ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κλάσεις και αντικείμενα στην Java

Strings

Πίνακες

ΚΛΑΣΕΙΣ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΑ

Κλάση

- Μια κλάση είναι μία αφηρημένη περιγραφή αντικειμένων με κοινά χαρακτηριστικά και κοινή συμπεριφορά.
  - Ένα καλούπι/πρότυπο που παράγει αντικείμενα
- Ένα αντικείμενο είναι ένα στιγμιότυπο μίας κλάσης.
- Η κλάση ορίζει τον τύπο του αντικειμένου.
  - Τα χαρακτηριστικά του αντικειμένου
  - Τις ενέργειες που μπορεί να επιτελέσει.

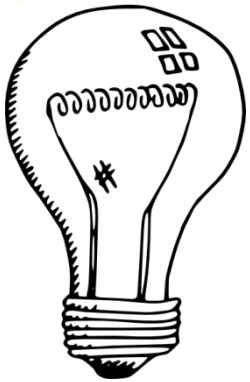
Πρακτικά στον κώδικα

- Μία κλάση Κ ορίζεται από
  - Κάποιες μεταβλητές τις οποίες ονομάζουμε πεδία
  - Κάποιες συναρτήσεις που τις ονομάζουμε μεθόδους.
    - Οι μέθοδοι «βλέπουν» τα πεδία της κλάσης

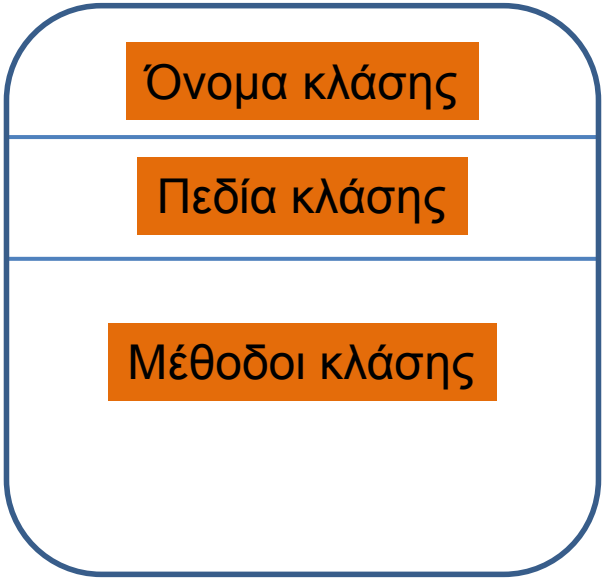
μέλη της κλάσης

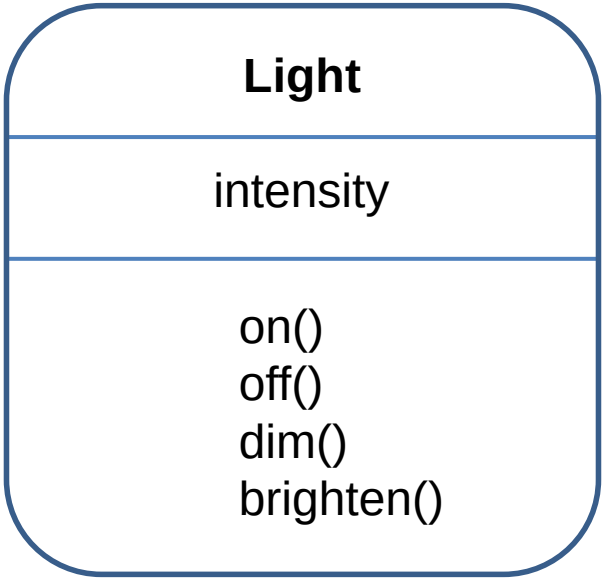
- Ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου Κ
  - Το αντικείμενο έχει συγκεκριμένες τιμές στα πεδία.
  - Στο πρόγραμμα έχουμε (συνήθως) πρόσβαση μόνο τις μεθόδους.
    - Μέσω των μεθόδων έχουμε πρόσβαση στα πεδία
  - Αν υπάρχουν κάποια πεδία στα οποία έχουμε πρόσβαση αυτά τα λέμε properties.

Γενική μορφή της κλάσης



Θέλουμε να κάνουμε ένα πρόγραμμα που να διαχειρίζεται τα φώτα σε διάφορα δωμάτια και θα υλοποιεί και ένα dimmer

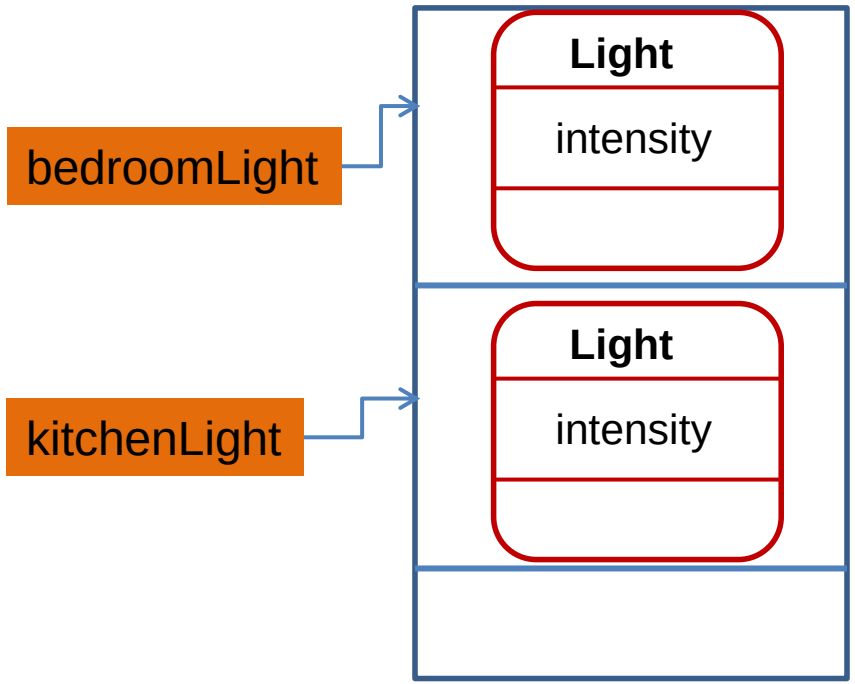




Δημιουργία αντικειμένων

```
Light bedroomLight = new Light();
Light kitchenLight = new Light();
```

- Με την εντολή new δημιουργούμε ένα καινούριο αντικείμενο της κλάσης και του δίνουμε ένα όνομα.
- Η new δεσμεύει χώρο μνήμης για το αντικείμενο
  - Μας επιστρέφει την διεύθυνση του χώρου που δεσμεύτηκε.
- Η μεταβλητή που ορίζουμε "δείχνει" σε αυτό τον χώρο μνήμης



Κλήση μεθόδων

- Ή πρόσβαση που έχουμε στα αντικείμενα είναι (κατά κύριο λόγο) μέσα από τις μεθόδους τους.
- Η κλήση μιας μεθόδου
  - `<όνομα αντικειμένου>.<όνομα μεθόδου>`
- Π.χ.

```
Light bedroomLight = new Light();
bedroomLight.on();
bedroomLight.brighten();
bedroomLight.dim();
bedroomLight.off();
```

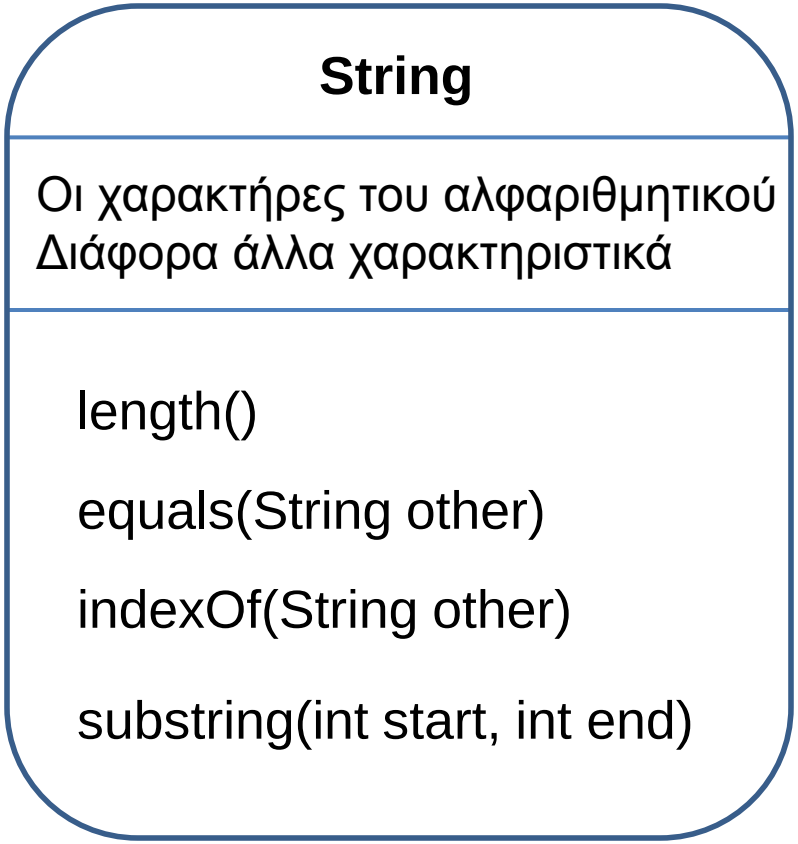
Κλήση μεθόδων

- Για να καλέσουμε μια μέθοδο μιας κλάσης θα πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης
- Εξαίρεση: οι στατικές μέθοδοι
  - Μπορούν να κληθούν χρησιμοποιώντας το όνομα της κλάσης
- Παράδειγμα:
  - Η μέθοδος main που έχουμε δει καλείται χωρίς να έχουμε δημιουργήσει αντικείμενο
  - Γι αυτό και πρέπει να οριστεί ως static.

ΥΠΑΡΧΟΥΣΕΣ ΚΛΑΣΕΙΣ

Strings

- Έχουμε ήδη χρησιμοποιήσει κλάσεις και αντικείμενα όταν χρησιμοποιούμε Strings



Η ακριβής αναπαράσταση του αλφαριθμητικού δεν έχει και τόσο σημασία εφόσον εμείς χρησιμοποιούμε μόνο τις μεθόδους.

String αντικείμενα

- Ένα String αντικείμενο είναι μια μεταβλητή τύπου String.
  - Τρεις διαφορετικοί τρόποι να δώσουμε τιμή σε ένα String object

```
import java.util.Scanner;

class StringExample{
  public static void main(String[] args){
    Scanner input = new Scanner(System.in);

    String x = input.next();
    String z = new String("java");
    String y = "java”;
  }
}
```

String μέθοδοι

- Υπάρχουν πολλές χρήσιμες μέθοδοι της κλάσης String.
  - length(): μήκος του String
  - equals(String x): τσεκάρει για ισότητα του String που καλεί την μέθοδο με το String x
  - trim(): αφαιρεί κενά στην αρχή και το τέλος του string.
  - split(char delim): σπάει το string σε πίνακα από strings με βάσει τον χαρακτήρα delim.
  - indexOf(String s): Επιστρέφει την θέση της πρώτης εμφάνισης του s μέσα στο String που καλεί την μέθοδο
  - substring(int start, int end): Επιστρέφει το υπο-string μέσα στο String που καλεί την μέθοδο μεταξύ των θέσεων start και end
  - Κλπ.

Παράδειγμα

```
class StringExample{
  public static void main(String[] args){
    String x = new String("introduction to java programming");
    String y = "java";

    int offset = x.indexOf(y);
    int end = x.length();
    x = x.substring(offset,end);
    System.out.println(x);
  }
}
```

Τα Strings είναι αμετάβλητα (immutable) αντικείμενα
Η τελευταία ανάθεση δημιουργεί ένα καινούριο αντικείμενο και το αναθέτει στην μεταβλητή x

Αμετάβλητα αντικείμενα

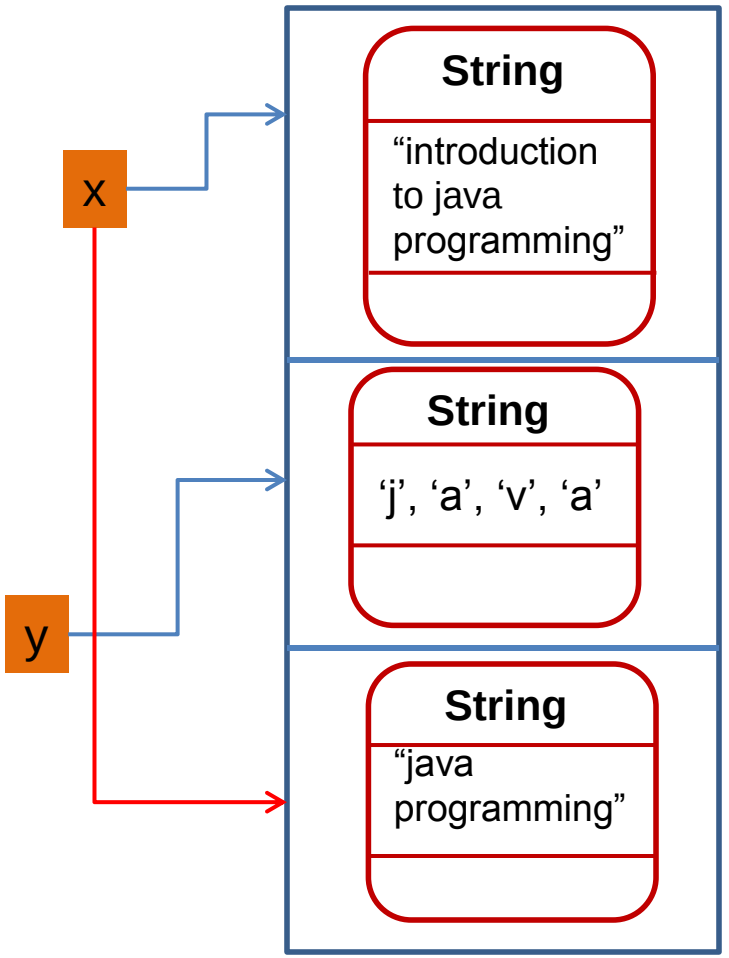
- Τα αμετάβλητα αντικείμενα (immutable objects) είναι αντικείμενα των οποίων η εσωτερική κατάσταση (ουσιαστικά τα πεδία τους) δεν μπορεί να μεταβληθεί.
- Τα Strings είναι αμετάβλητα αντικείμενα
  - Αυτό σημαίνει ότι δεν μπορούμε να αλλάξουμε τα περιεχόμενα ενός αντικειμένου String
    - Π.χ., δεν μπορούμε να αλλάξουμε ένα χαρακτήρα ενός String
  - Ότι αλλαγή κάνουμε έχει αποτέλεσμα να δημιουργείται ένα καινούριο String και να εκχωρείται στην μεταβλητή μας.

Αμετάβλητα αντικείμενα

```
String x = new String("introduction to java programming");
String y = "java";
x = x.substring(offset,end);
```

Τα Strings είναι αμετάβλητα (immutable) αντικείμενα

Η τελευταία ανάθεση δημιουργεί ένα καινούριο αντικείμενο και το αναθέτει στην μεταβλητή x



Ισότητα String

Τι θα εκτυπωθεί?
(μια λογική συνθήκη τυπώνει true/false ανάλογα αν είναι αληθής/ψευδής)

```
import java.util.Scanner;

class StringEquality{
  public static void main(String[] args){
    String x = new String(“java”);
    String y = new String("java");
    String z = y;

    System.out.println("1. "+ (x == y));
    System.out.println("2. "+ (y == z));
    System.out.println("3. "+ (z == x));
    System.out.println("4. "+ x.equals(y));
    System.out.println("5. "+ y.equals(z));
    System.out.println("6. "+ z.equals(x));
  }
}
```

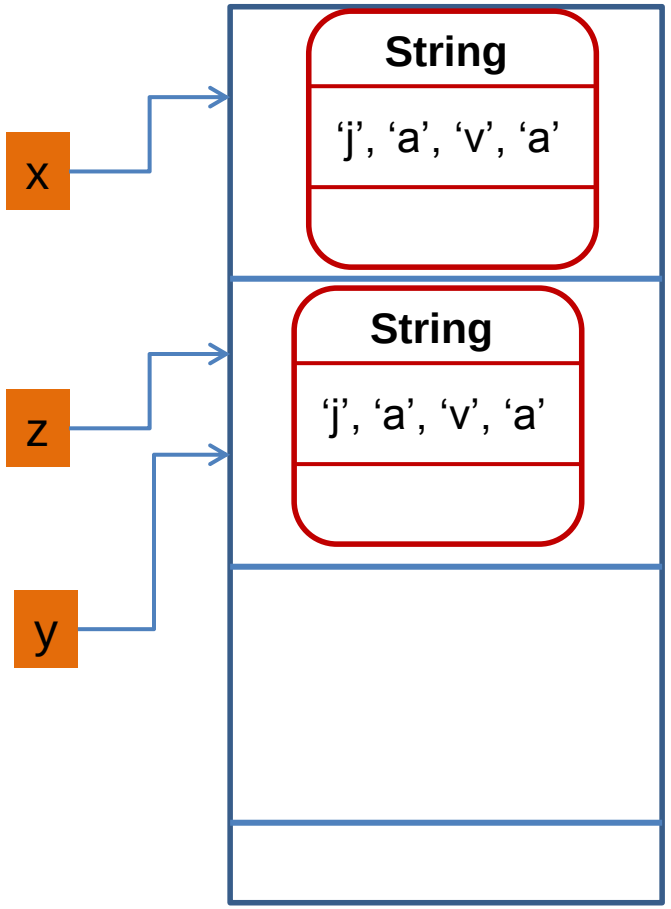
1. false
2. true
3. false
4. true
5. true
6. true

Για την σύγκριση Strings **ΠΑΝΤΑ** χρησιμοποιούμε την μέθοδο equals.

Εξήγηση

```
String x = new String("java");
String z = new String("java");
String y = z;
```

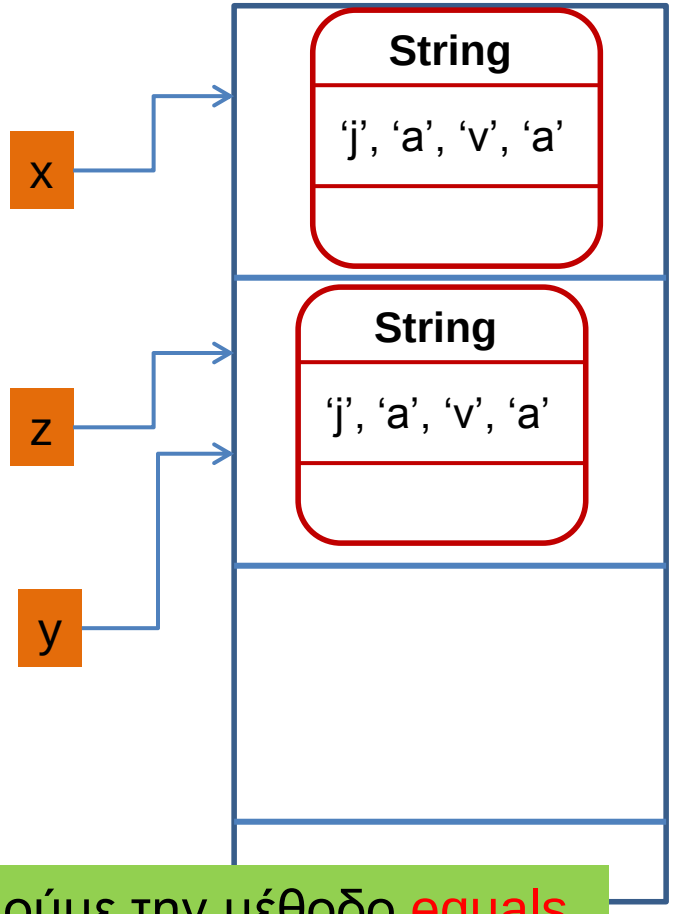
- Όταν δημιουργούμε ένα String αντικείμενο δεσμεύουμε χώρο στη μνήμη για το αντικείμενο
- Η μεταβλητή που ορίζουμε «δείχνει» σε αυτό το χώρο μνήμης
- Η εκχώρηση μεταξύ αντικειμένων τα κάνει να δείχνουν στην ίδια θέση μνήμης



Εξήγηση

```
String x = new String("java");
String z = new String("java");
String y = z;
System.out.println("2. "+ (y == z));
```

- Ο τελεστής == μεταξύ δύο αντικειμένων εξετάζει αν δείχνουν στην ίδια θέση μνήμης.
- Γι αυτό (`y == z`) επιστρέφει true.
- Όλα αυτά θα είναι πιο ξεκάθαρα όταν θα μιλήσουμε για αναφορές.



Για την σύγκριση Strings **ΠΑΝΤΑ** χρησιμοποιούμε την μέθοδο equals.

String σταθερές

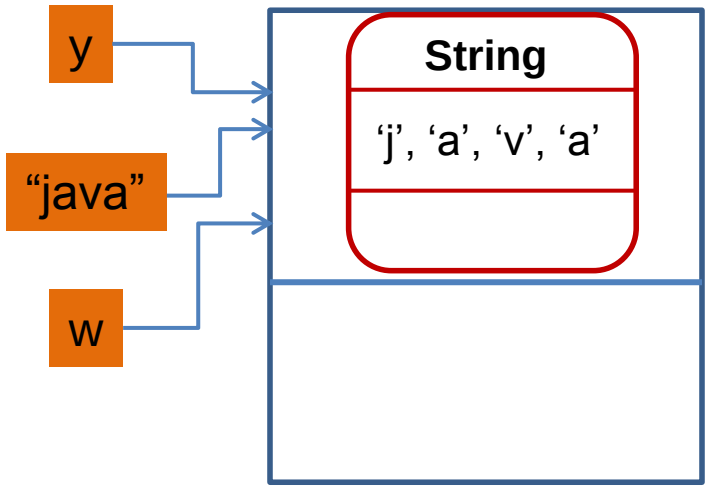
- Οι String τιμές είναι κι αυτές αντικείμενα και μπορούμε να καλέσουμε τις μεθόδους τους

```
import java.util.Scanner;

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

    int offset = “java programming".indexOf(“pro”);
    int end = "java programming".length();
    String z = "java programming".substring(offset,end);
    System.out.println(z);

  }
}
```

String σταθερές

- Ο ορισμός της σταθεράς **"java"** δημιουργεί ένα αντικείμενο με αυτό το όνομα
  - Intern String
- Οι εκχωρήσεις **y = "java"** και **w = "java"** κάνει τις μεταβλητές **y** και **w** να δείχνουν σε αυτό το αντικείμενο.
- Γι αυτό και η σύγκριση **y == w** επιστρέφει true.
- Θα τα ξαναδούμε αυτά όταν θα μιλήσουμε για αναφορές.

```
String y = "java";
String w = "java";
System.out.println((y == w);
```



Scanner

- Δημιουργία αντικειμένου Scanner
  - `Scanner input = new Scanner(System.in);`

- Μέθοδοι της Scanner:
  - **`next()`**: επιστρέφει το επόμενο String από την είσοδο (όλοι οι χαρακτήρες από το σημείο που σταμάτησε την προηγούμενη φορά μέχρι να βρει white space: κενο, tab, αλλαγή γραμμής)
  - **`nextInt()`**: διαβάζει το επόμενο String και το μετατρέπει σε int και επιστρέφει ένα int αριθμό.
  - **`nextDouble()`**: διαβάζει το επόμενο String και το μετατρέπει σε double και επιστρέφει τον double αριθμό.
  - **`nextLine()`**: Διαβάζει ότι υπάρχει μέχρι να βρει newline και το επιστρέφει ως String.

Wrapper classes

- Για κάθε βασικό τύπο η Java έχει και μία wrapper class:
  - Integer class
  - Double class
  - Boolean class
- Οι κλάσεις αυτές έχουν κάποιες μεθόδους και πεδία που μπορεί να μας είναι χρήσιμα
  - Κατά κύριο λόγο μετατροπή από και προς string
  - Τη μέγιστη και την ελάχιστη τιμή κάθε τύπου
- Κάποιες από τις μεθόδους είναι στατικές
  - Μπορούμε να τις καλέσουμε χωρίς να έχουμε αντικείμενο.

Παράδειγμα

```
class WrapperTest{
  public static void main(String args[])
  {
    int i = Integer.valueOf("2");
    double d = Double.parseDouble("2.5");
    System.out.println(i*d);
    Integer x = 5;
    Double y = 2.5;
    String s = x.toString() + y.toString();
    System.out.println(s);
    System.out.println(Integer.MAX_VALUE);
  }
}
```

ΠΙΝΑΚΕΣ

Πίνακες

- Πολλές φορές έχουμε πολλές μεταβλητές του ίδιου τύπου που συσχετίζονται και θέλουμε να τις βάλουμε μαζί.
  - Τα ονόματα των φοιτητών σε μία τάξη
  - Οι βαθμοί ενός φοιτητή για όλα τα εργαστήρια.

- Για το σκοπό αυτό χρησιμοποιούμε τους πίνακες.

Πίνακες

- Ορισμός πίνακα:

```
int[] myArray1  = {10,20};
int myArray2[] = new int[2];
int[] myArray3;
```

Ορίζει ένα πίνακα ακεραίων δύο θέσεων με αρχικές τιμές 10,20

Ορίζει ένα πίνακα ακεραίων δύο θέσεων χωρίς αρχικές τιμές (αρχικοποιούνται αυτόματα στο μηδέν)

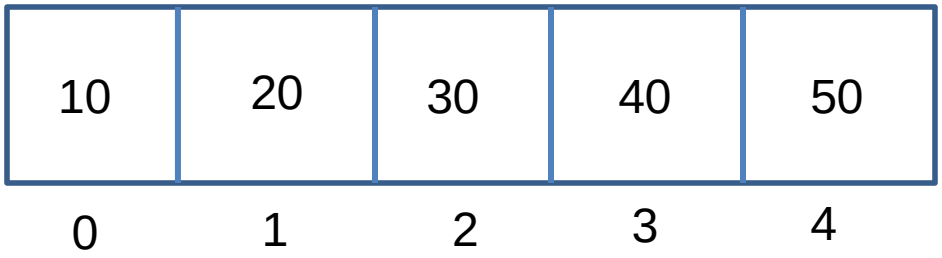
Ορίζει μια μεταβλητή που είναι πίνακας από ακεραίους αλλά δεν δεσμεύει χώρο για τον πίνακα

- Οι πίνακες ορίζονται με ένα μέγεθος (length) και αυτό δεν αλλάζει εκτός αν ορίσουμε ξανά τον πίνακα.

- Στη Java ένας πίνακας είναι ένα αντικείμενο και έχει properties
  - `System.out.println(myArray2.length);`
  - Τυπώνει το μέγεθος του πίνακα.

Πρόσβαση των στοιχείων του πίνακα

- Προσοχή! Τα στοιχεία του πίνακα αριθμούνται από το `0…length-1` (ΟΧΙ `1…length`)
  - `int myArray[] = {10,20,30,40,50};`



- Για να προσπελαύνουμε το δεύτερο στοιχείο του πίνακα
  - `myArray[1] += 5;`
  - `System.out.prinln(myArray[1]);`

Διατρέχοντας ένα πίνακα

- Στην Java έχουμε δύο τρόπους να διατρέχουμε ένα πίνακα

Διατρέχουμε τα στοιχεία

```
for (<array type> element: array)
{
   … do something with element…
}
```

```
int array[] = {1,3,5,7};
for (int element: array)
{
  System.out.println(element)
}
```

Διατρέχουμε τις θέσεις του πίνακα

```
for (int i = 0; i < array.length; i ++)
{
   … do something with array[i]…
}
```

```
int array[] = {1,3,5,7};
for (int i = 0; i < array.length; i ++)
{
  System.out.println(array[i])
}
```

Πίνακες

```
public class TestArrays1 {
   public static void main(String [] args){

      int arr0[];  // int[] arr0;

      int arr1 [] = {1, 2, 3, 4};
      for (int i = 0; i < arr1.length; i ++){
           System.out.println(arr1[i]);
      }

      int arr2[] = new int [10];
      for (int i = 0; i < arr2.length; i ++){
           arr2[i] = i+1;
      }
      arr0 = arr2;

   }
}
```

Εναλλακτικό συντακτικό

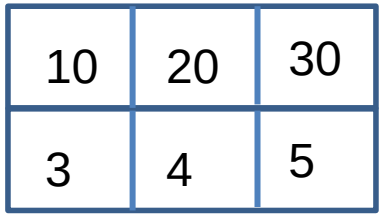
Παράδειγμα

- Τυπώστε όλα τα στοιχεία του πίνακα και όλα τα ζεύγη από στοιχεία στον πίνακα

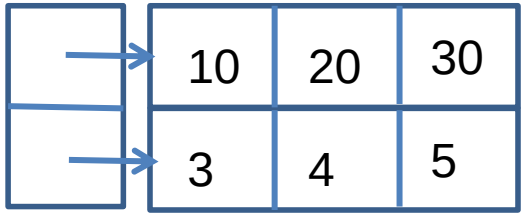
```
class ScanArray
{
    public static void main(String [] args)
    {
        double [] array = {5.3, 3.4, 2.3, 1.2, 0.1};

        // Print all elements
        for (double element: array){
            System.out.println(element);
        }

        // Print all pairs of elements
        for (int i = 0; i < array.length; i ++){
            for (int j = i+1; j < array.length; j ++){
                System.out.println(array[i] + " " + array[j]);
            }
        }
    }
}
```

Πολυδιάστατοι πίνακες

- Μπορούμε να ορίσουμε και πολυδιάστατους πίνακες
  - `int[][] myArray1 = {{10,20,30},{3,4,5}};`
  - `int[][] myArray2 = new int[2][3];`

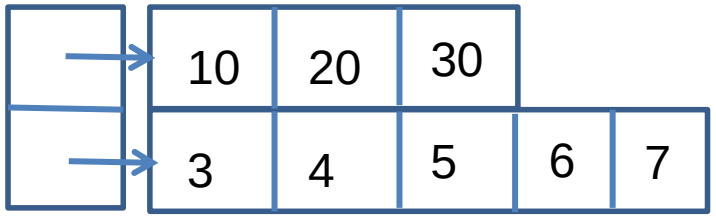


- Ένας δισδιάστατος πίνακας είναι ένας πίνακας από αντικείμενα-πίνακες.
  - `int[][] myArray3 = new int[2][]`
  - `myArray3[0] = new int[3]`
  - `myArray3[1] = new int[3]`



Η κάθε γραμμή είναι ένας πίνακας και πρέπει να αρχικοποιηθεί

- Ο πίνακας μπορεί να είναι ασύμμετρος
  - `myArray3[1] = new int[5]`



- Τι παίρνω για τα παρακάτω?
  - `System.out.println(myArray3.length);`
  - `System.out.println(myArray3[1].length);`

Πίνακες

```
public class TestArrays2 {
        public static void main(String [] args){
            int[][] arr3 = {{1, 2, 3}, {3, 4, 5}};

            int[][] arr4 = new int [10][20];

            arr4 = arr3;

            System.out.println(arr4.length + " "
                                    + arr4[0].length);

            int arr5[][] = new int[2][];
            arr5[0] = new int[3];
            arr5[1] = new int[5];

        }
}
```

Πίνακας με αρχικές τιμές

Πίνακας 10x20

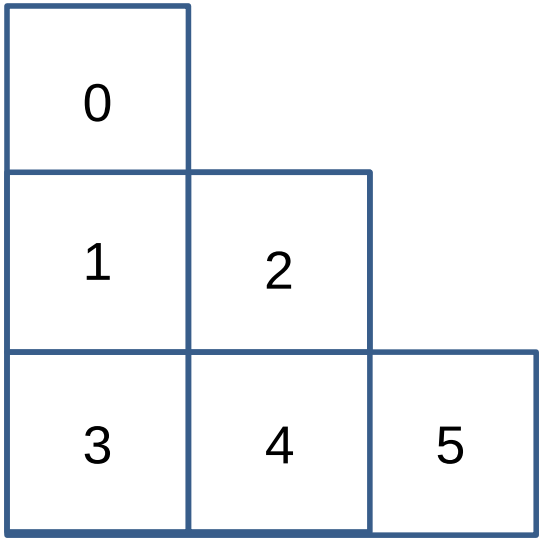
Ο πίνακας arr4 γίνεται ίδιος με τον arr3. Δηλαδή δείχνει στον ίδιο χώρο μνήμης.

Τυπώνει “2 3”

Ασύμμετρος πίνακας

Αρχικοποίηση πινάκων

- Δημιουργήστε τους παρακάτω πίνακες:
  - Ένα μονοδιάστατο πίνακα με n θέσεις με τις τιμές 0..n-1
  - Ένα δισδιάστατο πίνακα με nxn θέσεις με τις τιμές 0…n^2-1
  - Ένα κάτω διαγώνιο πίνακα nxn (π.χ. παρακάτω 3x3)
- Το μέγεθος του πίνακα θα το δίνει ο χρήστης



```
import java.util.Scanner;

class ArrayInitialization
{
	public static void main(String[] args){
		Scanner input = new Scanner(System.in);
		int n = input.nextInt();

		int[] array1d = new int[n];
		for (int i = 0; i < n; i ++){
			array1d[i] = i;
		}
		for (int i = 0; i < n; i ++){
			System.out.print(array1d[i] + " ");
		}
		System.out.println();
	}
}
```

```
import java.util.Scanner;

class ArrayInitialization
{
	public static void main(String[] args){
		Scanner input = new Scanner(System.in);
		int n = input.nextInt();

		int[][] array2d = new int[n][n];
		for (int i = 0; i < n; i ++){
			for (int j = 0; j < n; j ++){
				array2d[i][j] = i*n+j;
			}
		}
		for (int i = 0; i < n; i ++){
			for (int j = 0; j < n; j ++){
				System.out.print(array2d[i][j] + " " );
			}
			System.out.println();
		}
	}
}
```

```
import java.util.Scanner;

class ArrayInitialization
{
        public static void main(String[] args){
                Scanner input = new Scanner(System.in);
                int n = input.nextInt();

                int[][] lowerDiagonal = new int[n][];
                for (int i = 0; i < n; i ++){
                    lowerDiagonal[i] = new int[i+1];
                    for (int j = 0; j < i+1; j ++){
                       lowerDiagonal[i][j] = i*(i+1)/2 + j;
                    }
                }
                for (int i = 0; i < n; i ++){
                    for (int j = 0; j < i+1; j ++){
                       System.out.print(lowerDiagonal[i][j] + " ");
                       }
                    System.out.println();
                }
        }
}
```

Παράδειγμα με strings και πίνακες

- Φτιάξτε ένα πρόγραμμα που να διαβάζει μία γραμμή από κείμενο και να ψάχνει μία λέξη που δίνουμε σαν όρισμα μέσα σε αυτή τη γραμμή.

➢ **`java LookFor hello`**

- Περιμένει να διαβάσει μια γραμμή από κείμενο και ψάχνει τη λέξη hello μέσα στο κείμενο.

```
import java.util.Scanner;

class LookFor
{
    public static void main(String args[])
    {
       String name = "default";
       if (args.length == 1)
       {
              name = args[0];
       }
       Scanner input = new Scanner(System.in);
       String line = input.nextLine();
       String [] words = line.split(" ");
       for (int i =0; i < words.length; i ++)
       {
           if (name.equals(words[i])){
              System.out.println(name + " found it at " + i);
           }
       }
    }
}
```

Τα command-line ορίσματα του προγράμματος αποθηκεύονται στον πίνακα από Strings που είναι όρισμα στην main()

Η μέθοδος split της κλάσης String με όρισμα ένα delimiter string σπάει το String με βάση το delimiter και επιστρέφει ένα πίνακα από Strings

Στην περίπτωση αυτή σπάμε το line με βάση το κενό και παίρνουμε τις λέξεις.

Η κλάση ArrayList

- Η κλάση ArrayList ορίζει ένας δυναμικό πίνακα με μεταβλητό μέγεθος ανάλογα με τον αριθμό των στοιχείων που τοποθετούμε
  - Το ArrayList μπορεί να κρατάει αντικείμενα οποιουδήποτε τύπου.

- Συντακτικό:
  - `import java.util.ArrayList;`
  - `ArrayList<Βασικος Τύπος> myList;`

- Ο βασικός τύπος είναι οποιοσδήποτε μια οποιαδήποτε κλάση.
  - Αυτός είναι ο τύπος των δεδομένων που αποθηκεύει ο πίνακας μας.
  - Για να αποθηκεύσουμε βασικούς τύπους χρειαζόμαστε την wrapper class.

- Παραδείγματα:
  - `ArrayList<Integer> myList;` // λίστα από ακεραίους
  - `ArrayList<String> myList;` // λίστα από String
  - `ArrayList<Person> myList;` // λίστα από αντικείμενα Person

ArrayList

- Constructors
  - `ArrayList<T> myList = new ArrayList<T>();`
- Μέθοδοι
  - `add(T x)`: προσθέτει το στοιχειό `x` στο τέλος του πίνακα.
  - `add(int i, T x)`: προσθέτει το στοιχειό `x` στη θέση `i` και μετατοπίζει τα υπόλοιπα στοιχεία κατά μια θέση.
  - `get(int i)`: επιστρέφει το αντικείμενο τύπου `T` στη θέση `i`.
  - `remove(int i)`: αφαιρεί το στοιχείο στη θέση `i`
  - `remove(T x)`: αφαιρεί το στοιχείο `x`
  - `set(int i, T x)`: θέτει στην θέση `i` την τιμή `x` αλλάζοντας την προηγούμενη
  - `size()`: ο αριθμός των στοιχείων του πίνακα.
- Διατρέχοντας τον πίνακα:
  - `ArrayList<T> myList = new ArrayList<T>();`
  - `for(T x: myList){…}`

Παράδειγμα

- Ζητάμε από την είσοδο ακεραίους αριθμούς μέχρι ο χρήστης να δώσει το -1. Αποθηκεύστε τους αριθμούς σε ένα πίνακα και τυπώστε τους

- Δεν ξέρουμε εκ των προτέρων πόσους αριθμούς θα πρέπει να αποθηκεύσουμε.
  - Θα χρησιμοποιήσουμε ArrayList αντί για πίνακα.

```
import java.util.ArrayList;
import java.util.Scanner;

class ArrayListTest
{
    public static void main(String[] args){
       ArrayList<Integer> numbers = new ArrayList<Integer>();
       Scanner input = new Scanner(System.in);
       int x = input.nextInt();
       while (x != -1){
           numbers.add(x);
           x = input.nextInt();
       }

       for (Integer y: numbers){
           System.out.print(y + " ");
       }
       System.out.println();
    }
}
```