Αντικειμενοστρεφής Προγραμματισμός
Διάλεξη – 6 : ΠΙΝΑΚΕΣ

Κων. Κόκκινος

ΠΙΝΑΚΕΣ (ARRAYS)

- Είναι χώροι της μνήμης για προσωρινή αποθήκευση δεδομένων του ίδιου τύπου.
- Οι πίνακες είναι δομές δεδομένων που τις συναντάμε σχεδόν σε όλες τις γλώσσες προγραμματισμού με μόνη διαφορά ότι **στη Java οι πίνακες είναι αντικείμενα**.
- Οι πίνακες έχουν προκαθορισμένο μέγεθος. Σε κάθε κελί του πίνακα μπορεί να αποθηκευτεί μία τιμή, η οποία μπορεί να είναι είτε αρχικός τύπος της Java (int, double, κλπ.), είτε αναφορά σε κάποιο αντικείμενο.
- Η επεξεργασία των στοιχείων του πίνακα γίνεται με εύκολο τρόπο. Η αναφορά στα δεδομένα κάποιου κελιού του πίνακα γίνεται με το όνομα του και ένα ή περισσότερους δείκτες, ανάλογα με τις διαστάσεις του.
- Υπάρχουν οι μονοδιάστατοι, δισδιάστατοι και πολυδιάστατοι πίνακες.

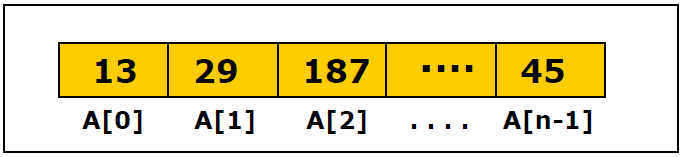
Μονοδιάστατοι Πίνακες

ΟΡΙΣΜΟΣ ΜΟΝΟΔΙΑΣΤΑΤΩΝ ΠΙΝΑΚΩΝ

- Οι πίνακες μιας διάστασης θεωρούνται ως γραμμικοί ενιαίοι χώροι για την αποθήκευση στοιχείων του ίδιου τύπου, δηλαδή μεταβλητές του ίδιου βασικού τύπου ή αναφορές σε αντικείμενα της ίδιας κλάσης.
- Τα κελιά του πίνακα θεωρούνται ως συνεχόμενα, άρα με δείκτες συνεχόμενους που αρχίζουν πάντα από το μηδέν:
- **A[0], A[1], A[2],…, A[n-1].**

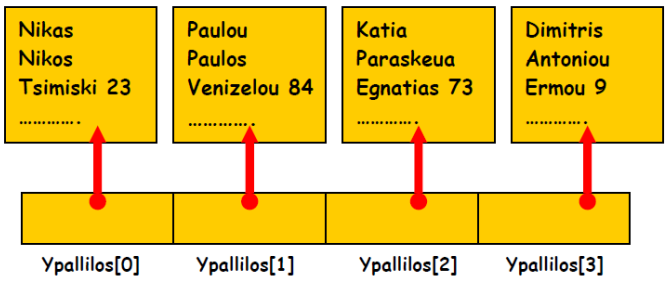
Πίνακες με τιμές βασικών τύπων

- Οι τιμές των βασικών τύπων της Java αποθηκεύονται στα κελιά του πίνακα.
- **Παράδειγμα με αποθήκευση ακεραίων αριθμών**



Πίνακες αντικειμένων

- Σε αντίθεση με τους πίνακες βασικών τύπων, όταν δημιουργείται ένας πίνακας αντικειμένων, αυτά δεν αποθηκεύονται στα κελιά του πίνακα.
- Στον πίνακα αποθηκεύονται μόνο οι αναφορές στα αντίστοιχα αντικείμενα (που μπορεί να έχουν τιμές – όπως στο παράδειγμα παρακάτω – ή να είναι μηδέν).



Δήλωση και χρήση ενός μονοδιάστατου πίνακα 1/2

- Στην Java, για να χρησιμοποιήσουμε ένα πίνακα πρέπει πρώτα να τον δηλώσουμε, όπως συμβαίνει και στις απλές μεταβλητές. Η δήλωση του πίνακα γίνεται με μία εντολή της μορφής:

```
<τύπος δεδομένων> <όνομα μεταβλητής πίνακα[]>; ή
<τύπος δεδομένων[]> <όνομα μεταβλητής πίνακα>;
```

- *Όπου:*

Τύπος δεδομένων, είναι ο τύπος των στοιχείων του θα περιέχει ο πίνακας και όνομα μεταβλητής πίνακα, το όνομα του πίνακα. Με τον ορισμό αυτό ορίζεται ένας πίνακας με μηδενικό περιεχόμενο, δηλαδή ο πίνακας ακόμη δεν υπάρχει στην φυσική του μορφή.

Παράδειγμα: `int numbers[];` ή `int[] numbers;`

Δήλωση και χρήση ενός μονοδιάστατου πίνακα 2/2

- Η δήλωση του πίνακα είναι το πρώτο βήμα για την χρήση του. Το δεύτερο βήμα είναι να δεσμευτεί ο χώρος της μνήμης που απαιτείται για τον πίνακα. Αυτό επιτυγχάνεται με τη δημιουργία ενός αντικειμένου (όπως και με τα άλλα αντικείμενα της Java) του δηλωθέντος τύπου του πίνακα με την εντολή **new**:

```
<όνομα μεταβλητής πίνακα> = new <τύπος δεδομένων>[];
```

Παράδειγμα: `numbers = new int[10];`

Συνήθως συνδυάζουμε τις δύο εντολές σε μία:

```
τύπος δεδομένων  <όνομα - μεταβλητής πίνακα[ ]> =  new τύπος [μέγεθος];  ή
τύπος δεδομένων[ ]  <όνομα - μεταβλητής πίνακα> =  new τύπος [μέγεθος];
```

Παράδειγμα: Δημιουργία του πίνακα numbers που θα μπορεί να αποθηκεύσει 10 ακέραιους.

```
int numbers[] = new int[10];  ή
int[] numbers = new int[10];
```

Παραδείγματα με πίνακες 1D (1/3)

- **numbers[3] = 24;** Ορίζει την τιμή 24 στο 4ο κελί του πίνακα numbers.
- **numbers[6] = numbers[3];** Εκχωρεί το περιεχόμενο του 4ου κελιού στο 7ο κελί. Στο παράδειγμα μας τον αριθμό 24.
- **System.out.println(numbers[6]);** Εμφανίζει το περιεχόμενο του 7ου κελιού, δηλαδή τον αριθμό 24.
- **int numbers[] = {10, 12, 30, 40, 55, 60, 63};** ή **int[] numbers = {10, 12, 30, 40, 55, 60, 63};** Εκχωρεί απευθείας τιμές στα κελιά του πίνακα, δηλαδή: numbers[0] = 10, …, numbers[6] = 63. Επίσης εκχωρούνται String τιμές μέσα σε διπλά εισαγωγικά, και τύπου char μέσα σε μονά εισαγωγικά:

String[] categories = {"Action", "Comedy", "Drama"}; ή

String categories[] = {"Action", "Comedy", "Drama"};

char[] epiloges = {'a','b','c','d','e','f'}; ή

char epiloges[] = {'a','b','c','d','e','f'};

Προσοχή!!! Σε αυτές τις περιπτώσεις δεν χρειάζεται η δήλωση της new.

Παραδείγματα με πίνακες 1D (2/3)

- Στο παρακάτω παράδειγμα δημιουργείται ο πίνακας numbers με 10 κελιά, στα οποία θα καταχωρηθούν οι τιμές του μετρητή **i** (0, 1, 2, .., 9) και στο τέλος θα εμφανιστούν τα αποτελέσματα. Η καταχώρηση και η εμφάνιση των στοιχείων θα γίνει με την εντολή ***for που ενδείκνυται για τέτοιες εργασίες πινάκων.***

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

    //dimiourgia tou pinaka numbers me 10 akeraious
    int[] numbers = new int[10];

    //eisodos stoixeivn tou pinaka
    for (int i=0; i<10; i++) numbers[i] = i;

    //emfanisi stoixeivn tou pinaka
    for (int i=0; i<10; i++) System.out.println("Keli: " + i + "  Timi: " + numbers[i]);
  }
}
```

Παραδείγματα με πίνακες 1D (3/3)

- Στο παρακάτω παράδειγμα θα υπολογίσουμε τον μέσο όρο αριθμών του μονοδιάστατου πίνακα numbers.

```
class Average {
  public static void main(String args[]) {
    double numbers[] = {2.1, 1.2, 4.5, 3.6, 4.7};
    double result = 0;
    int i;

    for(i=0; i<5; i++)
      result = result + numbers[i];

    System.out.println("O mesos oros einai: " + result / i);
  }
}
```

Το χαρακτηριστικό length στους πίνακες (1/2)

- Ένα χαρακτηριστικό *(instance variable)* του αντικειμένου πίνακας είναι η **length.** Η length καθορίζεται με την εντολή **new** και λαμβάνει τιμή ίση με το πλήθος των κελιών του πίνακα. Η length θα μας αποτρέψει να βγούμε εκτός των ορίων ενός πίνακα στις εντολές επανάληψης. Σε τέτοια λάθη η java δείχνει το μήνυμα **java.lang.ArrayIndexOutOfBoundsException.**. Έτσι είναι καλύτερα να χρησιμοποιούμε την length σαν συνθήκη τερματισμού της επανάληψης επεξεργασίας του πίνακα. Στο παρακάτω παράδειγμα η length χρησιμοποιείται σαν συνθήκη τερματισμού της εντολής for:

  Όπου, **numbers.length** δηλώνει το πλήθος των κελιών του πίνακα numbers.

- Στο παράδειγμα εισάγεται από το πληκτρολόγιο το μέγεθος του πίνακα (πλήθος κελιών) καθώς και οι τιμές των κελιών. Το πρόγραμμα σαν έξοδο, απλώς εμφανίζει τα περιεχόμενα των κελιών. Προσέξτε ιδιαίτερα την σύνταξη της εντολής - for με την χρήση της array.length.

Το χαρακτηριστικό length στους πίνακες (2/2)

```
import java.io.* ;
class PinakasArithmon {
  public static void main ( String[ ] args ) throws IOException {
        BufferedReader inData = new BufferedReader(new InputStreamReader(System.in));
        int[ ] array;
        System.out.print( "Posa kelia na exei o pinakas;" );
        int size = Integer.parseInt(inData.readLine());
        array = new int[size];
        for(int i=0; i < array.length; i++) {
              System.out.print( "Dose ena akeraio:" );
              array[i] = Integer.parseInt(inData.readLine());
        }
        for(int i=0; i < array.length; i++ ) {
              System.out.println( "Keli[ " + i + " ] = " + array[i]);
        }
  }
}
```

Πίνακες περισσότερων διαστάσεων

Η Java υποστηρίζει πίνακες περισσοτέρων διαστάσεων, υιοθετώντας τη φιλοσοφία: *πίνακες αποτελούμενοι από πίνακες.*

Δισδιάστατοι πίνακες (1/2)

- Για να ορίσουμε ένα δισδιάστατο πίνακα προσθέτουμε ένα νέο δείκτη:

τύπος δεδομένων <όνομα - μεταβλητής πίνακα[][]> =
new τύπος δεδομένων [πλήθος γραμμών][πλήθος στηλών];

ή

τύπος δεδομένων[][] <όνομα - μεταβλητής πίνακα> =
new τύπος δεδομένων [πλήθος γραμμών][πλήθος στηλών];

Παράδειγμα:

int Dis[][] = new int[3][4]; ή
int[][] Dis = new int[3][4];

Δείκτες	*0*	*1*	*2*	*3*
0	18		24	
1		65		90
2	32		39	

Δισδιάστατοι πίνακες (2/2)

ΠΑΡΑΔΕΙΓΜΑ

```
class Dis {
        public static void main(String args[])  {
                int Dis[][]= new int[3][4];
                int i, j, k = 0;
                for(i=0; i<3; i++) {
                        for(j=0; j<4; j++) {
                                Dis[i][j] = k;
                                k++;
                        }
                }
                for(i=0; i<3; i++) {
                        for(j=0; j<4; j++)
                                System.out.print(Dis[i][j] + " ");
                        System.out.println();
                }
        }
}
```

Ακανόνιστοι πίνακες (1/2)

- Κάθε γραμμή ενός δισδιάστατου πίνακα μπορεί να έχει οποιοδήποτε αριθμό στοιχείων (στηλών) ή αλλιώς διαφορετικές γραμμές μπορεί να έχουν διαφορετικό αριθμό στηλών *(ragged πίνακες)*.
- Γενικά στους πολυδιάστατους πίνακες, αρκεί να ορίσουμε τον αριστερό δείκτη – γραμμών ενώ τον δεξιό δείκτη - των στηλών μπορούμε να τον ορίσουμε χειρωνακτικά μέσα στο πρόγραμμα και με διαφορετική τιμή για κάθε γραμμή.
- Αυτό σημαίνει ότι μπορούμε να έχουμε διαφορετικές στήλες σε κάθε γραμμή για ένα πολυδιάστατο πίνακα. Δηλαδή, όταν ορίζουμε χειρωνακτικά τις διαστάσεις του πίνακα δεν χρειάζεται να ορίζουμε το ίδιο πλήθος κελιών για κάθε διάσταση.
- Στο παρακάτω παράδειγμα θα ορίσουμε ένα δισδιάστατο πίνακα στον οποίο τα μεγέθη της δεύτερης διάστασης δεν είναι ίσα.

Ακανόνιστοι πίνακες (2/2)

```
class Dis1 {
        public static void main(String args[]) {
                int Dis1[][] = new int[3][];
                Dis1[0] = new int[1];
                Dis1[1] = new int[2];
                Dis1[2] = new int[3];
                int i, j, k = 0;
                for(i=0; i<3; i++) {
                        for(j=0; j<i+1; j++) {
                                Dis1[i][j] = k;
                                k++;
                        }
                }
                for(i=0; i<3; i++) {
                        for(j=0; j<i+1; j++) {
                                System.out.print(Dis1[i][j] + " ");
                        }
                        System.out.println();
                }
        }
}
```

Πίνακες με αρχικοποίηση

```
class Matrix {
        public static void main(String args[]) {
                double m[][] = {
                        { 0*0, 1*0, 2*0, 3*0 },
                        { 0*1, 1*1, 2*1, 3*1 },
                        { 0*2, 1*2, 2*2, 3*2 },
                        { 0*3, 1*3, 2*3, 3*3 }
                };
                int i, j;
                for(i=0; i<4; i++) {
                        for(j=0; j<4; j++)
                                System.out.print(m[i][j] + " ");
                        System.out.println();
                }
        }
}
```

Τρισδιάστατοι πίνακες (1/2)

- Στο παρακάτω παράδειγμα θα ορίσουμε ένα τρισδιάστατο πίνακα (3 * 4 * 5), δηλαδή τρεις δισδιάστατους πίνακες 4 * 5 = 20 κελιών ο καθένας ή διαφορετικά 3 δισδιάστατους, 4 – γραμμών και 5- στηλών ο καθένας.
- Το κάθε κελί θα έχει σαν τιμή το γινόμενο των αντιστοίχων δεικτών του.

Τρισδιάστατοι πίνακες (2/2)

```
class Tris {
 public static void main(String args[]) {
        int tris[][][] = new int[3][4][5];
        int i, j, k;
        for(i=0; i<3; i++)
                for(j=0; j<4; j++)
                        for(k=0; k<5; k++)
                                tris[i][j][k] = i * j * k;
        for(i=0; i<3; i++) {
                for(j=0; j<4; j++) {
                        for(k=0; k<5; k++)
                                System.out.print(tris[i][j][k] + " ");
                                System.out.println();
                }
                System.out.println();
        }
 }
}
```

Πίνακας ως παράμετρος σε μέθοδο (1/3)

- Ένας πίνακας μπορεί να περάσει σαν παράμετρος εισόδου δεδομένων σε μία μέθοδο. **Μια μέθοδος με παράμετρο ένα πίνακα πρέπει να καθορίζει μόνο τον τύπο δεδομένων του πίνακα και όχι το μέγεθός του**. Π.χ. **public static void arithmoi(int[] a),** εδώ ένα πίνακα ακεραίων.
- **Όταν ένας ολόκληρος πίνακας δίνεται σαν όρισμα, τότε δεν χρησιμοποιούνται αγκύλες** (δες παρακάτω παράδειγμα). **Επίσης μια μέθοδος που καθορίζει έναν πίνακα ως παράμετρο, μπορεί να δεχτεί πίνακα μόνο του ίδιου τύπου και οποιουδήποτε μεγέθους**.
- Στο παρακάτω παράδειγμα θα περάσουμε τον πίνακα **myarr** σαν παράμετρο σε μία μέθοδο **print()** όπου απλά θα εμφανιστούν τα στοιχεία του πίνακα.

Πίνακας ως παράμετρος σε μέθοδο (2/3)

```
class AMethod {
  void print(int[] x)  {
    for (int index=0; index < x.length; index++)
      System.out.print( x[index] + " " );
    System.out.println();
  }
}
```

```
class MyArrayDemo {
  public static void main(String[] args) {
    AMethod operate = new AMethod();
    int[] myarr =  { 14, 1, -21, 13, 8, -7, 35, 80 } ;

    System.out.print("\n The array is : " );
    operate.print(myarr);
  }
}
```

Πίνακας ως παράμετρος σε μέθοδο (3/3)

```
class AMethod {
  void print( int[] x )  {
    for ( int index=0; index < x.length; index++ )
      System.out.print( x[index] + " " );
      System.out.println();
  }
  // εμφάνιση των στοιχείων από τιμή-start μέχρι-end.
  void printRange ( int[] x, int start, int end ) {
    for ( int index=start; index <= end ; index++ )
      System.out.print( x[index] + " " );
      System.out.println();
  }
}
```

```
class MyArrayDemo {
  public static void main(String[] args)  {
    AMethod operate = new AMethod();
    int[] myarr1 =  {14, 1, -21, 13, 8, -7, 35, 80} ;

    // εμφάνιση πέντε στοιχείων των: 1, -21, 13, 8, -7.
    operate.printRange(myarr1, 1, 5 );
  }
}
```

Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)* (1/5)

- Ας αναλύσουμε την περίπτωση
  1. της δημιουργίας ενός μονοδιάστατου πίνακα που θα περιέχει 3-αντικείμενα του τύπου Box (...στην πραγματικότητα 3-αναφορές στα αντικείμενα),
  2. τον οποίο θα στείλουμε σαν παράμετρο στην μέθοδο volume(), η οποία θα υπολογίζει τον όγκο του κάθε Box και θα τον τοποθετεί σε αντίστοιχο κελί ενός πίνακα τύπου-double[], τον οποίο θα επιστρέψει στο κυρίως πρόγραμμα.
  3. Τέλος θα εμφανίσουμε τα αποτελέσματα (όγκους) των 3-αντικειμένων (εμφάνιση του επιστρεφόμενου πίνακα), στο κυρίως πρόγραμμα.

Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)* (2/5)

- **Τι θα πρέπει να προσέξουμε:**
  - Ένας πίνακας αντικειμένων δημιουργείται όπως και ο πίνακας των βασικών τύπων με τον τελεστή new. Π.χ. **Box pinakas[] = new Box[3];**
  - Με την εντολή αυτή δημιουργείται ο πίνακας-pinakas[] που θα περιέχει **τις αναφορές 3 αντικειμένων** του τύπου Box (όμως προσοχή: δεν δημιουργεί τα ίδια τα αντικείμενα). Για να έχουμε πρόσβαση στα αντικείμενα και να τα επεξεργαστούμε θα πρέπει πρώτα να τα αρχικοποιήσουμε χρησιμοποιώντας τον **κατασκευαστή** της κλάσης Box ή να δώσουμε αρχικές τιμές μέσα στο πρόγραμμα, χωρίς **κατασκευαστή** , όπως κάναμε και στα πρώτα παραδείγματα των αντικειμένων τύπου Box

Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)* (3/5)

- pinakas[0]= new Box(10, 20, 15);
- pinakas[1]= new Box(3, 6, 9);
- pinakas[2]= new Box(4, 5, 6);
- Για να λάβουμε σε πίνακα τους όγκους των 3-κουτιών(objects), ορίζουμε ένα πίνακα τύπου double 3-κελιών και καλούμε την μέθοδο volume() με παράμετρο τον πίνακα:

  double Volumes[]=new double[3];

  Volumes=Box.volume(pinakas);

```
public static double[] volume(Box a[]) {
 double Vol[] = new double[3];
 for(int i=0; i<3; i++) Vol[i]=a[i].width * a[i].height * a[i].depth;
 return Vol;}
```

Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)* (4/5)

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

   Box(double w, double h ,double d){
        width  = w;
        height = h;
        depth  = d; }

   public static double[] volume(Box a[]) {
    double Vol[] = new double[3];
    for(int i=0; i<3; i++) Vol[i]=a[i].width * a[i].height * a[i].depth;
    return Vol;
}}
```

Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)* (5/5)

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

   Box pinakas[]=new Box[3];      //array of Box-objects

   pinakas[0]= new Box(10, 20, 15);  //arxikopoisi 3-object kai topothetisi anaforon sta kelia toy pinaka
   pinakas[1]= new Box(3, 6, 9);
   pinakas[2]= new Box(4, 5, 6);

   double Volumes[]=new double[3];
   Volumes=Box.volume(pinakas);  //klisi tis methodoy volume() me paramtro ton pinaka Box-objects
                                 //Pernoume tous ogkous ston pinaka Volumes()


    //emfanisi ton ogkvn 3-objects
   for(int i=0; i<3; i++){
        System.out.println(" O ogkos tou box [" + i + "]" + " einai = "+ Volumes[i]); }
}}
```

Η κλάση Arrays (1/4)

- Η κλάση **Arrays** περιέχει σημαντικές μεθόδους χειρισμού των πινάκων:
  - **Η μέθοδος arraycopy()** Μία μέθοδος της βιβλιοθήκης java.lang είναι η arraycopy(). Αντιγράφει ένα πίνακα από κάποιο άλλο πίνακα, ξεκινώντας από κάποια συγκεκριμένη θέση και για συγκεκριμένα στοιχεία του πίνακα. Η σύνταξη της εντολής είναι:
  - **static void arraycopy(<Obj.source>, <int sourceStart>, <Obj.target>, <int targetStart>,<int size>);**
  - *Όπου:*
    - *<Obj.source>*, Ο πίνακας από τον οποίο θα γίνει η αντιγραφή
    - *<int sourceStart>*, Από ποιο στοιχείο θα αρχίσει η αντιγραφή
    - *<Obj.target>*, Ο πίνακας αντίγραφο που θα παραχθεί
    - *<int targetStart>*, Από ποιο στοιχείο του αντιγράφου θα αρχίσει η αντιγραφή
    - *<int size>);* Πόσα στοιχεία θα αντιγράψει

Η κλάση Arrays (2/4)

```
import java.lang.System;     //προαιρετική χρήση

public class arrayofcopy {

public static void main(String[] args) {

char [] a1={'a','b','c','d','e','f'};
char [] b1={'x','y','z','u','v'};

System.arraycopy(a1,1,b1,2,3);
System.out.println(b1);
}
}
```

Η κλάση Arrays (3/4)

- **Η μέθοδος fill()** Η μέθοδος αυτή 'γεμίζει' τον πίνακα με κάποια τιμή. Π.χ.
  - **int[] arr = new int[10];**
  - **Arrays.fill(arr, 99);**
  - Σε αυτό το παράδειγμα ο πίνακας arr - 10 ακέραιων αριθμών – 'γεμίζει' με την τιμή 99 σε όλα του τα κελιά.
- **Arrays.fill(arr, 2, 4, 99);**
  - Σε αυτό το παράδειγμα ο πίνακας arr - 10 ακέραιων αριθμών – 'γεμίζει' με την τιμή 99 από την θέση 2 μέχρι την θέση 4 του πίνακα.

Η κλάση Arrays (4/4)

- **Η μέθοδος equals()** Ελέγχει την ισότητα στα περιεχόμενα των πινάκων, επιστρέφοντας true ή false, ανάλογα με το εάν τα περιεχόμενα τους είναι ίσα ή άνισα αντίστοιχα. Π.χ.
  - boolean[ ] myArrA; boolean[ ] myArrB;
  - myArrA = new boolean[ ]{true, false}; myArrB = new boolean[ ]{true, false};
  - **Boolean b = Arrays.equals(myArrA, myArrB);**
- **Η μέθοδος sort()** Η μέθοδος αυτή ταξινομεί τους πίνακες και των βασικών τύπων αλλά και των αντικειμένων. Η μέθοδος ταξινόμησης εφαρμόζεται είτε σε ολόκληρο τον πίνακα, είτε σε συγκεκριμένο εύρος κελιών.
  - **Arrays.sort(array);** Ταξινομεί τα στοιχεία ενός πίνακα βασικών τύπων κατά αύξουσα τάξη.
  - **Arrays.sort(array, from, to);** Ταξινομεί τα στοιχεία από το κελί (from) …μέχρι το (to-1) ενός πίνακα βασικών τύπων κατά αύξουσα τάξη.

Ορίσματα από την γραμμή εντολών (1/2)

- Τι σημαίνει όρισμα από την γραμμή εντολών??? (command line arguments)
- Όταν μεταγλωττίζουμε ένα πρόγραμμα (π.χ Test.java) στη γραμμή εντολών γράφουμε (javac Test.java)
- Στην προκειμένη περίπτωση, το όνομα του προγράμματος Test.java είναι το πρώτο όρισμα του προγράμματος javac που εκτελείται από την γραμμή εντολών
- Έστω ότι επιθυμούμε να κατασκευάσουμε ένα πρόγραμμα που να δέχεται ένα ή περισσότερα ορίσματα από τη γραμμή εντολών π.χ. το πρόγραμμα TestArguments το οποίο να εκτελείται από τη γραμμή εντολών ως εξής:
- Java TestArguments one two three
- Οι λέξεις one two three είναι το πρώτο, το δεύτερο και το τρίτο όρισμα του προγράμματος TestArguments. Τα ορίσματα αυτά αποθηκεύονται αυτόματα ως αλφαριθμητικά μέσα στη μεταβλητή args που χρησιμοποιούμε στο ορισμό της μεθόδου main

Ορίσματα από την γραμμή εντολών (2/2)

- Η μεταβλητή args είναι ορισμένη ως πίνακας αλφαριθμητικών. Στο παρακάτω παράδειγμα φαίνεται ένα πρόγραμμα που διαβάζει αυτά τα ορίσματα και μετά τα τυπώνει στην οθόνη.

```
class TestArguments {
    public static void main(String[ ] args) {
        for (int i=0; i<args.length; i++) {
            System.out.println(args[i]);
        }
    }
}
```