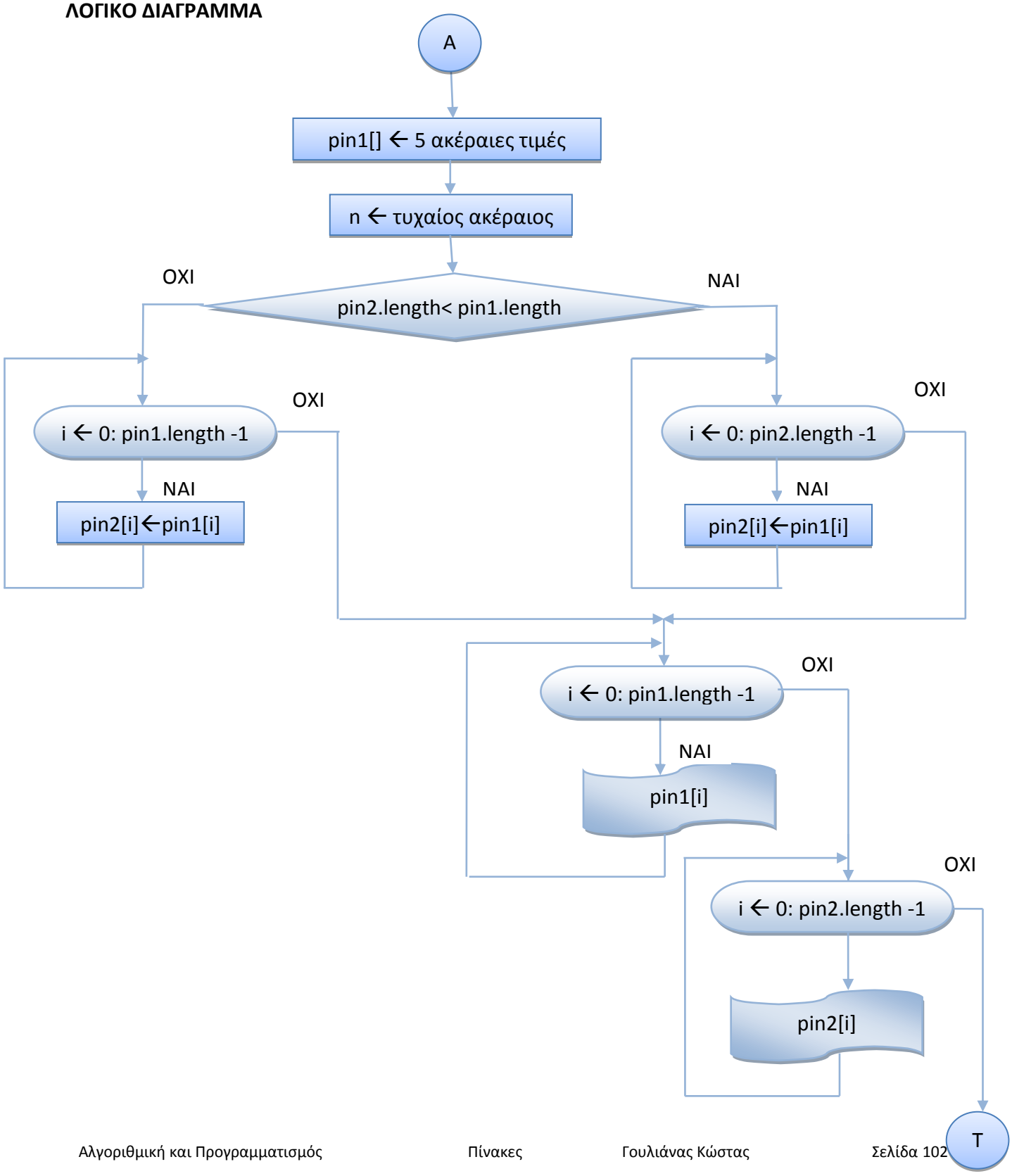
Έξοδος Προγράμματος :

```
run:
Πίνακας = 10 2 13 14 5
Μέγιστο στοιχείο = 14 στη θέση 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.5.3 Αντιγραφή Στοιχείων ενός Πίνακα σε κάποιον άλλο Πίνακα

- Να γραφεί Πρόγραμμα που να δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ για το δεύτερο δημιουργεί με τη μέθοδο `Math.random()` μια τυχαία ακέραια τιμή για το μέγεθος του πίνακα 2 μεταξύ του 1 και 10 και αντιγράφει ΟΣΑ στοιχεία του πίνακα 1 χωράνε στον πίνακα 2 και εμφανίζει τα στοιχεία των 2 πινάκων.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin1[]` με 5 θέσεις τους αριθμούς `1` μέχρι `5`
2. Εκχώρηση τυχαίας ακέραιας τιμής `n` στο μέγεθος του πίνακα `2`
3. Δήλωση πίνακα `pin2[]` με `n` θέσεις
4. Αντιγραφή στον πίνακα `2` των στοιχείων του πίνακα `1`
   Αν το μέγεθος του `pin2 <` μέγεθος του `pin1 (pin2.length < pin1.length)`
   Αντιγραφή **όσων** στοιχείων του πίνακα `pin1` χωράνε στον πίνακα `pin2`
   Διαφορετικά
   Αντιγραφή **ΟΛΩΝ** των στοιχείων του πίνακα `pin1` στον πίνακα `pin2`
5. Εμφάνιση των στοιχείων του πίνακα `pin1`
6. Εμφάνιση των στοιχείων του πίνακα `pin2`

Πρόγραμμα

```
public class PinCopy {
/* Το πρόγραμμα δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση
και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ για το δεύτερο
δημιουργεί με τη μέθοδο Math.random() μια τυχαία ακέραια τιμή για το
μέγεθος του πίνακα 2 μεταξύ του 1 και 10 και αντιγράφει ΟΣΑ στοιχεία του
πίνακα 1 χωράνε στον πίνακα 2 και εμφανίζει τα στοιχεία των 2 πινάκων.
*/
    public static void main(String[] args) {
        int i, j;

        // Δήλωση - Αρχικοποίηση πίνακα 1
        int pin1[]= {1,2,3,4,5};

         // Εκχώρηση τυχαίας ακέραιας τιμής στο μέγεθος του πίνακα 2
        int n = (int) ( 1 + Math.random()*10);
        System.out.println("n = " + n);

        // Δήλωση πίνακα 2
        int pin2[]= new int[n];

        // Αντιγραφή στον πίνακα 2 των στοιχείων του πίνακα 1
        if ( pin2.length < pin1.length)
         for (i = 0;i < pin2.length;i++)
          pin2[i] = pin1[i];
        else
         for (i = 0;i < pin1.length;i++)
          pin2[i] = pin1[i];

        // Εμφάνιση στοιχείων πίνακα 1
        System.out.print("Πίνακας 1 = ");
        for (i = 0;i < pin1.length;i++)
          System.out.print(pin1[i] + " " );
        System.out.println();

        // Εμφάνιση στοιχείων πίνακα 2
        System.out.print("Πίνακας 2 = ");
        for (i = 0;i < pin2.length;i++)
          System.out.print(pin2[i] + " " );
        System.out.println();
        }
}
```

Έξοδος Προγράμματος

```
run:
n = 3
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2 3

n = 5
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2 3 4 5

n = 8
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2 3 4 5 0 0 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Όπως φαίνεται στην έξοδο του προγράμματος, για **`n = 5`** και οι 2 πίνακες έχουν τα **ΙΔΙΑ** περιεχόμενα. Είναι όμως **ΔΙΑΦΟΡΕΤΙΚΟΙ** πίνακες. Η αλλαγή στα στοιχεία ενός πίνακα **ΔΕΝ** επηρεάζει τον άλλο.

6.5.4 Αντιγραφή Στοιχείων ενός Πίνακα σε κάποιον άλλο Πίνακα με κλήση μεθόδων

- Να γραφεί Πρόγραμμα που να δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ για το δεύτερο δημιουργεί με τη μέθοδο `Math.random()` μια τυχαία ακέραια τιμή για το μέγεθος του πίνακα 2 μεταξύ του 1 και 10 και αντιγράφει ΟΣΑ στοιχεία του πίνακα 1 χωράνε στον πίνακα 2 και εμφανίζει τα στοιχεία των 2 πινάκων.

Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin1[]` με 5 θέσεις τους αριθμούς 1 μέχρι 5
2. Εκχώρηση τυχαίας ακέραιας τιμής `n` στο μέγεθος του πίνακα 2
3. Δήλωση πίνακα `pin2[]` με `n` θέσεις
4. Αντιγραφή στον πίνακα 2 των στοιχείων του πίνακα 1 καλώντας τη μέθοδο `copyPin1toPin2()`
5. Εμφάνιση των στοιχείων του πίνακα `pin1` καλώντας τη μέθοδο `showPin()`
6. Εμφάνιση των στοιχείων του πίνακα `pin2` καλώντας τη μέθοδο `showPin()`

Πρόγραμμα

```
public class PinCopyMethods {
/* Το πρόγραμμα δημιουργεί 2 πίνακες, τον πρώτο με δυναμική αρχικοποίηση
και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 5, ενώ για το δεύτερο
δημιουργεί με τη μέθοδο Math.random() μια τυχαία ακέραια τιμή μεταξύ του
1 και 10 για το μέγεθος του πίνακα 2 και αντιγράφει καλώντας τη μέθοδο
copyPin1toPin2() ΟΣΑ στοιχεία του πίνακα 1 χωράνε στον πίνακα 2 και
εμφανίζει τα στοιχεία των 2 πινάκων καλώντας τη μέθοδο showPin() */
```

```
        static void showPin(int pin[] ){
        // Μέθοδος Εμφάνισης στοιχείων πίνακα
            for (int i = 0;i <= pin.length-1;i++)
            {
             System.out.print(pin [i] + " " );
            }
            System.out.println();
         }

        static int[] copyPin1toPin2(int n, int pin1[]){
        // Μέθοδος Αντιγραφής στον πίνακα 2 των στοιχείων του πίνακα 1
            int pin2[] = new int[n];
            if ( pin2.length < pin1.length)
             for (int i = 0;i <= pin2.length -1;i++)
              pin2[i] = pin1[i];
            else
             for (int i = 0;i <= pin1.length -1;i++)
              pin2[i] = pin1[i];
            return pin2;
        }


        public static void main(String[] args) {
            int i, j;

            // Δήλωση - Αρχικοποίηση πίνακα 1
            int pin1[]= {1,2,3,4,5};

            // Εκχώρηση τυχαίας ακέραιας τιμής στο μέγεθος του πίνακα 2
            int n = (int) ( 1 + Math.random()*10);
            System.out.println("n = " + n);

            // Δήλωση πίνακα 2
            int pin2[]= new int[n];

            // Αντιγραφή στον πίνακα 2 των στοιχείων του πίνακα 1
            pin2 = copyPin1toPin2( n,  pin1);

            //  Εμφάνιση στοιχείων πίνακα 1
          System.out.print("Πίνακας 1 = ");
          showPin( pin1);

            // Εμφάνιση στοιχείων πίνακα 2
          System.out.print("Πίνακας 2 = ");
            showPin( pin2);
          }
    }
```

Έξοδος Προγράμματος

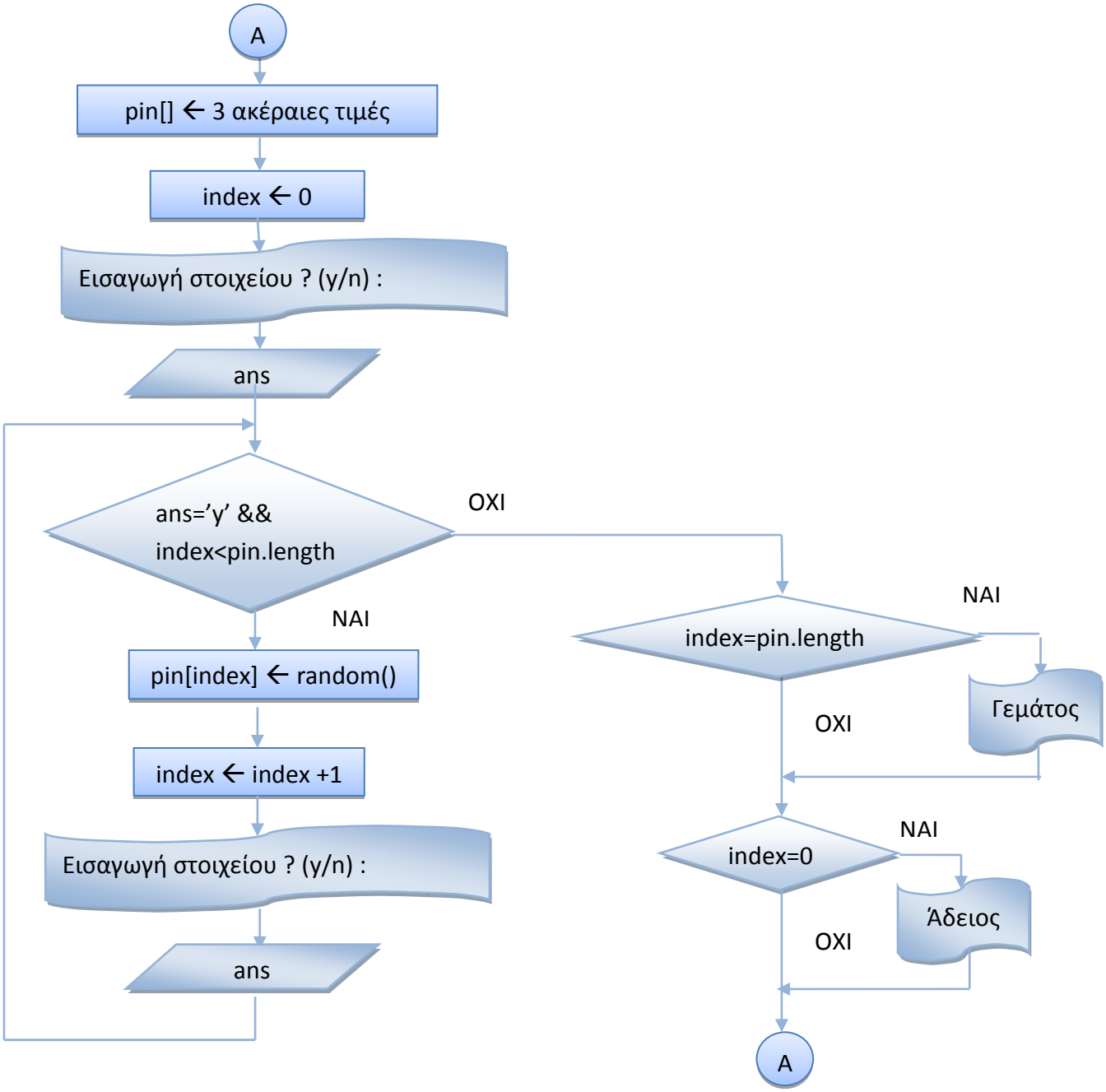
```
run:
n = 2
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2

n = 5
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2 3 4 5

n = 10
Πίνακας 1 = 1 2 3 4 5
Πίνακας 2 = 1 2 3 4 5 0 0 0 0 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.5.5 Δυναμικό Γέμισμα ενός Πίνακα – Έλεγχος αν Είναι Άδειος ή Γεμάτος

Σε αρκετές εφαρμογές θα χρειαστεί να γεμίζουμε δυναμικά τις θέσεις ενός πίνακα, οπότε θα πρέπει να ξέρουμε αν ο πίνακας έχει γεμίσει ή αν δεν έχει μπει κανένα στοιχείο.

- Να γραφεί Πρόγραμμα που να δημιουργεί έναν πίνακα που το μέγεθός του δίνεται με μια τυχαία ακέραια τιμή. Για όσο ο χρήστης επιλέγει να βάλει στοιχεία στον πίνακα δημιουργεί τυχαίες ακέραιες τιμές. Όταν ο πίνακας γεμίσει, εμφανίζει το μήνυμα "`Ο Πίνακας είναι γεμάτος`", ενώ αν ο χρήστης επιλέξει να μη βάλει κανένα στοιχείο στον πίνακα, εμφανίζει το μήνυμα "`Ο Πίνακας είναι άδειος`". Η απάντηση του χρήστη δίνεται με την κλήση της μεθόδου `readchar()`.

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ



Αλγόριθμος

1. Δήλωση πίνακα `pin[3]` με 3 θέσεις
2. Αρχική τιμή για τη θέση του τελευταίου στοιχείων του πίνακα `pin = 0 ( index = 0)`
3. Εμφάνιση μηνύματος "`Εισαγωγή στοιχείου ? (y/n) : `"
4. Διάβασμα απάντησης του χρήστη με την κλήση της μεθόδου `readchar()`.
5. **Για όσο** η απάντηση του είναι χρήστη είναι `'y' (ans == 'y')` και το `index` δεν είναι ίσο με το μέγεθος του πίνακα (`index < pin.length)`)
   Βάζουμε στη θέση `index` του πίνακα `pin` μια τυχαία ακέραια τιμή
   Αυξάνουμε το `index` κατά `1`
   Διάβασμα νέας απάντησης του χρήστη με την κλήση της μεθόδου `readchar()`.
6. Αν `index = pin.length,` Εμφάνιση μηνύματος "`Ο Πίνακας είναι γεμάτος`"
7. **Αν** `index = 0,` Εμφάνιση μηνύματος "`Ο Πίνακας είναι άδειος`"
   Διαφορετικά
   Εμφάνιση των στοιχείων του πίνακα

Πρόγραμμα

```
public class PinEmptyFull {
/* Το πρόγραμμα δημιουργεί έναν πίνακα 3 στοιχείων. Για όσο ο χρήστης επιλέγει
να βάλει στοιχεία στον πίνακα δημιουργεί τυχαίες ακέραιες τιμές. Όταν ο πίνακας
γεμίσει, εμφανίζει το μήνυμα "Ο Πίνακας είναι γεμάτος", ενώ αν ο χρήστης
επιλέξει να μη βάλει κανένα στοιχείο στον πίνακα, εμφανίζει το μήνυμα "Ο
Πίνακας είναι άδειος". Η απάντηση του χρήστη δίνεται με την κλήση της μεθόδου
readchar().
*/
    static char readchar() throws java.io.IOException{
        // Μήνυμα - Απάντηση για Εισαγωγή νέου στοιχείου
        char ans;
        System.out.print("Εισαγωγή στοιχείου ? (y/n) : ");
        do {
          ans = (char)System.in.read();
        }
        while (ans == '\n' | ans == '\r');
        return ans;
    }

    public static void main(String[] args)
                        throws java.io.IOException{

        // Εκχώρηση ακέραιας τιμής στο μέγεθος του πίνακα 2
        int n = 3;
        System.out.println("n =  " + n  );
        char ans = 'y';

        // Δήλωση πίνακα 2
        int pin[] = new int[n];
        int index = 0;

        // Γέμισμα πίνακα με τυχαίες ακέραιες τιμές

        ans = readchar();
        while ((ans == 'y') && ( index < pin.length)) {

           pin[index] = (int) ( Math.random()*10);
           System.out.println("Στοιχείο " + index + " = " + pin[index] );
           index++;
```

```
            // Μήνυμα - Απάντηση για Εισαγωγή νέου στοιχείου
            ans = readchar();
        }

        if ( index == pin.length)
            System.out.println("Ο Πίνακας είναι γεμάτος");

        if (index == 0)
            System.out.println("Ο Πίνακας είναι άδειος");
        else {
        // Εμφάνιση στοιχείων πίνακα 2
        System.out.print("Πίνακας = ");
        for (int i = 0;i < n;i++)
          System.out.print(pin[i] + " " );
        System.out.println();
        }
        }
    }
```

Έξοδος Προγράμματος :

```
run:
Εισαγωγή στοιχείου ? (y/n) : y
Στοιχείο 0 = 6
Εισαγωγή στοιχείου ? (y/n) : y
Στοιχείο 1 = 9
Εισαγωγή στοιχείου ? (y/n) : y
Στοιχείο 2 = 9
Εισαγωγή στοιχείου ? (y/n) : y
Ο Πίνακας είναι γεμάτος
Πίνακας = 6 9 9


n =  3
Εισαγωγή στοιχείου ? (y/n) : n
Ο Πίνακας είναι άδειος

n =  3
Εισαγωγή στοιχείου ? (y/n) : y
Στοιχείο 0 = 1
Εισαγωγή στοιχείου ? (y/n) : y
Στοιχείο 1 = 5
Εισαγωγή στοιχείου ? (y/n) : n
Πίνακας = 1 5 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.6 Αναφορές Πινάκων

Όταν δημιουργείται ένα αντικείμενο τύπου πίνακας με τον τελεστή **new**, στην πράξη το `pin` περιέχει τη διεύθυνση – αναφορά στο αντικείμενο τύπου πίνακας `pin`. Με τις εντολές

```
int pin1[] = {1, 2, 3, 4, 5};
int pin2[] = new int[5];
pin2 = pin1;
```

δε γίνεται αντιγραφή των περιεχομένων του πίνακα **pin1** στο **pin2**. Απλώς οι μεταβλητές αναφορές πινάκων **pin1** και **pin1** δείχνουν στο ίδιο αντικείμενο - πίνακας (περιέχουν την ίδια διεύθυνση), το **pin1**. Αν αλλάξω την τιμή σε ένα στοιχείο του πίνακα που δείχνει το **pin2**, π.χ. με την εντολή :

```
pin2[0] = 1000;
```

αυτό θα επηρεάσει τον πίνακα, στον οποίο δείχνει και το **pin1**, δηλαδή το **pin1[0]** θα περιέχει πλέον την τιμή **1000.**

Για να ανταλλάξω τα περιεχόμενα 2 πινάκων **pin1** και **pin2,** οι οποίοι έχουν δηλωθεί να είναι του ιδίου τύπου(π.χ. **int[5]**) μπορώ να χρησιμοποιήσω τις εντολές :

```
int temp[] = new int[5];
temp = pin1;
pin1 = pin2;
pin2 = temp;
```

με τις οποίες ανταλάσσω τις τιμές των μεταβλητών αναφοράς στους πίνακες **pin1** και **pin2.**

Οι **αναφορές πινάκων**, όταν περνάνε σαν παράμετροι σε μεθόδους, περνάνε **με τιμή**. Δηλαδή, οποιαδήποτε αλλαγή κι αν γίνει μέσα στη μέθοδο στη μεταβλητή αναφοράς δε γίνεται στην πραγματική, αλλά στην τυπική μεταβλητή αναφοράς.

Παρ' όλα αυτά **τα στοιχεία των πινάκων** στα οποίο δείχνουν οι μεταβλητές αναφοράς περνάνε με **αναφορά**. Μπορούμε δηλαδή να αλλάξουμε κάποια ή όλα τα στοιχεία του πίνακα μέσα στη μέθοδο και αυτό να επηρεάσει τον πίνακα.

6.6.1 Αντιγραφή – Ανταλλαγή Αναφορών Πινάκων και Στοιχείων ενός Πίνακα

- Να γραφεί Πρόγραμμα που να δημιουργεί `2` πίνακες ακεραίων `5` θέσεων και τους γεμίζει με τους ακέραιους `1-5` και `6-10`. Ανταλάσσει τις τιμές των αναφορών των πινάκων `pin1` και `pin2` και εμφανίζει τις νέες τιμές των πινάκων που έχουν αλλάξει. Ανταλάσσει τις τιμές των αναφορών των πινάκων `pin1` και `pin2` με την κλήση της μεθόδου `swapPin(pin1, pin2)` και εμφανίζει τις νέες τιμές των πινάκων που **ΔΕΝ έχουν αλλάξει**. Δημιουργεί ένα τυχαίο δείκτη μεταξύ `0` και `3` και καλεί τη μέθοδο `swapElement(pin1, index, index+1)` με την οποία ανταλάσσει τις τιμές των στοιχείων `pin1[index]` και `pin1[index+1]` και εμφανίζει τις νέες τιμές του πίνακα `pin1` που **έχουν αλλάξει**. Αποθηκεύει την αναφορά `pin1` στο `pin2`. Καλεί τη μέθοδο `swapElement(pin2, index, index+1)` με την οποία ανταλάσσει τις τιμές των στοιχείων `pin2[index]` και `pin2[index+1]` και εμφανίζει τις νέες τιμές των πινάκων `pin1` και `pin2` που έχουν αλλάξει και **έχουν τις ΙΔΙΕΣ τιμές**.

Αλγόριθμος

1. Δήλωση πίνακα `pin1`
2. Γέμισμα πίνακα `pin1` με τους αριθμούς `1-5`
3. Δήλωση πίνακα `pin2`
4. Γέμισμα πίνακα `pin2` με τους αριθμούς `6-10`
5. Εμφάνιση στοιχείων πίνακα `pin1`
6. Εμφάνιση στοιχείων πίνακα `pin2`
7. Ανταλλαγή Αναφορών `pin1, pin2`
8. Εμφάνιση στοιχείων πίνακα `pin1` μετά την ανταλλαγή
9. Εμφάνιση στοιχείων πίνακα `pin2` μετά την ανταλλαγή
10. Ανταλλαγή Αναφορών `pin1, pin2` με την κλήση της μεθόδου `swapPin()`
11. Εμφάνιση στοιχείων πίνακα `pin1` μετά την ανταλλαγή
12. Εμφάνιση στοιχείων πίνακα `pin2` μετά την ανταλλαγή
13. Δημιουργία τυχαίου δείκτη `index` στο `0:3`
14. Εμφάνιση δείκτη `index`
15. Κλήση μεθόδου ανταλλαγής των στοιχείων `pin1[index], pin1[index+1]`
16. Εμφάνιση στοιχείων πίνακα `pin1`
17. Εκχώρηση της αναφοράς `pin1` στο `pin2`
18. Εμφάνιση δείκτη `index`
19. Κλήση μεθόδου ανταλλαγής των στοιχείων `pin2[index], pin2[index+1]`
20. Εμφάνιση στοιχείων πίνακα `pin2`
21. Εμφάνιση στοιχείων πίνακα `pin1`

Πρόγραμμα

```
public class SwapPinakes {
/*   Πρόγραμμα το οποίο δημιουργεί 2 πίνακες ακεραίων 5 θέσεων και τους
γεμίζει με τους ακέραιους 1-5 και 6-10. Ανταλάσσει τις τιμές των αναφορών
των πινάκων pin1 και pin2 και εμφανίζει τις νέες τιμές των πινάκων που
έχουν αλλάξει. Ανταλάσσει τις τιμές των αναφορών των πινάκων pin1 και pin2
με την κλήση της μεθόδου swapPin(pin1, pin2) και εμφανίζει τις νέες τιμές
των πινάκων που ΔΕΝ έχουν αλλάξει. Δημιουργεί ένα τυχαίο δείκτη μεταξύ 0
και 4 και καλεί τη μέθοδο swapElement(pin1, index, index+1) με την οποία
ανταλάσσει τις τιμές των στοιχείων pin1[index] και pin1[index+1] και
εμφανίζει τις νέες τιμές του πίνακα pin1 που έχουν αλλάξει. Αποθηκεύει την
αναφορά pin1 στο pin2. Καλεί τη μέθοδο swapElement(pin2, index, index+1)
με την οποία ανταλάσσει τις τιμές των στοιχείων pin2[index] και
pin2[index+1] και εμφανίζει τις νέες τιμές των πινάκων pin1 και pin2 που
έχουν αλλάξει και έχουν τις ΙΔΙΕΣ τιμές.
*/

    static void swapPin(int pin1[], int pin2[]){
    // Μέθοδος ανταλλαγής των τιμών αναφοράς 2 πινάκων ακέραιων
        int[] temp = new int[5];
        temp = pin1;
        pin1 = pin2;
        pin2 = temp;
    }

    static void swapElement(int a[], int i, int j){
    // Μέθοδος ανταλλαγής των τιμών 2 στοιχείων ενός πίνακα ακέραιων
    int temp = a[i];
    a[i] = a[j];
    a[j] = temp;
    }

    static void showPin( int pin[] ){
    // Μέθοδος Εμφάνισης στοιχείων πίνακα
       for (int i = 0;i <= pin.length-1;i++)
        System.out.print(pin[i] + " " );
       System.out.println();
    }

    public static void main(String[] args) {
       int i;

        // Δήλωση πίνακα pin1
        int pin1[] = new int[5];

        // Γέμισμα πίνακα pin1 με τους αριθμούς 1-5
         for (i = 0;i <= pin1.length-1;i++)
            pin1[i] = i + 1;

        // Δήλωση πίνακα pin2
        int pin2[] = new int[5];

        // Γέμισμα πίνακα pin2 με τους αριθμούς 6-10
         for (i = 0;i <= pin2.length-1;i++)
            pin2[i] = i + 6;

        // Εμφάνιση στοιχείων πίνακα pin1
        System.out.print("\nΠίνακας pin1  = ");
        showPin(pin1);


        // Εμφάνιση στοιχείων πίνακα pin2
        System.out.print("Πίνακας pin2  = ");
        showPin(pin2);
```

```
        // Ανταλλαγή Αναφορών pin1, pin2
        int[] temp = new int[5];
        temp = pin1;
        pin1 = pin2;
        pin2 = temp;

        // Εμφάνιση στοιχείων πίνακα pin1 μετά την ανταλλαγή Αναφορών
        System.out.print("\nΠίνακας pin1 μετά την ανταλλαγή Αναφορών = ");
        showPin(pin1);

        // Εμφάνιση στοιχείων πίνακα pin2 μετά την ανταλλαγή Αναφορών
        System.out.print("Πίνακας pin2 μετά την ανταλλαγή Αναφορών = ");
        showPin(pin2);

        // Ανταλλαγή Αναφορών pin1, pin2 με κλήση μεθόδου swapPin
        swapPin( pin1, pin2);

        // Εμφάνιση στοιχείων πίνακα pin1 μετά την ανταλλαγή
        System.out.print("\nΠίνακας pin1 μετά την ανταλλαγή Αναφορών με τη" +
                         " μέθοδο swapPin = ");
        showPin(pin1);

        // Εμφάνιση στοιχείων πίνακα pin2 μετά την ανταλλαγή
        System.out.print("Πίνακας pin2 μετά την ανταλλαγή Αναφορών με τη"
                       + " μέθοδο swapPin = ");
        showPin(pin2);

       // Δημιουργία τυχαίου δείκτη 0:4
       int index = (int) ( Math.random()*4);

        // Εμφάνιση δείκτη
        System.out.print("\nΑνταλλαγή των τιμών των στοιχείων ");
        System.out.println("pin1[" + index + "], pin1[" + (index+1) + "]" );

        // Κλήση μεθόδου ανταλλαγής των στοιχείων pin1[index], pin1[index+1]
        swapElement( pin1, index, index+1);

        // Εμφάνιση στοιχείων πίνακα pin1
        System.out.print("Πίνακας pin1 μετά την ανταλλαγή στοιχείων  = ");
        showPin(pin1);

        //Εκχώρηση της αναφοράς pin1 στο pin2
        pin2 = pin1;

        System.out.println("\nΕκχώρηση της αναφοράς pin1 στο pin2");

        // Εμφάνιση δείκτη
        System.out.print("\nΑνταλλαγή των τιμών των στοιχείων ");
        System.out.println("pin2[" + index + "], pin2[" + (index+1) + "]" );

        // Κλήση μεθόδου ανταλλαγής των στοιχείων pin2[index], pin2[index+1]
        swapElement( pin2, index, index+1);

        // Εμφάνιση στοιχείων πίνακα pin2
        System.out.print("Πίνακας pin2 μετά την ανταλλαγή στοιχείων  = ");
        showPin(pin2);

         // Εμφάνιση στοιχείων πίνακα pin1
        System.out.print("Πίνακας pin1 μετά την ανταλλαγή στοιχείων  = ");
        showPin(pin1);
        }
}
```

Έξοδος Προγράμματος :

```
run:

Πίνακας pin1  = 1 2 3 4 5
Πίνακας pin2  = 6 7 8 9 10

Πίνακας pin1 μετά την ανταλλαγή Αναφορών = 6 7 8 9 10
Πίνακας pin2 μετά την ανταλλαγή Αναφορών = 1 2 3 4 5

Πίνακας pin1 μετά την ανταλλαγή Αναφορών με τη μέθοδο swapPin = 6 7 8 9 10
Πίνακας pin2 μετά την ανταλλαγή Αναφορών με τη μέθοδο swapPin = 1 2 3 4 5

Ανταλλαγή των τιμών των στοιχείων pin1[0], pin1[1]
Πίνακας pin1 μετά την ανταλλαγή των στοιχείων  = 7 6 8 9 10

Εκχώρηση της αναφοράς pin1 στο pin2

Ανταλλαγή των τιμών των στοιχείων pin2[0], pin2[1]
Πίνακας pin2 μετά την ανταλλαγή των στοιχείων  = 6 7 8 9 10
Πίνακας pin1 μετά την ανταλλαγή των στοιχείων  = 6 7 8 9 10
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.7 Πίνακες 2 Διαστάσεων

Πίνακας 2 διαστάσεων είναι μια διάταξη στοιχείων σε **γραμμές** και **στήλες** που το καθένα τους έχει κάποια θέση στην αντίστοιχη γραμμή και στήλη. Καταλαμβάνει μια περιοχή μνήμης με το λογικό όνομα του πίνακα, ενώ κάθε στοιχείο του ξεχωρίζει με **δύο δείκτες**, γραμμή - στήλη.

Παράδειγμα

Πίνακας 2 διαστάσεων, 3 γραμμών και 2 στηλών με περιεχόμενα τους αριθμούς 1-6.

	Στήλη 0	Στήλη 1
Γραμμή 0	1	2
Γραμμή 1	3	4
Γραμμή 2	5	6

Δήλωση Πίνακα 2 διαστάσεων

Η Δήλωση ενός Πίνακα 2 διαστάσεων στη Java γίνεται με τη δήλωση του **τύπου** των στοιχείων που θα αποθηκεύσει, το **όνομά** του, το **μέγεθός** του (αριθμός γραμμών και στηλών σε ξεχωριστά ζεύγη αγκυλών) και τον τελεστή **new**, επειδή οι πίνακες στη Java είναι αντικείμενα.

Παράδειγμα

```
int pin1[][] = new int[3][2];
```

όπου δηλώνουμε τον πίνακα pin1, ο οποίος θα αποθηκεύσει 6 ακέραιους αριθμούς.

Το ίδιο αποτέλεσμα έχουμε με την εντολή

```
int[][] pin1 = new int[3][2];
```

αρκεί να υπάρχουν οι αγκύλες στον τύπο ή στο όνομα του πίνακα.

Ο αριθμός των γραμμών και στηλών του πίνακα μπορεί να είναι οι τιμές δύο μεταβλητών. Οι επόμενες εντολές έχουν το ίδιο αποτέλεσμα :

```
int m = 3, n = 2;
int pin1[][] = new int[m][n];
```

Δημιουργία πίνακα 2 διαστάσεων με δυναμική αρχικοποίηση

Ένας πίνακας 2 διαστάσεων μπορεί να δημιουργηθεί με **δυναμική αρχικοποίηση**, όπου οι τιμές του περικλείονται σε ζεύγη αγκίστρων για την κάθε γραμμή και χωρίζονται με κόμμα. Σ' αυτή την περίπτωση **δε** χρειάζεται ο τελεστής **new**.

Παράδειγμα

```
int pin2[][] = {{1, 2}, {3, 4}, {5, 6}};
```

όπου δηλώνουμε τον πίνακα `pin2`, ο οποίος θα αποθηκεύσει τους ακέραιους αριθμούς από το 1 μέχρι το 6 σε 3 γραμμές από 2 στοιχεία η καθεμιά.

Πρόσβαση σε κάποιο στοιχείο πίνακα 2 διαστάσεων

Η **πρόσβαση** σε κάποιο στοιχείο του πίνακα γίνεται με το όνομά του και τη θέση του (δύο δείκτες γραμμής και στήλης) μέσα σε αγκύλες. Π.χ. το `pin1[2][1]` αναφέρεται στο στοιχείο του `pin1` που βρίσκεται στην τρίτη γραμμή και δεύτερη στήλη, δηλαδή το 6.

Ακανόνιστοι Πίνακες

Στους πίνακας `2` διαστάσεων είναι **υποχρεωτική η δήλωση ΜΟΝΟ της 1^{ης} διάστασης** (αριθμός γραμμών). Μ' αυτό τον τρόπο μπορούμε να έχουμε πίνακες με διαφορετικό αριθμό στοιχείων στην κάθε στήλη (ακανόνιστοι πίνακες).

Παράδειγμα

Με τις παρακάτω εντολές :

```
int pin4[][] = new int[3][];
pin4[0] = new int[3];
pin4[1] = new int[2];
pin4[2] = new int[1];
```

δηλώνουμε τον πίνακα `pin4`, ο οποίος θα αποθηκεύσει ακέραιους αριθμούς σε `3` γραμμές, με `3` στοιχεία στην πρώτη γραμμή, `2` στοιχεία στη δεύτερη γραμμή και `1` στοιχείο στην τρίτη γραμμή.

Στο επόμενο παράδειγμα δηλώνεται ένας ακανόνιστος πίνακας με δυναμική αρχικοποίηση :

Παράδειγμα

```
int pin3[][] = {{1, 2, 3}, {4, 5}, {6}};
```

όπου δηλώνουμε τον πίνακα `pin3`, ο οποίος θα αποθηκεύσει τους ακέραιους αριθμούς από το `1` μέχρι το `6` σε `3` γραμμές, με `3` στοιχεία στην πρώτη γραμμή, `2` στοιχεία στη δεύτερη γραμμή και `1` στοιχείο στην τρίτη γραμμή.

6.7.1 Παράδειγμα Δημιουργίας – Γεμίσματος 2 πινάκων 2 διαστάσεων με random τιμές και Δυναμική Αρχικοποίηση

- Να γραφεί Αλγόριθμος/Πρόγραμμμα το οποίο γεμίζει έναν πίνακα 2 διαστάσεων (3x2), με **δυναμική αρχικοποίηση** και δίνει τις τιμές από το 1 μέχρι το 6, δηλώνει έναν άλλον πίνακα (3x2), θέσεων, τον οποίο γεμίζει με τυχαίες τιμές και εμφανίζει τα περιεχόμενα των 2 πινάκων.

Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα `pin1`
2. Δήλωση - Γέμισμα πίνακα `pin2` με τυχαίες τιμές
   Για κάθε γραμμή `i = 0` μέχρι `(n – 1)`
      Για κάθε στήλη `j = 0` μέχρι `(n – 1)`
         Εκχωρούμε μια τυχαία τιμή μεταξύ 0 και 10 στο στοιχείο `pin2[i][j]`
3. Εμφάνιση στοιχείων πίνακα `pin1`
   Για κάθε γραμμή `i = 0` μέχρι `(n – 1)`
      Για κάθε στήλη `j = 0` μέχρι `(n – 1)`
         Εμφανίζουμε το στοιχείο `pin1[i]`
4. Εμφάνιση στοιχείων πίνακα `pin2`
   Για κάθε γραμμή `i = 0` μέχρι `(n – 1)`
      Για κάθε στήλη `j = 0` μέχρι `(n – 1)`
         Εμφανίζουμε το στοιχείο `pin2[i]`

Πρόγραμμα

```
public class Pin2 {
/*
    Το πρόγραμμα δημιουργεί 2 πίνακες 2 διαστάσεων ( 3x2 ), τον πρώτο με
δυναμική αρχικοποίηση και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 6,
ενώ το δεύτερο τον γεμίζει με τυχαίες τιμές από 0 μέχρι 10 και εμφανίζει
τα στοιχεία των 2 πινάκων.
    */
    public static void main(String[] args) {
        int i, j;

        // Δήλωση - Αρχικοποίηση πίνακα 1
        int pin1[][] = {{1,2},{3,4},{5,6}};

        // Εκχώρηση ακέραιας τιμής στα μεγέθη γραμμών-στηλών του πίνακα 2
        int m = 3, n = 2;

        // Δήλωση πίνακα 2
        int pin2[][] = new int[m][n];

        // Γέμισμα πίνακα 2 με τυχαίες ακέραιες τιμές
        for (i = 0;i < m;i++)
         for (j = 0;j < n;j++)
           pin2[i][j] = (int) ( Math.random()*10);


        // Εμφάνιση στοιχείων πίνακα 1
        System.out.println("Πίνακας 1");
```

```
        for (i = 0;i < 3;i++)
        {
         for (j = 0;j < 2;j++)
           System.out.print(pin1[i][j] + " " );
        System.out.println();
        }
        System.out.println();

        // Εμφάνιση στοιχείων πίνακα 2
        System.out.println("Πίνακας 2");
        for (i = 0;i < m;i++)
        {
         for (j = 0;j < n;j++)
           System.out.print(pin2[i][j] + " " );
        System.out.println();
        }
     }
}
```

Έξοδος Προγράμματος :

```
run:
Πίνακας 1
1 2
3 4
5 6

Πίνακας 2
7 5
3 6
6 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.7.2 Παράδειγμα Δημιουργίας – Γεμίσματος 2 πινάκων 2 διαστάσεων με random τιμές και Δυναμική Αρχικοποίηση με τη χρήση μεθόδων

- Να γραφεί Αλγόριθμος/Πρόγραμμμα το οποίο γεμίζει έναν πίνακα 2 διαστάσεων (3x2), με **δυναμική αρχικοποίηση** και δίνει τις τιμές από το 1 μέχρι το 6, δηλώνει έναν άλλον πίνακα (3x2), θέσεων, τον οποίο γεμίζει με τυχαίες τιμές και εμφανίζει τα περιεχόμενα των 2 πινάκων καλώντας μεθόδους για το γέμισμα του πίνακα των τυχαίων τιμών και την εμφάνιση των 2 πινάκων.

Αλγόριθμος

1. Δήλωση - Αρχικοποίηση πίνακα 1
2. Δήλωση - Γέμισμα πίνακα 2 με τυχαίες τιμές καλώντας τη μέθοδο `fillPin2()`
3. Εμφάνιση στοιχείων πίνακα 1 καλώντας τη μέθοδο `showPin1(pin1)`
4. Εμφάνιση στοιχείων πίνακα 2 καλώντας τη μέθοδο `showPin2(m, n, pin2)`

Πρόγραμμα

```
public class Pin2Methods {
/* Το πρόγραμμα δημιουργεί 2 πίνακες 2 διαστάσεων ( 3x2 ), τον πρώτο με
δυναμική αρχικοποίηση και τον γεμίζει με τις ακέραιες τιμές από 1 μέχρι 6,
ενώ το δεύτερο τον γεμίζει με τυχαίες τιμές από 0 μέχρι 10 καλώντας τη
μέθοδο fillPin2() και εμφανίζει τα στοιχεία των 2 πινάκων καλώντας τις
μεθόδους  showPin1(pin1) και showPin2(m, n, pin2)   */

    static void showPin1(int pin1[][] ){
    // Μέθοδος Εμφάνισης στοιχείων πίνακα 1
        System.out.println("Πίνακας 1 = ");
        for (int i = 0;i < 3;i++) {
         for (int j = 0;j < 2;j++)
          System.out.print(pin1[i][j] + " " );
         System.out.println();
        }
     }

    static void fillPin2(int m, int n, int pin2[][] ){
    // Μέθοδος δημιουργίας στοιχείων πίνακα1, πέρασμα μεταβλητής pin2 με
αναφορά
        for (int i = 0;i < m;i++)
         for (int j = 0;j < n;j++)
           pin2[i][j] = (int) ( Math.random()*10);
        }

    static void showPin2(int m, int n, int pin2[][] ){
    // Μέθοδος Εμφάνισης στοιχείων πίνακα 2
        System.out.println("Πίνακας 2 = ");
        for (int i = 0;i < 3;i++) {
         for (int j = 0;j < 2;j++)
          System.out.print(pin2[i][j] + " " );
         System.out.println();
        }
     }

    public static void main(String[] args) {
        int i, j;

        // Δήλωση - Αρχικοποίηση πίνακα 1
```

```
        int pin1[][] = {{1,2},{3,4},{5,6}};
        // Εκχώρηση ακέραιας τιμής στα μεγέθη γραμμών-στηλών του πίνακα 2
        int m = 3, n = 2;

        // Δήλωση πίνακα 2
        int pin2[][] = new int[m][n];

        // Γέμισμα πίνακα 2 με τυχαίες ακέραιες τιμές
        fillPin2( m,  n,  pin2);


        // Εμφάνιση στοιχείων πίνακα 1
        showPin1(pin1);
        System.out.println();

        // Εμφάνιση στοιχείων πίνακα 2
        showPin2( m,  n,  pin2);
    }
}
```

Έξοδος Προγράμματος :

```
run:
Πίνακας 1 =
1 2
3 4
5 6

Πίνακας 2 =
5 3
6 6
1 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

6.7.3 Το length για τους Πίνακες 2 Διαστάσεων

Το `length` και για τους πίνακες 2 διαστάσεων παίρνει την τιμή του **αριθμού** των **γραμμών** του πίνακα που δηλώνουμε με `new` ή με δυναμική αρχικοποίηση. Στους ακανόνιστους πίνακες, το όνομα του πίνακα με τον αριθμό γραμμής και το `length` δίνει τον αριθμό των **στοιχείων** της γραμμής.

Παράδειγμα 1

```
int pin3[][] = {{1,2},{3,4},{5,6}}; // pin3.length = 3
```

Παράδειγμα 2

```
int pin4[][] = new int[4][];        // pin4.length = 4
pin4[0] = new int[31];              // pin4[0].length = 31
pin4[1] = new int[28];              // pin4[1].length = 28
pin4[2] = new int[31];              // pin4[2].length = 31
pin4[3] = new int[30];              // pin4[3].length = 30
```

6.8 Μέθοδοι Χειρισμού Πινάκων των Κλάσεων System και Arrays

Η Java περιέχει τις κλάσεις `System` και `Arrays`, οι οποίες περιέχουν μεθόδους γεμίσματος, αντιγραφής, σύγκρισης και ταξινόμησης πινάκων.

Η κλάση `Arrays` περιέχει τις παρακάτω μεθόδους :

`Arrays.fill(<όνομα_πίνακα>,<τιμή>)`

η οποία γεμίζει όλες τις θέσεις ενός πίνακα `<όνομα_πίνακα>` με την τιμή `<τιμή>`

`Arrays.equals(<όνομα_πίνακα_1>,<όνομα_πίνακα_2>)`

η οποία ελέγχει αν οι πίνακες `<όνομα_πίνακα_1>` και `<όνομα_πίνακα_2>` έχουν τα ίδια περιεχόμενα

`Arrays.sort(<όνομα_πίνακα>)`

η οποία ταξινομεί τα στοιχεία του πίνακα `<όνομα_πίνακα>`

ενώ η κλάση `System` περιέχει τη μέθοδο :

`System.arraycopy(<όνομα_πίνακα_1>, <αρχή_1>,>,<όνομα_πίνακα_2>,>, <αρχή_2>,<αριθμός_στοιχείων>)`

η οποία αντιγράφει όσα στοιχεία θέλουμε `<αριθμός_στοιχείων>` του πίνακα `<όνομα_πίνακα_1>` αρχίζοντας από τη θέση `<αρχή_1>` στον πίνακα `<όνομα_πίνακα_2>` αρχίζοντας από τη θέση `<αρχή_2>`

Θα πρέπει να γίνει εισαγωγή της βιβλιοθήκης `java.util.Arrays` με την εντολή

```
import java.util.Arrays;
```

6.8.1 Παράδειγμα Χρήσης Μεθόδων Χειρισμού Πινάκων των Κλάσεων System και Arrays

- Να γραφεί Πρόγραμμα που να γεμίζει έναν πίνακα `pin1` 5 θέσεων με το `-999` με την κλήση της μεθόδου `Arrays.fill()`, γεμίζει έναν πίνακα `pin2` 5 θέσεων με τυχαίες ακέραιες τιμές, εμφανίζει τα περιεχόμενα των 2 πινάκων και ελέγχει αν είναι ίδια με την κλήση της μεθόδου `Arrays.equals()`, Ταξινομεί τα στοιχεία του πίνακα 2 με την κλήση της μεθόδου `Arrays.sort()`, Αντιγράφει τα περιεχόμενα του πίνακα 2 στον πίνακα 1 με την κλήση της μεθόδου `System.arraycopy()`, εμφανίζει τα περιεχόμενα των 2 πινάκων και ελέγχει αν είναι ίδια με την κλήση της μεθόδου `Arrays.equals()`.

Αλγόριθμος

1. Γέμισμα ενός πίνακα `pin1` 5 θέσεων με το `-999` με την κλήση της μεθόδου `Arrays.fill()`
2. Γέμισμα ενός πίνακα `pin2` 5 θέσεων με με τυχαίες ακέραιες τιμές
3. Εμφάνιση των στοιχείων του πίνακα `pin1`
4. Εμφάνιση των στοιχείων του πίνακα `pin2`
5. Έλεγχος αν τα περιεχόμενα των 2 πινάκων είναι ίδια με την κλήση της μεθόδου `Arrays.equals()`
6. Ταξινόμηση των στοιχείων του πίνακα 2 με την κλήση της μεθόδου `Arrays.sort()`
7. Εμφάνιση των στοιχείων του πίνακα `pin2`
8. Αντιγραφή των περιεχομένων του πίνακα 2 στον πίνακα 1 με την κλήση της μεθόδου `System.arraycopy()`
9. Εμφάνιση των στοιχείων του πίνακα `pin1`
10. Έλεγχος αν τα περιεχόμενα των 2 πινάκων είναι ίδια με την κλήση της μεθόδου `Arrays.equals()`

Πρόγραμμα

```
import java.util.Arrays;

public class PinArrays {
/* Πρόγραμμα που γεμίζει έναν πίνακα pin1 5 θέσεων με το -999 με την κλήση
της μεθόδου Arrays.fill, γεμίζει έναν πίνακα pin2 5 θέσεων με τυχαίες
ακέραιες τιμές, εμφανίζει τα περιεχόμενα των 2 πινάκων και ελέγχει αν είναι
ίδια με την κλήση της μεθόδου Arrays.equals, Ταξινομεί τα στοιχεία του πίνακα 2
με την κλήση της μεθόδου Arrays.sort, Αντιγράφει τα περιεχόμενα του πίνακα 2
στον πίνακα 1 με την κλήση της μεθόδου System.arraycopy, εμφανίζει τα
περιεχόμενα των 2 πινάκων και ελέγχει αν είναι ίδια με την κλήση της μεθόδου
Arrays.equals
 */
    public static void main(String[] args) {
        int i, j;

        // Δήλωση - Αρχικοποίηση πίνακα 1 - Arrays.fill
        int pin1[]= new int[5];
        Arrays.fill(pin1, -999);

        // Δήλωση - Αρχικοποίηση πίνακα 2 - Math.random
```

```
        int pin2[]= new int[5];
        for (i = 0;i < pin2.length;i++)
          pin2[i] = (int) (Math.random()*10);

        // Εμφάνιση στοιχείων πίνακα 1
        System.out.print("Πίνακας 1 = ");
        for (i = 0;i < pin1.length;i++)
          System.out.print(pin1[i] + " " );
        System.out.println();

        // Εμφάνιση στοιχείων πίνακα 2
        System.out.print("Πίνακας 2 = ");
        for (i = 0;i < pin2.length;i++)
          System.out.print(pin2[i] + " " );
        System.out.println();

        // Έλεγχος αν οι Πίνακες 1,2 έχουν ίδια περιεχόμενα Arrays.equals
        if ( Arrays.equals(pin1, pin2))
          System.out.println("Οι πίνακες 1, 2 έχουν τα ίδια περιεχόμενα" );
        else
          System.out.println("Οι πίνακες 1,2 δεν έχουν τα ίδια "
                  + "περιεχόμενα" );

        // Ταξινόμηση των στοιχείων του πίνακα 2 - Arrays.sort
        Arrays.sort(pin2);

         // Εμφάνιση στοιχείων πίνακα 2
        System.out.print("\nΠίνακας 2 μετά την Ταξινόμηση = ");
        for (i = 0;i < pin2.length;i++)
          System.out.print(pin2[i] + " " );
        System.out.println();

        // Αντιγραφή των στοιχείων του πίνακα 2 στον πίνακα 1
        // System.arraycopy
        System.arraycopy(pin2, 0, pin1, 0, 5);

        // Εμφάνιση στοιχείων πίνακα 1
        System.out.print("Πίνακας 1 μετά την Αντιγραφή = ");
        for (i = 0;i < pin1.length;i++)
          System.out.print(pin1[i] + " " );
        System.out.println();

        // Έλεγχος αν οι Πίνακες 1,2 έχουν  ίδια περιεχόμενα Arrays.equals
        if ( Arrays.equals(pin1, pin2))
          System.out.println("Οι πίνακες 1, 2 έχουν τα ίδια περιεχόμενα" );
        else
          System.out.println("Οι πίνακες 1, 2 δεν έχουν τα ίδια "
                  + "περιεχόμενα" );

      }
}
```

Έξοδος Προγράμματος :

```
run:
Πίνακας 1 = -999 -999 -999 -999 -999
Πίνακας 2 = 0 6 2 7 4
Οι πίνακες 1, 2 δεν έχουν τα ίδια περιεχόμενα

Πίνακας 2 μετά την Ταξινόμηση = 0 2 4 6 7
Πίνακας 1 μετά την Αντιγραφή = 0 2 4 6 7
Οι πίνακες 1, 2 έχουν τα ίδια περιεχόμενα
BUILD SUCCESSFUL (total time: 0 seconds)
```

7 ΚΛΑΣΕΙΣ – ΑΝΤΙΚΕΙΜΕΝΑ

- Να γραφεί Αλγόριθμος/Πρόγραμμμα το οποίο δίνει τιμές στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου για 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή.

Αλγόριθμος

5. Εισαγωγή Στοιχείων 1ου φοιτητή
6. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή
7. Εισαγωγή Στοιχείων 2ου φοιτητή
8. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή

Πρόγραμμα

```
public class Objects1 {
/*    Πρόγραμμμα το οποίο δίνει τιμές στον Αριθμό Μητρώου, τον Κωδικό
Μαθήματος,το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου για 2 φοιτητές και
Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή */
    public static void main(String[] args) {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου
    double Telikos; // Τελικός Βαθμός

    // Εισαγωγή Στοιχείων 1ου φοιτητή
    AM = 14013;
    KM = 6000232;
    Theory = 6.5;
    Erg = 8.2;
    // Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή
    Telikos = Theory * 0.6 + Erg * 0.4;
    System.out.println ( "Τελικός Βαθμός Φοιτητή 1 = " + Telikos );

    // Εισαγωγή Στοιχείων 2ου φοιτητή
    AM = 14014;
    KM = 6000232;
    Theory = 8.6;
    Erg = 8.0;
    // Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή
    Telikos = Theory * 0.6 + Erg * 0.4;
    System.out.println ( "Τελικός Βαθμός Φοιτητή 2 = " + Telikos );
   }
}
```

Έξοδος Προγράμματος :

```
run:
Τελικός Βαθμός Φοιτητή 1 = 7.18
Τελικός Βαθμός Φοιτητή 2 = 8.36
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.1 Δημιουργία Κλάσης η οποία περιέχει ΜΟΝΟ Δεδομένα σαν μέλη

Στο προηγούμενο πρόγραμμα, όλα τα στοιχεία Αριθμός Μητρώου, Κωδικός Μαθήματος, Βαθμός Θεωρίας, Βαθμός Εργαστηρίου και Τελικός Βαθμός αφορούν τους φοιτητές. Θα μπορούσαμε να ορίσουμε μια κατηγορία, κλάση με το όνομα `Foititis`, η οποία να περιέχει τα παραπάνω στοιχεία και να δημιουργούμε στιγμιότυπα – αντικείμενα, τα οποία και έχουν **φυσική υπόσταση**, καταλαμβάνουν μνήμη.

Κλάση είναι ένα **αφηρημένο πρότυπο**, το οποίο περιγράφει – καθορίζει τη μορφή ενός αντικειμένου. Μπορεί εκτός από δεδομένα να περιέχει και μεθόδους που ενεργούν πάνω στα δεδομένα. Η κλάση ορίζει ένα **ΝΕΟ** τύπο δεδομένων.

Τα δεδομένα και οι μέθοδοι πρόσβασης στα δεδομένα λέγονται **μέλη** της κλάσης. Η κλάση αποκτά πραγματική υπόσταση με τη δυναμική δημιουργία του αντικειμένου.

Ορισμός Κλάσης

```
public class <όνομα_κλάσης> {
Δηλώσεις μεταβλητών
...
Μέθοδοι
...
}
```

Παράδειγμα

```
public class Foititis {
      int AM;          // Αριθμός Μητρώου Φοιτητή
      int KM;          // Κωδικός Μαθήματος
      double Theory;   // Βαθμός Θεωρίας
      double Erg;      // Βαθμός Εργαστηρίου
}
```

Η κλάση `Foititis` περιέχει σαν μέλη – δεδομένα ΜΟΝΟ τις μεταβλητές που χρησιμοποιήσαμε στο προηγούμενο πρόγραμμα και αφορούν τον κάθε φοιτητή.

Δημιουργία Αντικειμένου

Για τη δημιουργία ενός αντικειμένου πρέπει να δηλώσουμε ότι το αντικείμενο που θα δημιουργήσουμε είναι τύπου της κλάσης στην οποία ανήκει και να το δημιουργήσουμε με τον τελεστή **new**. Ο γενικός τύπος για τη δημιουργία του είναι :

```
<όνομα_ κλάσης>  <όνομα_αντικειμένου>;
<όνομα_αντικειμένου> = new <όνομα_κλάσης>();
```

Παράδειγμα

```
Foititis f;
f = new Foititis ();
```

Η πρώτη εντολή δηλώνει πως το **f** είναι τύπου `Foititis` (Δήλωση Αναφοράς Αντικειμένου). Με τη δεύτερη εντολή **δημιουργείται** το αντικείμενο - στιγμιότυπο της κλάσης `Foititis`.

Το ίδιο αποτέλεσμα θα έχουμε και με την εντολή :

```
Foititis f = new Foititis ();
```

Το αντικείμενο στο οποίο αναφέρεται το `f` θα περιέχει **τα δικά του αντίγραφα** των μεταβλητών και μεθόδων (**μελών**) της κλάσης `Foititis`.

Πρόσβαση στα μέλη του αντικειμένου

Η πρόσβαση στα μέλη του αντικειμένου γίνεται με τον τελεστή **'.'**. Η γενική της μορφή είναι `<όνομα_αντικειμένου>.<μέλος>`.

Παράδειγμα

Με την εντολή

```
f.AM = 14013;
```

δίνουμε την τιμή `14013` στο πεδίο `AM` (Αριθμός Μητρώου) του αντικειμένου `f`.

7.1.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει ΜΟΝΟ Δεδομένα σαν μέλη

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί `2` αντικείμενα τύπου `Foititis`, δίνει τιμές στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου για τους `2` φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή.

Κλάση `Foititis`

Η Κλάση `Foititis` περιέχει τα πεδία (μέλη) Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου.

```
public class Foititis {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου
}
```

Αλγόριθμος κλάσης Objects2 – main()

1. Δημιουργία Αντικειμένου Φοιτητή 1
2. Δημιουργία Αντικειμένου Φοιτητή 2
3. Εισαγωγή Στοιχείων 1ου φοιτητή
4. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή
5. Εισαγωγή Στοιχείων 2ου φοιτητή
6. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή

Πρόγραμμα

```
public class Objects2 {
/*  Πρόγραμμμα το οποίο δημιουργεί 2 αντικείμενα τύπου Foititis, δίνει τιμές
στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου για τους 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό
του κάθε φοιτητή
*/
    public static void main(String[] args) {
    double Telikos;

    // Δημιουργία Αντικειμένου Φοιτητή 1
    Foititis f1 = new Foititis();

    // Δημιουργία Αντικειμένου Φοιτητή 2
    Foititis f2;
    f2 = new Foititis();

    // Εισαγωγή Στοιχείων Φοιτητή 1
    f1.AM = 14013;
    f1.KM = 6000232;
    f1.Theory = 6.5;
    f1.Erg = 8.2;

    // Υπολογισμός - Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    Telikos = f1.Theory * 0.6 + f1.Erg * 0.4;
    System.out.println ("Τελικός Βαθμός Φοιτητή f1 = " + Telikos );

    // Εισαγωγή Στοιχείων Φοιτητή 2
    f2.AM = 14014;
    f2.KM = 6000232;
    f2.Theory = 8.6;
    f2.Erg = 8.0;

    // Υπολογισμός - Εμφάνιση Τελικού Βαθμού Φοιτητή 2
    Telikos = f2.Theory * 0.6 + f2.Erg * 0.4;
    System.out.println ("Τελικός Βαθμός Φοιτητή f2 = " + Telikos );
   }
}
```

Έξοδος Προγράμματος :

```
run:
Τελικός Βαθμός Φοιτητή f1 = 7.18
Τελικός Βαθμός Φοιτητή f2 = 8.36
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.2 Επεξεργασία Δεδομένων της Κλάσης η οποία περιέχει Δεδομένα σαν μέλη με Μεθόδους στη `main()`

Τα δεδομένα μιας κλάσης, αφού είναι προσβάσιμα στην κλάση που βρίσκεται η `main()` μπορούμε να τα επεξεργαστούμε με μεθόδους, οι οποίες βρίσκονται στην ίδια κλάση που βρίσκεται η `main()`.

7.2.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει Δεδομένα και επεξεργασία τους με Μεθόδους στην κλάση που περιέχει τη `main()`

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί 2 αντικείμενα τύπου `Foititis`, δίνει τιμές στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου για τους 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή **με την κλήση των μεθόδων `findTelikos1()` και `findTelikos2()`** που έχουν οριστεί στην κλάση που βρίσκεται η `main()`. Η μέθοδος `findTelikos1()` θα **υπολογίζει** και θα **εμφανίζει** τον Τελικό Βαθμό του 1ου φοιτητή, ενώ η μέθοδος `findTelikos2()` θα **υπολογίζει** και θα **επιστρέφει** στη `main()` τον Τελικό Βαθμό του 2ου φοιτητή, τον οποίο και θα εμφανίζει.

Κλάση `Foititis`

Η Κλάση `Foititis` περιέχει :

- Τα πεδία (μέλη της κλάσης `Foititis`) Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου.

```
public class Foititis {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου
```

Κλάση `Objects2aMethods`

Η Κλάση `Objects2aMethods` περιέχει :

- **Τη Μέθοδο `main()`.**
- **Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.**
- **Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.**

Αλγόριθμος `main()` κλάσης `Objects2amethods`

1. Δημιουργία Αντικειμένου Φοιτητή 1
2. Εισαγωγή Στοιχείων 1ου φοιτητή
3. **Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με κλήση μεθόδου `findTelikos1()`**
4. Δημιουργία Αντικειμένου Φοιτητή 2
5. Εισαγωγή Στοιχείων 2ου φοιτητή
6. **Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με κλήση μεθόδου `findTelikos2()`**

Πρόγραμμα

```
public class Objects2aMethods {
/*  Πρόγραμμμα το οποίο δημιουργεί 2 αντικείμενα τύπου Foititis, δίνει τιμές
στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου για τους 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό
του κάθε φοιτητή με την κλήση των μεθόδων findTelikos1() και findTelikos2()*/
    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    static void findTelikos1(Foititis f){
          System.out.println( "Τελικός Βαθμός Φοιτητή = "
                + (f.Theory * 0.6 + f.Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    static double findTelikos2(Foititis f){
        return f.Theory * 0.6 + f.Erg * 0.4;
    }
    public static void main(String[] args) {

    // Δημιουργία Αντικειμένου Φοιτητή 1
    Foititis f1 = new Foititis();

    // Εισαγωγή Στοιχείων Φοιτητή 1
    f1.AM = 14013;
    f1.KM = 6000232;
    f1.Theory = 6.5;
    f1.Erg = 8.2;

// Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1 - κλήση μεθόδου findTelikos1
    findTelikos1(f1);

    // Δημιουργία Αντικειμένου Φοιτητή 2
    Foititis f2 = new Foititis();

    // Εισαγωγή Στοιχείων Φοιτητή 2
    f2.AM = 14014;
    f2.KM = 6000232;
    f2.Theory = 8.6;
    f2.Erg = 8.0;

// Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2 – κλήση μεθόδου findTelikos2
    System.out.println ("Τελικός Βαθμός Φοιτητή f2 = " + findTelikos2(f2) );
    }
}
```

Έξοδος Προγράμματος :

```
run:
Τελικός Βαθμός Φοιτητή = 7.18
Τελικός Βαθμός Φοιτητή f2 = 8.36
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.3 Δημιουργία Κλάσης η οποία περιέχει Δεδομένα και Μεθόδους σαν μέλη

Στο προηγούμενο παράδειγμα, οι μέθοδοι **`findTelikos1()`** και **`findTelikos2()`** επεξεργάζονται άμεσα **δεδομένα - μέλη** της κλάσης `Foititis` . Η `Java` δίνει τη δυνατότητα στην κλάση, εκτός από δεδομένα, να μπορεί να περιέχει και μεθόδους, οι οποίες ενεργούν πάνω στα δεδομένα, όπως φαίνεται στο επόμενο παράδειγμα. Ο κώδικας των μεθόδων γράφεται **μετά** τα δεδομένα. Τα δεδομένα είναι **ορατά** μέσα στην κλάση, οπότε **δεν απαιτείται** η χρήση του τελεστή '.', ο οποίος χρησιμοποιείται για την πρόσβαση σε κάποιο μέλος της κλάσης.

7.3.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει Δεδομένα και Μεθόδους σαν μέλη

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί 2 αντικείμενα τύπου `Foititis`, δίνει τιμές στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου για τους 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή **με την κλήση των μεθόδων `findTelikos1()` και `findTelikos2()`** που έχουν οριστεί στην κλάση `Foititis`. Η μέθοδος `findTelikos1()` θα **υπολογίζει** και θα **εμφανίζει** τον Τελικό Βαθμό του 1ου φοιτητή, ενώ η μέθοδος `findTelikos2()` θα **υπολογίζει** και θα **επιστρέφει** τον Τελικό Βαθμό του 2ου φοιτητή, τον οποίο και θα εμφανίζει.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία (μέλη) Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου.
- **Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()` .**
- **Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()` .**

```
public class Foititis {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                                + (Theory * 0.6 + Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }
}
```

Αλγόριθμος κλάσης `Objects3Methods` – **`main()`**

1. Δημιουργία Αντικειμένου Φοιτητή 1
2. Εισαγωγή Στοιχείων 1ου φοιτητή
3. **Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με κλήση μεθόδου `findTelikos1()`**
4. Δημιουργία Αντικειμένου Φοιτητή 2
5. Εισαγωγή Στοιχείων 2ου φοιτητή
6. **Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με κλήση μεθόδου `findTelikos2()`**

Πρόγραμμα

```
public class Objects3Methods {
/*  Πρόγραμμμα το οποίο δημιουργεί 2 αντικείμενα τύπου Foititis, δίνει τιμές
στον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου για τους 2 φοιτητές και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό
του κάθε φοιτητή με την κλήση των μεθόδων findTelikos1() και findTelikos2()
που έχουν οριστεί στην κλάση Foititis.
*/

    public static void main(String[] args) {

    // Δημιουργία Αντικειμένου Φοιτητή 1
    Foititis f1 = new Foititis();

    // Εισαγωγή Στοιχείων Φοιτητή 1
    f1.AM = 14013;
    f1.KM = 6000232;
    f1.Theory = 6.5;
    f1.Erg = 8.2;

     // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1 με κλήση μεθόδου
     // findTelikos1()
    f1.findTelikos1();

    // Δημιουργία Αντικειμένου Φοιτητή 2
    Foititis f2 = new Foititis();

    // Εισαγωγή Στοιχείων Φοιτητή 2
    f2.AM = 14014;
    f2.KM = 6000232;
    f2.Theory = 8.6;
    f2.Erg = 8.0;

     // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2 με κλήση μεθόδου
     // findTelikos2()
    System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = "
                          + f2.findTelikos2() );
}
}
```

Έξοδος Προγράμματος :

```
run:
Τελικός Βαθμός Φοιτητή = 7.18
Τελικός Βαθμός Φοιτητή f2 = 8.36
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.4 Μέθοδοι Κατασκευής - Δομητές

Όταν δημιουργείται ένα αντικείμενο με τον τελεστή `new`, ενεργοποιείται μια εξ ορισμού (`default`) μέθοδος κατασκευής-δομητής του αντικειμένου, η οποία γεμίζει τα πεδία με το μηδενικό στοιχείο του κάθε τύπου δεδομένων που έχει οριστεί σαν πεδίο της κλάσης. Αυτή την έννοια έχει π.χ. το `Foititis()` στην εντολή `Foititis f = new Foititis();`

Μπορούμε να παρέμβουμε στον εξ ορισμού τρόπο δημιουργίας του αντικειμένου και να χρησιμοποιήσουμε μεθόδους κατασκευής-δομητές, οι οποίες δίνουν τιμές (σταθερές ή παραμετρικά) σε όσα πεδία ενός αντικειμένου της κλάσης επιθυμούμε.

Οι μέθοδοι κατασκευής-δομητές έχουν όλες **το όνομα της κλάσης** (υπερφόρτωση) και **δεν έχουν τύπο** (ούτε `void`).

Οι τιμές που περνάνε παραμετρικά στις μεθόδους κατασκευής-δομητές μπορεί να είναι σταθερές ή μεταβλητές.

Παράδειγμα

Μέθοδος κατασκευής-δομητής που γεμίζει παραμετρικά όλα τα πεδία ενός αντικειμένου τύπου `Foititis` με τη δημιουργία του.

```
Foititis(int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
 }
```

Η εντολή για τη δημιουργία του αντικειμένου `f` μπορεί να έχει τώρα την παρακάτω μορφή :

```
Foititis f = new Foititis(14014, 6000232, 8.6, 8.0);

Foititis f = new Foititis(am, km, theory, erg);
```

όπου οι μεταβλητές παράμετροι **`am`**, **`km`**, **`theory`**, **`erg`** έχουν πάρει κάποιες τιμές στην καλούσα μέθοδο.

❖ Όταν χρησιμοποιούμε τα ίδια ονόματα για τις παραμέτρους και τα πεδία της κλάσης χρησιμοποιείται το `this`.

Παράδειγμα

```
Foititis(int AM, int KM, double Theory, double Erg){
        this.AM = AM;
        this.KM = KM;
        this.Theory = Theory;
        this.Erg = Erg;
 }
```

7.4.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει Δεδομένα και Μεθόδους σαν μέλη και Μεθόδους Κατασκευής - Δομητές

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί 3 αντικείμενα τύπου `Foititis`. **Ο πρώτος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες δίνουμε σε μεταβλητές στη `main()`, ο δεύτερος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής δημιουργείται με τον πρώτο δομητή ενώ οι υπόλοιπες τιμές δίνονται απ' ευθείας στα αντίστοιχα πεδία του αντικειμένου**. Και για τους 3 φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή με την κλήση των μεθόδων `findTelikos1()` και `findTelikos2()` που έχουν οριστεί στην κλάση `Foititis`.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία (μέλη) Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου.
- **Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, η οποία δίνει μια συγκεκριμμένη τιμή στον Κωδικό Μαθήματος.**
- **Τη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου.**
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.

```
public class Foititis {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Κατασκευής - Δομητής 1
    Foititis(){
        KM = 6000232;
    }

    // Μέθοδος Κατασκευής - Δομητής 2
    Foititis(int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                + (Theory * 0.6 + Erg * 0.4));
    }
```

```
    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }
}
```

Αλγόριθμος κλάσης ObjectsDomhtes – main()

1. Εισαγωγή Στοιχείων 1ου Φοιτητή σε Μεταβλητές
2. **Δημιουργία Αντικειμένου 1ου Φοιτητή με το Δομητή 2**
3. Εμφάνιση στοιχείων 1ου Φοιτητή
4. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου findTelikos1()
5. **Δημιουργία Αντικειμένου 2ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων με το Δομητή 2**
6. Εμφάνιση στοιχείων 2ου Φοιτητή
7. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου Φοιτητή με την κλήση της μεθόδου findTelikos2()
8. **Δημιουργία Αντικειμένου 3ου Φοιτητή με τον πρώτο δομητή**
9. Εισαγωγή Υπολοίπων Στοιχείων 3ου Φοιτητή με Πρόσβαση στα δεδομένα του αντικειμένου
10. Εμφάνιση στοιχείων 3ου Φοιτητή
11. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 3ου φοιτητή με την κλήση της μεθόδου findTelikos2()

Πρόγραμμα

```
public class ObjectsDomhtes {
/*  Πρόγραμμα το οποίο δημιουργεί 3 αντικείμενα τύπου foititis. Ο πρώτος
φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες
δίνουμε σε μεταβλητές στη main(), ο δεύτερος φοιτητής δημιουργείται με το
δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής
δημιουργείται με τον πρώτο δομητή και οι υπόλοιπες τιμές δίνονται απ' ευθείας
στα αντίστοιχα πεδία του αντικειμένου. Και για τους 3 φοιτητές το πρόγραμμα
εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το
Βαθμό Εργαστηρίου και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε
φοιτητή.
*/
    public static void main(String[] args) {
    double Telikos;

    // Εισαγωγή Στοιχείων Φοιτητή 1 -> Μεταβλητές -> Δομητής 2
    int am = 14013;
    int km = 6000232;
    double theory = 6.5;
    double erg = 8.2;

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δεύτερο δομητή
    Foititis f1 = new Foititis(am, km, theory, erg);

    // Εμφάνιση στοιχείων Φοιτητή 1
    System.out.println ( "\n\nΦοιτητής f1\n\nΑριθμός Μητρώου = " + f1.AM
                + "\nΚωδικός Μαθήματος = " + f1.KM + "\nΒαθμός Θεωρίας = "
                + f1.Theory + "\nΒαθμός Εργαστηρίου = "  + f1.Erg);

    // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    // Κλήση μεθόδου findTelikos1()
    f1.findTelikos1();
```

```
        // Δημιουργία Αντικειμένου Φοιτητή 2 με το δεύτερο δομητή
        // + Εισαγωγή Στοιχείων -> Δομητής
        Foititis f2 = new Foititis(14014, 6000232, 8.6, 8.0);

        // Εμφάνιση στοιχείων Φοιτητή 2
        System.out.println ( "\n\nΦοιτητής f2\n\nΑριθμός Μητρώου = " + f2.AM
                    + "\nΚωδικός Μαθήματος = " + f2.KM + "\nΒαθμός Θεωρίας = "
                    + f2.Theory + "\nΒαθμός Εργαστηρίου = "  + f2.Erg);

        // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2
        // Κλήση μεθόδου findTelikos2()
        System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = "
              + f2.findTelikos2() );

        // Δημιουργία Αντικειμένου Φοιτητή 3 με τον πρώτο δομητή
        Foititis f3 = new Foititis();

        // Εισαγωγή Υπολοίπων Στοιχείων Φοιτητή 3
        // Πρόσβαση στα δεδομένα του αντικειμένου
        f3.AM = 14015;
        f3.Theory = 4.6;
        f3.Erg = 4.0;

        // Εμφάνιση στοιχείων Φοιτητή 3
        System.out.println ( "\n\nΦοιτητής f3\n\nΑριθμός Μητρώου = " + f3.AM
                    + "\nΚωδικός Μαθήματος = " + f3.KM + "\nΒαθμός Θεωρίας = "
                    + f3.Theory + "\nΒαθμός Εργαστηρίου = "  + f3.Erg);

      // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 3
      // Κλήση μεθόδου findTelikos2
       System.out.println ( "Τελικός Βαθμός Φοιτητή f3 = "
              + f3.findTelikos2() );
    }
  }
```

Έξοδος Προγράμματος :

```
run:
Φοιτητής f1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή = 7.18


Φοιτητής f2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.6
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f2 = 8.36


Φοιτητής f3

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 4.0
Τελικός Βαθμός Φοιτητή f3 = 4.359999999999999
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.5 Εμφάνιση των δεδομένων με τη Μέθοδο toString()

Με τη δημιουργία ενός αντικειμένου κληρονομείται και η μέθοδος **toString(),** η οποία μας δίνει τη δυνατότητα να εμφανίζουμε – δίνοντας απλώς το όνομα (αναφορά) του αντικειμένου - όσα από τα δεδομένα του επιθυμούμε και με τον τρόπο που εμείς επιλέγουμε να εμφανισθούν. Η μέθοδος `toString()` είναι τύπου `String` (συμβολοσειρά) και επιστρέφει μια συμβολοσειρά που περιέχει μηνύματα και ονόματα μεταβλητών της κλάσης, τα οποία θα θέλαμε να εμφανιστούν με την εντολή `System.out.println()`.

Παράδειγμα

```
public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός
Μαθήματος = ";
      s +=  KM + "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός
Εργαστηρίου = "
            + Erg;
      return s;
    }
```

Το περιεχόμενο της μεταβλητής `s` που επιστρέφει η μέθοδος θα εμφανιστεί με την εντολή :

```
System.out.println ( f );
```

Το ίδιο αποτέλεσμα θα είχαμε, αν γράφαμε την παρακάτω **toString()** :

```
public String toString() {
  return "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " +
  KM + "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = "
  + Erg;
}
```

7.5.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει Δεδομένα και Μεθόδους σαν μέλη, Μεθόδους Κατασκευής – Δομητές και τη Μέθοδο toString()

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί `3` αντικείμενα τύπου `Foititis`. Ο πρώτος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες δίνουμε σε μεταβλητές στη `main()`, ο δεύτερος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής δημιουργείται με τον πρώτο δομητή και οι υπόλοιπες τιμές δίνονται απ' ευθείας στα αντίστοιχα πεδία του αντικειμένου. Και για τους `3` φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου **με την κλήση της μεθόδου toString()** και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή με την κλήση των μεθόδων `findTelikos1()` και `findTelikos2()` που έχουν οριστεί στην κλάση `Foititis`.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία (μέλη) Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου.
- Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, η οποία δίνει μια συγκεκριμμένη τιμή στον Κωδικό Μαθήματος.
- Τη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου.
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.
- **Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`.**

```
public class Foititis {
    int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;          // Κωδικός Μαθήματος
    double Theory;   // Βαθμός Θεωρίας
    double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Κατασκευής - Δομητής 1
    Foititis(){
        KM = 6000232;
    }

    // Μέθοδος Κατασκευής - Δομητής 2
    Foititis(int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                + (Theory * 0.6 + Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }


    // Μέθοδος Εμφάνισης Στοιχείων Φοιτητή
    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = ";
      s +=  KM + "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = "
            + Erg;
      return s;
    }
}
```

Αλγόριθμος κλάσης Objects5toString – main()

1. Εισαγωγή Στοιχείων 1ου Φοιτητή σε Μεταβλητές
2. Δημιουργία Αντικειμένου 1ου Φοιτητή με το Δομητή 2
3. **Εμφάνιση στοιχείων 1ου Φοιτητή με τη μέθοδο `toString()`**
4. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου `findTelikos1()`
5. Δημιουργία Αντικειμένου 2ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων με το Δομητή 2
6. **Εμφάνιση στοιχείων 2ου Φοιτητή με τη μέθοδο `toString()`**
7. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με κλήση μεθόδου `findTelikos2()`
8. Δημιουργία Αντικειμένου 3ου Φοιτητή με τον πρώτο δομητή
9. Εισαγωγή Υπολοίπων Στοιχείων 3ου Φοιτητή με Πρόσβαση στα δεδομένα του αντικειμένου
10. **Εμφάνιση στοιχείων 3ου Φοιτητή με τη μέθοδο `toString()`**
11. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 3ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`

Πρόγραμμα

```
public class Objects5toString {
/*  Πρόγραμμα το οποίο δημιουργεί 3 αντικείμενα τύπου foititis. Ο πρώτος
φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές,τις οποίες
δίνουμε σε μεταβλητές στη main(), ο δεύτερος φοιτητής δημιουργείται με το
δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής
δημιουργείται με τον πρώτο δομητή και οι υπόλοιπες τιμές δίνονται απ' ευθείας
στα αντίστοιχα πεδία του αντικειμένου. Και για τους 3 φοιτητές το πρόγραμμα
εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το
Βαθμό Εργαστηρίου με την κλήση της μεθόδου toString() και Υπολογίζει και
Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή.
*/
    public static void main(String[] args) {

    // Εισαγωγή Στοιχείων Φοιτητή 1 -> Μεταβλητές -> Δομητής
    int am = 14013;
    int km = 6000232;
    double theory = 6.5;
    double erg = 8.2;

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δεύτερο δομητή
    Foititis f1 = new Foititis (am, km, theory, erg);

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    // Κλήση μεθόδου findTelikos1()
    f1.findTelikos1();

    // Δημιουργία Αντικειμένου Φοιτητή 2 με το δεύτερο δομητή
    // + Εισαγωγή Στοιχείων -> Δομητής
    Foititis f2 = new Foititis(14014, 6000232, 8.6, 8.0);
```

```
        // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 2");
        System.out.println ( f2 );

        // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2
        // Κλήση μεθόδου findTelikos2()
        System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = "
              + f2.findTelikos2() );

        // Δημιουργία Αντικειμένου Φοιτητή 3 με τον πρώτο δομητή
        Foititis f3 = new Foititis();

        // Εισαγωγή Υπολοίπων Στοιχείων Φοιτητή 3
        // Πρόσβαση στα δεδομένα του αντικειμένου
        f3.AM = 14015;
        f3.Theory = 4.6;
        f3.Erg = 4.0;

        // Εμφάνιση Στοιχείων Φοιτητή 3 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 3");
        System.out.println ( f3 );

       // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 3
       // Κλήση μεθόδου findTelikos2
        System.out.println ( "Τελικός Βαθμός Φοιτητή f3 = "
              + f3.findTelikos2() );
    }
  }
```

Έξοδος Προγράμματος :

```
run:
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.6
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f2 = 8.36

Στοιχεία Φοιτητή 3

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 4.0
Τελικός Βαθμός Φοιτητή f3 = 4.359999999999999
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.6 Προσδιοριστικά Πρόσβασης στα Δεδομένα ενός Αντικειμένου

Ένα από τα χαρακτηριστικά της `Java` είναι η δυνατότητα να προστατεύσει τα δεδομένα μιας κλάσης από την αυθαίρετη χρήση κάποιου, ο οποίος θα χρησιμοποιήσει σε μια εφαρμογή τη συγκεκριμμένη κλάση δημιουργώντας τα αντίστοιχα αντικείμενα.

Όταν ένα πεδίο ή μια μέθοδος σε μια κλάση δεν έχει πριν τη δήλωση του τύπου του κανένα προσδιοριστικό πρόσβασης, θεωρείται **δημόσιο** (**`public`**) και έχει ελεύθερη πρόσβαση από οποιαδήποτε άλλη κλάση, χρησιμοποιώντας το **όνομα του αντικειμένου** και το **όνομα του πεδίου ή της μεθόδου**. Μπορεί κάποιος να δώσει οποιαδήποτε τιμή σε μια μεταβλητή (ακόμη και έναν αρνητικό Βαθμό Θεωρίας για παράδειγμα). Αν δεν ελέγξει στο πρόγραμμά του ότι η τιμή που δίνει σε μια μεταβλητή είναι αποδεκτή, θα προκύψουν απρόσμενα αποτελέσματα.

Με το προσδιοριστικό **`private`** πριν τη δήλωση του τύπου του πεδίου ή της μεθόδου, το πεδίο ή η μέθοδος γίνεται ιδιωτικό και προστατεύεται από την αυθαίρετη χρήση του. Η πρόσβαση σ' αυτή την περίπτωση **ΔΕΝ** μπορεί να γίνει με τη χρήση του ονόματός του, όπως στο `public` ή το `default` και απαιτούνται ειδικές **μέθοδοι πρόσβασης**, οι οποίες **ΠΡΕΠΕΙ** να είναι **μέλη** της κλάσης.

Σ' αυτές της μεθόδους μπορούν να μπουν και οι έλεγχοι που απαιτούνται, ώστε οι τιμές που θα δίνονται να είναι **έγκυρες**. Ο σχεδιαστής της κλάσης μ' αυτό τον τρόπο διασφαλίζει τη σωστή χρήση του πεδίου ή της μεθόδου.

Οι μέθοδοι πρόσβασης συνήθως έχουν το όνομα **`get<όνομα_μέλους_κλάσης>()`** για τη χρήση της τιμής ενός μέλους και **`set<όνομα_μέλους_κλάσης>()`** για την ανάθεση της τιμής σε ένα μέλος της κλάσης.

Παράδειγμα

Η μέθοδος

```
double getTheory(){
   return Theory;
}
```

επιστρέφει την τιμή του πεδίου `Theory` (Βαθμός Θεωρίας), ενώ η μέθοδος

```
void setTheory(double th){
 Theory = th;
}
```

αναθέτει την τιμή που έχει η μεταβλητή **`th`** στο πεδίο **`Theory`**.

Σημείωση : Στη μέθοδο **`setTheory(th)`** θα μπορούσε να γίνει πριν την ανάθεση της τιμής της μεταβλητής **`th`** στο πεδίο **`Theory`** και έλεγχος εγκυρότητας της τιμής του Βαθμού Θεωρίας.

7.6.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει private Δεδομένα και τις αντίστοιχες Μεθόδους get και set, Μεθόδους σαν μέλη, Μεθόδους Κατασκευής – Δομητές και τη Μέθοδο toString()

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί 3 αντικείμενα τύπου `Foititis`. Ο πρώτος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες δίνουμε σε μεταβλητές στη `main()`, ο δεύτερος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής δημιουργείται με τον πρώτο δομητή και οι υπόλοιπες τιμές δίνονται **στα αντίστοιχα private πεδία** του αντικειμένου **με τη χρήση των μεθόδων `setAM()`, `setTheory()`, `setErg()`.** Και για τους 3 φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου με την κλήση της μεθόδου `toString()` και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή. **Για το φοιτητή 3 θα εμφανίζει τον Τελικό του Βαθμό με πρόσβαση στα πεδία `Theory` και `Erg` με τις μεθόδους `getTheory()` και `getErg()`.**

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου. **Όλα τα πεδία δηλώνονται `private`, εκτός του πεδίου Κωδικός Μαθήματος.**
- Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, η οποία δίνει μια συγκεκριμμένη τιμή στον Κωδικό Μαθήματος.
- Τη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου.
- **Τις μεθόδους `getAM()`, `getTheory()`, `getErg()`** για τη Χρήση των τιμών των `private` πεδίων.
- **Τις μεθόδους `setAM()`, `setTheory()`, `setErg()`** για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`.

```
public class Foititis{
    private int AM;          // Αριθμός Μητρώου Φοιτητή
    int KM;                  // Κωδικός Μαθήματος
    private double Theory;   // Βαθμός Θεωρίας
    private double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Κατασκευής - Δομητής 1
    Foititis(){
        KM = 6000232;
    }
```

```
    // Μέθοδος Κατασκευής - Δομητής 2
    Foititis (int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδοι get - Χρήση τιμών των private πεδίων
    int getAM(){
     return AM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }

    // Μέθοδοι set - Ανάθεση τιμών στα private πεδία
    void setAM(int am ){
     AM = am;
    }

    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                            + (Theory * 0.6 + Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }

    // Μέθοδος Εμφάνισης Στοιχείων Φοιτητή
    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg;
      return s;
    }
}
```

Αλγόριθμος κλάσης Objects6Private – main()

1. Εισαγωγή Στοιχείων 1ου Φοιτητή σε Μεταβλητές
2. Δημιουργία Αντικειμένου 1ου Φοιτητή με το Δομητή 2
3. Εμφάνιση στοιχείων 1ου Φοιτητή με τη μέθοδο `toString()`
4. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου `findTelikos1()`

5. Δημιουργία Αντικειμένου 2ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων με το Δομητή 2
6. Εμφάνιση στοιχείων 2ου Φοιτητή με τη μέθοδο `toString()`
7. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`
8. Δημιουργία Αντικειμένου 3ου Φοιτητή με τον πρώτο δομητή
9. Εισαγωγή Υπολοίπων Στοιχείων 3ου Φοιτητή με Πρόσβαση στα δεδομένα του αντικειμένου με τις **μεθόδους `setAM()`, `setTheory()`, `setErg()`.**
10. Εμφάνιση στοιχείων 3ου Φοιτητή με τη μέθοδο `toString()`
11. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 3ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`
12. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 3ου φοιτητή με με πρόσβαση στα πεδία `Theory` και `Erg` με τις μεθόδους **`getTheory()`** και **`getErg()`**

Πρόγραμμα

```
public class Objects6Private {
/*  Πρόγραμμα το οποίο δημιουργεί 3 αντικείμενα τύπου foititis. Ο πρώτος
φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες
δίνουμε σε μεταβλητές στη main(), ο δεύτερος φοιτητής δημιουργείται με το
δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής
δημιουργείται με τον πρώτο δομητή και οι υπόλοιπες τιμές δίνονται στα
αντίστοιχα πεδία του αντικειμένου με τη χρήση των μεθόδων setAM(),
setTheory(), setErg(). Και για τους 3 φοιτητές το πρόγραμμα εμφανίζει τον
Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου με την κλήση της μεθόδου toString() και Υπολογίζει και Εμφανίζει
τον Τελικό Βαθμό του κάθε φοιτητή. Για το φοιτητή 3 θα εμφανίζει τον Τελικό
του Βαθμό με πρόσβαση στα πεδία Theory και Erg με τις μεθόδους getTheory()
και getErg().
*/
    public static void main(String[] args) {

    // Εισαγωγή Στοιχείων Φοιτητή 1 -> Μεταβλητές -> Δομητής 2
    int am = 14013;
    int km = 6000232;
    double theory = 6.5;
    double erg = 8.2;

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δεύτερο δομητή
    Foititis f1 = new Foititis(am, km, theory, erg);

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    // Κλήση μεθόδου findTelikos1()
    f1.findTelikos1();

    // Δημιουργία Αντικειμένου Φοιτητή 2 με το δεύτερο δομητή
    // + Εισαγωγή Στοιχείων -> Δομητής
    Foititis f2 = new Foititis(14014, 6000232, 8.6, 8.0);


    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );
```

```
        // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2
        // Κλήση μεθόδου findTelikos2
        System.out.println ("Τελικός Βαθμός Φοιτητή f2 = " + f2.findTelikos2());

        // Δημιουργία Αντικειμένου Φοιτητή 3 με τον πρώτο δομητή
        Foititis f3 = new Foititis();

        // Ανάθεση τιμών στα υπόλοιπα private πεδία του φοιτητή 1 MONO με τις
        // μεθόδους set
        f3.setAM( 14015 );
        f3.setTheory( 4.6 );
        f3.setErg( 4.0 );

        // Εμφάνιση Στοιχείων Φοιτητή 3 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 3");
        System.out.println ( f3 );

       // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 3
       // Κλήση μεθόδου findTelikos2
       System.out.println ( "Τελικός Βαθμός Φοιτητή f3 = " + f3.findTelikos2() );

         // Υπολογισμός - Εμφάνιση Τελικού Βαθμού Φοιτητή χωρίς μέθοδο
        // Πρόσβαση στα δεδομένα του φοιτητή MONO με getTheory(), getErg()
        System.out.print ( "Τελικός Βαθμός Φοιτητή 3 με τις μεθόδους ");
        System.out.println("getTheory(),getErg() = " +
                           (f3.getTheory()*0.6 + f3.getErg()*0.4) );
    }
}
```

Έξοδος Προγράμματος :

```
run:
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f2 με τη Μέθοδο Telikos2 = 8.3

Στοιχεία Φοιτητή 3

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 4.0
Τελικός Βαθμός Φοιτητή f3 = 4.359999999999999
Τελικός Βαθμός Φοιτητή f3 με τις μεθόδους getTheory(),getErg() =
4.359999999999999
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.7 Δεδομένα final

Όταν μια κλάση περιέχει στα δεδομένα της ένα πεδίο, το οποίο δεν πρόκειται να αλλάξει τιμή, είναι δηλαδή **σταθερό**, μπορεί αυτό το πεδίο να δηλωθεί σαν **`final`**. Π.χ. το πεδίο `KM`, το οποίο αφορά τον Κωδικό Μαθήματος, ο οποίος είναι ίδιος για όλους τους φοιτητές και έχει συγκεκριμένη τιμή, μπορεί να δηλωθεί σαν `final` με την εντολή :

```
final int KM = 6000232;
```

Η τιμή του πεδίου στην παραπάνω δήλωση δίνεται με δυναμική αρχικοποίηση στην κλάση `Foititis` και **ΔΕΝ** μπορεί να αλλάξει.

Σ' αυτή την περίπτωση η πλήρης μέθοδος κατασκευής-δομητής δέχεται σαν παραμέτρους τιμές για **όλα τα υπόλοιπα πεδία, εκτός αυτού που έχει δηλωθεί σαν `final`, την τιμή του οποίου παίρνουν όλα τα αντικείμενα που θα δημιουργηθούν**.

7.7.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει private και final Δεδομένα και τις αντίστοιχες Μεθόδους get και set, Μεθόδους σαν μέλη, Μεθόδους Κατασκευής – Δομητές, και τη Μέθοδο toString()

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί `3` αντικείμενα τύπου `Foititis`. Ο πρώτος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας όλες τις τιμές, **εκτός από τον Κωδικό Μαθήματος, ο οποίος δηλώνεται σαν `final`,** τις οποίες δίνουμε σε μεταβλητές στη `main()`, ο δεύτερος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές **εκτός από τον Κωδικό Μαθήματος** και ο τρίτος φοιτητής δημιουργείται με τον πρώτο **κενό δομητή**, ενώ οι υπόλοιπες τιμές δίνονται στα αντίστοιχα `private` πεδία του αντικειμένου με τη χρήση των μεθόδων `setAM()`, `setTheory()`, `setErg()`. Και για τους `3` φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου με την κλήση της μεθόδου `toString()` και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου. Όλα τα πεδία δηλώνονται `private`, εκτός του πεδίου Κωδικός Μαθήματος, **ο οποίος δηλώνεται σαν `final`.**
- **Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, κενός δομητής για τη δημιουργία αντικειμένων με τιμή ΜΟΝΟ στο πεδίο τύπου `final`.**
- **Τη Μέθοδο Κατασκευής `Foititis(int am, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου, εκτός από τον Κωδικό Μαθήματος που δίνεται αυτόματα.**

- Τις μεθόδους `getAM(), getTheory(), getErg()` για τη Χρήση των τιμών των `private` πεδίων.
- Τις μεθόδους `setAM(), setTheory(), setErg()` για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`.

```
public class Foititis {
    private int AM;
    final int KM = 6000232;
    private double Theory, Erg;

    Foititis(){
    }

    Foititis(int am, double th, double erg){
        AM = am;
        Theory = th;
        Erg = erg;
    }

    int getAM(){
     return AM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }

    void setAM(int am ){
     AM = am;
    }

    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }

    void findTelikos1(){
     System.out.println( "Τελικός Βαθμός Φοιτητή = " + (Theory * 0.6 + Erg * 0.4));
    }

    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }

    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg;
      return s;
    }
}
```

Αλγόριθμος κλάσης Objects7Final**- main()**

1. Εισαγωγή Στοιχείων 1ου Φοιτητή (**εκτός του Κωδικού Μαθήματος**) σε Μεταβλητές
2. Δημιουργία Αντικειμένου 1ου Φοιτητή με το Δομητή 2
3. Εμφάνιση στοιχείων 1ου Φοιτητή με τη μέθοδο `toString()`
4. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου `findTelikos1()`
5. Δημιουργία Αντικειμένου 2ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων με το Δομητή 2
6. Εμφάνιση στοιχείων 2ου Φοιτητή με τη μέθοδο `toString()`
7. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`
8. **Δημιουργία Αντικειμένου 3ου Φοιτητή με τον πρώτο κενό δομητή**
9. Εισαγωγή Υπολοίπων Στοιχείων 3ου Φοιτητή με Πρόσβαση στα δεδομένα του αντικειμένου με τις μεθόδους `setAM(), setTheory(), setErg()`.
10. Εμφάνιση στοιχείων 3ου Φοιτητή με τη μέθοδο `toString()`
11. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 3ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`

Πρόγραμμα

```
public class Objects7Final {
/*  Πρόγραμμμα το οποίο δημιουργεί 3 αντικείμενα τύπου Foititis. Ο πρώτος
φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας τις τιμές, τις οποίες
δίνουμε σε μεταβλητές στη main(), ο δεύτερος φοιτητής δημιουργείται με το
δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο τρίτος φοιτητής
δημιουργείται με τον πρώτο κενό δομητή και οι υπόλοιπες τιμές δίνονται στα
αντίστοιχα πεδία του αντικειμένου με τη χρήση των μεθόδων setAM(),
setTheory(), setErg(). Και για τους 3 φοιτητές το πρόγραμμα εμφανίζει τον
Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου με την κλήση της μεθόδου toString() και Υπολογίζει και Εμφανίζει
τον Τελικό Βαθμό του κάθε φοιτητή.
*/
    public static void main(String[] args) {

    // Εισαγωγή Στοιχείων Φοιτητή 1 -> Μεταβλητές -> Δομητής
    int am = 14013;
    double theory = 6.5;
    double erg = 8.2;

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δεύτερο δομητή
    Foititis f1 = new Foititis(am, theory, erg);

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    // Κλήση μεθόδου findTelikos1()
    f1.findTelikos1();

    // Δημιουργία Αντικειμένου Φοιτητή 2 με το δεύτερο δομητή
    // + Εισαγωγή Στοιχείων -> Δομητής
    Foititis f2 = new Foititis(14014, 8.6, 8.0);
```

```
        // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 2");
        System.out.println ( f2 );

        // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2
        // Κλήση μεθόδου findTelikos2()
       System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = " + f2.findTelikos2() );

        // Δημιουργία Αντικειμένου Φοιτητή 3 με τον πρώτο κενό δομητή
        Foititis f3 = new Foititis();

        // Ανάθεση τιμών στα υπόλοιπα private πεδία του φοιτητή 1 MONO με τις
        // μεθόδους set
        f3.setAM( 14015 );
        f3.setTheory( 4.6 );
        f3.setErg( 4.0 );

        // Εμφάνιση Στοιχείων Φοιτητή 3 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 3");
        System.out.println ( f3 );

       // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 3
       // Κλήση μεθόδου findTelikos2()
       System.out.println ( "Τελικός Βαθμός Φοιτητή f3 = " + f3.findTelikos2() );
     }
  }
```

Έξοδος Προγράμματος :

```
run:
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f2 με τη Μέθοδο Telikos2 = 8.3

Στοιχεία Φοιτητή 3

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 4.0
Τελικός Βαθμός Φοιτητή f3 = 4.359999999999999

BUILD SUCCESSFUL (total time: 0 seconds)
```

7.8 Δεδομένα – Μέθοδοι static

Όταν ένα πεδίο ή μια μέθοδος σε μια κλάση αντικειμένων (όχι στην κλάση που περιέχει τη `main()`) έχει δηλωθεί σαν **`static`**, η πρόσβαση σ' αυτό το μέλος μπορεί να γίνει είτε με τον κλασσικό τρόπο **<όνομα_αντικειμένου).<μέλος>** είτε με το **<όνομα_κλάσης>.<μέλος>**, στην οποία ανήκει η μεταβλητή ή η μέθοδος `static`, **χωρίς να είναι απαραίτητη η δημιουργία αντικειμένου**.

Συνήθως οι εφαρμογές περιέχουν μια **ξεχωριστή κλάση** με μεθόδους ή και μεταβλητές `static`, οι οποίες απαιτούνται για την επεξεργασία των δεδομένων των αντικειμένων που θα δημιουργηθούν και οι οποίες δεν απαιτούν τη δημιουργία αντικειμένου, γιατί δεν αφορούν κάποια οντότητα, αλλά συνήθως την επεξεργασία δεδομένων πολλών αντικειμένων.

Οι τιμές στα πεδία `static` που δίνονται με δυναμική αρχικοποίηση είτε με ανάθεση τιμής ανατίθενται στο αντίστοιχο πεδίο **ΟΛΩΝ** των αντικειμένων της κλάσης, ακόμη κι αν αλλάξει η τιμή τους χρησιμοποιώντας το όνομα ενός από τα αντικείμενα (**καθολικές** μεταβλητές).

Οι μέθοδοι `static` μπορούν να προσπελάσουν **ΜΟΝΟ** `static` μεταβλητές.

Μπορεί σε μια κλάση να υπάρχει ένα `static block`, μια ομάδα εντολών, οι οποίες εκτελούνται πριν τη δημιουργία κάθε αντικειμένου.

7.8.1 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει private και static Δεδομένα και τις αντίστοιχες Μεθόδους get και set, Μεθόδους σαν μέλη, Μεθόδους Κατασκευής – Δομητές και τη Μέθοδο toString()

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί `2` αντικείμενα τύπου `Foititis`. Ο πρώτος φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές και ο δεύτερος φοιτητής δημιουργείται με τον πρώτο κενό δομητή, ενώ οι υπόλοιπες τιμές του δίνονται στα αντίστοιχα `private` πεδία του αντικειμένου με τη χρήση των μεθόδων `setAM()`, `setTheory()`, `setErg()`. Και για τους `2` φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, **τον Κωδικό Μαθήματος (ο οποίος δηλώνεται σαν `static`),** το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου με την κλήση της μεθόδου `toString()` και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό των δύο φοιτητών με την κλήση των μεθόδων `findTelikos1()` και `findTelikos2()` αντίστοιχα. Αλλάζει την τιμή του Κωδικού Μαθήματος για το φοιτητή `1` και εμφανίζει τη νέα τιμή του Κωδικού Μαθήματος και για τους 2 φοιτητές. Αλλάζει την τιμή του Κωδικού Μαθήματος χρησιμοποιώντας το όνομα της κλάσης `Foititis` και εμφανίζει τη νέα τιμή του Κωδικού Μαθήματος και για τους 2 φοιτητές. Εμφανίζει την αρχική τιμή της `static` μεταβλητής `betterGrade`, η οποία δηλώνεται σε μια κλάση με μεθόδους `static` (`ClassOf StaticMethods`), καλεί τη `static` μέθοδο `setBetterGrade()`, η οποία βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό και εμφανίζει τη νέα τιμή της μεταβλητής `betterGrade`.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου. Όλα τα πεδία δηλώνονται `private`, **εκτός από τον Κωδικό Μαθήματος, ο οποίος δηλώνεται σαν `static`.**
- Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, κενός δομητής.
- Τη Μέθοδο Κατασκευής `Foititis(int am, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου **εκτός από τον Κωδικό Μαθήματος**.
- Τις μεθόδους `getAM()`, `getTheory()`, `getErg()` για τη Χρήση των τιμών των `private` πεδίων.
- Τις μεθόδους `setAM()`, `setTheory()`, `setErg()` για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`.

```
public class Foititis {

    private int AM;          // Αριθμός Μητρώου Φοιτητή
    static int KM = 6000232; // Κωδικός Μαθήματος
    private double Theory;   // Βαθμός Θεωρίας
    private double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Κατασκευής - Δομητής 1
    Foititis (){
    }

    // Μέθοδος Κατασκευής - Δομητής 2
    Foititis(int am, double th, double erg){
        AM = am;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδοι get - Χρήση τιμών των private πεδίων
    int getAM(){
     return AM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }

    // Μέθοδοι set - Ανάθεση τιμών στα private πεδία
    void setAM(int am ){
     AM = am;
    }
```

```
    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                + (Theory * 0.6 + Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }

    // Μέθοδος Εμφάνισης Στοιχείων Φοιτητή
    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg;
      return s;
    }
}
```

Παρατήρηση

- Η κλάση `Foititis` θα μπορούσε να περιέχει την πλήρη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου, αλλά με τη δημιουργία ενός αντικειμένου με τον πλήρη δομητή θα άλλαζε η τιμή του `static` πεδίου `KM` για όλα τα αντικείμενα.

Κλάση `ClassOfStaticMethods`

- Η κλάση `ClassOfStaticMethods` περιέχει τη `static` καθολική μεταβλητή `betterGrade` και τη `static` μέθοδο `setBetterGrade()`, η οποία βρίσκει και επιστρέφει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό.

```
public class ClassOfStaticMethods {

    static int betterGrade = 0;   // Μεταβλητή static

    /* Μέθοδος static, η οποία βρίσκει και επιστρέφει το φοιτητή με το
    μεγαλύτερο Τελικό Βαθμό
    */

    static int setBetterGrade(Foititis f1, Foititis f2){
        if ( f1.findTelikos2() > f2.findTelikos2())
          betterGrade = 1;
        else
          betterGrade = 2;
        return betterGrade;
    }
}
```

Αλγόριθμος κλάσης `Objects8Static` – **`main()`**

1. Δημιουργία Αντικειμένου 1ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων (**εκτός του Κωδικού Μαθήματος**) με το Δομητή 2
2. Εμφάνιση στοιχείων 1ου Φοιτητή με τη μέθοδο `toString()`
3. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου `findTelikos1()`
4. Εισαγωγή Στοιχείων 2ου Φοιτητή (**εκτός του Κωδικού Μαθήματος**) σε Μεταβλητές
5. Δημιουργία Αντικειμένου 2ου Φοιτητή με το Δομητή 2
6. Εμφάνιση στοιχείων 2ου Φοιτητή με τη μέθοδο `toString()`
7. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με την κλήση της μεθόδου `findTelikos2()`
8. **Αλλαγή της τιμής του Κωδικού Μαθήματος για το φοιτητή 1**
9. Εμφάνιση της νέας τιμής του Κωδικού Μαθήματος και για τους 2 φοιτητές
10. **Αλλαγή της τιμής του Κωδικού Μαθήματος χρησιμοποιώντας το όνομα της κλάσης `Foititis`**
11. Εμφάνιση της νέας τιμής του Κωδικού Μαθήματος και για τους 2 φοιτητές
12. **Εμφάνιση της αρχικής τιμή της static μεταβλητής `betterGrade`, η οποία δηλώνεται σε μια κλάση με μεθόδους `static` (`ClassOfStaticMethods`)**
13. **Κλήση της `static` μεθόδου `setBetterGgrade()`, η οποία βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό.**
14. **Εμφάνιση της νέας τιμής της `static` μεταβλητής `betterGrade`.**

Πρόγραμμα

```
public class Objects8Static {
/*  Πρόγραμμμα το οποίο δημιουργεί 2 αντικείμενα τύπου Foititis. Ο πρώτος
φοιτητής δημιουργείται με το δεύτερο δομητή περνώντας απ' ευθείας τις
τιμές,ενώ ο δεύτερος φοιτητής δημιουργείται με τον πρώτο κενό δομητή και οι
υπόλοιπες τιμές – εκτός του Κωδικού Μαθήματος που δηλώνεται static – δίνονται
στα αντίστοιχα πεδία του αντικειμένου με τη χρήση των μεθόδων setAM(),
setTheory(), setErg(). Και για τους 2 φοιτητές το πρόγραμμα εμφανίζει τον
Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό
Εργαστηρίου με την κλήση της μεθόδου toString() και Υπολογίζει και Εμφανίζει
τον Τελικό Βαθμό του κάθε φοιτητή.
Αλλάζει την τιμή του Κωδικού Μαθήματος για το φοιτητή 1 και εμφανίζει τη νέα
τιμή του Κωδικού Μαθήματος και για τους 2 φοιτητές. Αλλάζει την τιμή του
Κωδικού Μαθήματος χρησιμοποιώντας το όνομα της κλάσης Foititis και εμφανίζει
τη νέα τιμή του Κωδικού Μαθήματος και για τους 2 φοιτητές. Εμφανίζει την
αρχική τιμή της static μεταβλητής betterGrade, η οποία δηλώνεται σε μια κλάση
με μεθόδους static ( Classofstaticmethods ), καλεί τη static μέθοδο
setBetterGrade(), η οποία βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό
και εμφανίζει τη νέα τιμή της static μεταβλητής betterGrade.
*/
    public static void main(String[] args) {
    double Telikos;

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δομητή/Εισαγωγή Στοιχείων
    Foititis f1 = new Foititis(14013, 6.5, 8.2);

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

  // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1-Κλήση μεθόδου findTelikos1
    f1.findTelikos1();
```

```
        // Εισαγωγή Στοιχείων Φοιτητή 2
        int am = 14014;
        double theory = 8.6;
        double erg = 8.0;

        // Δημιουργία Αντικειμένου Φοιτητή 2 με το δομητή
        Foititis f2 = new Foititis();

        // Ανάθεση τιμών στα private πεδία του φοιτητή 2 MONO με μεθόδους set
        f2.setAM( am );
        f2.setTheory( theory );
        f2.setErg( erg );

        // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString
        System.out.println ( "\nΣτοιχεία Φοιτητή 2");
        System.out.println ( f2 );

      // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2-Κλήση μεθόδου findTelikos2
       System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = " + f2.findTelikos2() );

        // Αλλαγή Τιμής static μεταβλητής ΚΜ φοιτητή f1 με όνομα αντικειμένου
        f1.KM = 8000123;

         // Εμφάνιση Κωδικού Μαθήματος Φοιτητή 1
        System.out.println ( "\nΚωδικός Μαθήματος φοιτητή f1 = " + f1.KM);

        // Εμφάνιση Κωδικού Μαθήματος Φοιτητή 2
        System.out.println ( "\nΚωδικός Μαθήματος φοιτητή f2 = " + f2.KM);

        // Αλλαγή Τιμής static μεταβλητής ΚΜ  φοιτητή f1-f2 με όνομα κλάσης
        Foititis.KM = 8000456;

         // Εμφάνιση Κωδικού Μαθήματος Φοιτητή 1
        System.out.println ( "\nΚωδικός Μαθήματος φοιτητή f1 = " + f1.KM);

        // Εμφάνιση Κωδικού Μαθήματος Φοιτητή 2
        System.out.println ( "\nΚωδικός Μαθήματος φοιτητή f2 = " + f2.KM);

        // Εμφάνιση Αρχικής Τιμής static μεταβλητής betterGrade
        System.out.println ( "\nΑρχική Τιμή betterGrade = "
                                        + ClassOfStaticMethods.betterGrade);

        // Κλήση Μεθόδου setBetterGrade()- Εμφάνιση νέας Τιμής betterGrade
        System.out.println ( "\nΚαλύτερο Τελικό Βαθμό έχει ο Φοιτητής #"
                              + ClassOfStaticMethods.setBetterGrade( f1, f2));
    }
}
```

Έξοδος Προγράμματος :

```
run:
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.6
```

```
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f2 = 8.36

Κωδικός Μαθήματος φοιτητή f1 = 8000123

Κωδικός Μαθήματος φοιτητή f2 = 8000123

Κωδικός Μαθήματος φοιτητή f1 = 8000456

Κωδικός Μαθήματος φοιτητή f2 = 8000456

Αρχική Τιμή betterGgrade = 0

Καλύτερο Τελικό Βαθμό έχει ο Φοιτητής #2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

- Σε προβλήματα που πρέπει να δημιουργηθούν αντικείμενα, στα οποία κάποιο πεδίο τους πρέπει να παίρνει σαν τιμές κάποιους διαδοχικούς αριθμούς, (π.χ. τα υποκαταστήματα μιας τράπεζας σ' όλη την Ελλάδα πρέπει να δίνουν τον επόμενο αριθμό σε κάθε καινούριο λογαριασμό ή η Γραμματεία να δίνει διαδοχικούς αριθμούς στον Αριθμό Μητρώου κάθε φοιτητή), συνήθως δηλώνουμε μια `static` μεταβλητή σε κάποια κλάση με `static` μεταβλητές και μεθόδους, την οποία μεταβλητή ενημερώνουμε κάθε φορά που πρέπει να δημιουργήσουμε ένα νέο αντικείμενο, όπως φαίνεται στο Παράδειγμα που ακολουθεί :

7.8.2 Παράδειγμα Δημιουργίας Κλάσης η οποία περιέχει private Δεδομένα και τις αντίστοιχες Μεθόδους get και set, Μεθόδους σαν μέλη, Μεθόδους Κατασκευής – Δομητές, τη Μέθοδο toString() και κλάση με static μεταβλητές και μεθόδους

- Να γραφεί Αλγόριθμος/Πρόγραμμα δημιουργεί 3 αντικείμενα τύπου `Foititis`. Ο πρώτος φοιτητής δημιουργείται με τον πρώτο κενό δομητή και οι υπόλοιπες τιμές δίνονται στα αντίστοιχα `private` πεδία του αντικειμένου με τη χρήση των μεθόδων `setAM()`, `setKM()`, `setTheory()`, `setErg()`. Ο Αριθμός Μητρώου είναι η αρχική τιμή της `static` μεταβλητής `currentAM` στην κλάση `ClassStaticVariableMethod`. Ο δεύτερος φοιτητής δημιουργείται με το δεύτερο πλήρη δομητή περνώντας απ' ευθείας τις υπόλοιπες τιμές, όπου σαν Αριθμός Μητρώου περνάει η νέα τιμή της `static` μεταβλητής `currentAM`, η οποία αυξάνεται κατά `1`, ενώ ο τρίτος φοιτητής δημιουργείται με το δεύτερο πλήρη δομητή περνώντας απ' ευθείας τις υπόλοιπες τιμές, οι οποίες δίνονται σε μεταβλητές στη `main()`, όπου σαν Αριθμός Μητρώου περνάει η νέα τιμή της `static` μεταβλητής `currentAM`, η οποία αυξάνεται κατά `1`. Και για τους `3` φοιτητές το πρόγραμμα εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το Βαθμό Εργαστηρίου με την κλήση της μεθόδου `toString()` και Υπολογίζει και Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή με την κλήση της μεθόδου επιστροφής τιμής `findTelikos2()`. Με την Κλήση της Μεθόδου `findBetterGrade()` βρίσκει το φοιτητή με το μεγαλύτερο βαθμό και εμφανίζει την τελική τιμή της `static` μεταβλητής `currentAM` στην κλάση `ClassStaticVariableMethod`.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας και Βαθμός Εργαστηρίου. Όλα τα πεδία δηλώνονται `private`.
- Τη Μέθοδο Κατασκευής/Δομητή `Foititis()`, κενός δομητής.
- Τη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου.
- Τις μεθόδους `getAM()`,`getKM()`,`getTheory()`, `getErg()` για τη Χρήση των τιμών των `private` πεδίων.
- Τις μεθόδους `setAM()`, `setKM()`, `setTheory()`, `setErg()` για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Υπολογισμού - Εμφάνισης Τελικού Βαθμού `findTelikos1()`.
- Τη Μέθοδο Υπολογισμού - Επιστροφής Τελικού Βαθμού `findTelikos2()`.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`.

```
public class Foititis {

    private int AM;          // Αριθμός Μητρώου Φοιτητή
    private  int KM ;        // Κωδικός Μαθήματος
    private double Theory;   // Βαθμός Θεωρίας
    private double Erg;      // Βαθμός Εργαστηρίου

    // Μέθοδος Κατασκευής - Δομητής 1
    Foititis(){
    }

    // Μέθοδος Κατασκευής - Δομητής 2
    Foititis(int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδοι get - Χρήση τιμών των private πεδίων
    int getAM(){
     return AM;
    }

    int getKM(){
     return KM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }
```

```
    // Μέθοδοι set - Ανάθεση τιμών στα private πεδία
    void setAM(int am ){
     AM = am;
    }

    void setKM(int km ){
     KM = km;
    }

    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }

    // Μέθοδος Υπολογισμού - Εμφάνισης Τελικού Βαθμού
    void findTelikos1(){
        System.out.println( "Τελικός Βαθμός Φοιτητή = "
                + (Theory * 0.6 + Erg * 0.4));
    }

    // Μέθοδος Υπολογισμού - Επιστροφής Τελικού Βαθμού
    double findTelikos2(){
        return Theory * 0.6 + Erg * 0.4;
    }

    // Μέθοδος Εμφάνισης Στοιχείων Φοιτητή
    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg;
      return s;
    }
}
```

Κλάση ClassStaticVariableMethod

- Η κλάση **ClassStaticVariableMethod** περιέχει τη `static` καθολική μεταβλητή **currentAM** και τη `static` μέθοδο `findBetterGrade()`, η οποία βρίσκει και επιστρέφει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό.

```
public class ClassStaticVariableMethod {
  static int currentAM = 145001;   // Καθολική Μεταβλητή static

    /* Μέθοδος static, η οποία βρίσκει και επιστρέφει το φοιτητή με το
    μεγαλύτερο Τελικό Βαθμό
    */
    static int findBetterGrade(Foititis f1, Foititis f2){
        int betterGrade;
        if ( f1.findTelikos2() > f2.findTelikos2())
          betterGrade = 1;
        else
          betterGrade = 2;
        return betterGrade;
    }
}
```

Αλγόριθμος κλάσης ObjectsClassStatic **– main()**

1. Δημιουργία Αντικειμένου 1ου Φοιτητή με τον κενό δομητή
2. Εισαγωγή Στοιχείων 1ου Φοιτητή (**εκτός του Αριθμού Μητρώου**) σε μεταβλητές της main()
3. Ανάθεση των τιμών στα private πεδία του αντικειμένου (**Αριθμός Μητρώου = currentAM)**
4. Εμφάνιση στοιχείων 1ου Φοιτητή με τη μέθοδο toString()
5. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 1ου φοιτητή με την κλήση της μεθόδου findTelikos2()
6. Ενημέρωση Τιμής static μεταβλητής currentAM
7. Δημιουργία Αντικειμένου 2ου Φοιτητή με το Δομητή 2, πέρασμα σταθερών τιμών (**Αριθμός Μητρώου = νέα τιμή currentAM)**
8. Εμφάνιση στοιχείων 2ου Φοιτητή με τη μέθοδο toString()
9. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με την κλήση της μεθόδου findTelikos2()
10. Ενημέρωση Τιμής static μεταβλητής currentAM
11. Εισαγωγή Στοιχείων 3ου Φοιτητή (**εκτός του Αριθμού Μητρώου**) σε μεταβλητές της main()
12. Δημιουργία Αντικειμένου 3ου Φοιτητή με το Δομητή 2, πέρασμα των τιμών παραμετρικά (**Αριθμός Μητρώου = νέα τιμή currentAM)**
13. Εμφάνιση στοιχείων 3ου Φοιτητή με τη μέθοδο toString()
14. Υπολογισμός - Εμφάνιση Τελικού Βαθμού 2ου φοιτητή με την κλήση της μεθόδου findTelikos2()
15. **Κλήση της static μεθόδου findBetterGgrade(), η οποία βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό.**
16. **Εμφάνιση της τελικής τιμής της static μεταβλητής currentAM.**

Πρόγραμμα

```
public class ObjectsClassStatic {
/*  Πρόγραμμμα το οποίο δημιουργεί 3 αντικείμενα τύπου Foititis. Ο πρώτος
φοιτητής δημιουργείται με τον πρώτο κενό δομητή και οι υπόλοιπες τιμές
δίνονται στα αντίστοιχα private πεδία του αντικειμένου με τη χρήση των
μεθόδων setAM(), setKM(), setTheory(), setErg(). Ο Αριθμός Μητρώου είναι η
αρχική τιμή της static μεταβλητής currentAM στην κλάση
ClassStaticVariableMethod. Ο δεύτερος φοιτητής δημιουργείται με το δεύτερο
πλήρη δομητή περνώντας απ' ευθείας τις υπόλοιπες τιμές, όπου σαν Αριθμός
Μητρώου περνάει η νέα τιμή της static μεταβλητής currentAM, η οποία αυξάνεται
κατά 1, ενώ ο τρίτος φοιτητής δημιουργείται με το δεύτερο πλήρη δομητή
περνώντας απ' ευθείας τις υπόλοιπες τιμές, οι οποίες δίνονται σε μεταβλητές
στη main(), όπου σαν Αριθμός Μητρώου περνάει η νέα τιμή της static μεταβλητής
currentAM, η οποία αυξάνεται κατά 1. Και για τους 3 φοιτητές το πρόγραμμα
εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας και το
Βαθμό Εργαστηρίου με την κλήση της μεθόδου toString() και Υπολογίζει και
Εμφανίζει τον Τελικό Βαθμό του κάθε φοιτητή με την κλήση της μεθόδου
επιστροφής τιμής Telikos2(). Με την Κλήση της Μεθόδου findBetterGrade()
βρίσκει το φοιτητή με το μεγαλύτερο βαθμό και εμφανίζει την τελική τιμή
της static μεταβλητής currentAM στην κλάση ClassStaticVariableMethod.
*/
    public static void main(String[] args) {

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δομητή/Εισαγωγή Στοιχείων
    Foititis f1 = new Foititis();
```

```
    // Εισαγωγή Στοιχείων Φοιτητή 1
    int km = 14014;
    double theory = 8.6;
    double erg = 8.0;

     // Ανάθεση τιμών στα private πεδία του φοιτητή 1 MONO με μεθόδους set
    f1.setAM( ClassStaticVariableMethod.currentAM );
    f1.setKM( km );
    f1.setTheory( theory );
    f1.setErg( erg );

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 1
    // Κλήση μεθόδου findTelikos2()
    System.out.println ( "Τελικός Βαθμός Φοιτητή f1 = "
                          + f1.findTelikos2());

    // Ενημέρωση Τιμής static μεταβλητής currentAM
    ClassStaticVariableMethod.currentAM++;

    // Δημιουργία Αντικειμένου Φοιτητή 2 με το δομητή
    Foititis f2 =
          new Foititis(ClassStaticVariableMethod.currentAM, 14014, 6.5, 8.2);

    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );

   // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 2
    // Κλήση μεθόδου findTelikos2()
    System.out.println ( "Τελικός Βαθμός Φοιτητή f2 = "
                          + f2.findTelikos2() );

    // Εισαγωγή Στοιχείων Φοιτητή 3
    km = 14014;
    theory = 5.5;
    erg = 5.2;

    // Δημιουργία Αντικειμένου Φοιτητή 3 με το δομητή
    // Ενημέρωση Τιμής static μεταβλητής currentAM
    Foititis f3 =
        new Foititis(++ClassStaticVariableMethod.currentAM, km, theory, erg);

    // Εμφάνιση Στοιχείων Φοιτητή 3 με κλήση μεθόδου toString
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f3 );

   // Υπολογισμός-Εμφάνιση Τελικού Βαθμού Φοιτητή 3
    // Κλήση μεθόδου findTelikos2()
    System.out.println ( "Τελικός Βαθμός Φοιτητή f3 = "
                          + f3.findTelikos2() );

   // Κλήση Μεθόδου findBetterGrade()
    System.out.println ( "\nΚαλύτερο Τελικό Βαθμό έχει ο Φοιτητής #"
                     + ClassStaticVariableMethod.findBetterGrade( f1, f2));

    // Εμφάνιση νέας Τιμής currentAM
    System.out.println ( "\nΤελική Τιμή static μεταβλητής currentAM = "
                               + ClassStaticVariableMethod.currentAM);
   }
}
```

Έξοδος Προγράμματος :

```
run:

Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 145001
Κωδικός Μαθήματος = 14014
Βαθμός Θεωρίας = 8.6
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός Φοιτητή f1 = 8.36

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 145002
Κωδικός Μαθήματος = 14014
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός Φοιτητή f2 = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 145003
Κωδικός Μαθήματος = 14014
Βαθμός Θεωρίας = 5.5
Βαθμός Εργαστηρίου = 5.2
Τελικός Βαθμός Φοιτητή f3 = 5.38

Καλύτερο Τελικό Βαθμό έχει ο Φοιτητής #1

Τελική Τιμή static μεταβλητής currentAM = 145003
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.9 Αναφορές Αντικειμένων

Όταν δημιουργείται ένα αντικείμενο **f** της κλάσης **Foititis** με τον τελεστή **new**, στην πράξη το `f` περιέχει τη διεύθυνση – αναφορά στο αντικείμενο. Σε αντίθεση με τους κλασικούς τύπους δεδομένων όπου χρησιμοποιώντας το όνομα της αντίστοιχης μεταβλητής απλού τύπου έχουμε πρόσβαση στο περιεχόμενά της, δε συμβαίνει το ίδιο με τα αντικείμενα. Για παράδειγμα οι εντολές

```
int a = 5;
int b = a;
```

έχουν σαν αποτέλεσμα να αντιγραφεί το περιεχόμενο της μεταβλητής `a` στη μεταβλητή `b`, οπότε και οι 2 μεταβλητές `a` και `b` να έχουν το ίδιο περιεχόμενο, αλλά περιέχουν διαφορετικές διευθύνσεις στη μνήμη. Αν π.χ. χρησιμοποιήσω την εντολή

```
b = 7;
```

το περιεχόμενο της μεταβλητής `b` είναι πλέον διαφορετικό απ' αυτό της `a`.

Με τα αντικείμενα τα πράγματα είναι διαφορετικά. Με τις εντολές

```
Foititis f1 = new Foititis (14013, 6.5, 8.2);
Foititis f2 = new Foititis (14014, 8.5, 3.2);
f2 = f1;
```

δε γίνεται αντιγραφή των περιεχομένων του αντικειμένου **f1** στο **f2**. Απλώς οι μεταβλητές αναφορές αντικειμένων **f1** και **f2** δείχνουν στο ίδιο αντικείμενο (περιέχουν την ίδια διεύθυνση), το **f1**. Αν αλλάξω την τιμή σε ένα πεδίο του αντικειμένου που δείχνει το `f2`, π.χ. με την εντολή :

```
f2.Theory = 4.0;
```

αυτό θα επηρεάσει το αντικείμενο, στο οποίο δείχνει και το **f1**, δηλαδή το `f1.Theory` θα περιέχει πλέον την τιμή `4.0`.

Για να ανταλλάξω τα περιεχόμενα 2 αντικειμένων **f1** και **f2** μπορώ να χρησιμοποιήσω τις εντολές :

```
Foititis temp = f1;
f1 = f2;
f2 = temp;
```

με τις οποίες ανταλλάσω τις τιμές των μεταβλητών αναφοράς στα αντικείμενα **f1** και **f2.**

Οι **αναφορές αντικειμένων**, όταν περνάνε σαν παράμετροι σε μεθόδους, περνάνε **με τιμή**. Δηλαδή, οποιαδήποτε αλλαγή κι αν γίνει μέσα στη μέθοδο στη μεταβλητή αναφοράς δε γίνεται στην πραγματική, αλλά στην τυπική μεταβλητή αναφοράς.

Παρ' όλα αυτά **το αντικείμενο** στο οποίο δείχνει η μεταβλητή αναφοράς περνάει με **αναφορά**. Μπορούμε δηλαδή να αλλάξουμε κάποια ή όλα τα πεδία του αντικειμένου μέσα στη μέθοδο και αυτό να επηρεάσει το αντικείμενο.

7.9.1 Παράδειγμα Αλλαγών σε Τιμές Μεταβλητών Αναφοράς Αντικειμένων και σε Παραμέτρους - Μεταβλητές Αναφοράς Αντικειμένων σε Μεθόδους

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί 2 αντικείμενα τύπου `Foititis`. **Η κλάση `Foititis` περιέχει και το πεδίο `telikosBathmos` για τον Τελικό Βαθμό**. Και οι δύο φοιτητές δημιουργούνται με το δομητή περνώντας απ' ευθείας τις τιμές. Και για τους 2 φοιτητές το πρόγραμμα **υπολογίζει τον Τελικό Βαθμό του κάθε φοιτητή** και εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας το Βαθμό Εργαστηρίου **και τον Τελικό Βαθμό με την κλήση της μεθόδου `toString()`. Ανταλλάσει** τις τιμές των αναφορών αντικειμένων `f1` και `f2` και εμφανίζει τις νέες τιμές των πεδίων τους. **Ανταλλάσει** τις τιμές των αναφορών αντικειμένων `f1` και `f2` με την κλήση της μεθόδου `swap(f1, f2)` και εμφανίζει τις τιμές των πεδίων τους που **ΔΕΝ έχουν αλλάξει**. **Ανταλλάσει** τις τιμές του πεδίου Αριθμός Μητρώου των αντικειμένων `f1` και `f2` με την κλήση της μεθόδου `swapAM(f1, f2)` και εμφανίζει τις τιμές των πεδίων τους που **έχουν αλλάξει**.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας Βαθμός Εργαστηρίου και **Τελικός Βαθμός.** Όλα τα πεδία δηλώνονται `private`.
- Τη Μέθοδο Κατασκευής `Foititis(int am, int km, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου, εκτός του **Τελικού Βαθμού**.
- Τις μεθόδους `getAM()`, `getKM()`, `getTheory()`, `getErg()`, `getTelikos()` για τη Χρήση των τιμών των `private` πεδίων.
- Τις μεθόδους `setAM()`, `setKM()`, `setTheory()`, `setErg()`, `setTelikos()` για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`, όπου **έχει προστεθεί** και η εμφάνιση του **Τελικού Βαθμού**.

```
public class Foititis {
    private int AM;                     // Αριθμός Μητρώου Φοιτητή
    private int KM;                     // Κωδικός Μαθήματος
    private double Theory;              // Βαθμός Θεωρίας
    private double Erg;                 // Βαθμός Εργαστηρίου
    private double telikosBathmos;      // Τελικός Βαθμός

    // Μέθοδος Κατασκευής - Δομητής
        Foititis(int am, int km, double th, double erg){
        AM = am;
        KM = km;
        Theory = th;
        Erg = erg;
    }

    // Μέθοδοι get - Χρήση τιμών των private πεδίων
    int getAM(){
     return AM;
    }
```

```
    int getKM(){
     return KM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }

    double getTelikosBathmos(){
     return telikosBathmos;
    }

    // Μέθοδοι set - Ανάθεση τιμών στα private πεδία
    void setAM(int am ){
     AM = am;
    }

    void setKM(int km ){
     AM = km;
    }

    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }

    void setTelikosBathmos(){
     telikosBathmos = Theory * 0.6 + Erg * 0.4;
    }

    // Μέθοδος Εμφάνισης Στοιχείων Φοιτητή + Τελικός Βαθμός
    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg
              + "\nΤελικός Βαθμός = " + telikosBathmos;
      return s;
    }
}
```

Κλάση ClassOfStaticMethods

- Η κλάση ClassOfStaticMethods περιέχει τη static μέθοδο swapF(), η οποία ανταλλάσσει τις τιμές των μεταβλητών αναφοράς 2 αντικειμένων και τη static μέθοδο swapAM(), η οποία ανταλλάσει τις τιμές των Αριθμών Μητρώου 2 αντικειμένων.

```
public class ClassOfStaticMethods {

    // Μέθοδος ανταλλαγής των τιμών αναφοράς 2 αντικειμένων
    static void swapF(Foititis f1, Foititis f2){
    Foititis temp = f1;
    f1 = f2;
    f2 = temp;
    }
```

```
    // Μέθοδος ανταλλαγής των τιμών του Αριθμού Μητρώου 2 αντικειμένων
    static void swapAM(Foititis f1, Foititis f2){
    int temp = f1.getAM();
    f1.setAM(f2.getAM());
    f2.setAM(temp);
    }
}
```

Αλγόριθμος κλάσης Objects9Swap – main()

1. Δημιουργία Αντικειμένου 1ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων (**εκτός του Τελικού Βαθμού**) με το Δομητή
2. Υπολογισμός **Τελικού Βαθμού** 1ου φοιτητή με την κλήση της μεθόδου **setTelikosBathmos()**
3. Εμφάνιση ΟΛΩΝ των στοιχείων του 1ου Φοιτητή με τη μέθοδο toString()
4. Δημιουργία Αντικειμένου 2ου Φοιτητή με ταυτόχρονη Εισαγωγή Στοιχείων (**εκτός του Τελικού Βαθμού**) με το Δομητή
5. Υπολογισμός **Τελικού Βαθμού** 2ου φοιτητή με την κλήση της μεθόδου **setTelikosBathmos()**
6. Εμφάνιση ΟΛΩΝ των στοιχείων του 2ου Φοιτητή με τη μέθοδο toString()
7. **Ανταλλαγή** των τιμών των **μεταβλητών αναφοράς** 2 αντικειμένων
8. Εμφάνιση ΟΛΩΝ των στοιχείων του 1ου Φοιτητή με τη μέθοδο toString()
9. Εμφάνιση ΟΛΩΝ των στοιχείων του 2ου Φοιτητή με τη μέθοδο toString()
10. **Ανταλλαγή** των τιμών των **μεταβλητών αναφοράς** 2 αντικειμένων **με κλήση της static μεθόδου swapF()**
11. Εμφάνιση ΟΛΩΝ των στοιχείων του 1ου Φοιτητή με τη μέθοδο toString()
12. Εμφάνιση ΟΛΩΝ των στοιχείων του 2ου Φοιτητή με τη μέθοδο toString()
13. Αλλαγή της τιμής του Αριθμού Μητρώου **Ανταλλαγή** των τιμών των **μεταβλητών αναφοράς** 2 αντικειμένων **με κλήση της static μεθόδου swapAM()**
14. Εμφάνιση ΟΛΩΝ των στοιχείων του 1ου Φοιτητή με τη μέθοδο toString()
15. Εμφάνιση ΟΛΩΝ των στοιχείων του 2ου Φοιτητή με τη μέθοδο toString()

Πρόγραμμα

```
public class Objects9Swap {
/*  Πρόγραμμμα το οποίο δημιουργεί 2 αντικείμενα τύπου Foititis. Και οι δύο
φοιτητές δημιουργούνται με το δεύτερο δομητή περνώντας απ' ευθείας τις τιμές.
Και για τους 2 φοιτητές το πρόγραμμα Υπολογίζει τον Τελικό Βαθμό του κάθε
φοιτητή και εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό
Θεωρίας το Βαθμό Εργαστηρίου και τον Τελικό Βαθμό με την κλήση της μεθόδου
toString(). Ανταλάσσει τις τιμές των αναφορών αντικειμένων f1 και f2 και
εμφανίζει τις νέες τιμές των πεδίων τους. Ανταλάσσει τις τιμές των αναφορών
αντικειμένων f1 και f2 με την κλήση της μεθόδου swapF(f1, f2) και εμφανίζει
τις τιμές των πεδίων τους που ΔΕΝ έχουν αλλάξει. Ανταλάσσει τις τιμές του
πεδίου Αριθμός Μητρώου των αντικειμένων f1 και f2 με την κλήση της μεθόδου
swapAM(f1, f2) και εμφανίζει τις τιμές των πεδίων τους που έχουν αλλάξει.
*/
   public static void main(String[] args) {

    // Δημιουργία Αντικειμένου Φοιτητή 1 με το δομητή/ Εισαγωγή Στοιχείων
    Foititis f1 = new Foititis(14013, 6000232, 6.5, 8.2);

     // Υπολογισμός Τελικού Βαθμού Φοιτητή 1 με μεθόδους get-set
    f1.setTelikosBathmos();
```

```
    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Δημιουργία Αντικειμένου Φοιτητή 2 με το δομητή/ Εισαγωγή Στοιχείων
    Foititis f2 = new Foititis(14014, 6000232, 8.5, 8.3);

    // Υπολογισμός Τελικού Βαθμού Φοιτητή 2 με μεθόδους get-set
    F2.setTelikosBathmos();

    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );

    //Ανταλλαγή αναφορών αντικειμένων
    Foititis temp = f1;
    f1 = f2;
    f2 = temp;

    // Εμφάνιση Στοιχείων Φοιτητών μετά την ανταλλαγή των f1, f2
    System.out.println ( "\nΣτοιχεία Φοιτητών μετά την ανταλλαγή f1, f2");

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );

    // Κλήση μεθόδου swapF() - πέρασμα αναφορών αντικειμένων ΜΕ ΤΙΜΗ,
    // καμιά αλλαγή
    ClassOfStaticMethods.swapF(f1, f2);

    // Εμφάνιση Στοιχείων Φοιτητών μετά την κλήση της μεθόδου swapF(f1, f2)
    System.out.println ( "\nΣτοιχεία Φοιτητών μετά την κλήση της μεθόδου "
            + "swapF(f1, f2 )");

    // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );

    // Κλήση μεθόδου swapAM() - πέρασμα αναφορών αντικειμένων, αλλαγή ΑΜ
    ClassOfStaticMethods.swapAM(f1, f2);

    // Εμφάνιση Στοιχείων Φοιτητών μετά την κλήση  μεθόδου swapAM(f1,f2)
    System.out.println ( "\nΣτοιχεία Φοιτητών μετά την κλήση της μεθόδου "
            + "swapAM(f1, f2)");

     // Εμφάνιση Στοιχείων Φοιτητή 1 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 1");
    System.out.println ( f1 );

    // Εμφάνιση Στοιχείων Φοιτητή 2 με κλήση μεθόδου toString()
    System.out.println ( "\nΣτοιχεία Φοιτητή 2");
    System.out.println ( f2 );
  }
}
```

Έξοδος Προγράμματος :

```
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός = 7.18

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.3
Τελικός Βαθμός = 8.42

Στοιχεία Φοιτητών μετά την ανταλλαγή f1, f2
Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.3
Τελικός Βαθμός = 8.42

Στοιχεία Φοιτητή 2
Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός = 7.18

Στοιχεία Φοιτητών μετά την κλήση της μεθόδου swapF(f1, f2)
Στοιχεία Φοιτητή 1
Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.3
Τελικός Βαθμός = 8.42

Στοιχεία Φοιτητή 2
Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός = 7.18

Στοιχεία Φοιτητών μετά την κλήση της μεθόδου swapAM(f1, f2)

Στοιχεία Φοιτητή 1
Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.5
Βαθμός Εργαστηρίου = 8.3
Τελικός Βαθμός = 8.42

Στοιχεία Φοιτητή 2
Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.5
Βαθμός Εργαστηρίου = 8.2
Τελικός Βαθμός = 7.1
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.10 Πίνακες Αντικειμένων

Στις περισσότερες εφαρμογές απαιτείται η δημιουργία ενός πίνακα αντικειμένων, όταν τα αντικείμενα που θα χρειαστούμε είναι πολλά. Για το σκοπό αυτό θα χρειαστεί να δημιουργήσουμε **πρώτα** το **αντικείμενο** τύπου πίνακας και **μετά** να δημιουργήσουμε τα **αντικείμενα** του πίνακα.

Παράδειγμα

```
Foititis f[] = new Foititis[3]; // Δημιουργία Πίνακα 3 φοιτητών

f[0] = new Foititis(14014, 8.5, 8.3); // Δημιουργία Αντικειμένου Φοιτητή 0
                                      με το δομητή/Εισαγωγή Στοιχείων
```

7.10.1 Παράδειγμα Δημιουργίας Πίνακα Αντικειμένων – Υπολογισμός Τελικού/Μεγαλύτερου Τελικού Βαθμού – Ανταλλαγή Περιεχομένων 2 Αντικειμένων στον Πίνακα με χρήση μεθόδων

- Να γραφεί Αλγόριθμος/Πρόγραμμα το οποίο δημιουργεί έναν **πίνακα 3 αντικειμένων** τύπου `Foititis`. Η κλάση `Foititis` περιέχει και το πεδίο `telikosBathmos` για τον Τελικό Βαθμό. Και οι τρεις φοιτητές δημιουργούνται με το δομητή περνώντας απ' ευθείας τις τιμές, εκτός του πεδίου `KM` (το οποίο δηλώνεται σαν `static` και παίρνει τιμή με δυναμική αρχικοποίηση) και του Τελικού Βαθμού. Το πρόγραμμα υπολογίζει τον Τελικό Βαθμό ΟΛΩΝ των φοιτητών με την κλήση της μεθόδου `findTelikosBathmosAll(f)`, εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας, το Βαθμό Εργαστηρίου και τον Τελικό Βαθμό με την κλήση της μεθόδου `displayStudents(f)` και βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό με την κλήση της μεθόδου `findBetterGrade(f)` και τον εμφανίζει. Κατόπιν δημιουργεί έναν τυχαίο ακέραιο αριθμό `index` μεταξύ `0` και `f.length-2` και καλεί τη μέθοδο `swapF(f, index, index+1)` με την οποία ανταλάσσει τα περιεχόμενα των `f[index]`, `f[index+1]` και εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας, το Βαθμό Εργαστηρίου και τον Τελικό Βαθμό με την κλήση της μεθόδου `displayStudents(f)`. Όλες οι παραπάνω **μέθοδοι** θα γραφούν σε μια **ξεχωριστή κλάση με `static` μεθόδους** και όχι στην κλάση που βρίσκεται η `main()`.

Κλάση `Foititis`

Η Κλάση **`Foititis`** περιέχει :

- Τα πεδία Αριθμός Μητρώου Φοιτητή, Κωδικός Μαθήματος, Βαθμός Θεωρίας Βαθμός Εργαστηρίου και **Τελικός Βαθμός.** Όλα τα πεδία δηλώνονται `private`, εκτός του πεδίου `KM` το οποίο δηλώνεται σαν `static` και παίρνει τιμή με δυναμική αρχικοποίηση.

- Τη Μέθοδο Κατασκευής `Foititis(int am, double th, double erg)`, η οποία δίνει παραμετρικά τις τιμές σε όλα τα πεδία του αντικειμένου, εκτός του **Κωδικού Μαθήματος** και του **Τελικού Βαθμού**.
- Τις μεθόδους `getAM()`, `getKM()`, `getTheory()`, `getErg()`, `getTelikosBathmos()` για τη Χρήση των τιμών των `private` πεδίων.
- Τις μεθόδους `setAM()`, `setKM()`, `setTheory()`, `setErg()`, `setTelikosBathmos()` για την Ανάθεση τιμών στα `private` πεδία.
- Τη Μέθοδο Εμφάνισης των Στοιχείων του κάθε Φοιτητή `toString()`, όπου **έχει προστεθεί** και η εμφάνιση του **Τελικού Βαθμού**.

```
public class Foititis {
    private int AM;
    static int KM = 6000232;
    private double Theory;
    private double Erg;
    private double telikosBathmos;

    Foititis() {
     }

    Foititis(int am, double th, double erg){
        AM = am;
        Theory = th;
        Erg = erg;
    }

    int getAM(){
     return AM;
    }

    int getKM(){
     return KM;
    }

    double getTheory(){
     return Theory;
    }

    double getErg(){
     return Erg;
    }

    double getTelikosBathmos(){
     return telikosBathmos;
    }

    void setAM(int am ){
     AM = am;
    }

    void setKM(int km ){
     KM = km;
    }

    void setTheory(double th){
     Theory = th;
    }

    void setErg(double erg){
     Erg = erg;
    }
```

```
    void setTelikosBathmos(){
        telikosBathmos = Theory * 0.6 + Erg * 0.4;
    }

    public String toString() {
      String s = "\nΑριθμός Μητρώου = " + AM + "\nΚωδικός Μαθήματος = " + KM;
      s += "\nΒαθμός Θεωρίας = " + Theory + "\nΒαθμός Εργαστηρίου = " + Erg;
      s += "\nΤελικός Βαθμός = " + telikosBathmos;
      return s;
    }
}
```

Κλάση ClassOfMethods

Περιέχει τις παρακάτω `static` μεθόδους :

`displayStudents(Foititis f[]) -` Μέθοδος Εμφάνισης των Στοιχείων όλων των Φοιτητών

`findTelikosBathmosAll(Foititis f[]) -` Μέθοδος Υπολογισμού του Τελικού Βαθμού όλων των Φοιτητών

`findBetterGrade(Foititis f[]) -` Μέθοδος Εύρεσης του Μεγίστου Τελικού Βαθμού

`swapF(Foititis f[], int i, int j)-` Μέθοδος Αλλαγής του Περιεχομένου των στοιχείων `i` και `j` ενός Πίνακα Αντικειμένων Φοιτητών

```
public class ClassOfMethods {
/* Κλάση που περιέχει ΜΟΝΟ τις παρακάτω static μεθόδους :
   displayStudents(Foititis f[]) - Μέθοδος Εμφάνισης των Στοιχείων όλων
                                   των Φοιτητών
    findTelikosBathmosAll(Foititis f[]) - Μέθοδος Υπολογισμού του
                                   Τελικού Βαθμού όλων των Φοιτητών
    findBetterGrade(Foititis f[]) -    Μέθοδος Εύρεσης του Μεγίστου Τελικού
                                    Βαθμού
    swap(Foititis f[], int i, int j)-   Μέθοδος Αλλαγής του Περιεχομένου των
                                    στοιχείων i και j ενός Πίνακα
                                     Αντικειμένων Φοιτητών
    */

    static int betterGrade = 0;   // Μεταβλητή static

    // Μέθοδος Εμφάνισης των Στοιχείων όλων των Φοιτητών
    static void displayStudents(Foititis f[]){
     // Εμφάνιση Στοιχείων Φοιτητών με κλήση μεθόδου toString
     for (int i = 0;i<f.length;i++) {
        System.out.println ( "\nΣτοιχεία Φοιτητή " + i);
        System.out.println ( f[i] );
     }
```

```
    System.out.println ( "\n");
    }

    static void findTelikosBathmosAll(Foititis f[]){
    // Μέθοδος Υπολογισμού του Τελικού Βαθμού όλων των Φοιτητών
    for (int i = 0;i<f.length;i++)
     f[i].setTelikosBathmos();
    }

    // Μέθοδος Εύρεσης του Μεγίστου Τελικού Βαθμού
    static int findBetterGrade(Foititis f[]){

        double maxTelikos = f[0].getTelikosBathmos() ;
        for (int i = 1;i<=f.length-1;i++) {
        if ( f[i].getTelikos() > maxTelikos){
          maxTelikos = f[i].getTelikosBathmos();
          betterGrade = i;
        }
    }
        return betterGrade;
}

    static void swap(Foititis f[], int i, int j){
    // Μέθοδος Αλλαγής του Περιεχομένου των στοιχείων i και j ενός Πίνακα
    // Αντικειμένων Φοιτητών
     Foititis temp = new Foititis();
     temp = f[i];
     f[i] = f[j];
     f[j] = temp;
    }
}
```

Αλγόριθμος κλάσης `PinObjects` – `main()`

1. Δημιουργία Πίνακα Αντικειμένων `3` Φοιτητών
2. Δημιουργία `3` Αντικειμένων τύπου `Foititis` με ταυτόχρονη Εισαγωγή Στοιχείων (**εκτός του Κωδικού Μαθήματος και του Τελικού Βαθμού**) με τον πλήρη Δομητή
3. Υπολογισμός **Τελικού Βαθμού** για τους `3` φοιτητές με την κλήση της μεθόδου `findTelikosBathmosAll(Foititis f[])`
4. Εμφάνιση ΟΛΩΝ των στοιχείων των `3` Φοιτητών με τη μέθοδο `displayStudents(Foititis f[])`
5. Εύρεση του φοιτητή με το μεγαλύτερο Τελικό Βαθμό με την κλήση της μεθόδου `findBetterGrade(Foititis f[])`
6. Εμφάνιση της θέσης του φοιτητή με το μεγαλύτερο Τελικό Βαθμό
7. Δημιουργία ενός τυχαίου ακέραιου αριθμού `index` μεταξύ `0` και `f.length-2`
8. Ανταλλαγή των περιεχομένων των `f[index], f[index+1]` με την κλήση της μεθόδου `swapF(Foititis f[], int i, int j)`
9. Εμφάνιση ΟΛΩΝ των στοιχείων των `3` Φοιτητών με τη μέθοδο `displayStudents(Foititis f[])`

Πρόγραμμα

```
public class PinObjects {
/*
Να γραφεί Αλγόριθμος/Πρόγραμμμα το οποίο δημιουργεί έναν πίνακα 3 αντικειμένων τύπου Foititis. Η κλάση Foititis περιέχει και το πεδίο telikosBathmos για τον Τελικό Βαθμό. Και οι τρεις φοιτητές δημιουργούνται με το δομητή περνώντας απ' ευθείας τις τιμές, εκτός του πεδίου ΚΜ το οποίο δηλώνεται σαν static και παίρνει τιμή με δυναμική αρχικοποίηση και του Τελικού Βαθμού. Το πρόγραμμα υπολογίζει τον Τελικό Βαθμό του κάθε φοιτητή με την κλήση της μεθόδου findTelikosBathmosAll(f), εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας, το Βαθμό Εργαστηρίου και τον Τελικό Βαθμό με την κλήση της μεθόδου displayStudents(f) και βρίσκει το φοιτητή με το μεγαλύτερο Τελικό Βαθμό με την κλήση της μεθόδου findBetterGrade(f) και τον εμφανίζει. Κατόπιν δημιουργεί έναν τυχαίο ακέραιο αριθμό index μεταξύ 0 και f.length-2 και καλεί τη μέθοδο swap(f, index, index+1) με την οποία ανταλάσσει τα περιεχόμενα των f[index], f[index+1] και εμφανίζει τον Αριθμό Μητρώου, τον Κωδικό Μαθήματος, το Βαθμό Θεωρίας, το Βαθμό Εργαστηρίου και τον Τελικό Βαθμό με την κλήση της μεθόδου displayStudents(f).
*/
    public static void main(String[] args) {

    // Ανάθεση Τιμής στο Μέγεθος του Πίνακα
    int n = 3;

    // Δημιουργία Πίνακα 3 φοιτητών
  Foititis f[] = new Foititis[n];

  // Δημιουργία Αντικειμένων Φοιτητών με το δομητή
  f[0] = new Foititis(14013, 6.0, 6.4);
  f[1] = new Foititis(14014, 8.6, 8.0);
  f[2] = new Foititis(14015, 4.6, 5.0);

// Υπολογισμός Τελικού Βαθμού Φοιτητών με κλήση μεθόδου findTelikosBathmosAll
  ClassOfMethods.findTelikosBathmosAll(f);

  // Εμφάνιση Στοιχείων Φοιτητών με κλήση μεθόδου displayStudents
  ClassOfMethods.displayStudents(f);

  // Εύρεση Μεγίστου Τελικού Βαθμού με κλήση μεθόδου findBetterGrade
  int thesiMax = ClassOfMethods.findBetterGrade(f);
  System.out.println ( "Μέγιστο Τελικό Βαθμό = "
    + f[thesiMax].getTelikosBathmos() + " έχει ο Φοιτητής " + thesiMax);

  // Δημιουργία τυχαίου ακεραίου αριθμού μεταξύ 0 και f.length-2
  int index = (int) (Math.random()*2);

  // Εμφάνιση τυχαίου ακεραίου αριθμού μεταξύ 0 και f.length-2
  System.out.println( "\n\nΑνταλλαγή f[" + index + "] and f[" + (index+1) + "]");

  // Αλλαγή του Περιεχομένου των στοιχείων index και index + 1 του Πίνακα
  // Αντικειμένων Φοιτητών
  ClassOfMethods.swapF(f, index, (index + 1) );

  // Εμφάνιση Στοιχείων Φοιτητών με κλήση μεθόδου displayStudents()
  ClassOfMethods.displayStudents(f);

   }
}
```

Έξοδος Προγράμματος :

```
run:

Στοιχεία Φοιτητή 0

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.0
Βαθμός Εργαστηρίου = 6.4
Τελικός Βαθμός = 6.16

Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.6
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός = 8.36

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 5.0
Τελικός Βαθμός = 4.76


Μέγιστο Τελικό Βαθμό = 8.36 έχει ο Φοιτητής 1


Ανταλλαγή f[1] and f[2]

Στοιχεία Φοιτητή 0

Αριθμός Μητρώου = 14013
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 6.0
Βαθμός Εργαστηρίου = 6.4
Τελικός Βαθμός = 6.16

Στοιχεία Φοιτητή 1

Αριθμός Μητρώου = 14015
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 4.6
Βαθμός Εργαστηρίου = 5.0
Τελικός Βαθμός = 4.76

Στοιχεία Φοιτητή 2

Αριθμός Μητρώου = 14014
Κωδικός Μαθήματος = 6000232
Βαθμός Θεωρίας = 8.6
Βαθμός Εργαστηρίου = 8.0
Τελικός Βαθμός = 8.36
BUILD SUCCESSFUL (total time: 0 seconds)
```

8 ΣΥΜΒΟΛΟΣΕΙΡΕΣ - STRINGS

Οι **Συμβολοσειρές – Strings** στη Java είναι αντικείμενα και **όχι Πίνακες** Χαρακτήρων. Η Δήλωση μιας Συμβολοσειράς γίνεται με τη δήλωση του **τύπου** String των στοιχείων που θα αποθηκεύσει, το **όνομά** της και τη σταθερά τύπου String μέσα σε " " με ή χωρίς τον τελεστή **new**.

Παράδειγμα

```
String s1 = "Kostas Goulianas";
String s2 = new String("KOSTAS GOULIANAS");
String s3;
s3 = new String(s2);
```

όπου δηλώνουμε τη Συμβολοσειρά s1, χωρίς τον τελεστή new, η οποία θα αποθηκεύσει το όνομα "Kostas Goulianas", τη Συμβολοσειρά s2, με τον τελεστή new, η οποία θα αποθηκεύσει το όνομα "KOSTAS GOULIANAS" και τη Συμβολοσειρά s3, με τον τελεστή new, η οποία θα αποθηκεύσει τη Συμβολοσειρά s2, δηλαδή το όνομα "KOSTAS GOULIANAS". Όλες οι παραπάνω Συμβολοσειρές έχουν δημιουργηθεί με **δυναμική αρχικοποίηση**.

Εμφάνιση Συμβολοσειρών

Η Εμφάνιση των Συμβολοσειρών σταθερών ή μεταβλητών γίνεται με την εντολή System.out.println().

Παράδειγμα

```
String s1 = "Kostas Goulianas";
String s2 = new String("KOSTAS GOULIANAS");
String s3 = new String(s2);
System.out.println("Kostas Goulianas");
System.out.println("s1 = " + s1);
System.out.println("s2 = " + s2);
System.out.println("s3 = " + s3);
```

Έξοδος Προγράμματος

```
run:
Kostas Goulianas
s1 = Kostas Goulianas
s2 = KOSTAS GOULIANAS
s3 = KOSTAS GOULIANAS
BUILD SUCCESSFUL (total time: 0 seconds)
```

Μήκος Συμβολοσειράς

Για να βρούμε το **Μήκος** μιας Συμβολοσειράς χρησιμοποιούμε τη Μέθοδο **`length()`**, αφού η Συμβολοσειρά είναι αντικείμενο. **Η μέθοδος `length()` χρειάζεται παρενθέσεις**.

Παράδειγμα

```
String s1 = "Kostas Goulianas";
System.out.println("s1 = " + s1);
System.out.println("Το όνομα " + s1 + " έχει " + s1.length()
                        + " χαρακτήρες");
```

Έξοδος Προγράμματος

```
run:
s1 = Kostas Goulianas
Το όνομα Kostas Goulianas έχει 16 χαρακτήρες
BUILD SUCCESSFUL (total time: 0 seconds)
```

8.1 Μέθοδοι Χειρισμού Συμβολοσειρών

Η κλάση `String` περιέχει τις παρακάτω μεθόδους για σύγκριση, αναζήτηση κάποιου χαρακτήρα, εξαγωγή Υπο-Συμβολοσειράς (`substring`) κ.λ.π..

Η Μέθοδος `equals()`

Με τη Μέθοδο `equals()` συγκρίνουμε 2 Συμβολοσειρές. Αν είναι ίδιες, επιστρέφει `true`, διαφορετικά `false`. Η σύνταξή της είναι :

```
<Συμβολοσειρά-1>.equals(<Συμβολοσειρά-2>)
```

Παράδειγμα

```
String s1 = "Kostas Goulianas";
String s2 = new String("KOSTAS GOULIANAS");
if (s1.equals(s2)) System.out.println(" s1 = " + s1 + " = s2 = "
+ s2);
    else System.out.println(s1.equals(s2));

String s3 = new String(s2);
if (s2.equals(s3)) System.out.println(" s2 = " + s2 + " = s3 = "
                                        + s3);
```

Έξοδος Προγράμματος

```
run:
false
s2 = KOSTAS GOULIANAS = s3 = KOSTAS GOULIANAS
BUILD SUCCESSFUL (total time: 0 seconds)
```

Η Μέθοδος `compareTo()`

Με τη Μέθοδο `compareTo()` συγκρίνουμε 2 Συμβολοσειρές. Αν η πρώτη είναι μεγαλύτερη από τη δεύτερη, επιστρέφει μια τιμή μεγαλύτερη του μηδενός, αν η πρώτη είναι μικρότερη από τη δεύτερη, επιστρέφει μια τιμή μικρότερη του μηδενός και αν η πρώτη είναι ίση με τη δεύτερη, επιστρέφει την τιμή μηδέν. Η σύνταξή της είναι :

```
<Συμβολοσειρά-1>.compareTo(<Συμβολοσειρά-2>)
```

Παράδειγμα

```
String s1 = "Kostas Goulianas";
String s2 = new String("KOSTAS GOULIANAS");
String s3 = new String(s2);

if (s1.compareTo(s2) < 0 )
    System.out.println("s1 = " + s1 + " < s2 = " + s2);
else
    if (s1.compareTo(s2) > 0 )
        System.out.println("s1 = " + s1 + " > s2 = " + s2);
    else
        System.out.println("s1 = " + s1 + " = s2 = " + s2);

if (s2.compareTo(s3) < 0 )
        System.out.println("s2 = " + s2 + " < s3 = " + s3);
else
        if (s2.compareTo(s3) > 0 )
             System.out.println("s2 = " + s2 + " > s3 = " + s3);
        else
             System.out.println("s2 = " + s2 + " = s3 = " + s3);
```

Έξοδος Προγράμματος

```
run:
s1 = Kostas Goulianas > s2 = KOSTAS GOULIANAS
s2 = KOSTAS GOULIANAS = s3 = KOSTAS GOULIANAS
BUILD SUCCESSFUL (total time: 0 seconds)
```

Η Μέθοδος `substring()`

Με τη Μέθοδο `substring()` εξάγουμε ένα τμήμα μιας Συμβολοσειράς σε μια άλλη μεταβλητή τύπου `String`. Η αρχική Συμβολοσειρά παραμένει ως έχει. Η σύνταξή της είναι :

```
<Συμβολοσειρά>.substring(<Θέση Πρώτου Χαρακτήρα>,<Θέση Τελευταίου Χαρακτήρα> + 1)
```

Παράδειγμα

```
String s1 = "Kostas Goulianas";
System.out.println("s1 = " + s1);

// Εξαγωγή Τμήματος Συμβολοσειράς - Μέθοδος substring (Αρχή, Τέλος + 1)
String first = s1.substring(0, 6);
System.out.println("first name = " + first);
String last = s1.substring(7, 16);
System.out.println("last name = " + last);
```

Έξοδος Προγράμματος

```
run:
s1 = Kostas Goulianas
first name = Kostas
last name = Goulianas
BUILD SUCCESSFUL (total time: 0 seconds)
```

Η Μέθοδος `charAt()`

Με τη Μέθοδο `charAt()` εξάγουμε ένα χαρακτήρα μιας Συμβολοσειράς, αν δώσουμε τη θέση του. Η σύνταξή της είναι :

```
<Συμβολοσειρά>.charAt(<Θέση Χαρακτήρα>)
```

Παράδειγμα

```
String s1 = "Kostas Goulianas";
System.out.println("s1 = " + s1);

// Δημιουργία τυχαίου ακεραίου αριθμού στο 0-15
int index = (int)(Math.random()*15);

// Εξαγωγή Χαρακτήρα Συμβολοσειράς - Μέθοδος charAt(θέση)
System.out.println("Ο " + index + "-ος χαρακτήρας της Συμβολοσειράς " + s1 + " είναι : " + s1.charAt(index));
```

Έξοδος Προγράμματος

```
run:
s1 = Kostas Goulianas
Ο 5-ος χαρακτήρας της Συμβολοσειράς Kostas Goulianas είναι : s
BUILD SUCCESSFUL (total time: 0 seconds)
```

Οι Μέθοδοι `indexOf()` και `lastIndexOf()`

Με τις Μεθόδους `indexOf()` και `lastIndexOf()` βρίσκουμε τη Θέση του **πρώτου χαρακτήρα** της **πρώτης** και **τελευταίας** εμφάνισης ενός `substring` σε μια Συμβολοσειρά. Αν η δεν υπάρχει η `substring` στη Συμβολοσειρά, επιστρέφουν την τιμή `-1`. Η σύνταξή τους είναι :

```
<Συμβολοσειρά>.indexOf(<substring>)
<Συμβολοσειρά>.lastindexOf(<substring>)
```

Παράδειγμα

```
String s1 = "Kostas Goulianas";
System.out.println("s1 = " + s1);

String first = s1.substring(0, 6);
System.out.println("first name = " + first);
String last = s1.substring(7, 16);
System.out.println("last name = " + last);

// Εύρεση της θέσης του "Kostas", "Goulianas", "KOSTAS", "as" στη Συμβολοσειρά "Kostas Goulianas"

System.out.println("Ο 1-ος χαρακτήρας της Υπο-Συμβολοσειράς " + first + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.indexOf(first));

System.out.println("Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της " + "Υπο-Συμβολοσειράς " + first + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.lastIndexOf(first));

System.out.println("\nΟ 1-ος χαρακτήρας της Υπο-Συμβολοσειράς " + last + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.indexOf(last));

System.out.println("Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της " + "Υπο-Συμβολοσειράς " + last + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.lastIndexOf(last));

System.out.println("\nΟ 1-ος χαρακτήρας της Υπο-Συμβολοσειράς KOSTAS" + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.indexOf("KOSTAS"));

System.out.println("Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της " + " Υπο-Συμβολοσειράς KOSTAS στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.lastIndexOf("KOSTAS"));

System.out.println("\nΟ 1-ος χαρακτήρας της Υπο-Συμβολοσειράς as"  + " στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.indexOf("as"));
```

```
System.out.println("Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της " + " Υπο-Συμβολοσειράς as στη Συμβολοσειρά " + s1 + " βρέθηκε στη θέση : " + s1.lastIndexOf("as"));
```

Έξοδος Προγράμματος

```
run:
s1 = Kostas Goulianas
first name = Kostas
last name = Goulianas

Ο 1-ος χαρακτήρας της Υπο-Συμβολοσειράς Kostas στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: 0

Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της Υπο-Συμβολοσειράς Kostas στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: 0

Ο 1-ος χαρακτήρας της Υπο-Συμβολοσειράς Goulianas στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: 7

Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της Υπο-Συμβολοσειράς Goulianas στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: 7

Ο 1-ος χαρακτήρας της Υπο-Συμβολοσειράς KOSTAS στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: -1

Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της  Υπο-Συμβολοσειράς KOSTAS στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: -1

Ο 1-ος χαρακτήρας της Υπο-Συμβολοσειράς as στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση: 4

Ο 1-ος χαρακτήρας της τελευταίας εμφάνισης της  Υπο-Συμβολοσειράς as στη Συμβολοσειρά Kostas Goulianas βρέθηκε στη θέση : 14
BUILD SUCCESSFUL (total time: 0 seconds)
```

8.2 Συνένωση Συμβολοσειρών - String Concatenation

Η συνένωση Συμβολοσειρών γίνεται με τον τελεστή '+'.

Παράδειγμα

```
String first = "Kostas";
String last = "Goulianas";
String onoma = first + "      " + last;
System.out.println("Συμβολοσειρά με όνομα κενά και επώνυμο = " +
onoma);
```

Έξοδος Προγράμματος

```
run:
Συμβολοσειρά με όνομα κενά και επώνυμο = Kostas      Goulianas
BUILD SUCCESSFUL (total time: 0 seconds)
```

8.3 Πίνακες Συμβολοσειρών

Οι Πίνακες Συμβολοσειρών δηλώνονται όπως και οι άλλοι πίνακες. Τα στοιχεία τους όμως είναι συμβολοσειρές.

Παράδειγμα

```
// Δήλωση - Αρχικοποίηση Πίνακα Συμβολοσειρών
String name[] = {"Kostas", "Goulianas"};

// Εμφάνιση Στοιχείων Πίνακα Συμβολοσειρών
for ( int i = 0;i < name.length;i++)
   System.out.println("Στοιχείο " + i + " Πίνακα = " + name[i]);
```

Έξοδος Προγράμματος

```
run:
Στοιχείο 0 Πίνακα = Kostas
Στοιχείο 1 Πίνακα = Goulianas
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

Στον πίνακα το `length` χρησιμοποιείται χωρίς τις παρενθέσεις `()`.

8.4 Η κλάση StringBuffer

Η κλάση `StringBuffer` μας δίνει τη δυνατότητα, αν και οι Συμβολοσειρές γενικά είναι αμετάβλητες, δηλαδή δεν μπορεί να αλλάξει το περιεχόμενό τους, να αλλάξουμε το περιεχόμενό τους με τη χρήση της μεθόδου `setCharAt()`. Η σύνταξή της είναι :

```
<Συμβολοσειρά>.setCharAt (<Θέση Χαρακτήρα>,< Νέος Χαρακτήρας>)
```

Παράδειγμα

```
StringBuffer NAME = new StringBuffer("Kostas Goulianas");

// Εμφάνιση Αρχικής Συμβολοσειράς
System.out.println("\nΑρχική Συμβολοσειρά = " + NAME );

// Αλλαγή G σε g στο Επώνυμο με τη μέθοδο setCharAt()
NAME.setCharAt(7, 'g');
System.out.println("Η νέα Συμβολοσειρά μετά την Αλλαγή του G σε
g " + "στο Επώνυμο με τη μέθοδο setCharAt() = " + NAME);
```

Έξοδος Προγράμματος

```
run:
Αρχική Συμβολοσειρά = Kostas Goulianas
Η νέα Συμβολοσειρά μετά την Αλλαγή του G σε g στο Επώνυμο με τη μέθοδο
setCharAt() = Kostas goulianas
BUILD SUCCESSFUL (total time: 0 seconds)
```

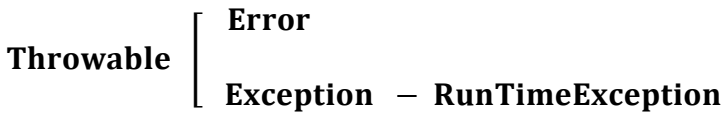
Παρατήρηση

Για να έχουμε πρόσβαση στη μέθοδο `setCharAt()` της κλάσης `StringBuffer` θα πρέπει να εισάγουμε το πακέτο `java.lang` που την περιέχει με την εντολή :

```
import java.lang.*;
```

9 ΕΞΑΙΡΕΣΕΙΣ - EXCEPTIONS

Με τον όρο **Εξαιρέσεις** (**Exceptions**) στη Java χαρακτηρίζουμε τα **σφάλματα** που μπορεί να προκύψουν κατά την εκτέλεση ενός προγράμματος, όπως διαίρεση με το μηδέν, προσπάθεια πρόσβασης σε στοιχείο ενός πίνακα με τιμή δείκτη εκτός των ορίων που έχουν δηλωθεί κ.λ.π.. Σε όλες τις Γλώσσες Προγραμματισμού, σε περίπτωση σφάλματος διακόπτεται η εκτέλεση του προγράμματος και εμφανίζονται ένα ή περισσότερα μηνύματα με πληροφορίες για το είδος και το σημείο του κώδικα στο οποίο προέκυψε το σφάλμα. Η Java δίνει τη δυνατότητα στο χρήστη να διαχειριστεί το σφάλμα που μπορεί να προκύψει, έτσι ώστε το πρόγραμμα να συνεχίσει κανονικά την εκτέλεσή του. Αυτό γίνεται με το **Χειριστή Εξαιρέσεων**, ένα block εντολών, οι οποίες καλούνται όταν προκύψει κάποιο σφάλμα και το χειρίζονται ανάλογα. Η Java περιέχει μια βιβλιοθήκη χειρισμού των πιο συνηθισμένων σφαλμάτων, αλλά δίνει και στο χρήστη τη δυνατότητα να χειριστεί τα σφάλματα με όποιον τρόπο επιθυμεί. Οι Εξαιρέσεις είναι **αντικείμενα**, οπότε ανήκουν σε κάποια κλάση. Οι Εξαιρέσεις που ενδιαφέρουν τον προγραμματιστή (γιατί υπάρχουν και σφάλματα που αφορούν την εικονική μηχανή της Java, τα οποία ανήκουν στην κλάση **Error**) είναι τα σφάλματα που προγράμματος, τα οποία ανήκουν στην κλάση **Exception** και πιο πολύ αυτά που συμβαίνουν κατά το χρόνο εκτέλεσης του προγράμματος (**Run-Time Exceptions**), τα οποία ανήκουν στην κλάση **RunTimeException**. Όλες οι εξαιρέσεις ανήκουν στην κλάση **Throwable**., ενώ οι κλάσεις **Error** και **Exception** είναι υπο-κλάσεις της κλασης **Throwable** και η κλάση **RunTimeException** είναι υπο-κλάση της κλασης **Exception** όπως φαίνεται στο επόμενο σχήμα :



Ο **Χειριστής Εξαιρέσεων** περιλαμβάνει τις παρακάτω εντολές - λέξεις κλειδιά :

try : block εντολών, στο οποίο μπορεί να **προκύψει** μια εξαίρεση.

catch : block εντολών, οι οποίες θα **χειριστούν** την εξαίρεση, αν προκύψει.

finally : προαιρετικό block εντολών, οι οποίες θα **εκτελεστούν**, **οπωσδήποτε**, υπάρξει δεν υπάρξει εξαίρεση.

throws : εντολή, η οποία **προωθεί** μια εξαίρεση στην καλούσα μέθοδο να τη χειριστεί.

Ο χειρισμός των εξαιρέσεων έχει την παρακάτω μορφή :

```
try{
    block εντολών, στο οποίο μπορεί να προκύψει μια εξαίρεση.
}
catch ( <Τύπος_Εξαίρεσης_1> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
catch ( <Τύπος_Εξαίρεσης_2> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
…
catch ( <Τύπος_Εξαίρεσης_n> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
finally{
    προαιρετικό block εντολών, οι οποίες θα εκτελεστούν, οπωσδήποτε.
}
```

9.1 Παράδειγμα Εξαίρεσης που προκαλείται σε εντολή της main()

Στο επόμενο πρόγραμμα προσπαθούμε να αλλάξουμε το περιεχόμενο της θέσης 5 ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις 0 μέχρι **a.length**−1 = 4 :

```
package exceptions1;
public class Exceptions1 {

    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};
        //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων
        a[5] = 99;
    }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions1.Exceptions1.main(Exceptions1.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

Με το τρέξιμο του προγράμματος προκύπτει η εξαίρεση ArrayIndexOutOfBoundsException στη γραμμή 7 της main(), η οποία ανήκει στην κλάση Exceptions1, η οποία ανήκει στο πακέτο exceptions1 (exceptions1.Exceptions1.main).

9.2 Παράδειγμα Εξαίρεσης σε εντολή επανάληψης της main()

Στο επόμενο πρόγραμμα προσπαθούμε να εμφανίσουμε το περιεχόμενο των θέσεων 0 μέχρι 5 ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις 0 μέχρι **a.length**−1=4, οπότε προκαλείται η εξαίρεση **ArrayIndexOutOfBoundsException** :

```
package exceptions2;
public class Exceptions2 {
  public static void main(String[] args) {
    int a[] = {1,2,3,4,5};
     //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων κατά την Εμφάνιση
    for ( int i = 0; i <= a.length; i++ )
       System.out.println("a[" + i + "] = " + a[i]);
    }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions2.Exceptions2.main(Exceptions2.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

9.2.1 Χειρισμός της Εξαίρεσης σε εντολή επανάληψης της main()

Στο προηγούμενο παράδειγμα η προσπάθεια πρόσβασης στα στοιχεία στις θέσεις i = **a.length** = 5 και i = **a.length + 1** = 6 του πίνακα a εκτός των ορίων της δήλωσης των θέσεων του πίνακα (0 μέχρι **a.length - 1**) με την εντολή

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί 2 φορές την εξαίρεση ArrayIndexOutOfBoundsException. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε **την εντολή που θα προκαλέσει την εξαίρεση** στο block try και **τις εντολές χειρισμού της εξαίρεσης** στο block catch , στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα καθώς και το περιεχόμενο του αντικειμένου της εξαίρεσης :

```
package exceptions2a;
public class Exceptions2a {
 public static void main(String[] args) {
  int a[] = {1,2,3,4,5};
  int i;
  //Εμφάνιση στοιχείων πίνακα -
  //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων σε block try
     for ( i = 0; i <= a.length + 1; i++ )
       try {
          System.out.println("a[" + i + "] = " + a[i]);
       }
```

```
    // Χειρισμός Εξαίρεσης - Εμφάνιση αντίστοιχου μηνύματος
    catch (IndexOutOfBoundsException obj){
      System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων ");
      System.out.println("Είδος Εξαίρεσης : " + obj);
    }
  }
}
```

Έξοδος Προγράμματος

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων Δήλωσης
Είδος Εξαίρεσης : java.lang.ArrayIndexOutOfBoundsException: 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων Δήλωσης
Είδος Εξαίρεσης : java.lang.ArrayIndexOutOfBoundsException: 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

Για να χειριστούμε **ΟΛΕΣ** τις εξαιρέσεις που μπορεί να συμβούν μέσα σε μια εντολή επανάληψης, θα πρέπει **η εντολή που πρόκειται να προκαλέσει την εξαίρεση** να ανήκει στο `block try` **και όχι η εντολή επανάληψης**, η οποία περιέχει την εντολή που πρόκειται να προκαλέσει την εξαίρεση.

9.3 Παράδειγμα Εξαίρεσης σε μέθοδο που καλείται από τη `main()`

Στο επόμενο πρόγραμμα προσπαθούμε **με την κλήση της μεθόδου `displayA()`** να εμφανίσουμε το περιεχόμενο των θέσεων `0` μέχρι `5` ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις `0` μέχρι `4`, οπότε προκαλείται η εξαίρεση `ArrayIndexOutOfBoundsException` :

```
package exceptions3Method;
public class Exceptions3Method {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα  -
  //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων - Μήνυμα Λάθους
  for ( int i = 0; i <= a.length; i++ )
       System.out.println("a[" + i + "] = " + a[i]);
  }

  public static void main(String[] args) {
  int a[] = {1,2,3,4,5};

  //Κλήση μεθόδου displayA()
  displayA(a);
  }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions3Method.Exceptions3Method.displayA(Exceptions3Method.java:7)
    at exceptions3Method.Exceptions3Method.main(Exceptions3Method.java:14)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

9.3.1 Χειρισμός της Εξαίρεσης Μέσα στη Μέθοδο που Προκαλείται

Και σε αυτό το παράδειγμα η προσπάθεια πρόσβασης στο στοιχείο στη θέση `i` = **`a.length`** = 5 του πίνακα `a` εκτός των ορίων της δήλωσης των θέσεων του πίνακα (`0` μέχρι **`a.length - 1`**) με την εντολή στη μέθοδο **`displayA()`**

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί την εξαίρεση `ArrayIndexOutOfBoundsException`. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε στη μέθοδο `displayA()` **την εντολή που θα προκαλέσει την εξαίρεση** στο `block try` και **τις εντολές χειρισμού της εξαίρεσης** στο `block catch`, στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα :

```
package exceptiions3MethodTry;
public class Exceptiions3MethodTry {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα  - Χειρισμός Εξαίρεσης στη μέθοδο
   for ( int i = 0; i <= a.length; i++ )
    try {
       System.out.println("a[" + i + "] = " + a[i]);
    }

    catch (IndexOutOfBoundsException obj){
      System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
    }
  }

  public static void main(String[] args) {
    int a[] = {1,2,3,4,5};

    //Κλήση μεθόδου displayA()
    displayA(a);
    }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.3.2 Χειρισμός της Εξαίρεσης της Μεθόδου στη `main()`

Και σε αυτό το παράδειγμα η προσπάθεια πρόσβασης σε στοιχείο (`i` = **`a.length`** = 5) του πίνακα `a` εκτός των ορίων της δήλωσης των θέσεων του πίνακα (`0` μέχρι **`a.length - 1`**) με την εντολή στη μέθοδο **`displayA()`**

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί την εξαίρεση `ArrayIndexOutOfBoundsException`. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε **την εντολή κλήσης της μεθόδου, η οποία περιέχει τις εντολές που θα προκαλέσουν την εξαίρεση** στο `block try` και **τις εντολές χειρισμού της εξαίρεσης** στο `block catch` , στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα :

```
package exceptions3MethodTryMain;
public class Exceptions3MethodTryMain {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα - Χειρισμός Εξαίρεσης στη main
  for ( int i = 0; i <= a.length; i++ )
     System.out.println("a[" + i + "] = " + a[i]);
  }

  public static void main(String[] args) {
   int a[] = {1,2,3,4,5};

   //Χειρισμός Εξαίρεσης στην Κλήση της μεθόδου displayA()
   try {
    displayA(a);
   }
   catch (IndexOutOfBoundsException obj){
     System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
        }
   }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.4 Παράδειγμα 2 Εξαιρέσεων σε εντολή επανάληψης της main()

Στο επόμενο πρόγραμμα προσπαθούμε να εμφανίσουμε το περιεχόμενο των θέσεων 0 μέχρι 5 ενός πίνακα δια του αντιστοίχου δείκτη i. Η προσπάθεια πρόσβασης στο στοιχείο στη θέση i = **a.length** = 5 του πίνακα a εκτός των ορίων της δήλωσης των θέσεων του πίνακα (0 - **a.length - 1**) προκαλεί την εξαίρεση ArrayIndexOutOfBoundsException, ενώ η διαίρεση του στοιχείου a[0] δια του i = 0 προκαλεί την εξαίρεση ArithmeticException, η οποία προκαλεί και τον τερματισμό του προγράμματος με την εκτέλεση της πρώτης εντολής **System.out.println()** στην εντολή **for :**

```
package exceptionstwoexceptions;
public class ExceptionsTwoExceptions {
    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};
        //Εμφάνιση στοιχείων πίνακα
        for ( int i = 0; i <= a.length; i++ )
           System.out.println("a[" + i + "]/" + i + " = " + (a[i]/i));
     }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at exceptionstwoexceptions.ExceptionsTwoExceptions.main(ExceptionsTwoExceptions.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.4.1 Χειρισμός των 2 Εξαιρέσεων στη main()

Για να χειριστούμε τις 2 πιθανές εξαιρέσεις (ArithmeticException και ArrayIndexOutOfBoundsException) που μπορεί να προκύψουν πρέπει να συμπεριλάβουμε στο πρόγραμμα 2 εντολές catch εμφανίζοντας και το αντίστοιχο μήνυμα :

```
package exeptions2Catch;
public class Exeptions2Catch {

    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};

        //Εμφάνιση στοιχείων πίνακα  - Χειρισμός Εξαίρεσης
        for ( int i = 0; i <= a.length; i++ )

         try {
           System.out.println("a[" + i + "]/" + i + " = " + (a[i]/i));
         }


         catch (ArithmeticException obj){
            System.out.println("Διαίρεση με το μηδέν");
         }

         catch (ArrayIndexOutOfBoundsException obj){
            System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
         }
     }
}
```

Έξοδος Προγράμματος

```
run:
Διαίρεση με το μηδέν
a[1]/1 = 2
a[2]/2 = 1
a[3]/3 = 1
a[4]/4 = 1
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 1 second)
```

9.5 Η λέξη-κλειδί throws

Αν μια μέθοδος μπορεί να προκαλέσει μια εξαίρεση, την οποία δεν μπορεί ή δεν θέλουμε να τη χειριστεί η ίδια η μέθοδος, θα πρέπει να **προωθήσει** την εξαίρεση στην καλούσα μέθοδο με τον όρο `throws`. Αυτό γίνεται στην υπογραφή της μεθόδου, όπου **μετά τη λίστα των παραμέτρων** γράφεται η λέξη `throws` με τα πιθανά είδη εξαιρέσεων. Ο γενικός τύπος της υπογραφής της μεθόδου σ' αυτή την περίπτωση είναι :

```
<Τύπος_Επιστροφής> <Όνομα_Μεθόδου>(Λίστα_Παραμέτρων)
                   throws Λίστα_Εξαιρέσεων
```

9.5.1 Προώθηση Εξαίρεσης μεθόδου στην καλούσα μέθοδο `main()`

Αν θέλουμε να διαβάσουμε ένα χαρακτήρα στη `main()` καλώντας τη μέθοδο `prompt()`, θα **πρέπει** να **προωθήσουμε** με το `throws` την εξαίρεση στην είσοδο δεδομένων που μπορεί να προκύψει στη μέθοδο `prompt()`, στην καλούσα μέθοδο, τη `main()` , όπως φαίνεται στο επόμενο πρόγραμμα.

```
package exceptionsthrows;
public class ExceptionsThrows {
    public static char prompt()
                        throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args) {
      char ch;
      ch = prompt();
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source
code - unreported exception java.io.IOException; must be caught or declared
to be thrown
    at exceptionsthrows.ExceptionsThrows.main(ExceptionsThrows.java:11)
Java Result: 1
BUILD SUCCESSFUL (total time: 2 seconds)
```

Παρατήρηση

Το πρόγραμμα εμφανίζει το παραπάνω μήνυμα, γιατί ενώ υπάρχει πιθανότητα να προκληθεί στη μέθοδο `prompt()` μια εξαίρεση στην είσοδο δεδομένων (`java.io.IOException`), η μέθοδος `prompt()` την προωθεί στη `main()` με την εντολή **`throws java.io.IOException`**, αλλά η καλούσα μέθοδος **δεν τη χειρίζεται**. Ο χειρισμός μπορεί να γίνει με 2 τρόπους :

- Να την **προωθήσει** και η καλούσα μέθοδος.
- Να τη **χειριστεί** η καλούσα μέθοδος με τη χρήση `try – catch`.

9.5.2 Προώθηση Εξαίρεσης μεθόδου και από την καλούσα μέθοδο `main()`

Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrows1;
public class ExceptionThrows1 {
    public static char prompt()
                        throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args)
                         throws java.io.IOException {
      char ch;
      ch = prompt();

      // Εμφάνιση Χαρακτήρα
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 9 seconds)
```

9.5.3 Χειρισμός Εξαίρεσης μεθόδου από την καλούσα μέθοδο `main()`

Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrows2;
public class ExceptionThrows2 {
    public static char prompt()
                        throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args)
                         {
      char ch;
      // Έλεγχος για Πιθανή Εξαίρεση IOException
      try {
           ch = prompt();
       }

       catch (java.io.IOException obj){
            System.out.println("Εξαίρεση σε Είσοδο Δεδομένων");
            ch = 'a';
       }

       // Εμφάνιση Χαρακτήρα
       System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 3 seconds)
```

9.5.4 Χειρισμός Εξαίρεσης μεθόδου Μέσα την καλούσα μέθοδο

Σ' αυτή την περίπτωση δε χρειάζεται να προωθήσουμε την εξαίρεση στην καλούσα μέθοδο, γιατί **τη χειρίζεται η ίδια η μέθοδος**. Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrowstry;
public class ExceptionThrowsTry {
    public static char prompt(){
    char ch = 'a';

    // Έλεγχος για Πιθανή Εξαίρεση IOException
    try {
     System.out.print("Δώσε ένα χαρακτήρα : ");
     ch = (char) System.in.read();
    }
```

```
    catch (java.io.IOException obj){
            System.out.println("Εξαίρεση σε Είσοδο Δεδομένων");
    }
    return ch;
    }

    public static void main(String[] args)
                        {
      char ch;
      // Κλήση Μεθόδου για Διάβασμα Χαρακτήρα
      ch = prompt();

      // Εμφάνιση Χαρακτήρα
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 3 seconds)
```

9.5.5 Ελεγμένες και Μη Ελεγμένες Εξαιρέσεις και ο Όρος `throws`

Οι εξαιρέσεις που μπορεί να συμβούν κατά την εκτέλεση ενός προγράμματος δεν είναι απαραίτητο να συμπεριλαμβάνονται στη λίστα `throws`, γιατί η `Java` θεωρεί πως είναι φυσικό να συμβούν τέτοιου είδους εξαιρέσεις, οπότε δε χρειάζεται να ελεγχθεί, αν συμβεί μια τέτοια εξαίρεση σε μια μέθοδο, ή αν θα τη χειριστεί και λέγονται **μη ελεγμένες** εξαιρέσεις. Οι πιο συνηθισμένες εξαιρέσεις αυτού του είδους είναι :

Εξαίρεση	**Σημασία**
`ArithmeticException`	Αριθμητικό Σφάλμα, όπως Διαίρεση με μηδέν
`ArrayIndexOutOfBoundsException`	Δείκτης Πίνακα εκτός ορίων
`IllegalArgumentException`	Λάθος παράμετρος στην κλήση μεθόδου
`NegativeArraySizeException`	Αρνητικό Μέγεθος Πίνακα
`NumberFormatException`	Λάθος μορφή δεδομένων

Υπάρχουν όμως και οι **ελεγμένες** εξαιρέσεις, όπως η `java.io.IOException` που **πρέπει** να συμπεριλαμβάνονται στη λίστα `throws` μιας μεθόδου, **αν δεν τις χειρίζεται η μέθοδος**.

10 ΕΙΣΟΔΟΣ – ΕΞΟΔΟΣ ΔΕΔΟΜΕΝΩΝ

Οι περισσότερες εφαρμογές της Java δε στηρίζονται σε προγράμματα, τα οποία δέχονται είσοδο δεδομένων απ' το πληκτρολόγιο, αλλά σε applets, εφαρμογές που βασίζονται στην παραθυρική επικοινωνία με το χρήστη. Γι' αυτό το λόγο, το μέρος του συστήματος Εισόδου/Εξόδου της Java (I/O, Input/Output System) δε χρσιμοποιείται πολύ στον εμπορικό κώδικα, αλλά περισσότερο για διδακτικούς λόγους. Το Σύστημα Εισόδου/Εξόδου της Java στηρίζεται στα streams δεδομένων (**δέσμες** ή **ρεύματα** δεδομένων), τα οποία συνδέονται με μια φυσική συσκευή, πληκτρολόγιο, οθόνη ή αρχείο. Υπάρχουν δύο τύποι streams, byte και χαρακτήρων. Τα streams χαρακτήρων προστέθηκαν στις τελευταίες εκδόσεις της Java. Τα byte streams παρέχουν έναν εύκολο τρόπο στο χειρισμό δεδομένων και είναι πολύ αποτελεσματικά στην ανάγνωση και εγγραφή δυαδικών δεδομένων από και σε αρχεία. Τα streams χαρακτήρων σχεδιάστηκαν για το χειρισμό εισόδου και εξόδου χαρακτήρων. Στηρίζονται στα byte streams, οπότε μετατρέπουν τα bytes σε χαρακτήρες. Αν και το σύστημα Εισόδου/Εξόδου της Java περιλαμβάνει πολλές και διάφορες κλάσεις και υποκλάσεις, ο απλός χρήστης που δεν έχει ακόμη γνώση της κληρονομικότητας θα πρέπει απλώς να γνωρίζει ότι η εντολή System.out.println() για παράδειγμα είναι η κλήση της μεθόδου println() του αντικειμένου System.out, το οποίο αναφέρεται στο στάνταρ stream εξόδου, την οθόνη. Παρόμοια, το αντικείμενο System.in, αναφέρεται στο στάνταρ stream εισόδου, το πληκτρολόγιο.

10.1 Byte Streams

Οι κλάσεις που χρησιμοποιούνται για είσοδο και έξοδο δεσμών bytes (byte streams) είναι οι κλάσεις InputStream και OutputStream αντίστοιχα. Το System.in είναι αντικείμενο τύπου InputStream, ενώ το System.out είναι αντικείμενο τύπου OutputStream. Τα αντικείμενα αυτά συνδέονται και με τις αντίστοιχες μεθόδους. Έτσι, η μορφή που μπορεί να πάρει η κλήση της μεθόδου για διάβασμα χαρακτήρων σε μορφή bytes είναι :

`int read()` : Επιστρέφει σε ακέραιο το περιεχόμενο ενός byte, χρειάζεται διανομή.
`int read(byte b[])` : Επιστρέφει τον αριθμό των bytes που διάβασε με επιτυχία στον πίνακα b[] από bytes
`int read(byte b[], int offset, int numOfBytes)` : Διαβάζει έναν πίνακα από bytes από το b[offset] μέχρι το numOfBytes

ενώ για την εμφάνιση των χαρακτήρων – bytes χρησιμοποιούνται οι μέθοδοι :

`void write(int b)` : Εμφανίζει ένα χαρακτήρα
`void write(byte b[])` : Εμφανίζει έναν πίνακα από bytes

`void write(byte b[], int offset, int numOfBytes)` : Εμφανίζει όσους χαρακτήρες επιλέξουμε από έναν πίνακα από `bytes`

Παράδειγμα 1

Με την εντολή

```
char ch = (char) System.in.read();
```

διαβάζουμε ένα `byte`, το μετατρέπουμε σε χαρακτήρα με **διανομή** `(char)` και αποθηκεύουμε το χαρακτήρα στη μεταβλητή `ch`.

Παράδειγμα 2

Με το επόμενο πρόγραμμα δηλώνουμε έναν πίνακα από `bytes`, τον `bytesPin[]`, στον οποίο αποθηκεύουμε ένα όνομα, το οποίο εμφανίζουμε με την εντολή **`System.out.write()`** :

```
public class ReadBytePin {

    public static void main(String[] args) {
        int numOfBytes = 0;
        byte bytesPin[] = new byte[100];
        System.out.print("Enter Your Name : ");

          try {
          numOfBytes = System.in.read(bytesPin);
          System.out.println("You Entered " + numOfBytes + " Characters");
          System.out.print("Your Name Is : ");
          System.out.write(bytesPin, 0, numOfBytes);
          System.out.println();
        }

        catch (java.io.IOException e) {
         System.out.println("Error Reading Your Name");
        }
    }
    }
```

Έξοδος Προγράμματος

```
run:
Enter Your Name : Kostas Goulianas
You Entered 17 Characters
Your Name Is : Kostas Goulianas
BUILD SUCCESSFUL (total time: 2 seconds)
```

Παρατήρηση

Τα στοιχεία του πίνακα **`bytesPin`** μπορούν να εμφανιστούν και με την εντολή `System.out.print()` και `System.out.println()`.

10.2 Streams Χαρακτήρων

Το System.in (η είσοδος δεδομένων δηλαδή) είναι αντικείμενο τύπου InputStream, μιας κλάσης που χρησιμοποιείται για είσοδο και έξοδο δεσμών bytes (byte streams). Για να χρησιμοποιήσουμε είσοδο και έξοδο δεδομένων που βασίζονται σε streams χαρακτήρων, θα πρέπει να χρησιμοποιήσουμε τις αντίστοιχες κλάσεις που βασίζονται σε streams χαρακτήρων, έτσι ώστε το System.in που είναι byte stream να μετατραπεί σε stream χαρακτήρων. Αυτό γίνεται με τη δημιουργία ενός αντικειμένου τύπου InputStreamReader, όπου το System.in περνάει σαν παράμετρος. Αυτό το νέο αντικείμενο περνάει σαν παράμετρος στη δημιουργία ενός αντικειμένου ενδιάμεσης μνήμης τύπου BufferedReader. Πιο αναλυτικά :

Με την εντολή :

```
InputStreamReader isr = new InputStreamReader(System.in);
```

το System.in που είναι byte stream μετατρέπεται σε stream χαρακτήρων.

Με την εντολή :

```
BufferedReader br = new BufferedReader(isr);
```

το αντικείμενο τύπου **BufferedReader** br είναι πλέον ένα stream χαρακτήρων, το οποίο συνδέεται με το πληκτρολόγιο, μέσω του System.in.

Το ίδιο αποτέλεσμα έχουμε με ΜΙΑ εντολή :

```
BufferedReader br = new BufferedReader(new
                                    InputStreamReader(System.in));
```

Το αντικείμενο τύπου **BufferedReader** συνδέεται και με τις αντίστοιχες μεθόδους. Έτσι, η μορφή που μπορεί να πάρει η κλήση της μεθόδου για διάβασμα χαρακτήρων σε μορφή χαρακτήρων είναι :

br.read() για το διάβασμα ενός χαρακτήρα
br.readLine() για το διάβασμα μιας Γραμμής Χαρακτήρων (Συμβολοσειράς).

ενώ για την εμφάνιση των χαρακτήρων ή των Συμβολοσειρών χρησιμοποιούνται οι μέθοδοι print() και println().

Παρατήρηση

Στα επαγγελματικά προγράμματα η εμφάνιση γίνεται με τη δημιουργία ενός αντικειμένου τύπου PrintWriter και τη χρήση των μεθόδων print() και println().

Παράδειγμα 1

Με το παρακάτω πρόγραμμα διαβάζουμε χαρακτήρες με τη μέθοδο **`read()`** μέχρι να δώσουμε τελεία και τους εμφανίζουμε.

Πρόγραμμα

```
package readchars;
import java.io.*;
public class ReadChars {

public static void main(String[] args)
                            throws java.io.IOException{
        char ch;

        System.out.print("Enter Your Name, Finish With . : ");
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        do {
            ch = (char)br.read();
            System.out.print(ch);
        }
        while ( ch != '.');
        System.out.println();
}
}
```

Έξοδος Προγράμματος

```
run:
Enter Your Name, Finish With . : Kostas Goulianas.
Kostas Goulianas.
BUILD SUCCESSFUL (total time: 9 seconds)
```

Παράδειγμα 2

Με το επόμενο πρόγραμμα δηλώνουμε μια μεταβλητή **`line`** τύπου `String`, στην οποία αποθηκεύουμε τη συμβολοσειρά που διαβάζουμε με τη μέθοδο **`readLine()`** .

Πρόγραμμα

```
package readstrings;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class ReadStrings {

public static void main(String[] args)
                            throws java.io.IOException{
        String line;

        System.out.println("Enter Lines, Finish With stop : ");
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        do {
            line = br.readLine();
            }
        while ( !line.equals("stop") );
   //   System.out.println();
}
}
```

Έξοδος Προγράμματος

```
run:
Enter Lines, Finish With stop :
line 1
line 2
line 3
stop
BUILD SUCCESSFUL (total time: 18 seconds)
```

Παρατήρηση 2

Και στα δύο παραδείγματα θα πρέπει να εισάγουμε τις κλάσεις **BufferedReader** και **InputStreamReader** με τις εντολές :

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
```

ή με την εντολή :

```
import java.io.*;
```

10.3 Μετατροπή Συμβολοσειρών σε Απλούς Τύπους - Wrappers

Η μέθοδος `println()` στην εντολή `System.out.println()` μετατρέπει αυτόματα το περιεχόμενο μιας μεταβλητής σε αναγνώσιμη μορφή. Δεν μπορεί να γίνει όμως το ίδιο και με τις συμβολοσειρές που διαβάζονται με τη μέθοδο **`readLine()`**, ώστε να μετατραπούν σε δεδομένα απλών τύπων (παράδειγμα η συμβολοσειρά 123 να μετατραπεί στον ακέραιο αριθμό 123) , γιατί οι απλοί τύποι δεν είναι αντικείμενα. Χρειάζονται λοιπόν οι αντίστοιχοι **μετατροπείς** σε τύπους `double`, `float`, `long`, `int`, `short`, `byte`, `char` και `boolean`. Για το σκοπό αυτό χρησιμοποιούνται οι αντίστοιχες κλάσεις για κάθε απλό τύπο με τις αντίστοιχες μεθόδους, οι οποίες είναι οι παρακάτω :

Μετατροπέας - Wrapper	**Αποτέλεσμα**
`Double.parseDouble(<string>)`	Μετατροπή του `<string>` σε `double`
`Float.parseFloat(<string>)`	Μετατροπή του `<string>` σε `float`
`Long.parseLong(<string>)`	Μετατροπή του `<string>` σε `long`
`Integer.parseInt(<string>)`	Μετατροπή του `<string>` σε `int`
`Short.parseShort(<string>)`	Μετατροπή του `<string>` σε `short`
`Byte.parseByte(<string>)`	Μετατροπή του `<string>` σε `byte`

10.3.1 Πρόγραμμα με Μεθόδους Μετατροπής String σε Δεδομένα Απλών Τύπων

Να γίνει πρόγραμμα που θα ζητάει απ' το χρήστη να εισάγει έναν ακέραιο αριθμό από το πληκτρολόγιο, θα μετατρέπει τη συμβολοσειρά σε ακέραιο και θα εμφανίζει την τιμή του.

Πρόγραμμα

```
package readinteger;
import java.io.*;
public class ReadInteger {
/* Πρόγραμμα που διαβάζει μια συμβολοσειρά απ' το πληκτρολόγιο, τη μετατρέπει
   σε ακέραιο αριθμό, τον οποίο και εμφανίζει
*/
public static void main(String[] args)
                              throws java.io.IOException{
        String line;
        int number;
        System.out.print("Δώσε έναν ακέραιο αριθμό : ");
        // Δημιουργία Αντικειμένου για Μετατροπή Εισόδου σε stream Χαρακτήρων
        BufferedReader br = new BufferedReader(new
                          InputStreamReader(System.in));
        // Εισαγωγή Συμβολοσειράς
        line = br.readLine();
        // Μετατροπή string Εισόδου σε Ακέραιο
        number = Integer.parseInt( line );
        // Εμφάνιση Τιμής Ακεραίου Μετά τη Μετατροπή
        System.out.println("Ο αριθμός που έδωσες είναι : " + number);
}
}
```

Έξοδος Προγράμματος

```
run:
Δώσε έναν ακέραιο αριθμό : 12345
Ο αριθμός που έδωσες είναι : 12345
BUILD SUCCESSFUL (total time: 4 seconds)

run:
Δώσε έναν ακέραιο αριθμό : 123.45
Exception in thread "main" java.lang.NumberFormatException: For input string:
"123.45"
    at
java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.parseInt(Integer.java:615)
    at readinteger.ReadInteger.main(ReadInteger.java:18)
Java Result: 1
BUILD SUCCESSFUL (total time: 4 seconds)

run:
Δώσε έναν ακέραιο αριθμό : kostas
Exception in thread "main" java.lang.NumberFormatException: For input string:
"kostas"
    at
java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.lang.Integer.parseInt(Integer.java:580)
    at java.lang.Integer.parseInt(Integer.java:615)
    at readinteger.ReadInteger.main(ReadInteger.java:18)
Java Result: 1
BUILD SUCCESSFUL (total time: 4 seconds)
```

Παρατήρηση

Η εισαγωγή πραγματικού αριθμού και συμβολοσειράς προκάλεσε ένα σφάλμα NumberFormatException, το οποίο πρέπει να ελεγχθεί, όπως φαίνεται στο επόμενο παράδειγμα :

10.3.2 Πρόγραμμα με Μεθόδους Μετατροπής String σε Δεδομένα Απλών Τύπων και Έλεγχο για Πιθανή Εξαίρεση

Να τροποποιηθεί το προηγούμενο πρόγραμμα που ζητάει απ' το χρήστη να εισάγει έναν ακέραιο αριθμό από το πληκτρολόγιο, μετατρέπει τη συμβολοσειρά σε ακέραιο και εμφανίζει την τιμή του, έτσι ώστε **να κάνει και τον έλεγχο για εξαιρέσεις**.

Πρόγραμμα

```
package readintegertrycatch;
import java.io.*;
public class ReadIntegerTryCatch {
/* Πρόγραμμα που διαβάζει μια συμβολοσειρά απ' το πληκτρολόγιο, τη μετατρέπει
σε ακέραιο αριθμό, τον οποίο και εμφανίζει κάνοντας και τον έλεγχο για πιθανή
εξαίρεση */
public static void main(String[] args) {
        String line;
        int number;
        System.out.print("Δώσε έναν ακέραιο αριθμό : ");
        // Δημιουργία Αντικειμένου για Μετατροπή Εισόδου σε stream Χαρακτήρων
        BufferedReader br = new BufferedReader(new
                        InputStreamReader(System.in));
        try {
        // Εισαγωγή Συμβολοσειράς
        line = br.readLine();
        // Μετατροπή string Εισόδου σε Ακέραιο
        number = Integer.parseInt( line );
        // Εμφάνιση Τιμής Ακεραίου Μετά τη Μετατροπή
        System.out.println("Ο αριθμός που έδωσες είναι : " + number);
        }
        catch ( Exception obj){
            System.out.println("Σφάλμα Εισόδου - Εξόδου");
        }
}
}
```

Έξοδος Προγράμματος

```
run:
Δώσε έναν ακέραιο αριθμό : 12345
Ο αριθμός που έδωσες είναι : 12345
BUILD SUCCESSFUL (total time: 4 seconds)

run:
Δώσε έναν ακέραιο αριθμό : 123.45
Σφάλμα Εισόδου - Εξόδου
BUILD SUCCESSFUL (total time: 6 seconds)

run:
Δώσε έναν ακέραιο αριθμό : kostas
Σφάλμα Εισόδου - Εξόδου
BUILD SUCCESSFUL (total time: 6 seconds)
```

Παρατήρηση

Η εισαγωγή του αριθμού και ο έλεγχος για πιθανό σφάλμα θα μπορούσαν να ανήκουν σε μια `static` μέθοδο, όπως φαίνεται στο επόμενο παράδειγμα :

10.3.3 Πρόγραμμα με `static` Μέθοδο για Διάβασμα Ακεραίου

Να τροποποιηθεί το προηγούμενο πρόγραμμα που ζητάει απ' το χρήστη να εισάγει έναν ακέραιο αριθμό από το πληκτρολόγιο, μετατρέπει τη συμβολοσειρά σε ακέραιο και εμφανίζει την τιμή του, κάνοντας και τον έλεγχο για εξαιρέσεις, ώστε το διάβασμα και η μετατροπή της συμβολοσειράς σε ακέραιο να γίνεται με μια μέθοδο. Η `main()` απλώς θα εμφανίζει το μήνυμα εισαγωγής του αριθμού και θα εμφανίζει την τιμή του .

Πρόγραμμα

```
package readintegertrycatchmethod;
import java.io.*;
public class ReadIntegerTryCatchMethod {
/*
Πρόγραμμα που διαβάζει μια συμβολοσειρά απ' το πληκτρολόγιο, τη μετατρέπει
σε ακέραιο αριθμό, τον οποίο και εμφανίζει κάνοντας και τον έλεγχο για πιθανή
εξαίρεση με την κλήση της μεθόδου readInteger()
*/

    static int readInteger(){
     String line;
     // Δημιουργία Αντικειμένου για Μετατροπή Εισόδου σε stream Χαρακτήρων
        BufferedReader br = new BufferedReader(new
                          InputStreamReader(System.in));
        try {
        // Εισαγωγή Συμβολοσειράς
        line = br.readLine();
        // Μετατροπή string Εισόδου σε Ακέραιο
        int i = Integer.parseInt( line );
        return i;
        }
        catch ( Exception obj){
            System.out.println("Σφάλμα Εισόδου - Εξόδου");
            return -1;
        }
       }

    public static void main(String[] args) {

        int number;

        System.out.print("Δώσε έναν ακέραιο αριθμό : ");

        // Κλήση της μεθόδου readInteger()
        number =  readInteger();

        // Εμφάνιση Τιμής Ακεραίου Μετά τη Μετατροπή
        System.out.println("Ο αριθμός που έδωσες είναι : " + number);
        }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε έναν ακέραιο αριθμό : 12345
Ο αριθμός που έδωσες είναι : 12345
BUILD SUCCESSFUL (total time: 4 seconds)

run:
Δώσε έναν ακέραιο αριθμό : 123.45
Σφάλμα Εισόδου - Εξόδου
Ο αριθμός που έδωσες είναι : -1
BUILD SUCCESSFUL (total time: 5 seconds)

run:
Δώσε έναν ακέραιο αριθμό : kostas
Σφάλμα Εισόδου - Εξόδου
Ο αριθμός που έδωσες είναι : -1
BUILD SUCCESSFUL (total time: 4 seconds)
```

10.3.4 Κλάση με static Μεθόδους Μετατροπής String σε Δεδομένα Απλών Τύπων και Έλεγχο για Πιθανή Εξαίρεση

Να γίνει μια κλάση, η οποία θα περιέχει μεθόδους για εισαγωγή δεδομένων οποιουδήποτε απλού τύπου από το πληκτρολόγιο. Να χρησιμοποιηθούν οι μετατροπείς και να γίνει έλεγχος για εξαιρέσεις.

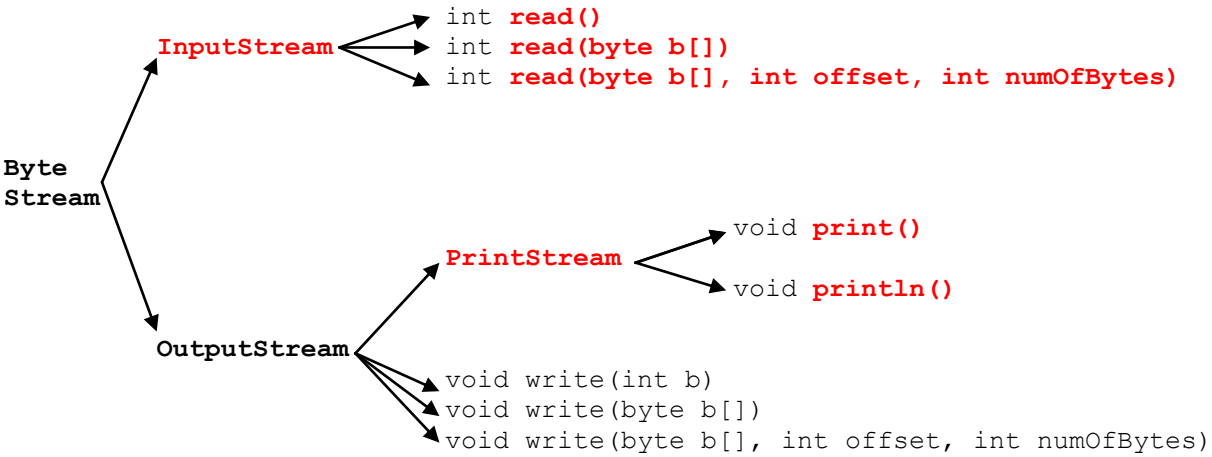
Κλάση `UserInput`

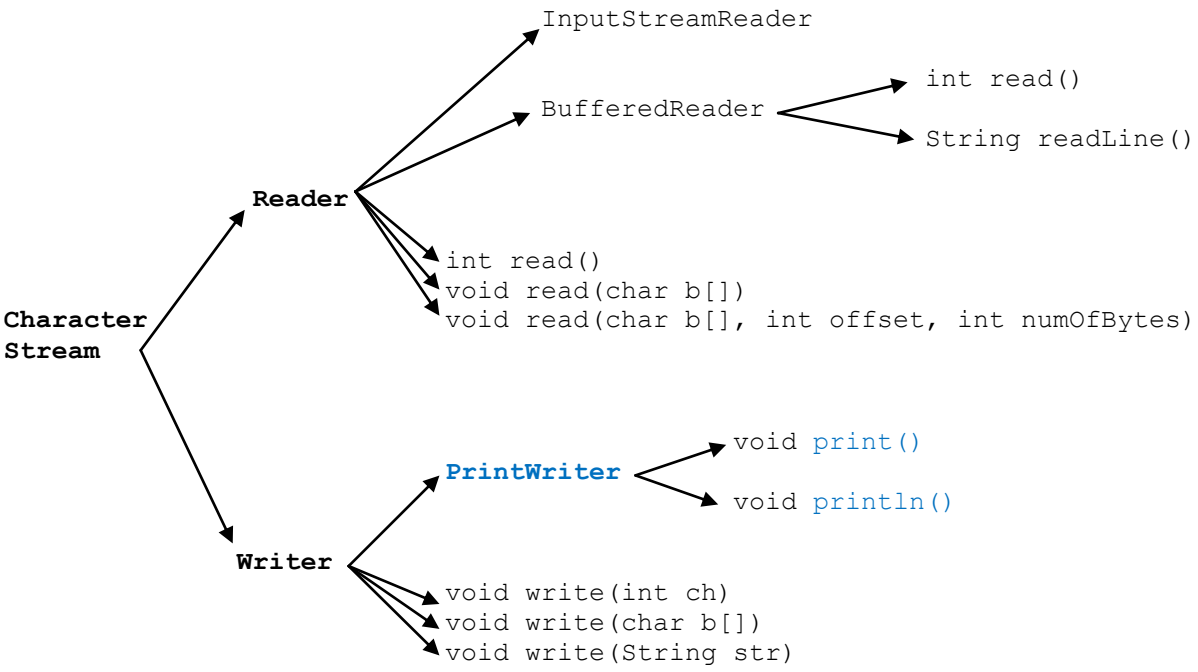
```
import java.io.*;

class UserInput { //Class gia eisagogi dedomenwn apo to pliktrologio
    static String getString() { //Methodos gia eisagogi String
          String line;
          InputStreamReader input = new InputStreamReader(System.in);
          BufferedReader in = new BufferedReader(input);
          try
          {
                line = in.readLine();
                return line;
          }
          catch(Exception e)
          {
                return "Exception";
          }
    }

    static byte getByte() { //Methodos gia eisagogi short
          String line;
          InputStreamReader input = new InputStreamReader(System.in);
          BufferedReader in = new BufferedReader(input);
          try{
                line = in.readLine();
                byte b = Byte.parseByte(line);
                return b;
          }
          catch(Exception e){
                return -1;
          }
     }
```

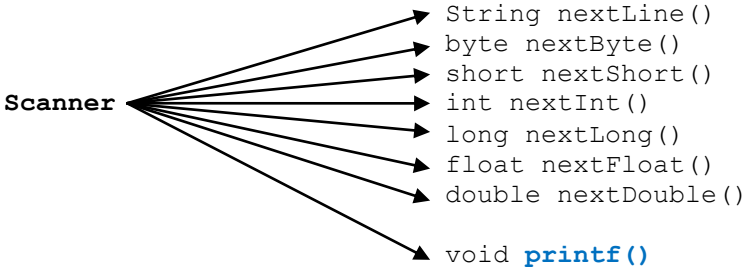
```
static short getShort() { //Methodos gia eisagogi short
      String line;
      InputStreamReader input = new InputStreamReader(System.in);
      BufferedReader in = new BufferedReader(input);
      try{
            line = in.readLine();
            short s = Short.parseShort(line);
            return s;
      }
      catch(Exception e){
            return -1;
      }
 }



static int getInteger() {//Methodos gia eisagogi Integer
      String line;
      InputStreamReader input = new InputStreamReader(System.in);
      BufferedReader in = new BufferedReader(input);
      try{
            line = in.readLine();
            int i = Integer.parseInt(line);
            return i;
      }
    catch(Exception e){
      return -1;
    }
 }



static short getLong() { //Methodos gia eisagogi long
      String line;
      InputStreamReader input = new InputStreamReader(System.in);
      BufferedReader in = new BufferedReader(input);
      try{
            line = in.readLine();
            long l = Long.parseLong(line);
            return l;
      }
      catch(Exception e){
            return -1;
      }
 }



static float getFloat() { //Methodos gia eisagogi float
      String line;
      InputStreamReader input = new InputStreamReader(System.in);
      BufferedReader in = new BufferedReader(input);
      try{
            line = in.readLine();
            float f = Float.parseFloat(line);
            return f;
      }
      catch(Exception e){
            return -1;
      }
 }
```

```
    static double getDouble() { //Methodos gia eisagogi double
          String line;
          InputStreamReader input = new InputStreamReader(System.in);
          BufferedReader in = new BufferedReader(input);
          try{
                line = in.readLine();
                double d = Double.parseDouble(line);
                return d;
          }
          catch(Exception e){
                return -1;
          }
     }



    static boolean getBoolean() { //Methodos gia eisagogi String
          String line;
          InputStreamReader input = new InputStreamReader(System.in);
          BufferedReader in = new BufferedReader(input);
          try
          {
                line = in.readLine();
                return (Boolean.valueOf(line)).booleanValue();
          }
          catch(Exception e)
          {
                return false;
          }
    }

}
```

10.4 Σχεδιάγραμμα Κλάσεων, Υποκλάσεων και Μεθόδων για τις κλάσεις Byte Streams, Streams Χαρακτήρων και Scanner







System.in – Είσοδος - Πληκτρολόγιο – Αντικείμενο Τύπου **InputStream**
System.out – Έξοδος - Οθόνη – Αντικείμενο Τύπου **PrintStream**

10.5 Είσοδος – Έξοδος Δεδομένων με τη Χρήση της Κλάσης Scanner

Η Κλάση **Scanner** παρέχει έναν πιο απλό τρόπο για είσοδο δεδομένων απ' το πληκτρολόγιο και έναν πιο ελεγχόμενο τρόπο εμφάνισης του περιεχομένου κάποιων μεταβλητών. Για να τη χρησιμοποιήσουμε, θα πρέπει πρώτα να την εισάγουμε με την εντολή

```
import java.util.Scanner;
```

Μετά, μπορούμε να συνδέσουμε με το πληκτρολόγιο, μέσω του `System.in`, ένα αντικείμενο τύπου **Scanner** με την εντολή

```
Scanner <object>  = new Scanner(System.in);
```

Το αντικείμενο `<object>` τύπου `Scanner` συνδέεται και με τις αντίστοιχες μεθόδους. Οι μέθοδοι που διαθέτει η κλάση `Scanner` για εισαγωγή δεδομένων είναι :

Μέθοδος	**Αποτέλεσμα**
`nextLine()`	Διάβασμα Συμβολοσειράς
`nextByte()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `byte`
`nextShort()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `short`
`nextInt()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `int`
`nextLong()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `long`
`nextFloat()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `float`
`nextDouble()`	Διάβασμα Αριθμού, Αποθήκευση σε Μεταβλητή `double`

Η μέθοδος για την εμφάνιση των δεδομένων (εκτός των μεθόδων `print()` και `println()`) είναι η μέθοδος `printf()`, η οποία, σε αντίθεση με τις 2 προηγούμενες μεθόδους εμφάνισης, δίνει τη δυνατότητα στο χρήστη να επιλέξει τον **τρόπο εμφάνισης** των δεδομένων που είναι αποθηκευμένα σε κάποιες μεταβλητές, των οποίων το περιεχόμενο θέλει να εμφανίσει. Η σύνταξή της διαφέρει από τη μέθοδο `println()`. Αντί να υπάρχουν **μηνύματα** και **ονόματα μεταβλητών** συνδεδεμένα με το '+', όπως στην εντολή `System.out.println()`, στην εντολή `System.out.printf()` υπάρχει μια συμβολοσειρά με **μηνύματα** και **κωδικούς εμφάνισης** των τιμών των μεταβλητών και ακολουθεί η **λίστα των μεταβλητών**, τις τιμές των οποίων θέλουμε να εμφανίσουμε, **χωρισμένες με κόμμα**. Ο κωδικός εμφάνισης των τιμών μεταβλητών τύπου `byte`, `short`, `int` και `long` είναι `%d`, ενώ ο κωδικός εμφάνισης των τιμών μεταβλητών τύπου `float` και `double` είναι `%f`. Βάζοντας έναν ακέραιο αριθμό μεταξύ του `%` και του `d`, στον κωδικό `%d`, ο ακέραιος αριθμός που περιέχει η μεταβλητή που αντιστοιχεί σ' αυτό τον κωδικό θα καταλάβει τόσες θέσεις στην οθόνη, όσες και ο αριθμός στον κωδικό `%d`. Αν ο ακέραιος αριθμός που περιέχει η μεταβλητή δεν έχει τόσα ψηφία, οι αριστερές θέσεις θα γεμίσουν με τα αντίστοιχα κενά. Π.χ. για τον αριθμό `123` με κωδικό `%5d`, όπου ο αριθμός `5` στον κωδικό `%5d` = <αριθμός_ψηφίων_στην_οθόνη> είναι μεγαλύτερος του `3`, θα εμφανιστεί το `^^123`, όπου `^` = κενό. Αν ο <αριθμός_ψηφίων_στην_οθόνη> είναι μικρότερος των ψηφίων του αριθμού, θα εμφανιστεί ο αριθμός χωρίς κενά. Αντίστοιχα, στον κωδικό `%f` μεταξύ του `%` και του `f`, μπορούν να υπάρχουν `2` αριθμοί χωρισμένοι με τελεία, όπου ο πρώτος δηλώνει τις **συνολικές θέσεις** που θα καταλάβει ο αριθμός κινητής υποδιαστολής στην οθόνη, ενώ ο δεύτερος τον **αριθμό των δεκαδικών ψηφίων**. Π.χ. ο κωδικός `%30.20f` σημαίνει πως ο

αριθμός που περιέχει η μεταβλητή που αντιστοιχεί σ' αυτό τον κωδικό θα καταλάβει 30 θέσεις συνολικά στην οθόνη, απ' τις οποίες οι 20 αντιστοιχούν σε δεκαδικά ψηφία. Αν τα μη μηδενικά δεκαδικά ψηφία του αριθμού είναι λιγότερα από όσα έχουμε προσδιορίσει, οι υπόλοιπες θέσεις γεμίζουν με μηδενικά, ενώ, αν είναι λιγότερα τα ψηφία πριν την υποδιαστολή, οι υπόλοιπες θέσεις γεμίζουν με κενά.

10.5.1 Παράδειγμα Εισόδου – Εξόδου Δεδομένων με τη Χρήση της Κλάσης Scanner

Να γραφεί πρόγραμμα που διαβάζει δεδομένα τύπου byte, short, int, long, float, double και String με τη χρήση της κλάσης Scanner και τα εμφανίζει με τη χρήση της μεθόδου printf().

Πρόγραμμα

```
package scannerclass;
import java.util.Scanner;
public class ScannerClass {
/*  Πρόγραμμα που εισάγει διάφορα δεδομένα με τη χρήση της κλάσης Scanner
    και τα εμφανίζει με τη χρήση της μεθόδου printf() */
    public static void main(String[] args) {
    // Δημιουργία αντικειμένου τύπου Scanner
    Scanner ob = new Scanner(System.in);

    // Εισαγωγή Δεδομένων διαφόρων τύπων
    System.out.print("Δώσε ένα Όνομα - Μέθοδος nextLine() : " );
    String name1 = ob.nextLine();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή byte : ");
    byte b = ob.nextByte();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή short : ");
    short s=ob.nextShort();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή int : ");
    int i=ob.nextInt();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή long : ");
    long l=ob.nextLong();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή float : ");
    float f = ob.nextFloat();
    System.out.print("Δώσε έναν αριθμό για μεταβλητή double : ");
    double d = ob.nextDouble();

    // Εμφάνιση τιμής μεταβλητής String με τη χρήση της μεθόδου println()
    System.out.println("\nΤο Όνομα με τη nextLine() είναι :" + name1 );

    // Εμφάνιση τιμών υπολοίπων μεταβλητών με τη χρήση της μεθόδου printf()
    System.out.printf("\nΕμφάνιση τιμών υπολοίπων μεταβλητών με τη χρήση της
    μεθόδου printf()\n");
    System.out.printf("Η τιμή της byte Μεταβλητής με format 5d είναι :
    %5d\n", b);
    System.out.printf("Η τιμή της short Μεταβλητής με format 10d είναι :
    %10d\n", s);
    System.out.printf("Η τιμή της int Μεταβλητής με format 15d είναι :
    %15d\n", i);
    System.out.printf("Η τιμή της long Μεταβλητής με format 20d είναι :
    %20d\n", l);
    System.out.printf("Η τιμή της float Μεταβλητής με format 18.16f είναι :
    %18.16f\n", f);
    System.out.printf("Η τιμή της double Μεταβλητής με format 30.20f είναι :
    %30.20f\n", d);
     }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα Όνομα - Μέθοδος nextLine() : Georgios Georgiou
Δώσε έναν αριθμό για μεταβλητή byte : 12
Δώσε έναν αριθμό για μεταβλητή short : 123
Δώσε έναν αριθμό για μεταβλητή int : 1234
Δώσε έναν αριθμό για μεταβλητή long : 123456789
Δώσε έναν αριθμό για μεταβλητή float : 123,45
Δώσε έναν αριθμό για μεταβλητή double : 123456789,12345

Το Όνομα με τη nextLine() είναι :Georgios Georgiou

Εμφάνιση τιμών υπολοίπων μεταβλητών με τη χρήση της μεθόδου printf()
Η τιμή της byte Μεταβλητής με format 5d είναι :    12
Η τιμή της short Μεταβλητής με format 10d είναι :        123
Η τιμή της int Μεταβλητής με format 15d είναι :            1234
Η τιμή της long Μεταβλητής με format 20d είναι :             123456789
Η τιμή της float Μεταβλητής με format 18.16f είναι : 123.4499969482421900
Η τιμή της double Μεταβλητής με format 30.20f είναι :
123456789.12345000000000000000
BUILD SUCCESSFUL (total time: 44 seconds)
```

10.5.2 Παράδειγμα Δημιουργίας Μεθόδου Εισόδου Δεδομένων τύπου int

Να γραφεί πρόγραμμα που να καλεί τη μέθοδο `scanInteger()` της κλάσης `ScannerUserInput`, η οποία **διαβάζει** έναν αριθμό τύπου `int` με τη χρήση της κλάσης `Scanner` και τον **εμφανίζει** με τη χρήση της μεθόδου `printf()`.

Κλάση ScannerInput - main()

```
package scannerinput;
public class ScannerInput {
    public static void main(String[] args) {
        System.out.print("Δώσε έναν ακέραιο αριθμό : ");
        int i = ScannerUserInput.scanInteger();
        System.out.printf("Ο ακέραιος αριθμός είναι : : %15d\n", i);
    }
}
```

Κλάση ScannerUserInput – Μέθοδος scanInteger()

```
package scannerinput;
import java.util.Scanner;
public class ScannerUserInput {

    static int scanInteger(){
    // Δημιουργία αντικειμένου τύπου Scanner
    Scanner ob = new Scanner(System.in);
      try{
        int i = ob.nextInt();
        return i;
      }
      catch(Exception e){
          return -1;
      }
}
}
```

Έξοδος Προγράμματος

```
run:
Δώσε έναν ακέραιο αριθμό : 12345
Ο ακέραιος αριθμός είναι : :           12345
BUILD SUCCESSFUL (total time: 5 seconds)

run:
Δώσε έναν ακέραιο αριθμό : 5.0
Ο ακέραιος αριθμός είναι : :              -1
BUILD SUCCESSFUL (total time: 3 seconds)

run:
Δώσε έναν ακέραιο αριθμό : abc
Ο ακέραιος αριθμός είναι : :              -1
BUILD SUCCESSFUL (total time: 6 seconds)
```

ΑΣΚΗΣΗ 10.1

Με τη χρήση της **Κλάσης `Scanner`,** να συμπληρωθεί η κλάση **`ScannerUserInput`** με τις υπόλοιπες μεθόδους για εισαγωγή δεδομένων οποιουδήποτε απλού τύπου από το πληκτρολόγιο κάνοντας και τον έλεγχο για εξαιρέσεις.