Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programming)

Wrapper Classes – Λίστες *(Lists)* - ArrayList Τοποθέτηση Δεδομένων – Κλήση Μεθόδων

Παναγιώτης Σφέτσος, PhD
http://aetos.it.teithe.gr/~sfetsos/
sfetsos@it.teithe.gr

Περιεχόμενα Μαθήματος

- **Wrapper Classes**
- **Λίστες (Lists) - ArrayList**
- **Τοποθέτηση Δεδομένων**
- **Κλήση Μεθόδων**

Wrapper Classes *(1/12)*

Στο πακέτο: java.lang

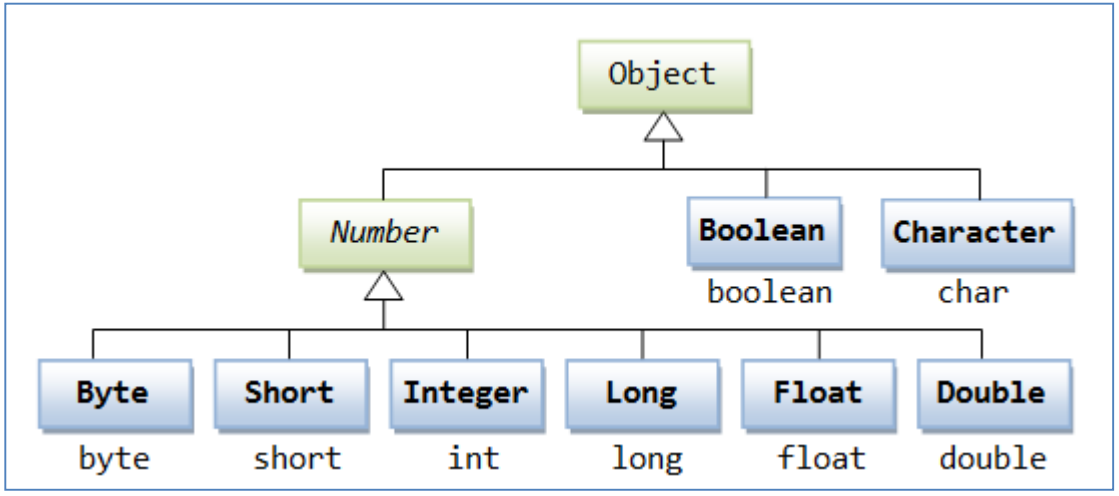
- Για καθένα από τους 8 – βασικούς τύπους δεδομένων υπάρχει μια αντίστοιχη προκαθορισμένη κλάση (Integer, Double, κλπ.). Οι κλάσεις αυτές ονομάζονται **wrapper classes** γιατί ***περικλείουν ένα βασικό τύπο***, έτσι ώστε μια μεταβλητή να μπορεί να ***αναπαρασταθεί ως αντικείμενο*** του αντίστοιχου τύπου.

```
int i = 43;   //στην int μεταβλητή - i δίνουμε την τιμή 43
Integer x = new Integer(53);  //δημιουργία αντικειμένου τύπου Integer με τιμή 53
                              //που ανατίθεται στη μεταβλητή x (αναφορά στο αντικείμενο)
```

- **Δύο κύριοι σκοποί:**
  - Να παρέχει μηχανισμό δημιουργίας αντικειμένων βασικών τύπων για είσοδο σε: ArrayList, Vector, HashMap, κλπ.
  - Να παρέχει χρήσιμα ***χαρακτηριστικά*** και ***λειτουργίες*** όπως τις MAX_VALUE, MIN_VALUE, Integer.parseInt(), Float.parseFloat(), κλπ.
  - Με τις parseX-type(), μετατρέπονται αριθμητικά Strings σε πραγματικούς αριθμούς.

Wrapper Classes *(2/12)*

Από το Java API: Ιεραρχία των Wrapper Κλάσεων



- Όλες οι wrapper κλάσεις (εκτός της Character) δέχονται το ***όρισμα του δομητή*** υπό μορφή String. Π.χ.

  Integer obj = new Integer(32);

  Integer obj1 = new Ineger(“32”);

Wrapper Classes *(3/12)*

- Η μέθοδος valueOf() των Wrapper κλάσεων δέχεται μια τιμή ή String και επιστρέφει ένα αντικείμενο της κλάσης:

- **Integer i1 = Integer.valueOf(13);**
- **Integer i2 = Integer.valueOf(“13”);**

- **Boolean b1 = Boolean. valueOf(true);**
- **Boolean b2 = Boolean. valueOf(“true”);**

- **Long l1 = Long.valueOf(36500000L);**
- **Long l1 = Long.valueOf(“36500000L”);**

Wrapper Classes *(4/12)*

Βασικός Τύπος	Wrapper Class	Μέθοδος Μετατροπής
byte	Byte	public static byte parseByte(String)
short	Short	public static short parseShort(String)
int	Integer	public static integer parseInt(String)
long	Long	public static long parseLong(String)
float	Float	public static float parseFloat(String)
double	Double	public static double parseDouble(String)
char	Character	
boolean	Boolean	public static boolean parseBoolean(String)

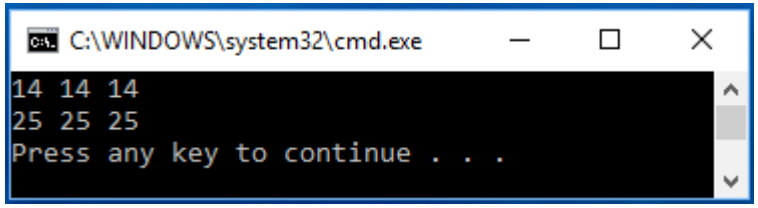
Wrapper Classes *(5/12)*

Μερικές σημαντικές μέθοδοι της Wrapper class - Integer

Βασικός Τύπος	Σκοπός
parseInt(s)	Επιστρέφει την int - τιμή του αριθμητικού String s.
toString(i)	Επιστρέφει ένα String αντικείμενο του int - i.
byteValue()	Επιστρέφει την τιμή του int σαν byte.
doubleValue()	Επιστρέφει την τιμή του int σαν double.
floatValue()	Επιστρέφει την τιμή του int σαν float.
shortValue()	Επιστρέφει την τιμή του int σαν short.
longValue()	Επιστρέφει την τιμή του int σαν long.
int compareTo(int i)	Συγκρίνει την αρ. τιμή του αντικειμένου με αυτήν του i. Επιστρέφει 0 αν οι τιμές είναι ίσες, θετική τιμή αν το αντικείμενο έχει μεγαλύτερη τιμή, ή αρνητική τιμή αν το αντικείμενο έχει μικρότερη τιμή.
static int compare(int n1, int n2)	Συγκρίνει όπως και πριν αλλά τώρα τις τιμές n1 και n2. Επιστρέφει όπως και άνω 0, θετική ή αρνητική τιμή.
boolean equals(Object iObj)	Επιστρέφει true αν το Integer αντικείμενο είναι ίσο με το αντικείμενο – iObj.

Wrapper Classes *(6/12)*

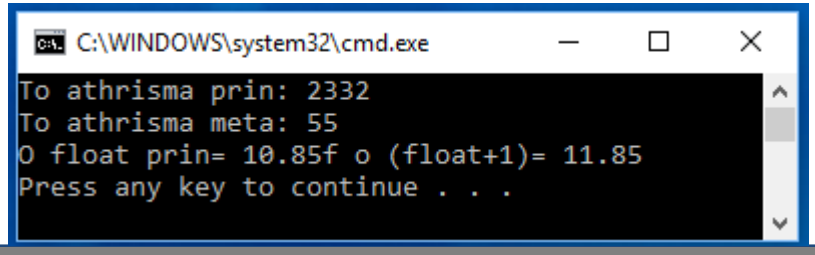
```
class WrapperExample3{
  public static void main(String args[]){
    //Metatropi int se Integer
    int x=14;
    Integer y=Integer.valueOf(x);//Metatropi int se Integer
    //autoboxing, o compiler ektelei esoterica Integer.valueOf(a)
    Integer z=x;
    System.out.println(x+" "+y+" "+z);

    // Metatropi Integer se int
   Integer c=new Integer(25);
   int g=c.intValue();// Metatropi Integer se int
    //unboxing, o compiler ektelei esoterica Integer.valueOf(a)
    int h=c;
    System.out.println(c+" "+g+" "+h); }}
```



Wrapper Classes *(7/12)*

```
class WraperDemo1 {
  public static void main(String[] args) {
    String  s[] = {"23", "32"};
    //concatenation 2332
    System.out.println("To athrisma prin: "+ s[0] + s[1]);
    //metatropi tou arithmitikou String se Integer
    int x=Integer.parseInt(s[0]);
    //metatropi tou arithmitikou String se Integer
    int y=Integer.parseInt(s[1]);
    int z=x+y;
    System.out.println("To athrisma meta: "+z);
    String k="10.85f";
    float f=Float.parseFloat(k);
    System.out.println("O float prin= " + k + " o (float+1)=
"+(f+1));}}
```

Τι θα συμβεί με αυτά τα Strings;
"18y"
"DEF"
"R"

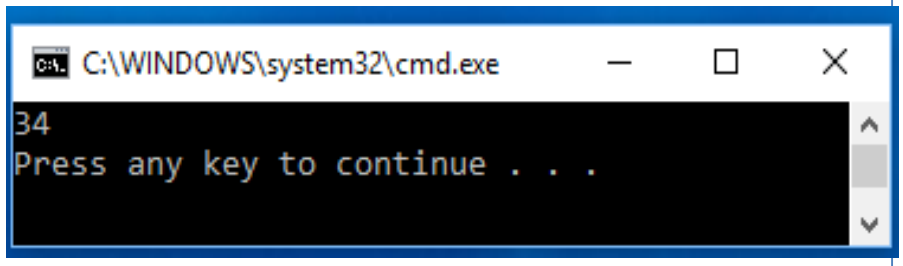


Wrapper Classes *(8/12)*

Java Autoboxing και Unboxing

- **Autoboxing:** Αυτόματη μετατροπή βασικού τύπου στην αντίστοιχη wrapper class.

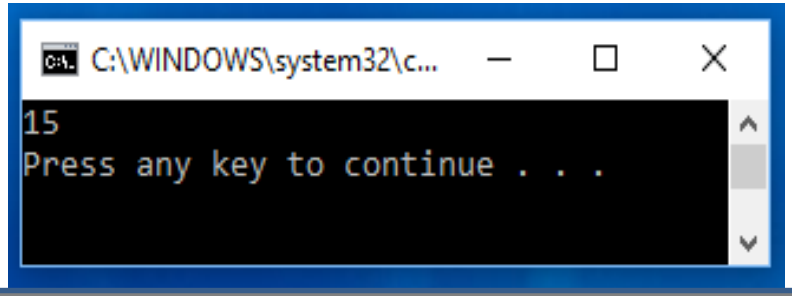
```
class AutoboxingExample1
{
  public static void AMethod(Integer num){
      System.out.println(num);
   }

public static void main(String[] args) {
/* H parametros pou einai aplos vasikos typos(34) tha
 * metatrapei se Integer antikeimeno kata tin ektelesi
 * tou programmatos(Runtime) */
  AMethod(34);
   }
}
```



Wrapper Classes *(9/12)*

- **Unboxing:** Αυτόματη μετατροπή αντικειμένου μιας wrapper class στον αντίστοιχο βασικό τύπο.

```
class UnboxingExample1 {
  public static void AMethod(int num){
    System.out.println(num); }

public static void main(String[] args) {
    Integer x = new Integer(15);

    /* H parametros pou einai Integer antikeimeno tha
    *  metatrapei ston antistoixo aplo vasiko typo int 15
    *  kata tin ektelesi tou programmatos(Runtime) */
    AMethod(x);
  }
}
```



Wrapper Classes *(10/12)*

- **Παραδείγματα Autoboxing - Unboxing:**

```
Integer x = 25; //Autoboxing (int σε Integer)
Long y = 32L;   //Autoboxing (long σε Long)

ArrayList<Integer> ar = new ArrayList<Integer>();
ar.add(28); //Autoboxing: int σε Integer
ar.add(35); //Autoboxing: int σε Integer

Integer i = new Integer(5);
int x = i;  //Unboxing: αντικείμενο σε βασικό τύπο – int

Integer y = new Integer(5);
int x = y.intValue(); //Unboxing: αντικείμενο σε βασικό τύπο – int
```

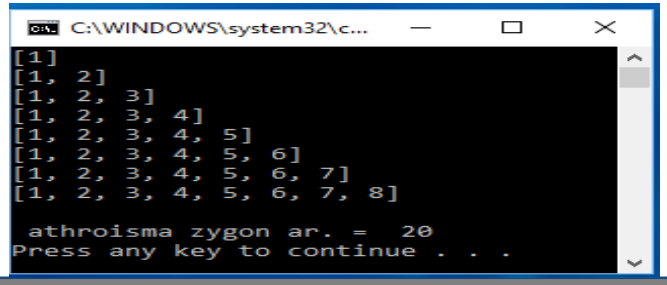
Wrapper Classes *(11/12)*

- **Παραδείγματα Autoboxing - Unboxing:**

```
import java.util.*;
class WrapperExample4 {
  public static void main(String args[]) {
   ArrayList<Integer> ar = new ArrayList<Integer>();
   for (int i = 1; i < 9; i ++){
     ar.add(i); //o compiler ektelei esoterika ar.add(Integer.valueOf(i));
     System.out.println(ar);}
   int sum = WrapperExample4.sumZyga(ar);
   System.out.println("\n athroisma zygon ar. =  " + sum); }

 public static int sumZyga(ArrayList<Integer> ar) {
    int s = 0;
    for (Integer i : ar)
     if (i % 2 == 0) s += i; //o compiler ektelei esoterika [s += i.intValue();]
        return s; } }
```

Autoboxing

Unboxing



Wrapper Classes *(12/12)*

Παραδείγματα με τις μεθόδους: compareTo(), equals(), compare()

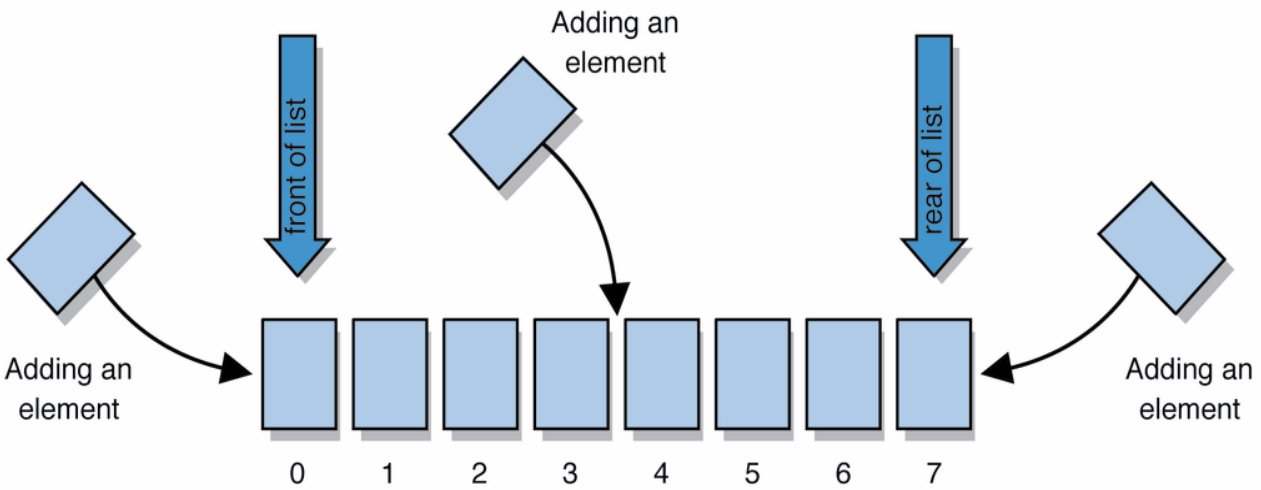
```
class WrapperDemo2 {
  public static void main(String args[]){
   Integer Obj1 = new Integer(32);
   Integer Obj2 = new Integer("32");
   Integer Obj3= new Integer(45);



   System.out.println("compareTo: Obj1= "+Obj1+" Obj2= " + Obj2+", apotelesma= "+
                                                  Obj1.compareTo(Obj2));
   System.out.println("compareTo: Obj1= "+Obj1+" Obj3= " + Obj3+", apotelesma= " +
                                                  Obj1.compareTo(Obj3));
   System.out.println("equals: Obj1= "+Obj1+ " Obj2= "+Obj2+ ", apotelesma= " +
                                                  Obj1.equals(Obj2));
   System.out.println("equals: Obj1= "+Obj1+ " Obj3= "+Obj3+ ", apotelesma= "  +
                                                  Obj1.equals(Obj3));
   Float f1 = new Float("2.25f");
   Float f2 = new Float("20.43f");
   Float f3 = new Float(2.25f);
   System.out.println("compare: ton f1= "+f1+ " kai f2= "+f2+", apotelesma= "
                                                 +Float.compare(f1,f2));
   System.out.println("compare: ton f1= "+f1+ " kai f3= "+f3+", apotelesma= "
                                                 +Float.compare(f1,f3));
   Float f = Obj1.floatValue() + f1;
   System.out.println("Prosthesi: Obj1= "+Obj1+ " kai f1= "+ f1+ ", apotelesma= " +f);}}
```

```
C:\WINDOWS\system32\cmd.exe
compareTo: Obj1= 32 Obj2= 32, apotelesma= 0
compareTo: Obj1= 32 Obj3= 45, apotelesma= -1
equals: Obj1= 32 Obj2= 32, apotelesma= true
equals: Obj1= 32 Obj3= 45, apotelesma= false
compare: ton f1= 2.25 kai f2= 20.43, apotelesma= -1
compare: ton f1= 2.25 kai f3= 2.25, apotelesma= 0
Prosthesi: Obj1= 32 kai f1= 2.25, apotelesma= 34.25
Press any key to continue . . .
```

Λίστες - Lists

Λίστα : μια συλλογή *(collection)* από διατεταγμένα στοιχεία.

- Κάθε στοιχείο ανιχνεύεται από ένα **δείκτη** (0 έως n-1)
- Η λίστα έχει δυναμικά αυξανόμενο **μέγεθος** *(size)* – το πλήθος των στοιχείων της
- Τα στοιχεία μπορούν να **εισαχθούν/διαγραφούν** σε/από **οποιαδήποτε θέση** (αρχή, μέση, τέλος, κλπ.).
- Μια σημαντική λίστα είναι το αντικείμενο **ArrayList**



ArrayList (1/4)

ArrayList

Είναι ένας **δυναμικός χώρος μνήμης** για συλλογή (*collection*) διατεταγμένων αντικειμένων/στοιχείων.

- Ακολουθεί την generics – φιλοσοφία.
- Μπορεί να οριστεί με αρχικό μέγεθος, να **αυξάνει** (προσθήκη στοιχείων) ή να **μειώνεται** ανάλογα (διαγραφή στοιχείων).
- Πρέπει να καθορίζεται ο τύπος των στοιχείων της λίστας μέσα στις τριγωνικές αγκύλες <>:

```
ArrayList<Type> name = new ArrayList<Type>();
```

π.χ.

```
ArrayList<String> list = new ArrayList<String>(20);
```

ArrayList (2/4)

- Ο τύπος των στοιχείων πρέπει να είναι τύπος – object (αντικείμενα wrapper κλάσεων) και όχι βασικός τύπος,

  δηλ., δεν μπορεί να υπάρξει ένας τέτοιος ορισμός:

  ```
  ArrayList<int> list = new ArrayList<int>();  // Λάθος ορισμός
  ```

- Μπορούμε να έχουμε βασικούς τύπους χρησιμοποιώντας αντικείμενα των κλάσεων των βασικών τύπων (wrapper classes), δηλ.:

  ```
  ArrayList<Integer> list = new ArrayList<Integer>();
  ```

Wrapper classes

- Όπως αναλύσαμε στις προηγούμενες διαφάνειες: ***τα αντικείμενα αυτών των κλάσεων κρατούν τιμές των βασικών τύπων.***

ArrayList (3/4)

- Μετά τον ορισμό της λίστας με τύπο – wrapper, μπορούμε να χειριστούμε τις βασικές τιμές με μεθόδους και κώδικα:

```
ArrayList<Double> vathmoi = new ArrayList<Double>();
vathmoi.add(3.2);
vathmoi.add(2.7);
...
```

- Μια μέθοδος μπορεί να δεχτεί σαν παράμετρο μια ArrayList ή να την επιστρέψει σαν αποτέλεσμα από μια μέθοδο (return):

public static void name(ArrayList<Type> name)

Παράδειγμα: Μέθοδος που διαγράφει τους ζυγούς αριθμούς από την λίστα list.

ArrayList (4/4)

```
public static void DiagrafiZygonArithmon(ArrayList<Integer> list) {
    for (int i = list.size() - 1; i >= 0; i--) {
        int n = list.get(i);
        if (n % 2 == 0) {
            list.remove(i); } }
  }
```

Τρείς διαφορετικοί δομητές:

ArrayList() - μια κενή λίστα.
ArrayList(Collection c) – η λίστα αρχικοποιείται με τα αντικείμενα της συλλογής c.
ArrayList(int capacity) – η λίστα θα έχει αρχικό μέγεθος – capacity.

- Πρέπει να εισάγεται (import) το πακέτο java.util **(import java.util.*)** ή μόνο το πακέτο ArrayList **(import java.ArrayList;)**

Σημαντικές Μέθοδοι της ArrayList (1/2)

boolean add(Object o)	Προσθέτει το αντικείμενο - ο στο τέλος της λίστας
void add(int index, Object o)	Εισάγει το αντικείμενο-ο στην θέση index
void clear()	Διαγράφει όλα τα στοιχεία της λίστας
int indexOf(Object o)	Επιστρέφει την θέση της πρώτης θέσης εύρεσης του στοιχείου (-1 αν δεν το βρει)
Object get(int index)	Επιστρέφει το στοιχείο της συγκεκριμένης θέσης
Object remove(int index)	Διαγράφει το στοιχείο της συγκεκριμένης θέσης
Object set(int index, Object element)	Αντικαθιστά το στοιχείο στην συγκεκριμένη θέση με το συγκεκριμένο στοιχείο
int size()	Επιστρέφει το πλήθος των στοιχείων της λίστας
String toString()	Επιστρέφει ένα String που αντιπροσωπεύει την λίστα με την μορφή "[3, 42, -7, 15]"
boolean addAll(Collection c) **boolean addAll**(index, Collection c)	Προσθέτει όλα τα στοιχεία της συλλογής c στο τέλος της λίστας ή τα εισάγει στην συγκεκριμένη θέση

Σημαντικές Μέθοδοι της ArrayList (2/2)

boolean contains(Object o)	Επιστρέφει true αν η λίστα περιέχει το στοιχείο
boolean containsAll(Collection c)	Επιστρέφει true αν η λίστα περιέχει όλα τα στοιχεία της συλλογής c
boolean equals(Collection c)	Επιστρέφει true αν η συλλογή c περιέχει τα στοιχεία της λίστας
int lastIndexOf(Object o)	Επιστρέφει την τελευταία θέση του στοιχείου στη λίστα (-1 αν δεν το βρει)
Object remove(int index)	Διαγράφει το στοιχείο στην συγκεκριμένη θέση από την λίστα
protected void removeRange(int fromIndex, int toIndex)	Διαγράφει τα στοιχεία της λίστας που βρίσκονται στα όρια fromIndex έως toIndex
Object[] toArray()	Επιστρέφει τα στοιχεία της λίστας σαν πίνακα

.....Περισσότερα στην τεκμηρίωση

Χρήση των σημαντικότερων μεθόδων *(1/4)*

Το μέγεθος της ArrayList

- Μέγεθος μιας ArrayList είναι ο συνολικός αριθμός των στοιχείων της.

```
int size = s.size();
```

Εύρεση της θέσης ενός στοιχείου της ArrayList

- Με την χρήση της indexOf():

```
int index = s.indexOf("Skiathos");
  //εύρεση του στοιχείου Skiathos στη λίστα
```

Χρήση της απλής for ή της foreach για την προσπέλαση μιας λίστας

```
for (int i = 0; i < stringList.size(); i++)
    String stoiheio = s.get(i);
    System.out.println("Stoiheio " + i + " : " + stoiheio);  }
```

Χρήση των σημαντικότερων μεθόδων (2/4)

```
for(String stoiheio : s){
        System.out.println("Stoiheio : " + stoiheio); }
```

Έλεγχος αν ένα ArrayList είναι άδειο

- Με δύο τρόπους:
  1) με την μέθοδο **isEmpty():**

```
boolean result = s.isEmpty();
//η isEmpty() επιστρέφει true αν η λίστα είναι άδεια
```

  2) με την μέθοδο **size()**:

```
if(s.size() == 0) {System.out.println("H lista einai adeia"); }
```

Χρήση των σημαντικότερων μεθόδων *(3/4)*

Διαγραφή στοιχείου από μια ArrayList

Με δύο τρόπους:

1) διαγραφή στοιχείου σε **συγκεκριμένη θέση** (όταν την γνωρίζουμε):

π.χ. **s.remove(0);**

2) διαγραφή **συγκεκριμένου στοιχείου**: π.χ. **s.remove(stoiheio);**

Αντιγραφή όλων των στοιχείων μιας λίστας σε μια άλλη

Με την μέθοδο **addAll(collection c),** π.χ.:

```
ArrayList<String> clone1 = new ArrayList<String>();
clone1.addAll(s);
```

Αντικατάσταση ενός στοιχείου σε συγκεκριμένη θέση

Με την μέθοδο **set(int i, O object)**, π.χ.: **s.set(7, "Santorini");**

Χρήση των σημαντικότερων μεθόδων *(4/4)*

Διαγραφή όλων των στοιχείων της λίστας

Με την μέθοδο **clear()**, π.χ.: s.clear();

Δημιουργία μιας ArrayList από απλό Array

Με την μέθοδο **Arrays.asList(T... a),** π.χ.:

```
ArrayList s = Arrays.asList(new String[]{“Skiathos”, “Kriti”, “Halkidiki”, “Santorini”});
//μετά την δημιουργία μπορούμε να αλλάξουμε και τα στοιχεία
```

Μετατροπή μιας ArrayList σε απλό Array

Με την μέθοδο **toArray(T[] a)**, π.χ.:

```
String[] listArray = new String[s.size()];
String[] aploArray = s.toArray(listArray);
```

Προσπέλαση στα στοιχεία της ArrayList με τον Iterator και την While

Η **Iterator** και **ListIterator** έχουν δύο μεθόδους που χρησιμοποιούμε με την while για να προσπελάσουμε τα στοιχεία:

- την μέθοδο **hasNext()**, που επιστρέφει true όσο υπάρχει επόμενο στοιχείο στην λίστα και
- την μέθοδο **next()** που επιστρέφει το επόμενο στοιχείο

```
Iterator<String> iterator = Lista.iterator();
while(iterator.hasNext()){
    System.out.println(iterator.next());
}
ListIterator<String> listIterator = Lista.listIterator();
while(listIterator.hasNext()){
    System.out.println(listIterator.next());
}
```

Διαφορές με τα απλά arrays

Στην **κατασκευή:**

```
String[] names = new String[3];
ArrayList<String> list = new ArrayList<String>(); // ή ArrayList<String>(3)
```

Στην **αποθήκευση τιμών:**

```
names[0] = "Sakis";
list.add("Sakis");
```

Στην **ανάκτηση τιμών:**

```
String s = names[0];
String s = list.get(0);
```

Παραδείγματα

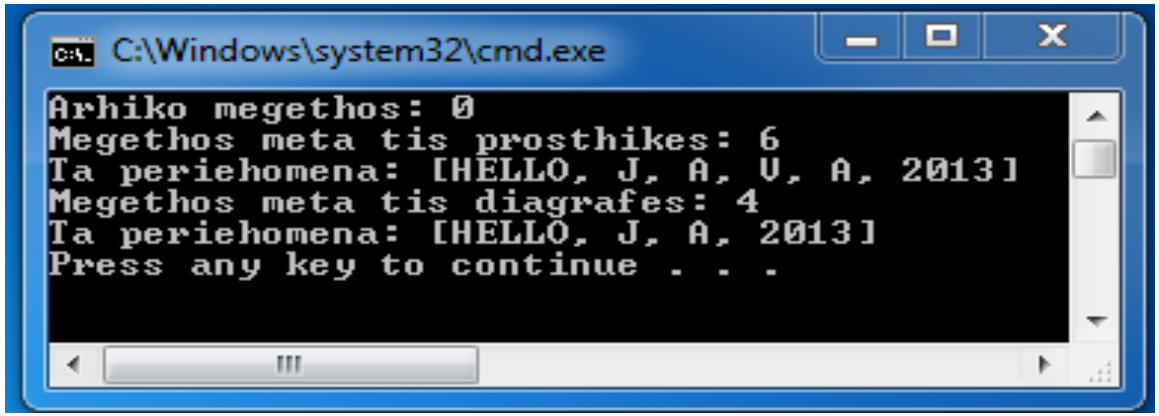
Παράδειγμα απλού ArrayList

Δοκιμάζουμε τις μεθόδους size(), add() και remove(). *Επειδή δεν χρησιμοποιούμε generic - ArrayList θα λάβουμε προειδοποιητικό μήνυμα κατά την μεταγλώττιση.*

```
import java.util.*;
class ArrayList1 {
   public static void main(String args[]) {
      ArrayList al = new ArrayList();
      System.out.println("Arhiko megethos: " + al.size());

      // prosthiki stoiheivn
      al.add("J");
      al.add("A");
      al.add("V");
      al.add("A");
      al.add(0, "HELLO");
```

Παραδείγματα

```
        al.add(new Integer(2013));
        System.out.println("Megethos meta tis prosthikes: " + al.size());
        // emfanisi tou array list
        System.out.println("Ta periehomena: " + al);
        // diagrafi stoiheivn tou array list
        al.remove("V");
        al.remove(2);
        System.out.println("Megethos meta tis diagrafes: " + al.size());
        System.out.println("Ta periehomena: " + al);
    }
}
```

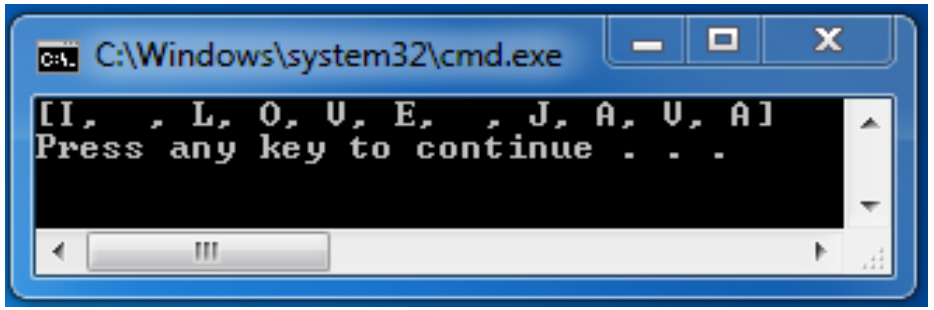


Παραδείγματα

Παράδειγμα generic - ArrayList

Μια πιο **ασφαλής παραλλαγή της ArrayList (generics)**.

```
import java.util.*;
class arraylist2  {
  public static void main(String args[])  {
      ArrayList<String> arr = new ArrayList<String>(10);
       arr.add("I"); arr.add(" "); arr.add("L"); arr.add("O");
      arr.add("V"); arr.add("E"); arr.add(" "); arr.add("J");
       arr.add("A"); arr.add("V"); arr.add("A");
        System.out.println(arr);
    }
  }
```



Διαχείριση της Μνήμης στη Java

- Κατά την εκτέλεση ενός προγράμματος, διαφορετικά μέρη της μνήμης (RAM) δεσμεύονται για κάθε κλάση, διεπαφή, αντικείμενο και μέθοδο. Τα δύο κύρια μέρη της είναι: η **Heap** και η **Stack**.

- Οι τοπικές μεταβλητές 'ζουν' στη **Stack** – μνήμη
  - Ο χώρος δεσμεύεται με την κλήση της μεθόδου
  - Αποδεσμεύεται με την επιστροφή της μεθόδου

- Τα λοιπά δεδομένα 'ζουν' στην **Heap** - μνήμη
  - Ο χώρος δεσμεύεται με την new
  - Αποδεσμεύεται αυτόματα και όχι από εμάς *(garbage collection)*

- Που αποθηκεύονται οι **βασικοί τύποι** και που τα **αντικείμενα των wrapper κλάσεων;**

Η μνήμη - Heap

Java Heap Space:

- Χώρος της μνήμης που χρησιμοποιείται, κατά την εκτέλεση του προγράμματος, για τις **στατικές πληροφορίες** *(classes και interfaces)* για τα **αντικείμενα** *(και τα πεδία τους).*
- Τα αντικείμενα δημιουργούνται πάντα στον χώρο της Heap.
- Η ***Garbage Collection*** *(συλλογή σκουπιδιών)* τρέχει στην Heap – μνήμη για να ελευθερώσει την μνήμη από τα αντικείμενα για τα οποία δεν υπάρχει κάποια αναφορά *(δεν χρησιμοποιούνται πλέον)*. Στα αντικείμενα της Heap έχουμε δημόσια πρόσβαση από οποιοδήποτε σημείο του προγράμματος μας.
- Πιο αργή μνήμη από την Stack, αλλά ο μεταγλωττιστής δεν χρειάζεται να γνωρίζει το μέγεθος και την διάρκεια ζωής των δεδομένων.

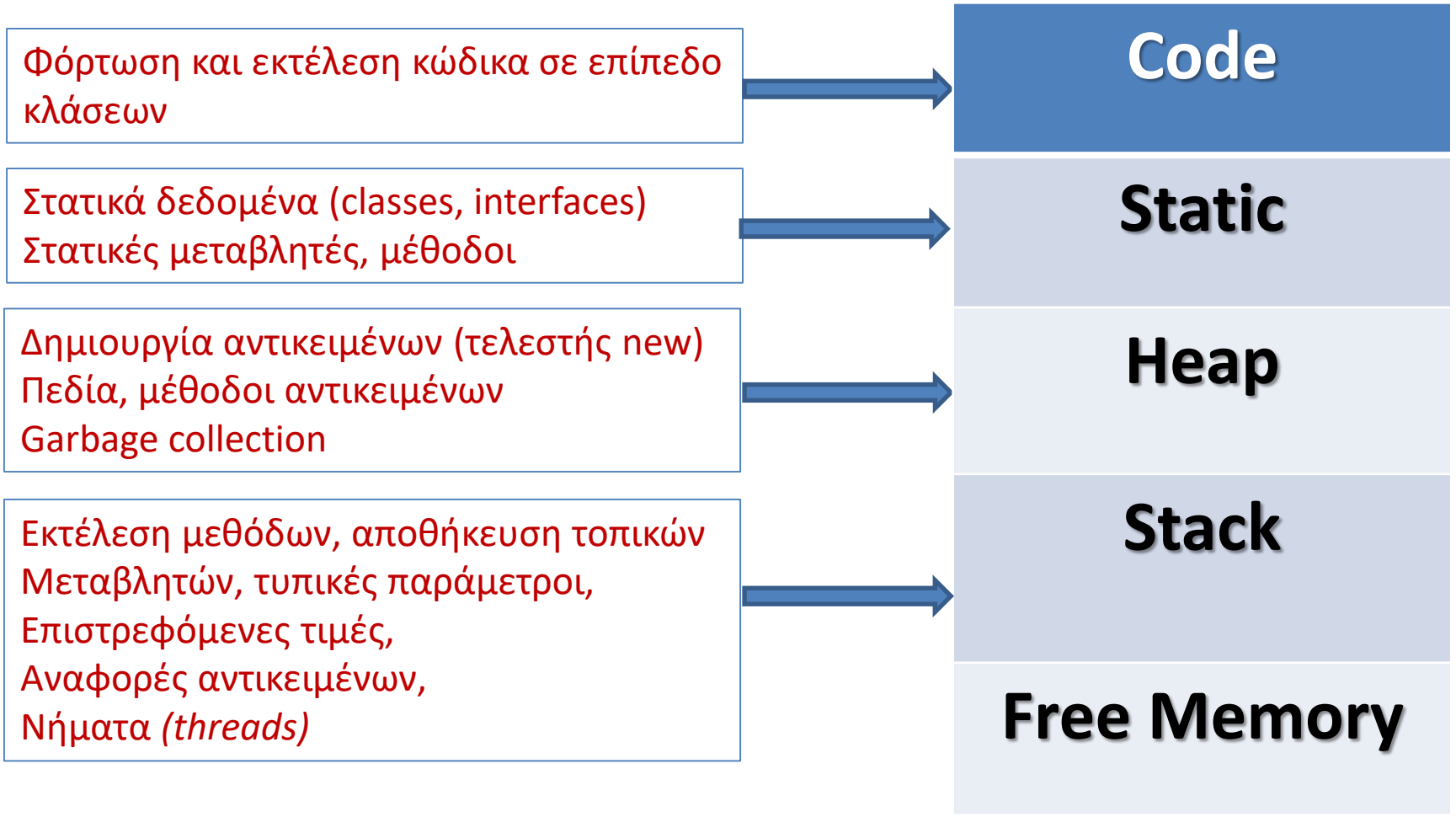
Η μνήμη - Stack

Java Stack Memory:

- Χώρος της μνήμης που χρησιμοποιείται για τις **πληροφορίες** της **μεθόδου**, όταν αυτή εκτελείται. Προσωρινή αποθήκευση των τοπικών μεταβλητών, τυπικών παραμέτρων, επιστρεφόμενη τιμή και που θα επιστρέψει.
- Στη Stack τοποθετούνται οι **αναφορές στα αντικείμενα της Heap** και εκτελούνται τα **νήματα** *(threads)* του προγράμματος.
- Όταν εκτελείται μια μέθοδος, τότε δημιουργείται μια **'εγγραφή ενεργοποίησης'** - ***activation record (AR)*** στη Stack για να αποθηκευτούν οι *τοπικές μεταβλητές* των βασικών τύπων, αλλά και οι *αναφορές σε αντικείμενα της μεθόδου*. Όταν τελειώσει η μέθοδος το κομμάτι ελευθερώνεται για χρήση από άλλη μέθοδο.
- Είναι συγκριτικά **μικρότερη** αλλά **γρηγορότερη** μνήμη από ότι η Heap. Ο επεξεργαστής έχει πρόσβαση σε αυτή την μνήμη μέσω ενός δείκτη, του *stack pointer.*
- Ο μεταγλωττιστής πρέπει να γνωρίζει *τον τύπο, το μέγεθος και την διάρκεια ζωής των δεδομένων* που θα αποθηκευτούν.

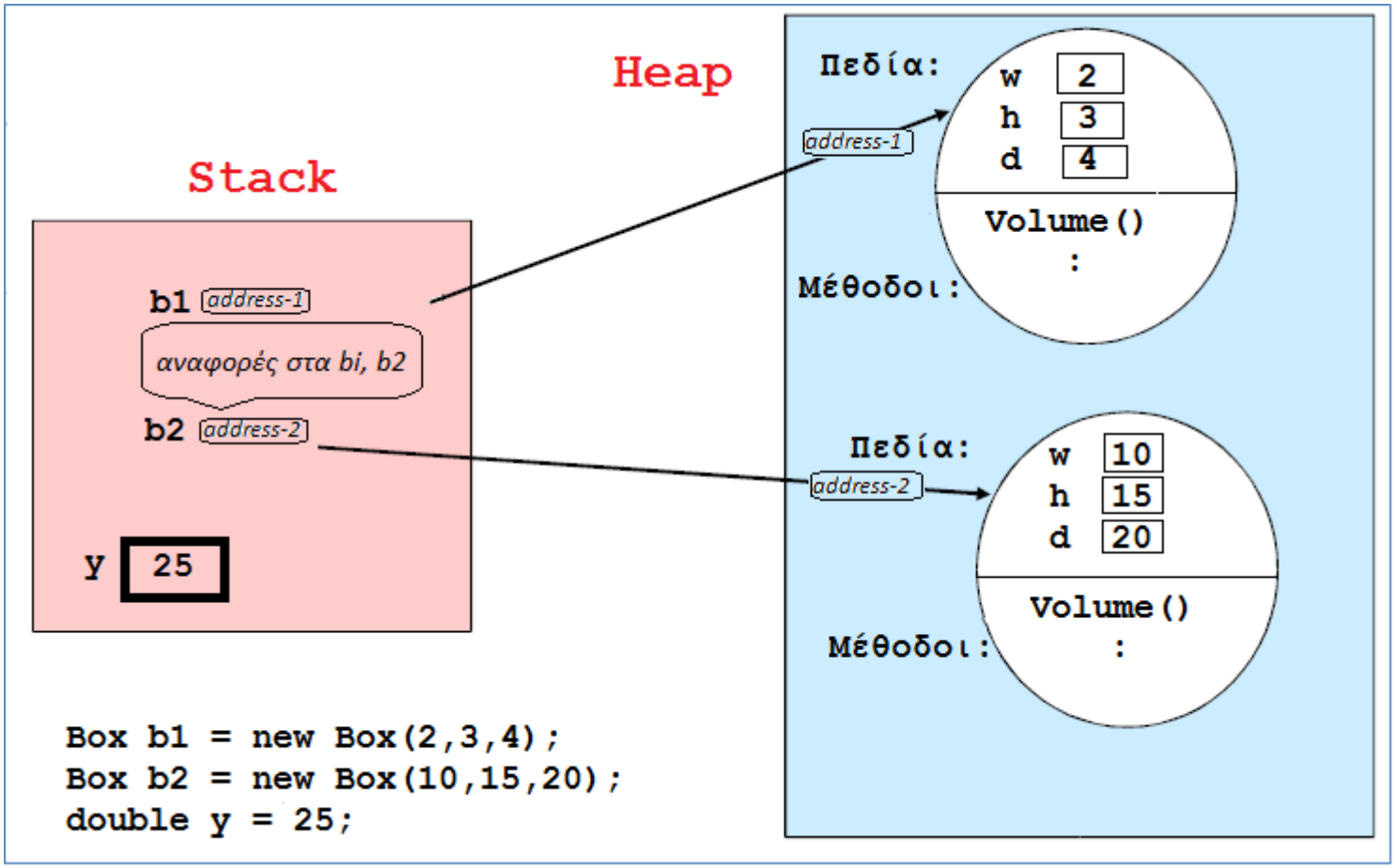
Κατανομή μνήμης *(1/4)*

Μια πρώτη απεικόνιση της διαθέσιμης μνήμης *(δεν παίζει ρόλο η σειρά…)*:



Κατανομή μνήμης *(2/4)*

- Μια μερική χρήση της **Stack** και **Heap** μνήμης

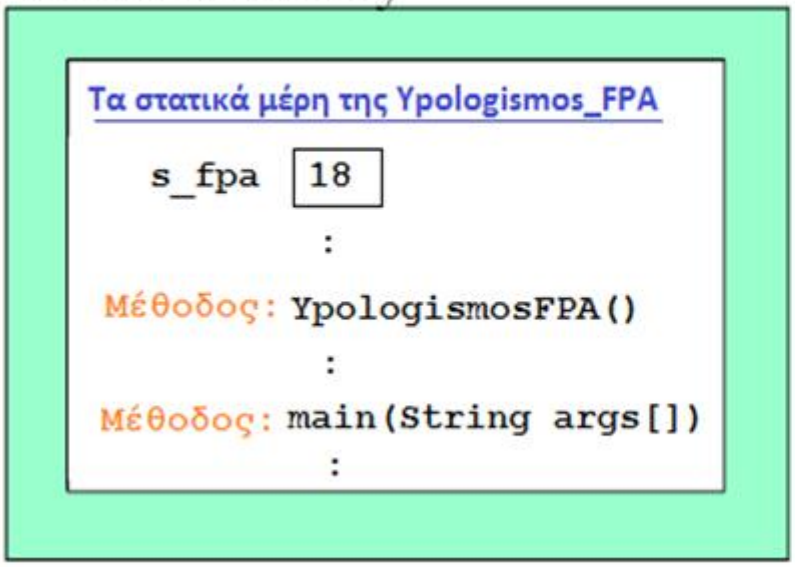


- Οι αναφορές *(references)* δείχνουν την **διεύθυνση του αντικειμένου** ή **null** *(καμία διεύθυνση)*

Κατανομή μνήμης *(3/4)*

Καταχωριτές *(Registers)* και Static storage:

- **Καταχωριτές *(Registers)*:** Ειδικός χώρος της μνήμης που χρησιμοποιείται μόνο από τον μεταγλωττιστή και κάποιες φορές από την VM, χωρίς να έχουμε πρόσβαση σε αυτή την μνήμη.
- **Στατική Μνήμη *(Static Memory)*:** Ειδικός χώρος της Heap – μνήμης όπου τοποθετούνται τα ***στατικά δεδομένα*** και οι ***στατικές μέθοδοι***.



Μια μερική χρήση της Static - μνήμης

```
class Ypologismos_FPA {
  :        :
public final static int s_fpa = 18;
  :        :
public static double YpologismosFPA() {..}
  :        :
public static void main(String args[]){..}
```

Κατανομή μνήμης *(4/4)*

Σύνοψη των Λειτουργιών

Call Stack / Runtime Stack

Θα αναλυθεί παρακάτω....

Call Frame ή Activation Record *(AR)*

Όλα τα δεδομένα με την **κλήση** και **εκτέλεση της μεθόδου** τοποθετούνται στη Stack και αποτελούν το AR, που 'σβήνει' με το τέλος της μεθόδου.

- τοπικές μεταβλητές
- τυπικές παράμετροι
- επιστρεφόμενη τιμή
- αναφορές αντικειμένων
- νήματα (threads)
- Compile time

Code area

- Φόρτωση και Εκτέλεση του κώδικα σε επίπεδο κλάσεων

Heap

Χώρος Αντικειμένων

- Δημιουργία αντικειμένων, τελεστής new
- Πεδία - Μέθοδοι *(κάθε αντικειμένου)*
- Garbage Collection

Στατική μνήμη

Στατικά Δεδομένα (Interfaces και classes)

- Στατικές μεταβλητές
- Στατικές μέθοδοι

Call Stack / Runtime Stack *(1/4)*

- Κατά την ***εκτέλεση μιας μεθόδου***, ο χώρος της μνήμης που χρησιμοποιείται για τα δεδομένα της μεθόδου, ονομάζεται **call stack** ή **runtime stack**.
- Όταν καλείται μια μέθοδος προς εκτέλεση, τότε δημιουργείται στη stack ένα **πλαίσιο κλήσης** *(call frame)* ή **εγγραφή ενεργοποίησης** *(activation record),* όπου ομαδοποιούνται όλα τα δεδομένα της μεθόδου.

Call Frame (Activation Record) →

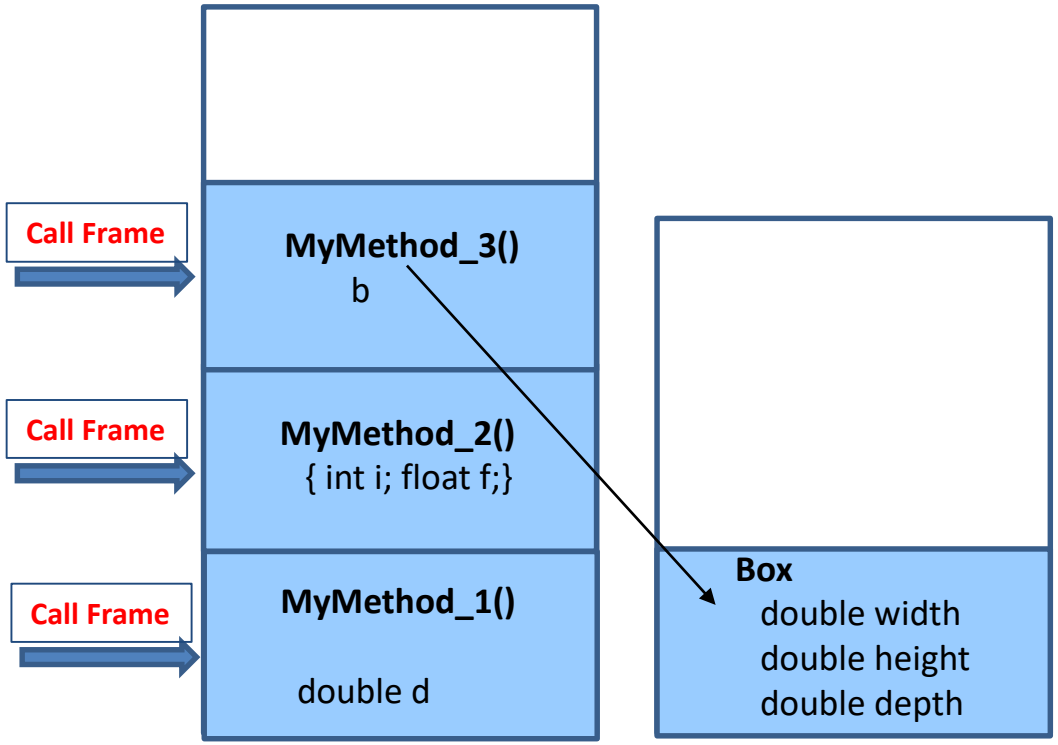
- **Επιστρεφόμενη τιμή μεθόδου**
- **Τοπικές μεταβλητές**
- **Τυπικές παράμετροι**
- **Επιστρεφόμενη διεύθυνση**

Call Stack – Call Frames *(2/4)*

Παράδειγμα:

```
class Box
{
  double width;
  double height;
   double depth;
:

}
```

```
class TestMemory
{
  MyMethod_1()
    { double d; }

  MyMethod_2()
    int i
    float f

  MyMethod_3()
    {
    Box b=new Box();
    }
```



class TestMemory

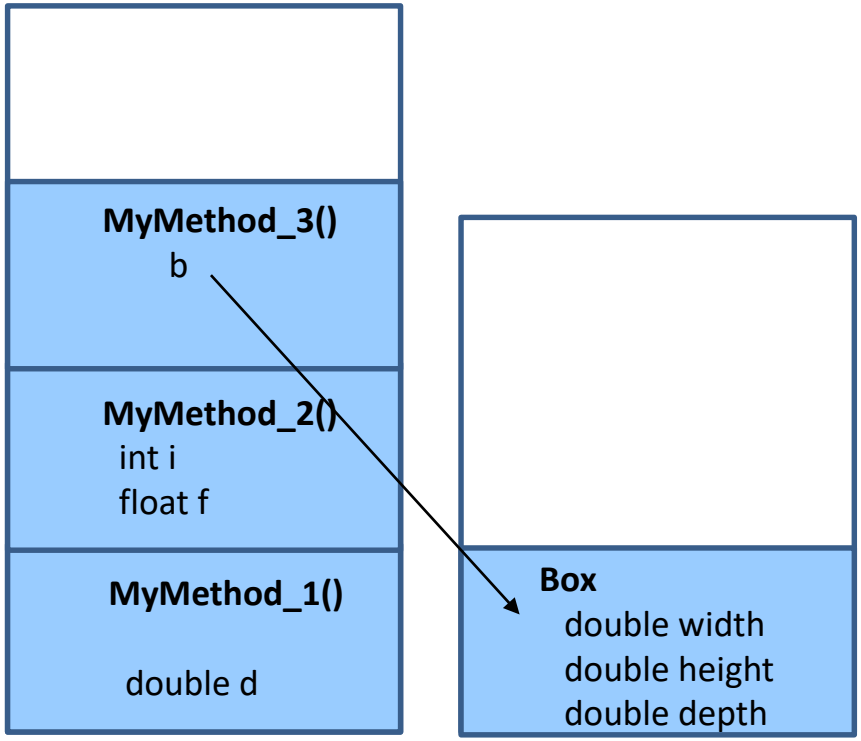
Μόλις τελειώσει η MyMethod_3()…..

Call Stack – Call Frames *(3/4)*

Παράδειγμα:

```
class Box
{
  double width;
  double height;
   double depth;
:

}
```

```
class TestMemory
{
  MyMethod_1()
    { double d; }

  MyMethod_2()
    { int i; float f;}

  MyMethod_3()
    {
    Box b=new Box();
    }
```



class Box

class TestMemory

Stack

Heap

Μόλις τελειώσει η MyMethod_3(), αρχίζει το **LIFO** (last-in-first-out)

Call Stack – Call Frames – Παράδειγμα *(4/4)*

Παράδειγμα:

```
class AddTwoValues {
 public static double add(double x, double y)
 {   double z;
     z=x + y;
    return z;  }

  public static void main(String args[]) {
    double d1=3.50;
    double d2=4.50;
    double r = add(d1, d2);
    System.out.println("To athroisma einai= " + r);
   }
}
```

Call Frame-add()

Call Frame-main()