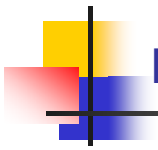
Πίνακες και Μέθοδοι

- Μονοδιάστατοι πίνακες
- Πολυδιάστατοι πίνακες
- Μέθοδοι
- Μέθοδοι Recursive
- Overloading



Μονοδιάστατοι πίνακες

- Οι πίνακες είναι μεταβλητές που έχουν ομαδοποιηθεί με ένα κοινό όνομα.

- Δημιουργία πινάκων με τη δήλωση new :

```
int[] a = new int[10];              // Πίνακας 10 ακεραίων αριθμών
double[] x = new double[20];   // Πίνακας 20 πραγματικών αριθμών
char[] c = new char[15];           // Πίνακας 15 χαρακτήρων
boolean[] b = new boolean[5]; // Πίνακας boolean 5 στοιχείων
String[] s = new String[8];        // Πίνακας 8 συμβολοσειρών
```

- Προσοχή στην αρίθμηση στοιχείων !

```
int[] a = new int[10];      ⟹  a[0], a[1], a[2] ….. a[9]
String[] s = new String[8]; ⟹  s[0], s[1], s[2] ….. s[7]
```

Μονοδιάστατοι πίνακες

- Τιμές στοιχείων πινάκων κατά την αρχικοποίηση:

Αριθμητικοί πίνακες	⟹	**0**
Πίνακες char	⟹	**'\0'**
Πίνακες boolean	⟹	**false**
Πίνακες String	⟹	**null**

- Απόδοση τιμών κατά την αρχικοποίηση

```
String[] names = {"Panos", "Giorgos", "Giannis", "Maria", "Nikos"};
```

αντιστοιχεί σε

```
String[] names = new String[5];
names[0]= "Panos";
names[1]= "Giorgos";
names[2]= "Giannis";
names[3]= "Maria";
names[4]= "Nikos";
```


Πολυδιάστατοι πίνακες

- Στη Java υπάρχει η δυνατότητα ορισμού πινάκων πολλών διαστάσεων. Για παράδειγμα ο πίνακας:

```
int[][] table={{1,2,3,4}, {5,6,7,8}, {9,10,11,12} };
```

αντιστοιχεί στον

1	2	3	4
5	6	7	8
9	10	11	12

δηλαδή

```
table[0][0]=1;      table[1][2]=7;
table[0][1]=2;      table[1][3]=8;
table[0][2]=3;      table[2][0]=9;
table[0][3]=4;      table[2][1]=10;
table[1][0]=5;      table[2][2]=11;
table[1][1]=6;      table[2][3]=12;
```

Παράδειγμα 1 με πίνακα

- Γράψτε ένα πρόγραμμα στο οποίο να δημιουργήσετε δέκα τυχαίους αριθμούς διπλής ακρίβειας και να τους αποθηκεύσετε σε κατάλληλο πίνακα. Στη συνέχεια χρησιμοποιείστε τον πίνακα για να τους εκτυπώσετε. Εκτυπώστε επίσης τον συνολικό αριθμό των στοιχείων του πίνακα που δημιουργήσατε χρησιμοποιώντας την μεταβλητή length. (Άσκηση 4.1.1)

Παράδειγμα 1 με πίνακα

```
import java.util.Random;

class TestArray
{
    public static void main(String[] arguments)
    {
        Random rndm = new Random();  // Dimiourgia antikeimenou rndm
        double[] x = new double[10]; // Dimiourgia pinaka me 10 stoixeia

        for(int i=0; i<10; i++)      // Apothikeysi ston pinaka 10 tixaion arithmon
            x[i]=rndm.nextDouble();

        for(int i=0; i<10; i++)      // Ektiposi tou pinaka
           System.out.println("x[" + i +"] = " + x[i]);

        System.out.println("O arithmos stoixeion tou pinaka einai : " + x.length);
    }
}
```

```
--:--  TestArray.java      (Java)--L1--All----------------------------------------
```

Παράδειγμα 1 με πίνακα

```
[student1@pc244 kef4]$
[student1@pc244 kef4]$
[student1@pc244 kef4]$ javac TestArray.java
[student1@pc244 kef4]$ java TestArray
x[0] = 0.503271039665935
x[1] = 0.9783879130795229
x[2] = 0.21364774265251452
x[3] = 0.29745598349172797
x[4] = 0.19127271972349125
x[5] = 0.9438726641065905
x[6] = 0.640012739787724
x[7] = 0.995834222249497
x[8] = 0.5741069262170253
x[9] = 0.16440516603059840
O arithmos stoixeion tou pinaka einai : 10
[student1@pc244 kef4]$
```


Παράδειγμα 2 με πίνακα

- Γράψτε ένα πρόγραμμα στο οποίο να δημιουργήσετε τον κατάλληλο διδιάστατο πίνακα ακεραίων αριθμών και αποθηκεύσετε σε αυτόν τα παρακάτω δεδομένα:

$$A = \begin{bmatrix} 15 & 22 & 43 & 10 \\ 56 & 98 & 34 & 43 \\ 82 & 65 & 19 & 39 \end{bmatrix}$$

Εκτυπώστε τον αριθμό των στηλών και των γραμμών του (χρησιμοποιείστε την μεταβλητή length). Εκτυπώστε τα στοιχεία του υπό μορφή στηλών-γραμμών. (Ασκηση 4.1.4)

Παράδειγμα 2 με πίνακα

```
class TwoDimArray
{
    public static void main(String[] arguments)
    {
        int[][] a = { {15, 22, 43, 10},
                      {56, 98, 34, 43},
                      {82, 65, 19, 39}};

        System.out.println("a.length    = " + a.length);
        System.out.println("a[0].length = " + a[0].length);


        for(int i=0; i<a.length; i++){
            for(int j=0; j<a[0].length; j++){
                System.out.print(a[i][j] + "   ");
            }
            System.out.println();
        }
    }
}
--:--  TwoDimArray.java      (Java)--L20--Top----------------------------------
```


Παράδειγμα 2 με πίνακα

```
[student1@pc244 kef4]$
[student1@pc244 kef4]$ javac TwoDimArray.java
[student1@pc244 kef4]$ java TwoDimArray
a.length    = 3
a[0].length = 4
15  22  43  10
56  98  34  43
82  65  19  39
[student1@pc244 kef4]$
```

Μέθοδοι

- Μια **μέθοδος** είναι μια ακολουθία δηλώσεων και εντολών συγκεντρωμένων σαν να αποτελούν ένα μίνι πρόγραμμα.

- Η χρήση μεθόδων δίνει **δομή** στα προγράμματά μας, κάνοντάς τα περισσότερο ευκολοδιάβαστα.

- Γράψτε μια μέθοδο η οποία να υπολογίζει την τιμή της συνάρτησης :

  $f(n) = \sqrt{3n^2 + 2n + 4}$

  Χρησιμοποιείστε την μέθοδο σε ένα πρόγραμμα για να τυπώσετε την τιμή της συνάρτησης για n=1, 2, 3….10. (Άσκηση 4.1.5)

Παράδειγμα 1 με Μέθοδο

```
class Fn
{
    public static void main(String[] arguments)
    {
        for(int n=1;n<=10;n++)
            System.out.println("n=" + n + " f(n)=" + fi(n));

    }


    static double fi(int i)
    {
        return Math.sqrt(3*i*i+2*i+4);
    }

}
-1:--  Fn.java          (Java)--L16--All----------------------------------
```

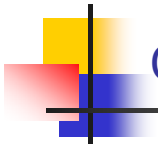
Παράδειγμα 1 με Μέθοδο

```
[student1@pc244 kef4]$
[student1@pc244 kef4]$ javac Fn.java
[student1@pc244 kef4]$ java Fn
n=1 f(n)=3.0
n=2 f(n)=4.47213595499958
n=3 f(n)=6.082762530298219
n=4 f(n)=7.745966692414834
n=5 f(n)=9.433981132056603
n=6 f(n)=11.135528725660043
n=7 f(n)=12.84523257866513
n=8 f(n)=14.560219778561036
n=9 f(n)=16.278820596099706
n=10 f(n)=18.0
[student1@pc244 kef4]$
```


Μέθοδοι Recursive

- Μια μέθοδος ονομάζεται **Recursive** όταν καλεί τον εαυτό της.
- Παράδειγμα υπολογισμού παραγοντικού n=1·2·3·4·…·n

```
static double paragontiko(int n)
{
    if(n<2)
            return 1.;
    return n*paragontiko(n-1);
}
```



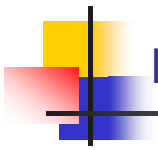
Overloading

- Στη Java είναι δυνατόν να έχουμε μεθόδους με τον ίδιο όνομα αλλά με διαφορετικό αριθμό ορισμάτων. Αυτό ονομάζεται **Overloading**.

- Παράδειγμα υπολογισμού μεγίστου δύο και τριών αριθμών:

```
static double max( double x, double y )
{
   ……….
}

static double max( double x, double y, double z )
{
  ……….
}
```

Παράδειγμα 2 με Μέθοδο

- Γράψτε μια μέθοδο η οποία να υπολογίζει την τιμή της συνάρτησης του παραγοντικού

  $n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \ldots (n-1) \cdot n$ με $0!=1$

- Χρησιμοποιείστε την μέθοδο σε ένα πρόγραμμα για να τυπώσετε την τιμή της συνάρτησης για n=1, 2, 3....8.
- Παρακάτω δίνονται τρεις διαφορετικές λύσεις στο πρόβλημα. Στην πρώτη μέθοδο f1() χρησιμοποιείται η εντολή for ενώ στη δεύτερη μέθοδο f2() η εντολή while. Ενδιαφέρον παρουσιάζει η τρίτη λύση με την μέθοδο f3() η οποία για να υπολογίσει το παραγοντικό καλεί τον εαυτό της. Όταν μία μέθοδος καλεί τον εαυτό της ονομάζεται **Recursive**.

Παράδειγμα 2 με Μέθοδο

```
class TestFactorial
{
    public static void main(String[] arguments)
    {
        for(int n=0;n<=8;n++)
            System.out.println(n + "! = " + f1(n)+ "\t" + f2(n)+ "\t" + f3(n));

    }

    static double f1(int n)     // Proti Methodos ypologismou paragontikou
    {
        double result=1.;
        for(int i=1;i<=n;i++)
            result*=i;

        return result;
    }
```

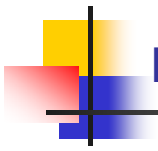

Παράδειγμα 2 με Μέθοδο

```
    static double f2(int n)   // Deyteri Methodos ypologismou paragontikou
    {
        double result=1.;
        while(n>1)
            result*=n--;

        return result;
    }

    static double f3(int n)   // Triti Methodos ypologismou paragontikou
    {                         // Recursive Methodos
        if (n<2)              // 0!=1 kai 1!=1
            return 1.;
        return n*f3(n-1);     // n!=n*(n-1)!
    }

}
--:--  TestFactorial.java      (Java)--L35--All----------------------------
```



Παράδειγμα 2 με Μέθοδο

```
[student1@pc244 kef4]$
[student1@pc244 kef4]$
[student1@pc244 kef4]$ javac TestFactorial.java
[student1@pc244 kef4]$ java TestFactorial
0! = 1.0        1.0     1.0
1! = 1.0        1.0     1.0
2! = 2.0        2.0     2.0
3! = 6.0        6.0     6.0
4! = 24.0       24.0    24.0
5! = 120.0      120.0   120.0
6! = 720.0      720.0   720.0
7! = 5040.0     5040.0  5040.0
8! = 40320.0    40320.0 40320.0
[student1@pc244 kef4]$
```