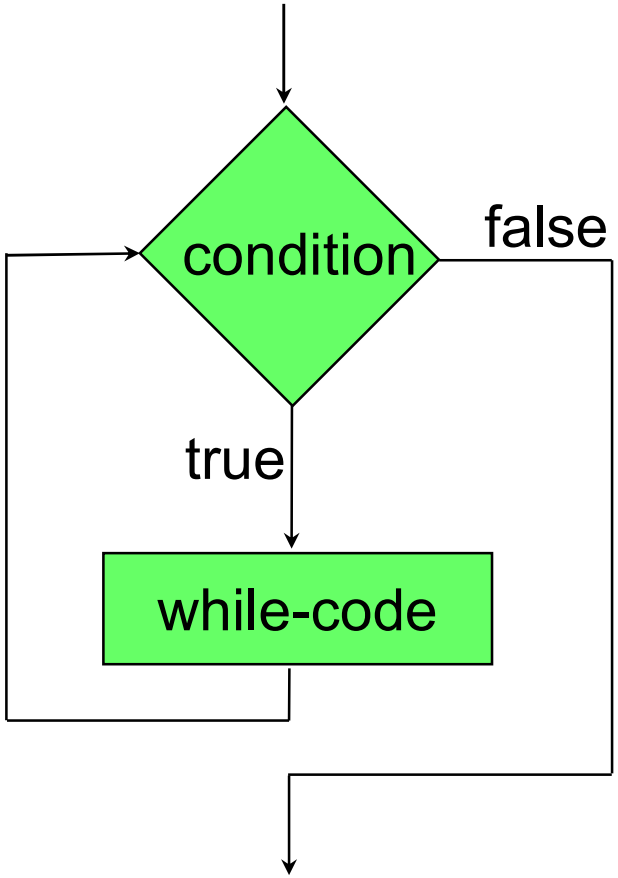
Επαναλήψεις - While statement

- Στην Java το while statement έχει το εξής συντακτικό

```
while(condition)
{
      …while-code block…
}
```

  - Αν η συνθήκη είναι αληθής τότε εκτελείται το block κώδικα while-code
  - Ο while-code block κώδικας υλοποιεί τις επαναλήψεις και αλλάζει την συνθήκη.
  - Στο τέλος του while-code block η συνθήκη αξιολογείται εκ νέου
  - Ο κώδικας επαναλαμβάνεται μέχρι η συνθήκη να γίνει ψευδής.



Παράδειγμα

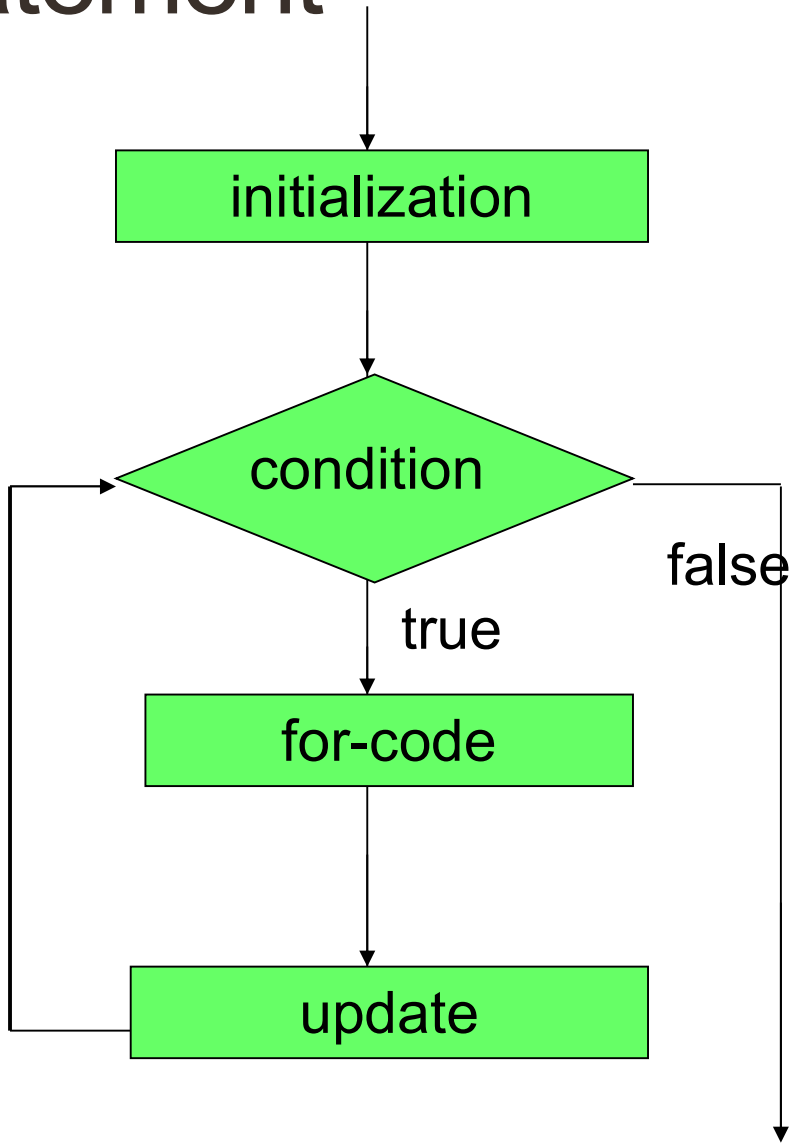
```
Scanner inputReader = new Scanner(System.in);
String input = inputReader.next();

while(input.equals("Yes"))
{
    System.out.println("Do you want to continue?");
    input = inputReader.next();
}
```

Επαναλήψεις – for statement

- Στην Java το for statement έχει το εξής συντακτικό

```
for(intialization;
    condition;
    update)
{
    …for-code block…
}
```

- Το όρισμα του for έχει 3 κομμάτια χωρισμένα με ;
  - Την αρχικοποίηση (initialization section): εκτελείται πάντα μία μόνο φορά
  - Τη λογική συνθήκη (condition): εκτιμάται πριν από κάθε επανάληψη.
  - Την ενημέρωση (update expression): υπολογίζεται μετά το κυρίως σώμα της επανάληψης.
  - Ο κώδικας επαναλαμβάνεται μέχρι η συνθήκη να γίνει ψευδής.



Παράδειγμα

```
for(int i = 0; i < 10; i = i+1)
{
    System.out.println("i = " + i);
}
```

Ορισμός της μεταβλητής `i`

Ανάθεση: υπολογίζεται η τιμή του `i+1` και ανατίθεται στη μεταβλητή `i`.

- Ισοδύναμο με while

```
int i = 0;
while(i < 10)
{
    System.out.println("i = " + i);
    i = i+1;
}
```

Παράδειγμα

```
for(int i = 0; i < 10; i ++)
{
     System.out.println("i = " + i);
     i = i+1;
}
```

i++ ισοδύναμο με το i = i+1

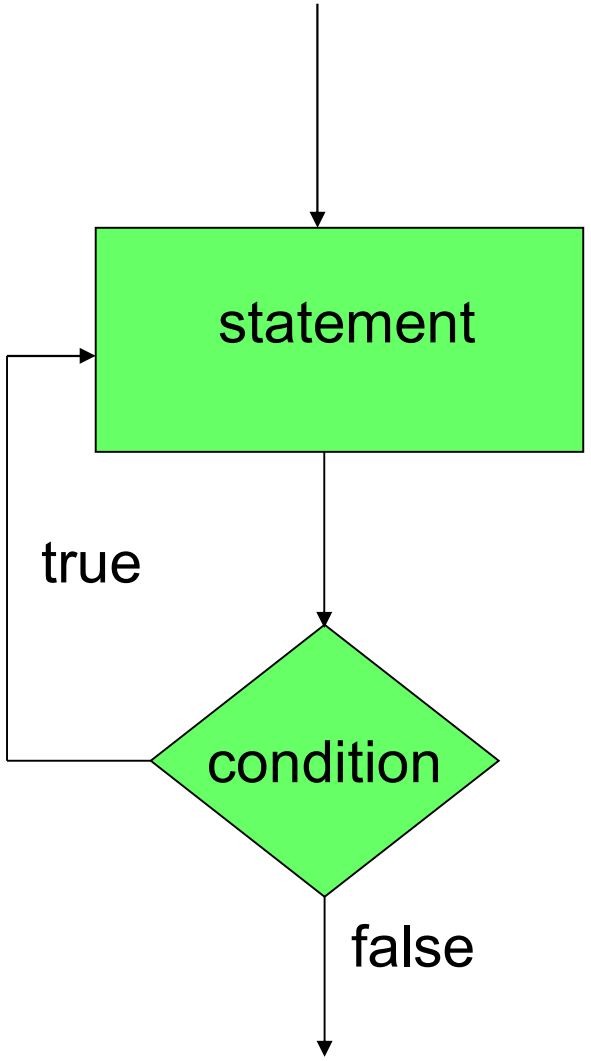
- Ισοδύναμο με while

```
int i = 0;
while(i < 10)
{
     System.out.println("i = " + i);
     i ++;
}
```

Το Do-While statement

- Ένα `do while` statement έχει το εξής συντακτικό:

```
Initialize
do
{
      …while-code block…

}while(condition)
```



- Το while code εκτελείται τουλάχιστον μία φορά; Μετά αν η συνθήκη είναι αληθής ο κώδικας εκτελείται ξανά.
- Το while code εκτελούν το βρόγχο και αλλάζουν την συνθήκη.

Παράδειγμα

- Κάνετε πρόγραμμα που παίρνει σαν είσοδο ένα αριθμό και υλοποιεί μια αντίστροφη μέτρηση. Αν ο αριθμός είναι θετικός η αντίστροφη μέτρηση γίνεται προς τα κάτω μέχρι το μηδέν, αν είναι αρνητικός γίνεται προς τα πάνω μέχρι το μηδέν. Η διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να δώσει την τιμή μηδέν.

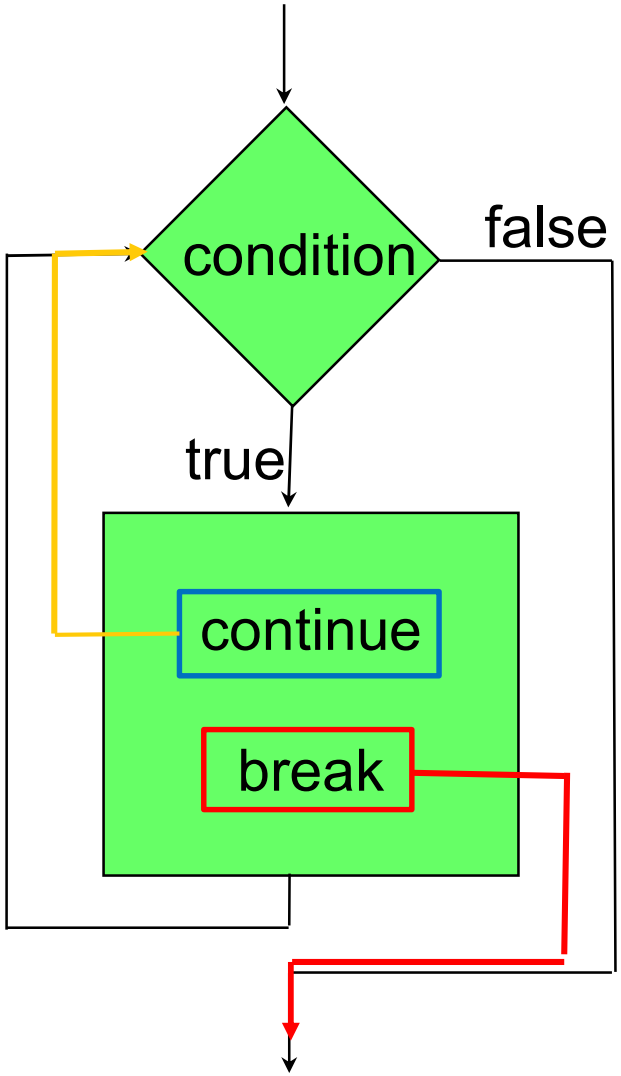
```
import java.util.Scanner;

class CountDown
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        while (inputInt != 0)
        {
            if (inputInt < 0 ){
                for (int i = inputInt; i < 0; i ++)
                {
                    System.out.println("Counter = " + i);
                }
            } else if (inputInt > 0){
                for (int i = inputInt; i > 0; i --)
                {
                    System.out.println("Counter = " + i);
                }
            }
            inputInt = reader.nextInt();
        }
    }
}
```

```
import java.util.Scanner;

class CountDown2
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt;
        do
        {
            inputInt = reader.nextInt();
            if (inputInt < 0 ){
                for (int i = inputInt; i < 0; i ++)
                {
                    System.out.println("Counter = " + i);
                }
            } else if (inputInt > 0){
                for (int i = inputInt; i > 0; i --)
                {
                    System.out.println("Counter = " + i);
                }
            }
        }while (inputInt != 0)
    }
}
```

Οι εντολές break και continue

- continue: Επιστρέφει τη ροή του προγράμματος στον έλεγχο της συνθήκης σε ένα βρόγχο.
  - Βολικό για τον έλεγχο συνθηκών πριν ξεκινήσει η εκτέλεση του βρόγχου, ή για πρόορη επιστροφή στον έλεγχο της συνθήκης
- break: Μας βγάζει έξω από την εκτέλεση του βρόχου από οποιοδήποτε σημείο μέσα στον κώδικα.
  - Βολικό για να σταματάμε το βρόγχο όταν κάτι δεν πάει καλά.
- Κάποιοι θεωρούν οι εντολές αυτές χαλάνε το μοντέλο του δομημένου προγραμματισμού.

Οι εντολές break και continue



Παράδειγμα

```
while (…)
{
  if (everything is ok){

    < rest of code>

  }// end of if
} // end of while loop
```

```
while (… && !StopFlag)
{
  < some code >

  if (I should stop){
    StopFlag = true;
  }else{
    < some more code>
  }
} // end of while loop
```

```
while (…)
{
  if (I don't like something){
      continue;
  }

  < rest of code>

} // end of while loop
```

```
while (…)
{
  < some code>

  if (I should stop){
      break;
  }

  < some code>
} // end of while loop
```

```
import java.util.Scanner;

class CountDownWithContinue
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        while (inputInt != 0)
        {
            if (inputInt%2 == 0){
                inputInt = reader.nextInt();
                continue;
            }
            if (inputInt < 0 ){
                for (int i = inputInt; i < 0; i ++)
                {
                    System.out.println("Counter = " + i);
                }
            } else if (inputInt > 0){
                for (int i = inputInt; i > 0; i --)
                {
                    System.out.println("Counter = " + i);
                }
            }
            inputInt = reader.nextInt();
        }
    }
}
```

Η αντίστροφη μέτρηση εκτελείται μόνο για περιττούς αριθμούς

```
import java.util.Scanner;

class CountDownWithBreak
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
         do
         {
             int inputInt = reader.nextInt();
             if (inputInt == 0){
                  break;
             }
             if (inputInt < 0 ){
                 for (int i = inputInt; i < 0; i ++)
                 {
                     System.out.println("Counter = " + i);
                  }
             } else if (inputInt > 0){
                 for (int i = inputInt; i > 0; i --)
                 {
                      System.out.println("Counter = " + i);
                  }
             }
         }while (true)
    }
}
```

Η συνθήκη αυτή ορίζει ένα ατέρμονο βρόγχο (infinite loop). Πρέπει μέσα στο πρόγραμμα να έχουμε μια εντολή break (ή return που θα δούμε αργότερα) για να μην κολλήσει το πρόγραμμα μας. Αυτή η κατασκευή είναι βολική όταν έχουμε πολλαπλές συνθήκες εξόδου.

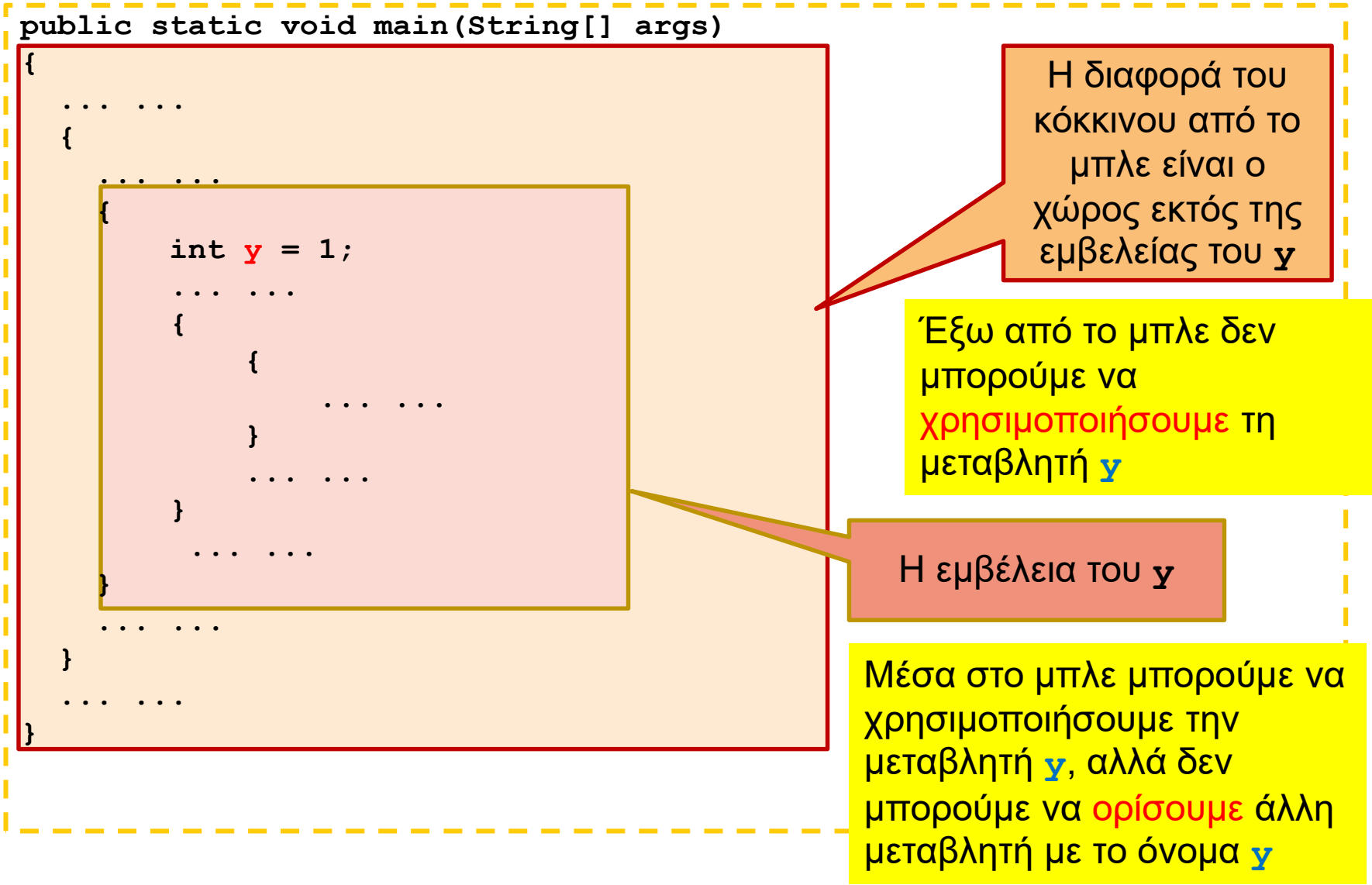
Εμβέλεια (scope) μεταβλητών

- Προσέξτε ότι η μεταβλητή `int i` πρέπει να οριστεί σε κάθε for, ενώ η `inputInt` πρεπει να οριστεί έξω από το while-loop αλλιώς ο compiler διαμαρτύρεται.
  - Προσπαθούμε να χρησιμοποιήσουμε μια μεταβλητή εκτός της εμβέλειας της
- Η κάθε μεταβλητή που ορίζουμε έχει εμβέλεια (scope) μέσα στο block το οποίο ορίζεται.
  - Τοπική μεταβλητή μέσα στο block.
- Μόλις βγούμε από το block η μεταβλητή χάνεται
  - Ο compiler δημιουργεί ένα χώρο στη μνήμη για το block το οποίο εκτελούμε, ο οποίος εξαφανίζεται όταν το block τελειώσει.
- Ένα block μπορεί να περιλαμβάνει κι άλλα φωλιασμένα blocks
  - Η μεταβλητή έχει εμβέλεια και μέσα στα φωλιασμένα blocks
  - Δεν μπορούμε να ορίσουμε μια άλλη μεταβλητή με το ίδιο όνομα σε ένα φωλιασμένο block

Παράδειγμα με το scope μεταβλητών

```
public static void main(String[] args)
{
        int y = 1;
        int x = 2;
        for (int i = 0; i < 3; i ++)
        {
            y = i;
            double x = i+1;
            int z = x+y;
            System.out.println("i = " + i);
            System.out.println("y = " + y);
            System.out.println("z = " + z);
        }
        System.out.println("i = " + i);
        System.out.println("z = " + z);
        System.out.println("y = " + y);
        System.out.println("x = " + x);
}
```

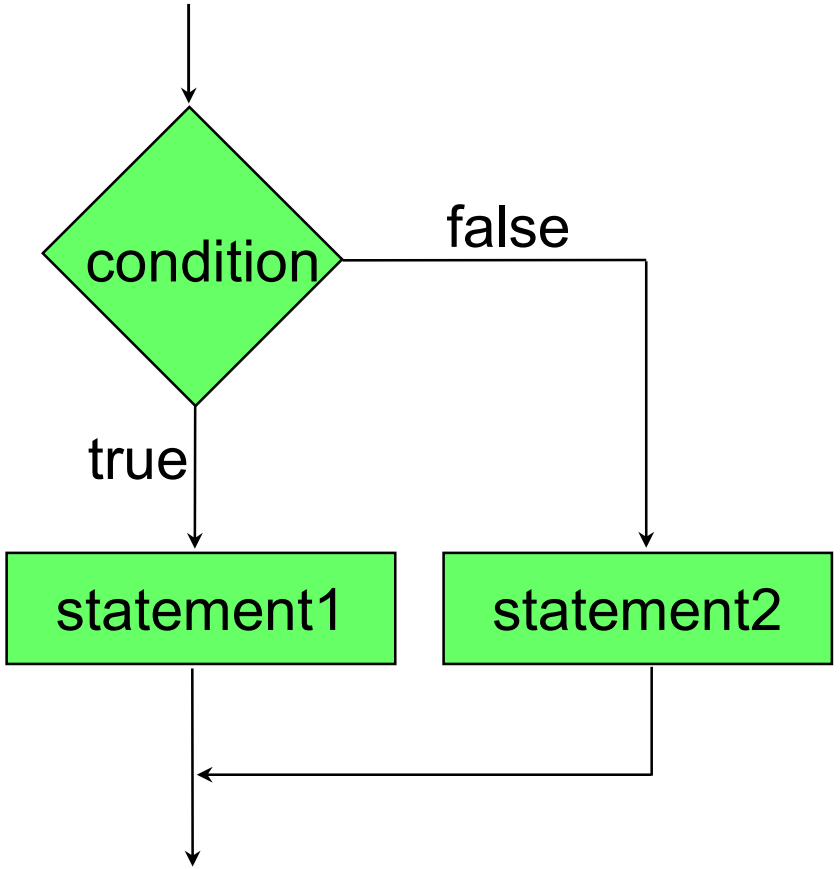
Ο κώδικας έχει λάθη σε τρία σημεία



Προχωρημένο: Κάθε block έχει το δικό του χώρο μνήμης. Σε ένα χώρο μνήμης μια μεταβλητή μπορεί να οριστεί μόνο μία φορά. Τα φωλιασμένα blocks έχουν και τις μεταβλητές των προγόνων.

Το if-else statement

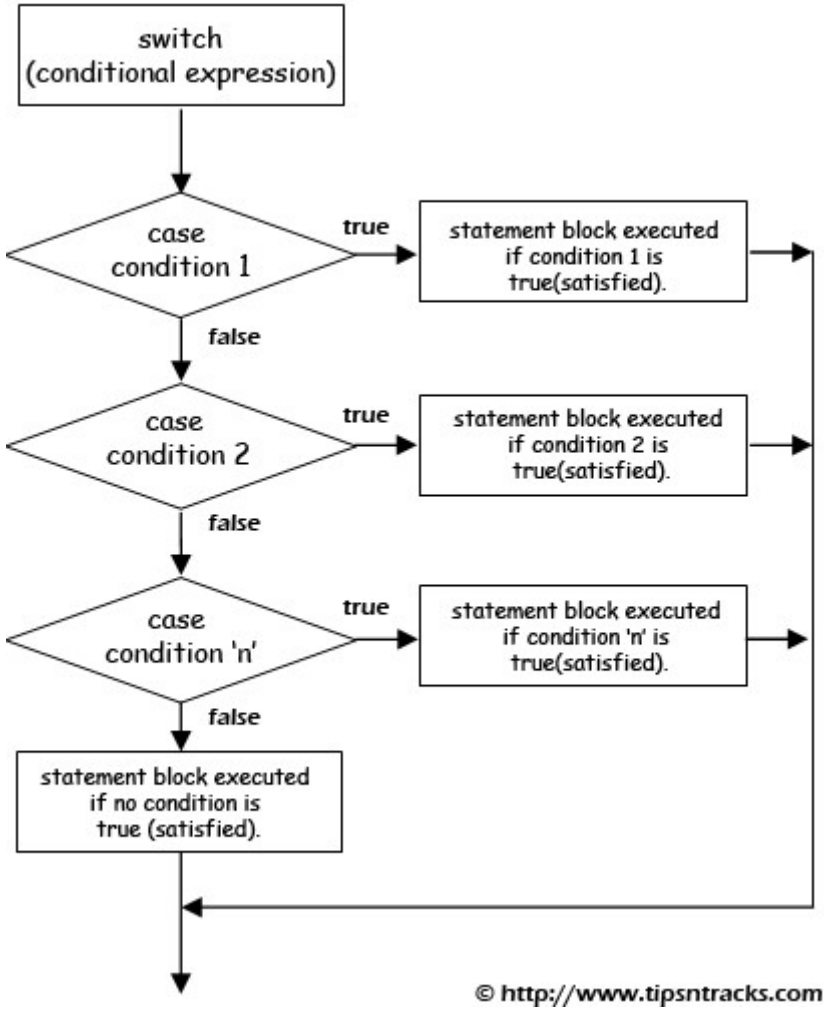
- Το if-else statement δουλεύει καλά όταν στο condition θέλουμε να περιγράψουμε μια επιλογή με δύο πιθανά ενδεχόμενα.
- Τι γίνεται αν η συνθήκη μας έχει πολλά ενδεχόμενα?



Switch statement

Συντακτικό:

```
switch (<condition expression>){
  case <condition 1>:
    code statements 1
    break;
  case <condition 2>:
    code statements 2
    break;
 case <condition 3>:
    code statements 3
    break;
  default:
    default statements
    break;
}
```



Ο κώδικας μέσα το switch είναι ένα μόνο block, και γι αυτό χρειαζόμαστε τα break

Παράδειγμα

- Ένα πρόγραμμα που να εύχεται καλημέρα σε τρεις διαφορετικές γλώσσες ανάλογα με την επιλογή του χρήστη.

```
import java.util.Scanner;

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

    switch(option){
      case "GR": // if (option.equals("GR")
        System.out.println("kalimera");
        break;
      case "EN": // if (option.equals("EN")
        System.out.println("good morning");
        break;
      case "FR": // if (option.equals("FR")
        System.out.println("bonjour");
        break;
      default:
        System.out.println("I do not speak this language.\n" +
                           "Greek, English, French only");
    }
  }
}
```

Αν θέλουμε να μπορούμε να απαντάμε και με μικρά?

```
import java.util.Scanner;

class SwitchTest2{
  public static void main(String args[])
  {
    Scanner input = new Scanner(System.in);
    String option = input.next();

    switch(option){
      case "GR":
      case "gr":
        System.out.println("kalimera");
        break;
      case "EN":
      case "en":
        System.out.println("good morning");
        break;
      case "FR":
      case "fr":
        System.out.println("bonjour");
        break;
      default:
        System.out.println("I do not speak this language.\n" +
                           "Greek, English, French only");
    }
  }
}
```

```
import java.util.Scanner;

class OtherSwitchTest
{
    public static void main(String args[])
    {
        Scanner input = new Scanner(System.in);
        System.out.println("Pick a curtain");
        int option = input.nextInt();
        switch (option-1)
        {
            case 0:
                System.out.println("You selected curtain 1. Zong!");
                break;
            case 1:
                System.out.println(
                        "You selected curtain 2. Congratulations!");
                break;
            case 2:
                System.out.println("You selected curtain 3. Zong!");
                break;
            default:
                System.out.println("Zong!");
        }
    }
}
```

4. ΚΛΑΣΕΙΣ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΑ ΣΤΗΝ JAVA

Κλάσεις και αντικείμενα στην Java

Strings και 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.

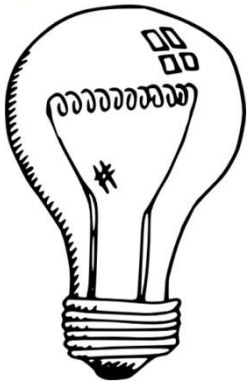
Πρακτικά στον κώδικα

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

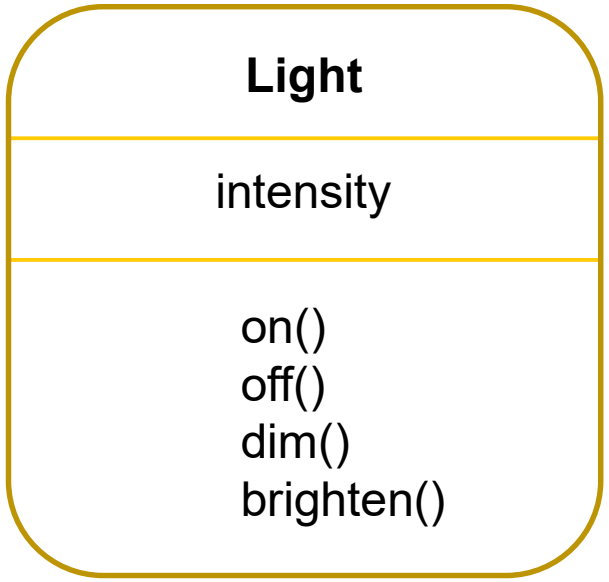
μέλη της κλάσης

- Ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου K
  - 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

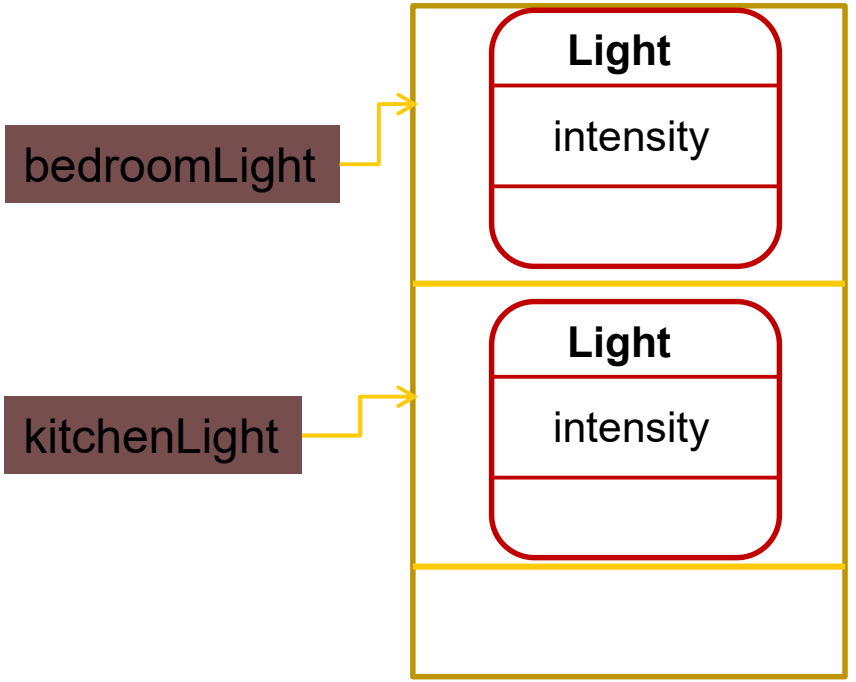


Δημιουργία αντικειμένων

```
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();
```

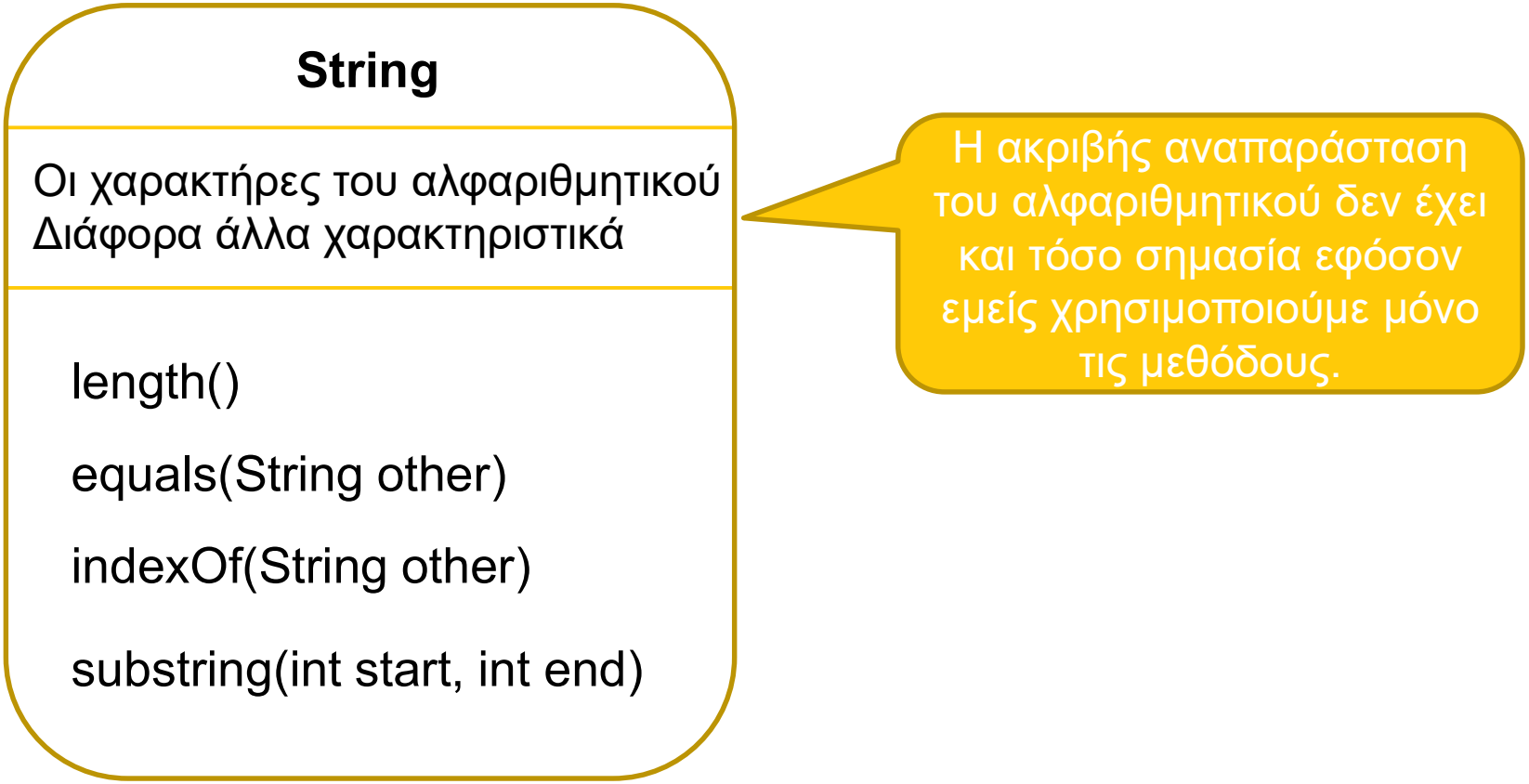
Κλήση μεθόδων

- 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



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

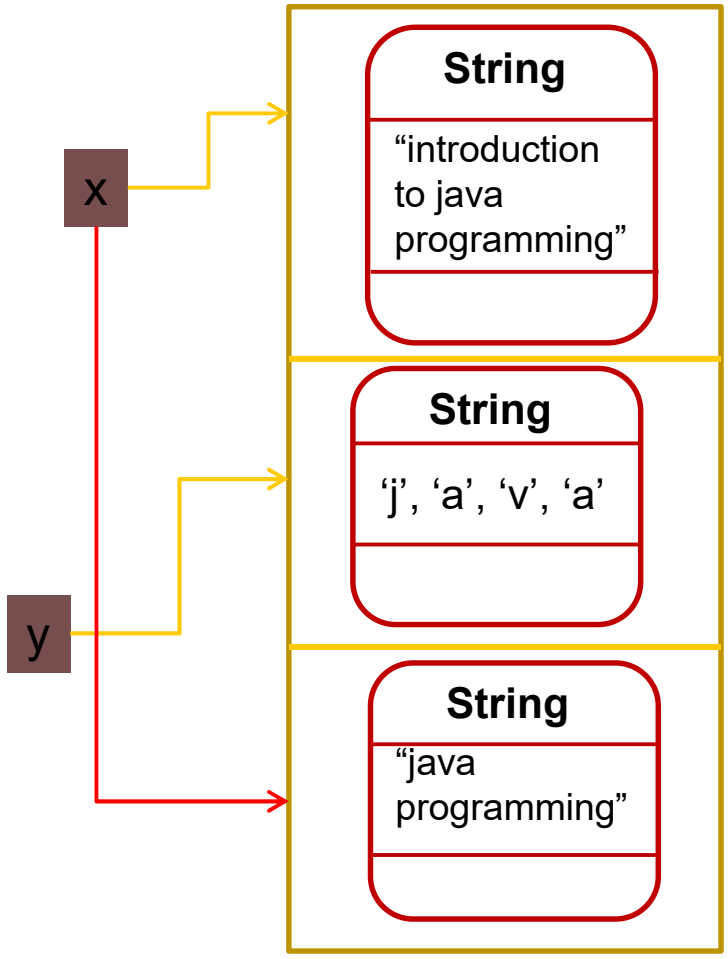
- Τα αμετάβλητα αντικείμενα (immutable objects) είναι αντικείμενα των οποίων η εσωτερική κατάσταση (ουσιαστικά τα πεδία τους) δεν μπορεί να μεταβληθεί.
- Τα Strings είναι αμετάβλητα αντικείμενα
  - Αυτό σημαίνει ότι δεν μπορούμε να αλλάξουμε τα περιεχόμενα ενός αντικειμένου String
    - Π.χ., δεν μπορούμε 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));
  }
}
```

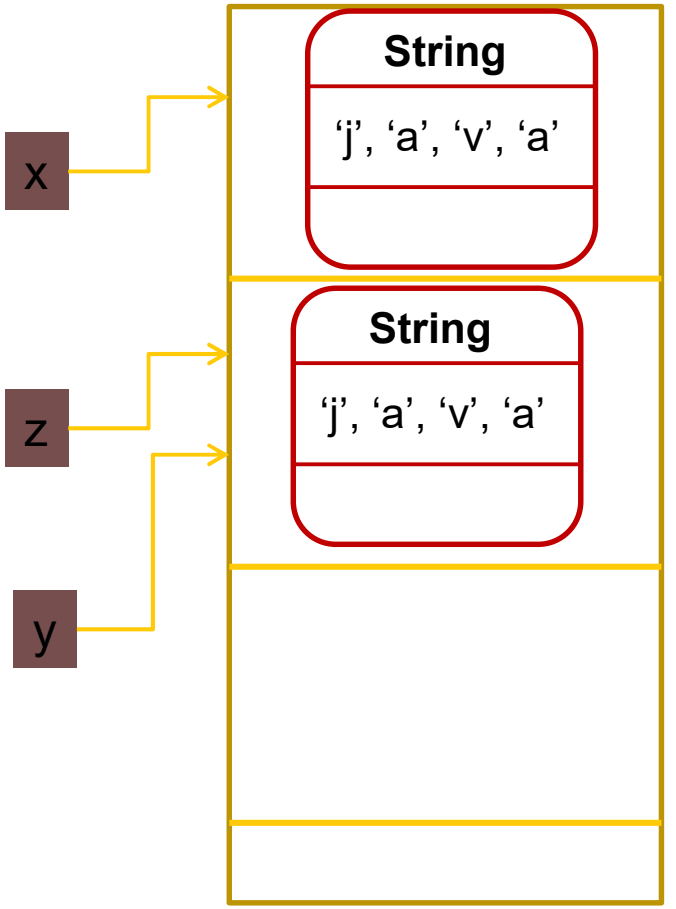
1. false
2. true
3. false
4. true
5. true
6. true

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

Εξήγηση

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

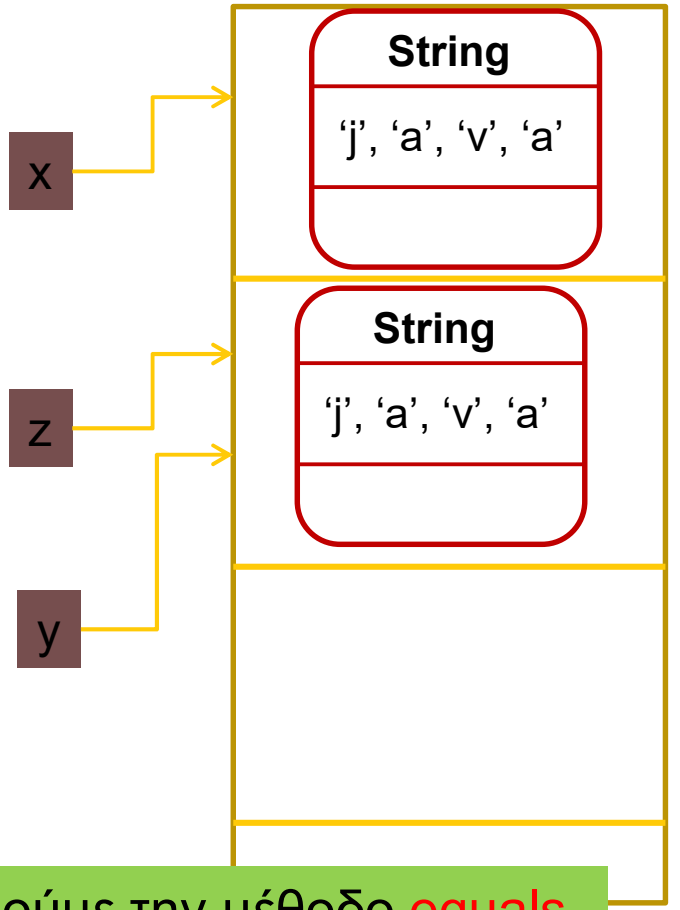
- Όταν δημιουργούμε ένα String αντικείμενο δεσμεύουμε χώρο στη μνήμη για το αντικείμενο
- Η μεταβλητή που ορίζουμε «δείχνει» σε αυτό το χώρο μνήμης
- Η εκχώρηση μεταξύ αντικειμένων τα κάνει να δείχνουν στην ίδια θέση μνήμης



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

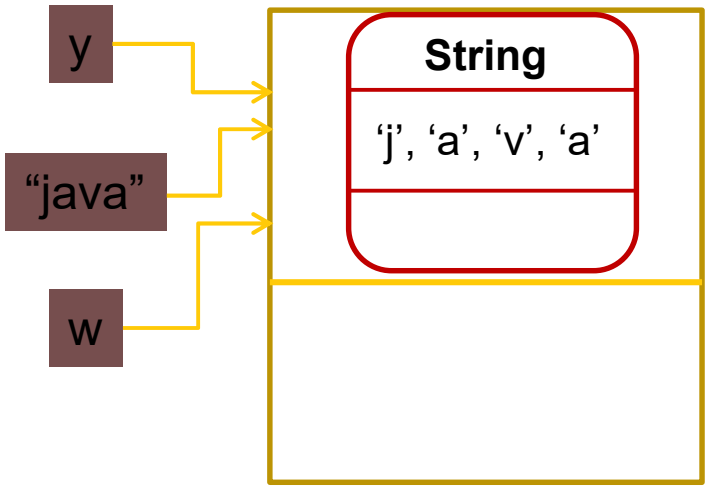
- Οι String τιμές είναι κι αυτές αντικείμενα και μπορούμε να καλέσουμε τις μεθόδους τους

```
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);
  }
}
```

ΜΑΘΗΜΑ 3

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)

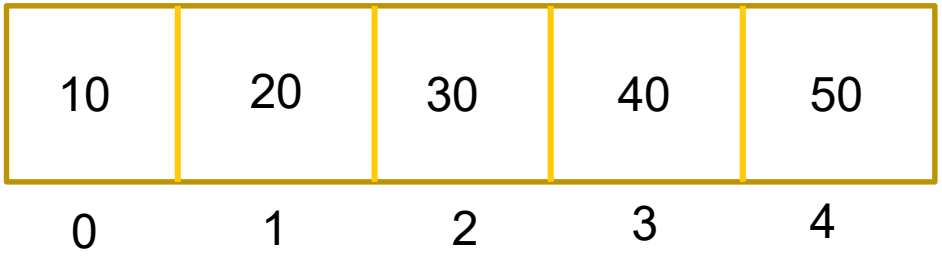
Ορίζει μια μεταβλητή που είναι πίνακας από ακεραίους αλλά δεν δεσμεύει χώρο για τον πίνακα

- 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

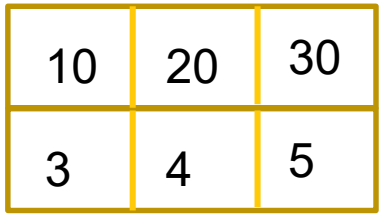
Παράδειγμα

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

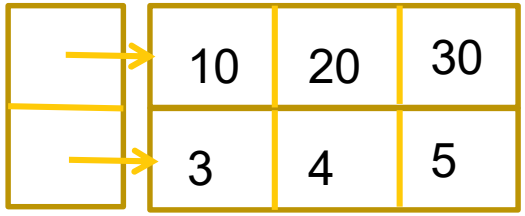
```
class ScanArray
{
    public static void main(String [] args)
    {
        double [] array = {5.3, 3.4, 2.3, 1.2, 0.1};

        // 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];`

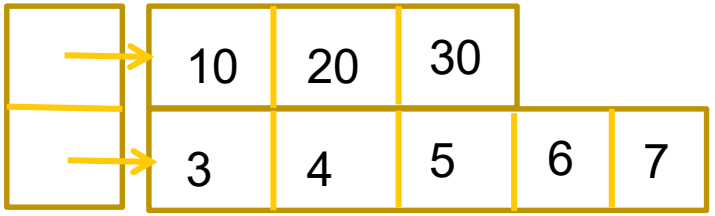


- Ένας δισδιάστατος πίνακας είναι ένας πίνακας από αντικείμενα-πίνακες.
  - `int[][] myArray3 = new int[2][]`
  - `myArray3[0] = new int[3]`
  - `myArray3[1] = new int[3]`



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

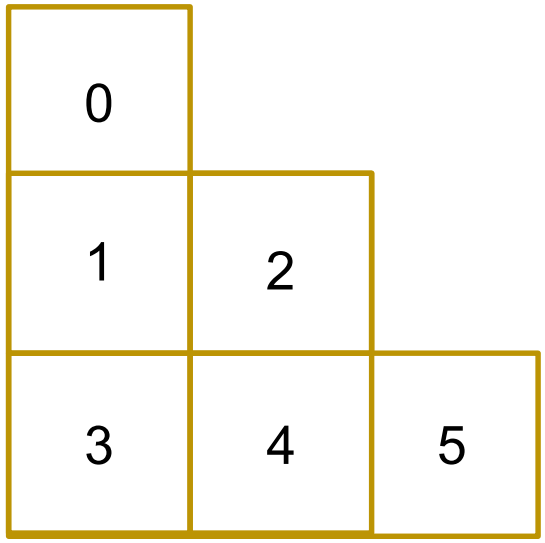
Ο πίνακας arr4 γίνεται ίδιος με τον arr3. Δηλαδή δείχνει στον ίδιο χώρο μνήμης.

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();
    }
}
```

5. ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΔΙΚΕΣ ΜΑΣ 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.

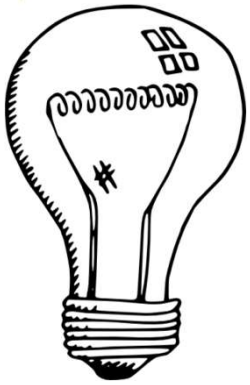
Πρακτικά στον κώδικα

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

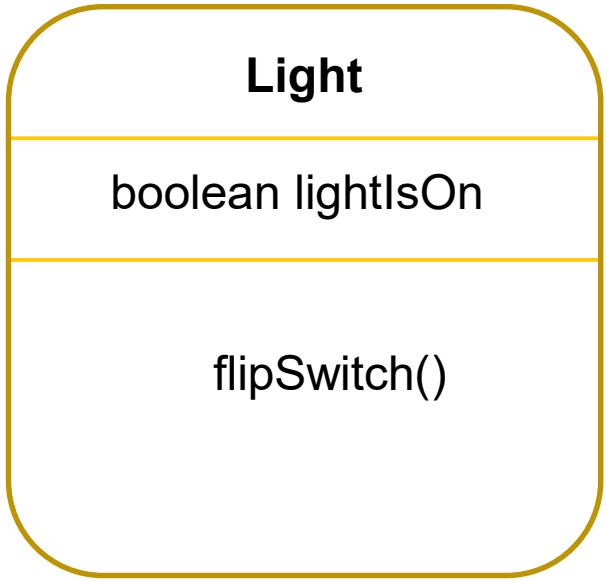
μέλη της κλάσης

- Ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου K
  - 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.

Δημιουργώντας φως



Θα φτιάξουμε μια κλάση που θα χειρίζεται ένα διακόπτη φωτός. Το φώς είναι είτε ανοιχτό είτε κλειστό και μπορούμε να ανοιγοκλείνουμε το φως



Light

```
class Light
{
        private boolean lightIsOn = false;

        public void flipSwitch()
        {
                lightIsOn = !lightIsOn;
        }
}

class HouseWithLights
{
        public static void main(String[] args)
        {
                Light bedroomLight = new Light();
                bedroomLight.flipSwitch();
        }
}
```

Ορισμός κλάσης

Ορισμός (και αρχικοποίηση) πεδίου

Ορισμός μεθόδου

Χρήση πεδίου

Ορισμός αντικειμένου

Κλήση μεθόδου

Κλάσεις και αντικείμενα

- Ορισμός κλάσης:

```
class <Όνομα Κλάσης>
{
        <Ορισμός πεδίων κλάσης>

        <Ορισμός μεθόδων κλάσης>
}
```

- Ορισμός αντικειμένου:

```
<Όνομα Κλάσης> myObject = new <Όνομα Κλάσης>();
```

  - Ο ορισμός του αντικειμένου γίνεται συνήθως μέσα στη main ή μέσα στη μέθοδο μίας άλλης κλάσης που χρησιμοποιεί το αντικείμενο

Τα keywords Public/Private

- Ότι είναι ορισμένο ως public σε μία κλάση είναι προσβάσιμο από μία άλλη κλάση που ορίζει ένα αντικείμενο τύπου Person
  - Π.χ., η μέθοδος flipSwitch() είναι προσβάσιμη από την κλάση HouseWithLights μέσω του αντικειμένου bedroomLight.
- Ότι είναι ορισμένο ως private σε μία κλάση δεν είναι προσβάσιμο από μία άλλη κλάση
  - Π.χ., το πεδίο lightIsOn δεν είναι προσβάσιμο από την κλάση HouseWithLights μέσω του αντικειμένου bedroomLight.
- Μπορούμε να έχουμε public και private πεδία και μεθόδους.
  - Κανόνας: Τα πεδία τα ορίζουμε (σχεδόν) **ΠΑΝΤΑ** private.
  - Οι κλάσεις που χρειάζονται να καλούνται από αντικείμενα είναι public αυτές που είναι βοηθητικές είναι private.
- Τα πεδία και οι μέθοδοι μίας κλάσης, ανεξάρτητα αν είναι public ή private, είναι προσβάσιμα από όλες τις μεθόδους και τα αντικείμενα της ίδιας κλάσης
  - Π.χ., το πεδίο lightIsOn είναι προσβάσιμο παντού μέσα στην κλάση Light, και σε οποιοδήποτε άλλο αντικείμενο τύπου Light

```
class Light
{
        private boolean lightIsOn = false;

        public void flipSwitch(){
                lightIsOn = !lightIsOn;
        }

        public void printState(){
                if (lightIsOn){
                        System.out.println("The light is on");
                }else {
                        System.out.println("The light is off");
                }
        }
}
```

Η κατάσταση ενός αντικειμένου προσδιορίζεται από τις τιμές που έχουν τα πεδία της κλάσης

```
class HouseWithLights
{
        public static void main(String[] args){
                Light bedroomLight = new Light();
                bedroomLight.flipSwitch();
                bedroomLight.printState();
                Light kitchenLight = new Light();
                kitchenLight.flipSwitch();
                kitchenLight.printState();
        }
}
```

Η μόνη πρόσβαση που έχουμε στην κατάσταση του αντικειμένου είναι μέσω των μεθόδων της κλάσης.

Dimmer

- Θέλουμε ο διακόπτης μας να μας δίνει την δυνατότητα να αυξομειώνουμε την ένταση.
  - Τι επιπλέον πεδία πρέπει να προσθέσουμε?
  - Τι επιπλέον μεθόδους χρειαζόμαστε?

```
class DimmerLight
{
    private boolean lightIsOn = false;
    private int intensity = 100;

    public void flipSwitch(){
        lightIsOn = !lightIsOn;
    }

    public void dim(){
        if (intensity > 0){
            intensity --;
        }
    }

    public void birghten(){
        if (intensity < 100){
            intensity ++;
        }
    }

    public void printState(){
        if (lightIsOn){
            System.out.println("The light is ON with intensity " + intensity);
        }else{
            System.out.println("The light is OFF");
        }
    }
}
```

```
class HouseWithDimmerLights
{
    public static void main(String[] args){
        DimmerLight bedroomLight =
                                new DimmerLight();
        bedroomLight.flipSwitch();
        bedroomLight.dim();
        bedroomLight.printState();
    }
}
```

Dimmer

- Κάθε φορά που αυξάνουμε ή μειώνουμε την ένταση θέλουμε να μας λέει και την κατανάλωση
  - (Κατανάλωση = ένταση * 0.1 λεπτά/ώρα)

```
class DimmerLight2
{
    private boolean lightIsOn = false;
    private int intensity = 100;

    public void flipSwitch(){
        lightIsOn = !lightIsOn;
    }

    public void dim(){
        if (intensity > 0){
            intensity --;
        }
        double consumption = intensity *0.1;
        System.out.print("Consumption = "+consumption);
    }

    public void birghten(){
        if (intensity < 100){
            intensity ++;
        }
        double consumption = intensity *0.1;
        System.out.print("Consumption = "+consumption);
    }

    public void printState(){
        if (lightIsOn){
            System.out.println("The light is ON with intensity " + intensity);
        }else{
            System.out.println("The light is OFF");
        }
    }
}
```

Οι μεταβλητές consumption είναι τοπικές μεταβλητές

Υπάρχουν μόνο μέσα στις μεθόδους dim και brighten και όταν τελειώσει η κλήση τους εξαφανίζονται.

Τοπικές μεταβλητές

- Είδαμε πρώτη φορά τις τοπικές μεταβλητές όταν μιλήσαμε για μεταβλητές που ορίζονται μέσα σε ένα λογικό block.
  - Παρόμοια είναι και για τις μεταβλητές μιας μεθόδου.

- Τοπικές μεταβλητές μιας μεθόδου είναι οι μεταβλητές που ορίζονται μέσα στον κώδικα της μεθόδου
  - Περιλαμβάνουν και τις μεταβλητές που κρατάνε τις παραμέτρους της μεθόδου
- Οι μεταβλητές αυτές έχουν εμβέλεια μόνο μέσα στην μέθοδο
  - Εξαφανίζονται όταν βγούμε από τη μέθοδο.

- Αντιθέτως τα πεδία της κλάσης διατηρούνται όσο υπάρχει το αντικείμενο, και έχουν εμβέλεια σε όλη την κλάση