Μ έ θ ο δ ο ι

Οι μέθοδοι είναι εκείνα τα τμήματα του κώδικα όπου εκτελούνται οι ουσιαστικές εργασίες του προγράμματος. Η γενική μορφή μιας μεθόδου είναι:

```
<τύπος>  <όνομα μεθόδου> (λίστα παραμέτρων)
{
  //κώδικας - σώμα της μεθόδου
}
```

Η μέθοδος αποτελείται από δύο μέρη: την **υπογραφή** *(signature)* της και το **σώμα** της (ο κώδικας). Ο τύπος, το όνομα της μεθόδου και οι παράμετροι αποτελούν την υπογραφή της μεθόδου.
Ο ***<τύπος>*** της μεθόδου μπορεί να είναι ένας από τους οκτώ βασικούς τύπους της Java ή κάποιος δικός μας και καθορίζει τον τύπο της επιστρεφόμενης τιμής (του αποτελέσματος).
Μια μέθοδος μπορεί να **επιστρέφει** (αφού πρώτα υπολογίσει) μια **τιμή** (του ίδιου τύπου – της δήλωσης) χρησιμοποιώντας τη λέξη κλειδί **return.** Αν απλώς εκτελεί λειτουργίες και πράξεις χωρίς να επιστρέφει τιμή (αλλά τη ροή του προγράμματος) τότε συντάσσεται με τη λέξη κλειδί **void** πριν από τον τύπο. Η λίστα παραμέτρων είναι μια ακολουθία από μεταβλητές με τους τύπους της, χωρισμένες με κόμμα. Είναι **τοπικές μεταβλητές** (local variables), δηλαδή ισχύουν μόνο όσο εκτελείται η συγκεκριμένη μέθοδος. Αυτό σημαίνει ότι τα ίδια ονόματα παραμέτρων μπορούν να χρησιμοποιηθούν και σε άλλες μεθόδους. Αν η μέθοδος δεν έχει παραμέτρους, αφήνουμε κενές τις παρενθέσεις. Μπροστά από κάθε μεταβλητή συντάσσεται ο τύπος της που προκαθορίζει τον τύπο της τιμής που θα λάβει όταν κληθεί από την αντίστοιχη εντολή. Η εντολή κλήσης περιέχει τις τιμές που θα "περάσουν" αντίστοιχα στις παραμέτρους της μεθόδου. Οι κλήσεις των μεθόδων είναι γνωστές και ως **μηνύματα** (messages).
Δύο είναι οι τύποι των μεθόδων: οι **μέθοδοι κλάσης** και οι **μέθοδοι στιγμιότυπου.** Οι μέθοδοι κλάσης δηλώνονται με τη λέξη κλειδί **static** πριν από τον τύπο και αυτό σημαίνει ότι μπορούν να εκτελέσουν τις λειτουργίες τους ***χωρίς να έχει δημιουργηθεί αντικείμενο της κλάσης***. Το αντίθετο συμβαίνει σε μια κλάση στιγμιότυπου. Δηλαδή για να εκτελεστεί ο κώδικας μιας κλάσης στιγμιότυπου θα πρέπει πρώτα να δημιουργηθεί ένα αντικείμενο της κλάσης.

Παράδειγμα:

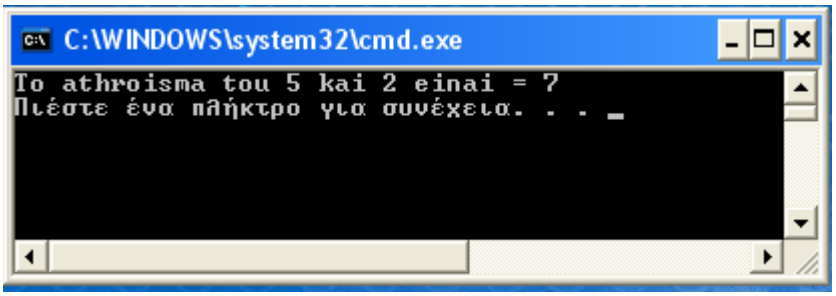
Θα γράψουμε μια μέθοδο που θα αθροίζει δύο αριθμούς και θα επιστρέφει το αποτέλεσμα στο κυρίως πρόγραμμα, όπου θα εμφανίζεται.

	Ορισμός μεθόδου	**Κλήση μεθόδου**
Υπογραφή Μεθόδου	**public static int add(int num1, int num2)**	**int sum=add(x, y)**
	λίστα παραμέτρων	
Σώμα μεθόδου	**{** int result = num1 + num2; **return** result; **}**	***παράμετροι***

Το πρόγραμμα *(προσοχή στη μέθοδο και την κλήση της)*

```
public class TestAdd1 {
   public static void main(String[] args) {
     int x = 5;
     int y = 2;
     int sum = add(x, y);      //κλήση της μεθόδου – πέρασμα των τιμών των παραμέτρων
     System.out.println("To athroisma tou " + x + " kai " + y + " einai = " + sum);
   }
   /* H methodos prosthesis 2 arithmvn */
   public static int add(int num1, int num2) {
     int result = num1 + num2;
     return result;
   }
 }
```

Το αποτέλεσμα:



Παραλλαγή της άσκησης με τη δημιουργία μεθόδου τύπου void

Θα γράψουμε την ίδια μέθοδο (που θα αθροίζει δύο αριθμούς), αλλά θα εμφανίζει το αποτέλεσμα στο σώμα της και θα **επιστρέφει τη ροή του προγράμματος** στο κυρίως πρόγραμμα. Επομένως η

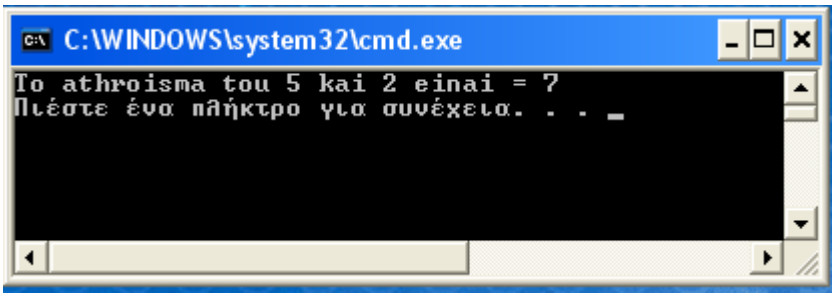
μέθοδος δεν θα έχει τύπο στην υπογραφή της, αλλά ούτε και return. Η μέθοδος θα είναι τύπου **void**.

	Ορισμός μεθόδου	**Κλήση μεθόδου**
Υπογραφή μεθόδου	**public static void add(int num1, int num2)**	**add(x, y)**
		παράμετροι
Σώμα μεθόδου	**{** ***λίστα παραμέτρων***	
	int result = num1 + num2;	
	System.out.println("To athroisma tou " +num1 + " kai " + num2 + " einai = " + result);	
	}	

Το πρόγραμμα *(προσοχή στη μέθοδο και την κλήση της)*

```
public class TestAdd2 {
   public static void main(String[] args) {
     int x = 5;
     int y = 2;
     add(x, y);      //κλήση της μεθόδου – πέρασμα των τιμών των παραμέτρων
   }

   /* H methodos prosthesis 2 arithmvn */
   public static void add(int num1, int num2) {
     int result = num1 + num2;
     System.out.println("To athroisma tou " +num1 + " kai " + num2 + " einai = " + result);
   }
 }
```

Το αποτέλεσμα:



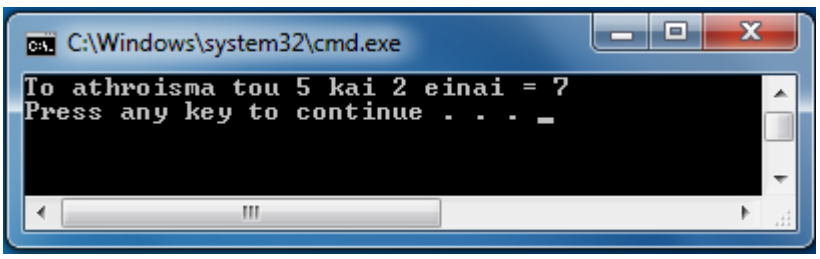
Μέθοδοι Στιγμιότυπου (Αντικειμένου)

Μέθοδοι Στιγμιότυπου ή Αντικειμένου απαιτούν πρώτα την δημιουργία Αντικειμένου. Η κλήση αυτών των μεθόδων γίνεται με την χρήση του Αντικειμένου.

Παράδειγμα μεθόδου τύπου return

```
class AddObj {
   int ar1;
   int ar2;

   AddObj(int x, int y)  {
     ar1 = x;
     ar2 = y;
   }

    /* H methodos prosthesis 2 arithmvn */
   public int add(int num1, int num2) {
    int result = num1 + num2;
    return result;}
 }

class TestAddObj {
   public static void main(String[] args) {
     int x=5; int y=2;
     AddObj obj = new AddObj(x, y);    //dhmioyrgia antikeimenoy typou AddObj
     int sum = obj.add(5, 2);  //κλήση της μεθόδου
     System.out.println("To athroisma tou " + x + " kai " + y + " einai = " + sum);
}}
```

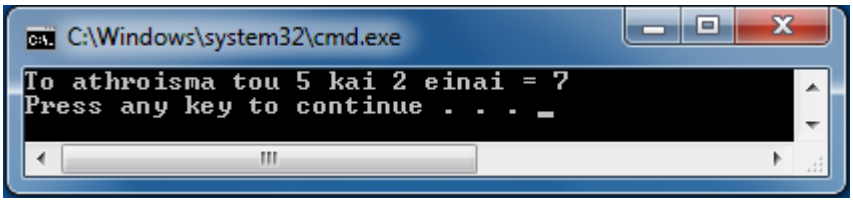
Το αποτέλεσμα:



Παράδειγμα μεθόδου τύπου void

```
class AddObj {
    int ar1;
    int ar2;

    AddObj(int x, int y)  {
     ar1 = x;
     ar2 = y;
     }

    /* H methodos prosthesis 2 arithmvn */
   public void add(int num1, int num2) {
     int result = num1 + num2;
     System.out.println("To athroisma tou " + num1 + " kai " +num2 + " einai = "
+ result);}
}

class TestAddObj {
    public static void main(String[] args) {
      int x=5; int y=2;
      AddObj obj = new AddObj(x, y);    //dhmioyrgia antikeimenoy typou AddObj
      obj.add(5, 2);     //κλήση της μεθόδου
}}
```

Το αποτέλεσμα:



ΠΕΡΑΣΜΑ ΠΑΡΑΜΕΤΡΩΝ

Οι παράμετροι που χρησιμοποιούνται κατά την κλήση μιας μεθόδου, ανάλογα με την χρήση των, ονομάζονται είτε **παράμετροι τιμής** *(call – by - value),* ή **παράμετροι αναφοράς** *(call – by – reference)*. Οι παράμετροι τιμής, που χρησιμοποιούνται μόνο για είσοδο τιμών, περνούν απλώς την τιμή τους στις αντίστοιχες παραμέτρους της μεθόδου. Οποιαδήποτε μεταβολή στην τιμή τέτοιων παραμέτρων μέσα στη μέθοδο δεν αντανακλάται στις αντίστοιχες παραμέτρους κλήσης.
Οι παράμετροι αναφοράς, είναι οι παράμετροι που περνούν στη μέθοδο μία σχέση απλής αναφοράς και όχι την ίδια την τιμή τους. Με τον τρόπο αυτό μια αλλαγή στην τιμή των, μέσα στη μέθοδο, αντανακλάται άμεσα και στις αναφερόμενες παραμέτρους της κλήσης.
Οι βασικοί τύποι της Java περνούν σαν παράμετροι τιμής, ενώ τα αντικείμενα σαν παράμετροι αναφοράς. Σε αντίθεση με άλλες γλώσσες προγραμματισμού, η Java δεν επιτρέπει να περαστεί κάποια μέθοδος σαν παράμετρος σε άλλη μέθοδο. Αυτό που επιτρέπεται είναι να περαστεί σαν παράμετρος το αντικείμενο και στη συνέχεια να κληθεί η επιθυμητή μέθοδος του αντικειμένου.

Πέρασμα παραμέτρων τιμής

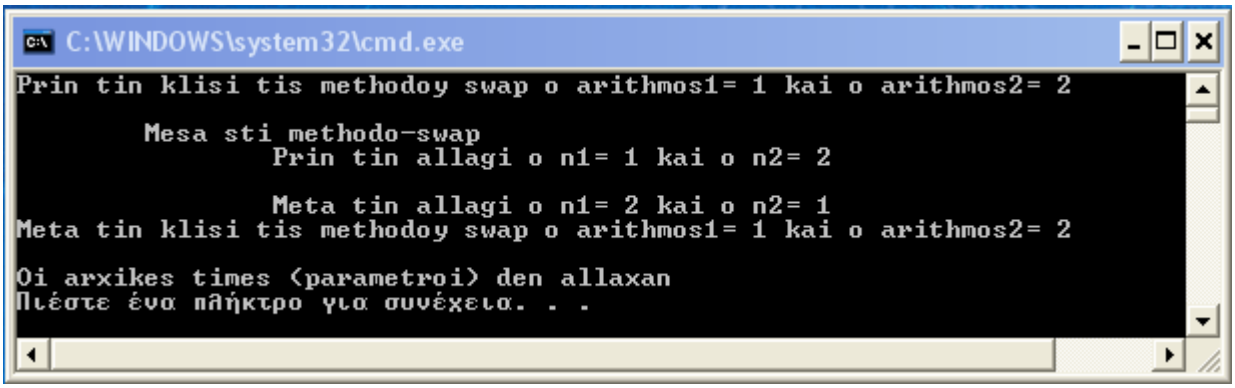
Παράδειγμα 1ο

Θα δούμε πως οι **παράμετροι τιμής διατηρούν τις αρχικές τους τιμές** μετά την κλήση μιας μεθόδου. Μέσα στη μέθοδο μπορούμε να τις αλλάξουμε (στο παράδειγμα swap), ή να εκτελέσουμε πράξεις με αυτές τις τιμές.

```
public class TestPassByValue {
   public static void main(String[] args) {
     int num1 = 1;
     int num2 = 2;

     System.out.println("Prin tin klisi tis methodoy swap o arithmos1= " + num1 + " kai o
       arithmos2= " + num2);

     // klisi tis methodou-swap
     swap(num1, num2);

     System.out.println("Meta tin klisi tis methodoy swap o arithmos1= " + num1 + " kai o
        arithmos2= " + num2);
     System.out.println();
     System.out.println("Oi arxikes times (parametroi) den allaxan");
   }
```

```
  /* metodos - swap */
  public static void swap(int n1, int n2) {
   System.out.println();
   System.out.println("\tMesa sti methodo-swap");
    System.out.println("\t\tPrin tin allagi o n1= " + n1 + " kai o n2= " + n2);

    // allagi tis n1 me n2
    int temp = n1;
    n1 = n2;
    n2 = temp;
    System.out.println();
    System.out.println("\t\tMeta tin allagi o n1= " + n1 + " kai o n2= " + n2);
  }
}
```

Το αποτέλεσμα:



Παράμετροι αναφοράς – Αντικείμενα σαν παράμετροι

Αντί για τους απλούς τύπους δεδομένων μπορούμε να χρησιμοποιήσουμε αντικείμενα σαν παραμέτρους τόσο στις μεθόδους όσο και στους δομητές. Τότε οι παράμετροι ονομάζονται **παράμετροι αναφοράς,** δηλαδή παράμετροι που περνούν στη μέθοδο μία σχέση απλής αναφοράς και όχι την ίδια την τιμή τους. Με αυτόν τον τρόπο μια αλλαγή στην τιμή των μέσα στη μέθοδο, αντανακλάται άμεσα και στις αναφερόμενες παραμέτρους της κλήσης.

Στο παρακάτω παράδειγμα θα δούμε την χρήση παραμέτρων αναφοράς:

```
class Test {
  int a, b;

  //constructor
  Test(int i, int j) {
    a = i;
    b = j;  }

  // πέρασμα αντικειμένου
  void meth(Test o) {
    o.a *=  2;
    o.b /= 2;
  }}

class CallByRef {
  public static void main(String args[]) {
    Test ob = new Test(15, 20);
    System.out.println("ob.a and ob.b before call: " + ob.a + " " + ob.b);
    ob.meth(ob);
    System.out.println("ob.a and ob.b after call: " + ob.a + " " + ob.b);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα :

```
a and b before call : 15  20
a and b after call : 30  10
```

Όπως βλέπουμε αλλαγές μέσα στην μέθοδο αντανακλώνται και στο ίδιο το αντικείμενο

Επιστροφή αντικειμένου σαν τιμή μεθόδου

Μία μέθοδος επιστρέφει όλους τους τύπους δεδομένων ακόμη και αντικείμενα. Στο παρακάτω παράδειγμα κάθε φορά που καλείται η incrByOne() δημιουργείται ένα νέο αντικείμενο που επιστρέφεται από την μέθοδο.

```
class Test {
  int a;
  Test(int i) {
    a = i;
  }

  Test incrByOne() {
    Test temp = new Test(a+1);
    return temp;
  }
}

class RetOb {
  public static void main(String args[]) {
    Test ob1 = new Test(2);
    Test ob2;
    ob2 = ob1.incrByOne();
    System.out.println("ob1.a: " + ob1.a);
    System.out.println("ob2.a: " + ob2.a);

    ob2 = ob2.incrByOne();
    System.out.println("ob2.a - new: " + ob2.a);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα :

```
Ob1.a: 2
Ob2.a: 3
Ob2.a - new: 4
```

Μία συνηθισμένη περίπτωση είναι εκείνη όπου ορίζεται ένα αντικείμενο από ένα ήδη υπάρχον (κλώνος).

Στην περίπτωση αυτή χρησιμοποιείται δομητής που δέχεται σαν παράμετρο ένα ήδη υπάρχον αντικείμενο. Στην παρακάτω παραλλαγή του παραδείγματος box θα δούμε πως ένα αντικείμενο ορίζει ένα άλλο.

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


  // δημιουργία κλώνου αντικειμένου
  Box(Box ob) {    // πέρασμα αντικειμένου στον δομητή
    width = ob.width;
    height = ob.height;
    depth = ob.depth; }

  // δομητής όταν γνωρίζουμε όλες τις διαστάσεις.
  Box(double w, double h, double d) {
    width = w;
    height = h;
    depth = d;
```

```
  }

  // δομητής όταν δεν καθορίζονται διαστάσεις.
  Box() {
    width = -1;
    height = -1;
    depth = -1;
   }

  // δομητής που χρησιμοποιείται όταν ορίζεται ο κύβος.
  Box(double len) {
    width = height = depth = len;
  }
  // υπολογισμός και επιστροφή του όγκου.
  double volume() {
    return width * height * depth;
  }
}

class OverloadCons2 {
  public static void main(String args[]) {

    // δημιουργία διαφορετικών αντικειμένων
    Box mybox1 = new Box(10, 20, 15);
    Box mybox2 = new Box();
    Box mycube = new Box(7);

    Box myclone = new Box(mybox1);

    double vol;

    // λήψη του όγκου του 1ου box.
    vol = mybox1.volume();
    System.out.println("Volume of mybox1 is " + vol);

    // λήψη του όγκου του 1ου box.
    vol = mybox2.volume();
    System.out.println("Volume of mybox2 is " + vol);

    // λήψη του όγκου του κύβου.
    vol = mycube.volume();
    System.out.println("Volume of cube is " + vol);

    // λήψη του όγκου του κλώνου.
    vol = myclone.volume();
    System.out.println("Volume of clone is " + vol);
  }}
```

Υπερφόρτωση Μεθόδων (Overloading Methods)

Στη java είναι δυνατόν να ορίσουμε μέσα σε μία κλάση δύο ή περισσότερες μεθόδους με το ίδιο όνομα αλλά με διαφορετικές παραμέτρους. Αυτό ονομάζεται **υπερφόρτωση μεθόδων** και στην περίπτωση αυτή ο **τύπος των δεδομένω**ν και ο **αριθμός των παραμέτρων** λαμβάνονται υπόψη

στον προσδιορισμό της μεθόδου που θα εκτελεστεί. Στο παρακάτω παράδειγμα θα δούμε πως υλοποιείται η υπερφόρτωση μεθόδων.

```
public class TestAdd1 {
   public static void main(String[] args) {
     int x = 5;
     int y = 2;
     double x1 = 2.0;
     double y1 = 4.0;

     int sum = add(x, y);          //klisi me dyo parametrous typou int
     System.out.println("To athroisma tou " + x + " kai " + y + " einai = " + sum);

     double sum1 = add(x1, y1);   //klisi me dyo parametrous typou double
     System.out.println("To athroisma tou " + x1 + " kai " + y1 + " einai = " + sum1);

     double sum2 = add(x1, y1, x);  //klisi me treis parametrous typou double
     System.out.println("To athroisma tou " + x1 + " kai " + y1 + " kai " + x + " einai = " +
        sum2);
   }

   /* H methodos prosthesis ton arithmvn x kai y */
   public static int add(int num1, int num2) {
     int result = num1 + num2;
     return result;
   }
 /* H methodos prosthesis ton arithmvn x1 kai y1 */
   public static double add(double num1, double num2) {
     double result = num1 + num2;
     return result;
   }
   /* H methodos prosthesis ton 3 arithmvn x1, y1 kai x */
     public static double add(double num1, double num2, double num3) {
       double result = num1 + num2 + num3;
       return result;  }}
```