# Πίνακες - Arrays

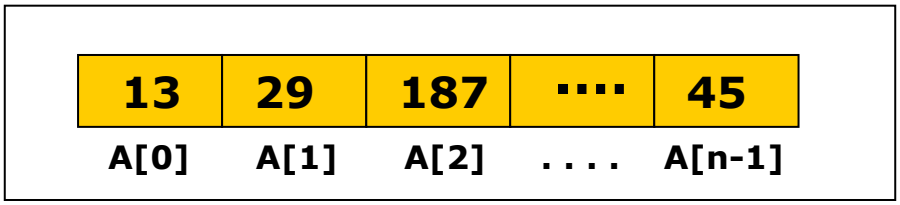
Είναι χώροι της μνήμης για προσωρινή αποθήκευση δεδομένων του ίδιου τύπου. Οι πίνακες είναι δομές δεδομένων που τις συναντάμε σχεδόν σε όλες τις γλώσσες προγραμματισμού με μόνη διαφορά ότι στη Java οι πίνακες είναι **αντικείμενα**. Οι πίνακες έχουν προκαθορισμένο μέγεθος. Σε κάθε κελί του πίνακα μπορεί 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, διδιάστατοι** και **πολυδιάστατοι πίνακες**.

## Μονοδιάστατοι Πίνακες

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]**.

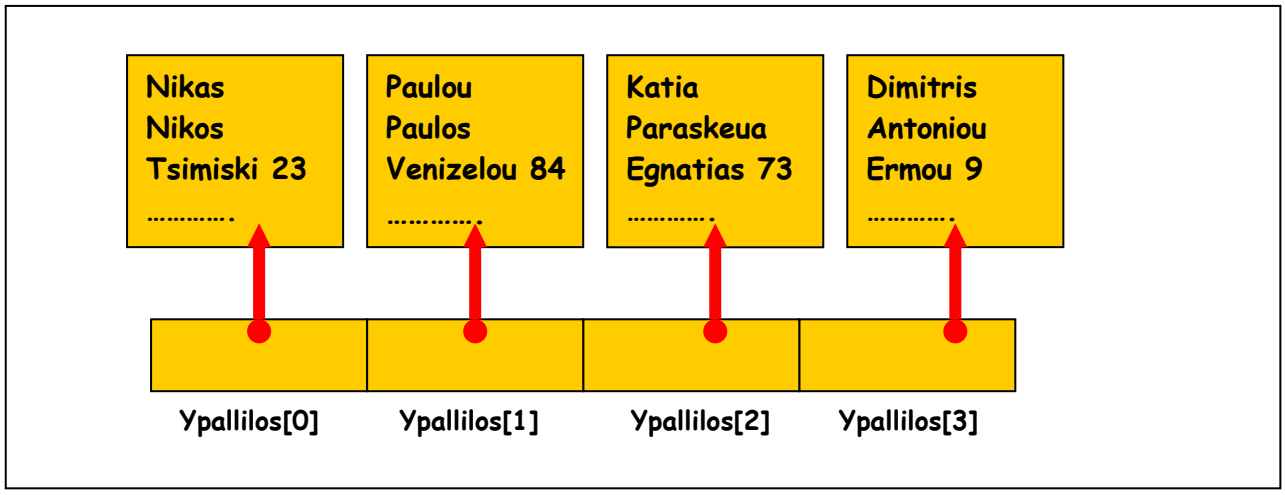
### Πίνακες με τιμές βασικών τύπων

Οι τιμές των βασικών τύπων της Java αποθηκεύονται στα κελιά του πίνακα.



## 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

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;**

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 αντικει-

μένου (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

**1) numbers[3] = 24;**

Ορίζει την τιμή 24 στο 4ο κελί του πίνακα numbers.

**2) numbers[6] = numbers[3];**

Εκχωρεί το περιεχόμενο του 4ου κελιού στο 7ο κελί. Στο παράδειγμα μας τον αριθμό 24.

**3) System.out.println(numbers[6]);**

Εμφανίζει το περιεχόμενο του 7ου κελιού, δηλαδή τον αριθμό 24.

**4) 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**

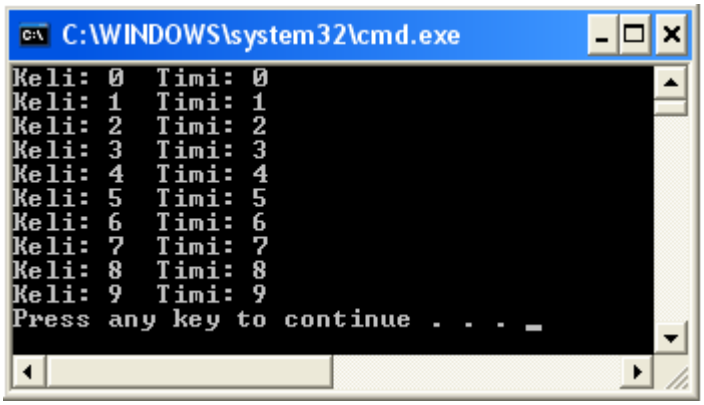
**char epiloges[] = {'a', 'b', 'c', 'd', 'e', 'f'};**

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

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 1o 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]);
  }
}
```

Το αποτέλεσμα του προγράμματος:




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);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το παρακάτω αποτέλεσμα:

```
O mesos oros einai: 3.22
```

## Το χαρακτηριστικό length στους πίνακες

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:

```
for(int i = 0; i < numbers.length; i++)
```

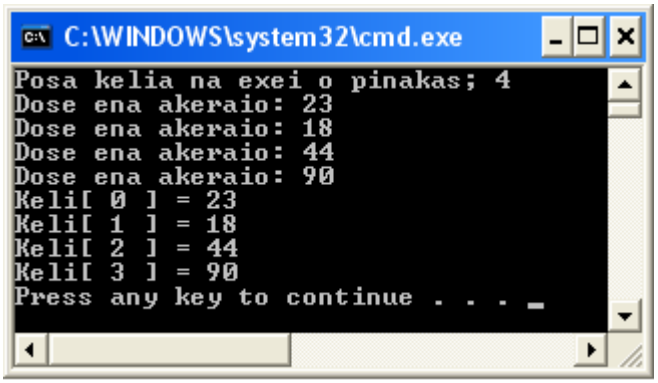
*Όπου,* **numbers.length** δηλώνει το πλήθος των κελιών του πίνακα numbers.

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.

```
import java.io.* ;

class PinakasArithmon  {
  public static void main ( String[] args ) throws IOException {
    BufferedReader inData = new BufferedReader(new InputStreamReader(System.in));
    int[] array;

    // είσοδος του πλήθους κελιών
    System.out.print( "Posa kelia na exei o pinakas; " );
    int size  = Integer.parseInt(inData.readLine());

    // ορισμός του πίνακα
    array  = new int[size];

    // είσοδος των δεδομένων
    for(int i=0; i < array.length; i++) {
      System.out.print( "Dose ena akeraio: " );
      array[i] = Integer.parseInt(inData.readLine());
    }
```

```
    // εμφάνιση των αποτελεσμάτων
    for(int i=0; i < array.length; i++ ) {
    System.out.println( "Keli[ " + i + " ] = " + array[i]);
    }
  }
}
```

Το αποτέλεσμα του προγράμματος:




## Πίνακες πολλαπλών διαστάσεων

Η Java υποστηρίζει πίνακες περισσοτέρων διαστάσεων, υιοθετώντας τη φιλοσοφία: *πίνακες αποτελούμενοι από πίνακες.* Για να ορίσουμε μια νέα διάσταση σε πίνακα προσθέτουμε στην εντολή ορισμού του ένα νέο δείκτη μέσα σε τετραγωνικές αγκύλες.

## Διδιάστατοι πίνακες

Για να ορίσουμε ένα διδιάστατο 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];
```

Ο πίνακας Dis έχει 12 – κελιά. Ο αριστερός δείκτης δείχνει τις γραμμές και ο δεξιός τις στήλες. Μία απεικόνιση του πίνακα θα μπορούσε να είναι:

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

Εκχώρηση μερικών τιμών στα κελιά του πίνακα Dis:

Dis[0][0] = 18;

Dis[0][2] = 24;

Dis[1][1] = 65;

Dis[1][3] = 90;

Dis[2][0] = 18;

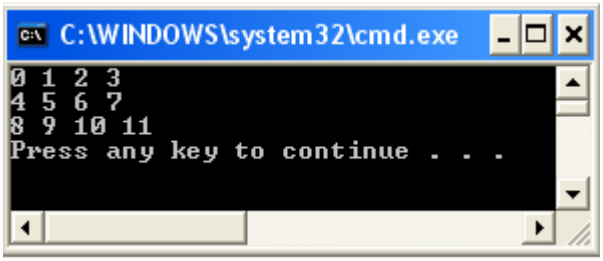
Dis[2][2] = 39;

## Παράδειγμα:

Στο παρακάτω παράδειγμα θα δημιουργήσουμε ένα τέτοιο διδιάστατο πίνακα (3 * 4 = 12 κελιών) και θα αριθμήσουμε αντίστοιχα το κάθε κελί. Στο τέλος θα εμφανίσουμε τα αποτελέσματα. ***Παρατηρήστε τη χρήση των εστιασμένων for – εντολών που ενδείκνυνται σαν οι πλέον κατάλληλες εντολές επανάληψης για την επεξεργασία διδιάστατων πινάκων.***

```
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();
    }
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα:

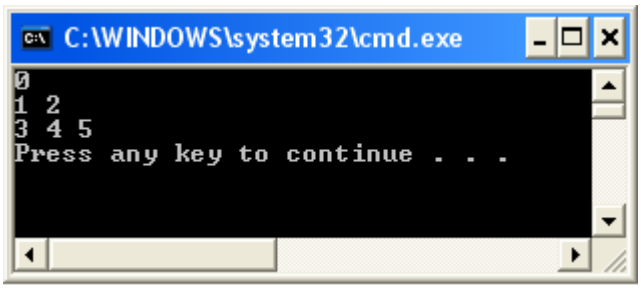



## Ragged πίνακες – Διαφορετικές στήλες για κάθε γραμμή

Κάθε γραμμή ενός διδιάστατου 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 διδιάστατο πίνακα στον οποίο τα μεγέθη της δεύτερης διάστασης δεν είναι ίσα.

```
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();
    }
  }
}
```

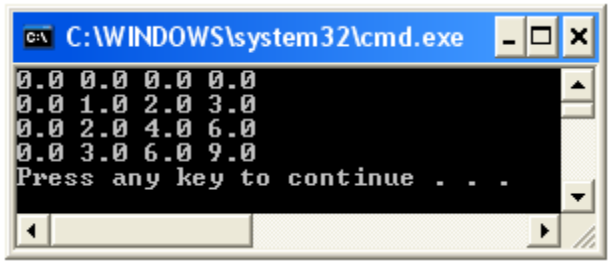
Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα :




Στο παρακάτω παράδειγμα θα δημιουργήσουμε ένα διδιάστατο πίνακα που θα έχει ως αρχικές τιμές στα κελιά του το γινόμενο των δεικτών του.

```
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();
    }
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα :

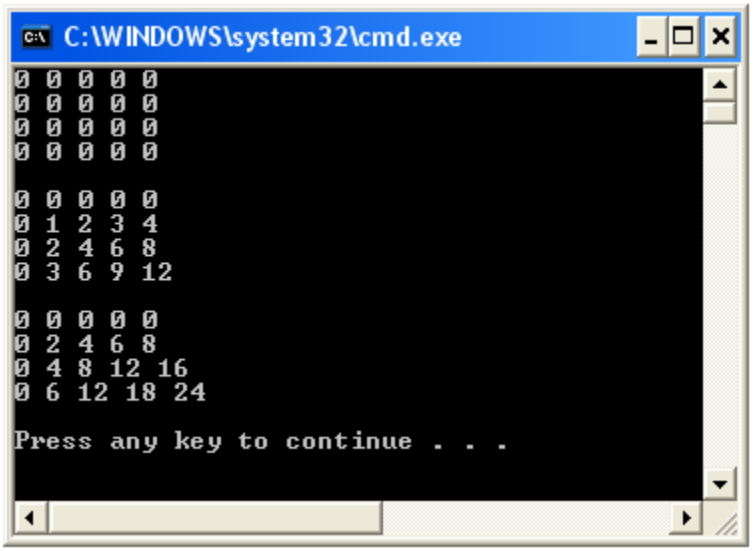



## Τρισδιάστατοι πίνακες

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

```
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();
    }
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα:




## Πίνακας ως παράμετρος σε μέθοδο

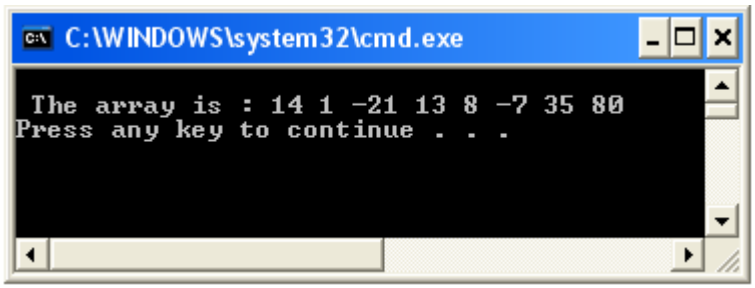
Ένας πίνακας μπορεί να περάσει σαν παράμετρος εισόδου δεδομένων σε μία μέθοδο. Μια μέθοδος με παράμετρο ένα πίνακα πρέπει να καθορίζει μόνο τον τύπο δεδομένων του πίνακα και όχι το μέγεθός του. Π.χ. **public static void arithmoi(int[] a),** εδώ ένα πίνακα ακεραίων.

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.

```
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);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το παρακάτω αποτέλεσμα :

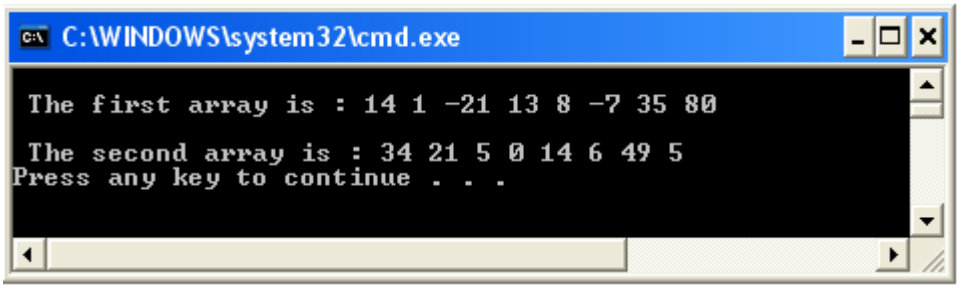



Μια παραλλαγή της προηγούμενης άσκησης είναι το πέρασμα στην ίδια μέθοδο ενός δεύτερου πίνακα με διαφορετικά δεδομένα.

```
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[] myarr1 =  { 14, 1, -21, 13, 8, -7, 35, 80 } ;
    int[] myarr2 =  { 34, 21, 5, 0, 14, 6, 49, 5 } ;
    System.out.print  ("\n The first array is : " );
    operate.print(myarr1);
    System.out.print  ("\n The second array is : " );
    operate.print(myarr2);
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα:




Μία άλλη επίσης παραλλαγή της άσκησης είναι και η παρακάτω όπου εμφανίζουμε ένα συγκεκριμένο πλήθος στοιχείων του πίνακα. Η μέθοδος printRange() δέχεται σαν παραμέτρους τον πίνακα και τις τιμές αρχής και τέλους εμφάνισης.

```
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 );
  }
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα:


Μία βελτιωμένη παραλλαγή της εντολής for είναι:

```
for (int i=start; i < end && i >= 0 && i < x.length; i++)
```

# Πίνακες Αντικειμένων *(αναφορών στα αντικείμενα)*

## Παράδειγμα – 1ο

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-doule[], τον οποίο θα επιστρέψει στο κυρίως πρόγραμμα. Τέλος (3) θα εμφανίσουμε τα αποτελέσματα (όγκους) των 3-αντικειμένων (εμφάνιση του επιστρεφόμενου πίνακα), στο κυρίως πρόγραμμα.

**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 **δομητή** της κλάσης Box ή να δώσουμε αρχικές τιμές μέσα στο πρόγραμμα, χωρίς δομητή, όπως κάναμε και στα πρώτα παραδείγματα των αντικειμένων τύπου Box (δες κεφ. 3). Με τον πλήρη δομητή:

```
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);
```

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);
```

Προσοχή στην σύνταξη της **μεθόδου volume().**

```
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;}
```

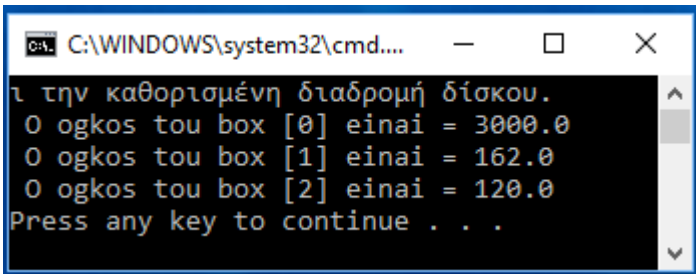
Ο πίνακας που θα επιστρέψει η μέθοδος είναι τύπου double[], ενώ θα δεχτεί σαν παράμετρο τον πίνακα αντικειμένων τύπου Box (εδώ με το όνομα a[]). Για τον υπολογισμό των 3-όγκων και την τοποθέτηση τους σε πίνακα θα χρειαστούμε ένα 'τοπικό' πίνακα, εδώ τον Vol[] ο οποίος **για κάθε κελί** (κάθε τιμή του i, σύνολο 3-τιμές) θα παίρνει την τιμή:

**Vol[i]=a[i].width * a[i].height * a[i].depth;**

**Μια πιθανή λύση:**

```
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;
}}

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]); }
}}
```

**Το αποτέλεσμα:**




## Παράδειγμα – 2ο

Υπολογισμός του Ακαθάριστου Μισθού των 4 υπαλλήλων (αντικειμένων) μιας εταιρίας. Τα πεδία του κάθε υπάλληλου είναι: (1) name, String, (2) meresErgasias, int, (3) hmeromisthio, double, και (4) yperories, double. Ο υπολογισμός του Ακαθάριστου Μισθού θα γίνει σε μια στατική μέθοδο την AKM() που θα δέχεται σαν παράμετρο τον πίνακα των αντικειμένων (υπαλλήλων) και σύμφωνα με την σχέση:

**AKM = meresErgasias* hmeromisthio + 0.2 * hmeromisthio * yperories**

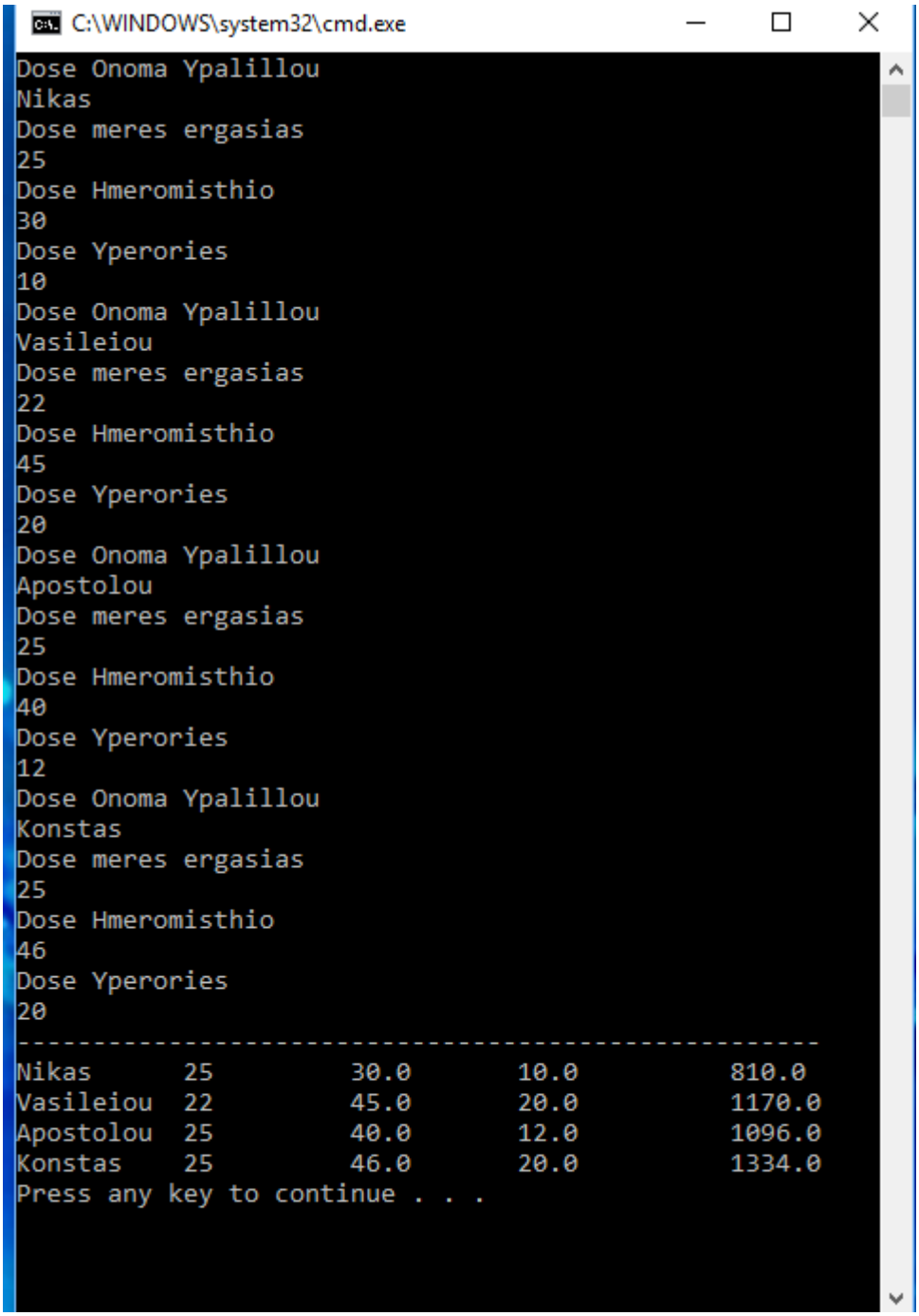
Οι ακαθάριστοι μισθοί θα αποθηκεύονται σε πίνακα τύπου double που θα επιστρέφεται στο κυρίως πρόγραμμα (main()). Το πρόγραμμα θα εμφανίζει όλα τα στοιχεία των υπαλλήλων και τους ακαθάριστους μισθούς τους υπό μορφή πίνακα.

**Μια πιθανή λύση:**

```
class Employee {
  private String name;
  private int meresErgasias;
  private double hmeromisthio;
  private double yperories;

  Employee(String s, int m, double h, double y)
   {name=s; meresErgasias=m; hmeromisthio=h; yperories=y;}
```

```
public static double[] AKM(Employee e[]) {
   double p[] = new double[4];
   for (int i=0;i<4; i++)
    p[i]=e[i].meresErgasias*e[i].hmeromisthio +
              0.2*e[i].hmeromisthio * e[i].yperories;
   return p;
 }

 public String toString()
  {return String.format("%-10s %-10s %-10s %-10s", name,
                        meresErgasias,hmeromisthio,yperories);}
 }
class ArrayOfObjects {
  public static void main(String args[]){
    Employee[] arr = new Employee[4];
      for( int i=0; i<4; i++ ){
          System.out.println("Dose Onoma Ypalillou");
          String s = UserInput.getString();
          System.out.println("Dose meres ergasias");
          int m = UserInput.getInteger();
          System.out.println("Dose Hmeromisthio");
          double h = UserInput.getInteger();
          System.out.println("Dose Yperories");
          double y = UserInput.getInteger();
          arr[i] = new Employee(s,m,h,y);
      }

     double p[] = new double[4];
     p = Employee.AKM(arr);
     System.out.println("-----------------------------------------");
     for( int i=0; i<4; i++ )
      System.out.println(arr[i]+"    "+p[i]);  } }
```

**Το αποτέλεσμα:**



# Η κλάση Arrays

Η κλάση **Arrays** περιέχει σημαντικές μεθόδους χειρισμού των πινάκων:

## 1) 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>, <Obs.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 |
| *<Obs.target>,* | Ο πίνακας αντίγραφο που θα παραχθεί |
| *<int targetStart>,* | Από ποιο στοιχείο του αντιγράφου θα αρχίσει η αντιγραφή |
| *<int size>);* | Πόσα στοιχεία θα αντιγράψει |

Στο παρακάτω απλό παράδειγμα θα δούμε την χρήση της arraycopy().

```
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);
}
}
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε το παρακάτω αποτέλεσμα:

```
xybcd
```

Ένα ακόμη παράδειγμα με τη χρήση της arraycopy:

```
public class ArrayCopyDemo  {
  public static void main(String[] args)  {
      int ar[] = {0,0,0,0,0,0,0,0,0,0};

      for (int i = 0; i < 10; ++i) {
        System.arraycopy(ar,0,ar,1,9);
        ar[0] = i;
      }
      System.out.print("ar = ");
      for (int i = 0; i < 10; ++i) {
        System.out.print(ar[i] + " ");
      }
      System.out.println("");
    }
 }
```

Αν εκτελέσουμε το πρόγραμμα θα πάρουμε τα παρακάτω αποτελέσματα :

```
ar = 9 8 7 6 5 4 3 2 1 0
```

## 2) Η μέθοδος fill()

Α) Η μέθοδος αυτή 'γεμίζει' τον πίνακα με κάποια τιμή. Π.χ.

**int[] arr1 = new int[10];**

**Arrays.fill(arr1, 99);**

Σε αυτό το παράδειγμα ο πίνακας arr1 - 10 ακέραιων αριθμών – 'γεμίζει' με την τιμή 99 σε όλα του τα κελιά.

Β) **Arrays.fill(arr1, 2, 4, 99);**

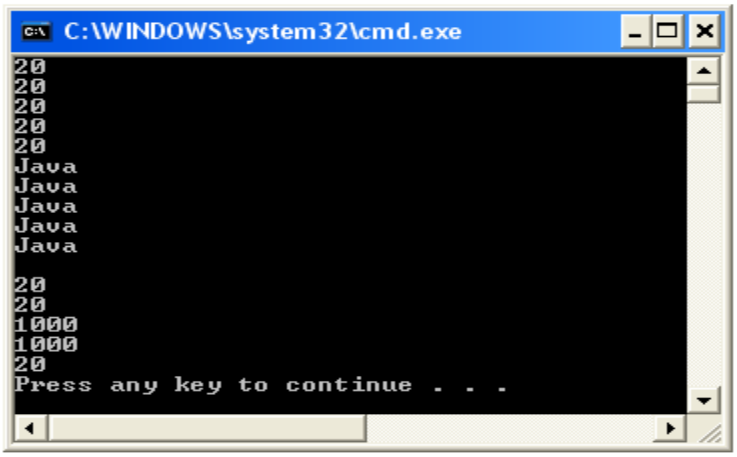
Σε αυτό το παράδειγμα ο πίνακας arr1 - 10 ακέραιων αριθμών – 'γεμίζει' με την τιμή 99 από την θέση 2 μέχρι την θέση 4 του πίνακα.

**Παράδειγμα:**

```
import java.util.*;
public class FillTest {
  public static void main(String args[]) {
    int array1[] = new int[5];
    String array2[] = new String[5];
    Arrays.fill(array1, 20);
    Arrays.fill(array2, "Java");
    for (int i=0, n=array1.length; i<n; i++) {
      System.out.println(array1[i]);
    }
    for (int i=0, n=array2.length; i<n; i++) {
```

```
      System.out.println(array2[i]);
    }
    System.out.println();
    Arrays.fill(array1, 2, 4, 1000);
    for (int i=0, n=array1.length; i<n; i++) {
      System.out.println(array1[i]);
    }
  }
}
```

Τα αποτελέσματα του προγράμματος:




## 3) 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[] myArr1;
boolean[] myArr2;
myArr1 = new boolean[]{true, false};
myArr2 = new boolean[]{true, false};
boolean b = Arrays.equals(myArr1, myArr2);  // επιστρέφει true
```

## 4) 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.

***(Δες περισσότερα στο API της Sun)***

**Παραδείγματα:**

```
String[] names = {"Roulis", "Takis", "Paulos"};
Arrays.sort(names);
System.out.println(Arrays.toString(names));

double[] myArr = {89.0, 36.2, 0.1, 193.2, 0.8};
Arrays.sort(myArr);
System.out.println(Arrays.toString(myArr));
```