

# ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

---

Εισαγωγή στη Java

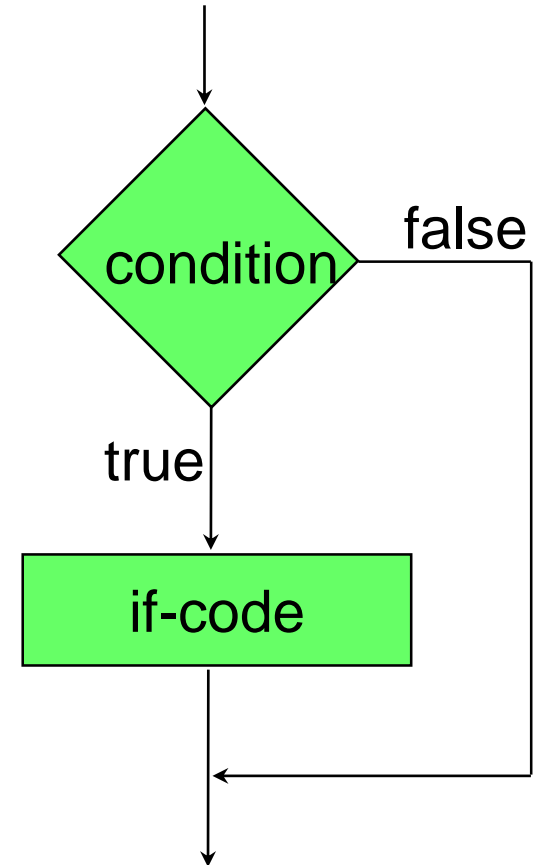
# Βρόγχοι – Το if-then Statement

- Στην Java το **if-then statement** έχει το εξής συντακτικό

Η παρένθεση είναι απαραίτητη

```
if (condition)
{
    ...if-code block...
}
```

- Αν η **συνθήκη** είναι **αληθής** τότε εκτελείται το block κώδικα if-code
- Αν η **συνθήκη** είναι **ψευδής** τότε το κομμάτι αυτό προσπερνιέται και συνεχίζεται η εκτέλεση.







```
import java.util.Scanner;

class IfTest1b
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        boolean inputIsPositive = (inputInt > 0);
        if (inputIsPositive) {
            System.out.println(inputInt +
                               " is positive");
        }
    }
}
```

Ακόμη και αν δεν το προσδιορίσουμε ελέγχει για true

# Programming Style: Λογικές μεταβλητές

- Συνηθίζεται όταν ορίζουμε **λογικές μεταβλητές** το **όνομα** τους να είναι αυτό που περιγράφει την περίπτωση που η μεταβλητή αποτιμάται **true**.

```
int x = 10;
boolean isPositive = (x > 0);
boolean isNegative = (x < 0);
boolean isNotPositive = !isPositive;
```

- Αυτό βολεύει για την εύκολη ανάγνωση του προγράμματος όταν χρησιμοποιούμε την μεταβλητή

```
if (isPositive) {
    System.out.println("Variable is positive");
}
```

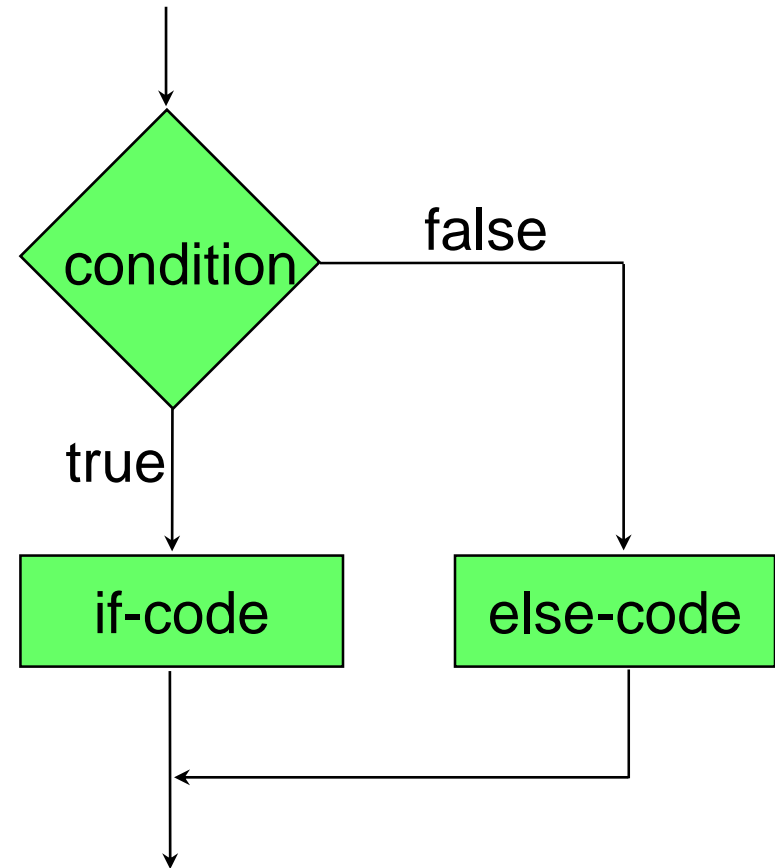
- Το ίδιο ισχύει και όταν αργότερα θα ορίζουμε **μεθόδους** που επιστρέφουν λογικές τιμές
  - Π.χ., για τα Strings υπάρχει η μέθοδος **equals** που γίνεται true όταν έχουμε ισότητα και η μέθοδος **isEmpty** που είναι true όταν έχουμε άδειο String.

# Βρόγχοι – Το if-then-else Statement

- Στην Java το **if-then-else statement** έχει το εξής συντακτικό

```
| if (condition) {  
|     ...if-code block...  
| }else{  
|     ...else-code block...  
| }
```

- Αν η **συνθήκη** είναι **αληθής** τότε εκτελείται το block κώδικα if-code
- Αν η **συνθήκη** είναι **ψευδής** τότε εκτελείται το block κώδικα else-code.
- Ο κώδικας του if-code block ή του else-code block μπορεί να περιέχουν ένα άλλο (**φωλιασμένο (nested)**) if statement
- **Προσοχή:** ένα **else** clause ταιριάζεται με το **τελευταίο** ελεύθερο **if** ακόμη κι αν η στοίχιση του κώδικα υπονοεί διαφορετικά.







## Έλεγχος πολλαπλών επιλογών

```
import java.util.Scanner;

class IfTest3
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        if (inputInt > 0){
            System.out.println(inputInt +
                               " is positive");
        }else if (inputInt < 0){
            System.out.println(inputInt +
                               " is negative");
        }else{
            System.out.println(inputInt + " is zero");
        }
    }
}
```

# Προσοχή!

## ΛΑΘΟΣ!

```
if( i == j )
    if ( j == k )
        System.out.print(
            "i equals k");
else
    System.out.print(
        "i is not equal to j");
```

Το else μοιάζει σαν να πηγαίνει με το μπλε else αλλά ταιριάζεται με το τελευταίο (πράσινο) if

## ΣΩΣΤΟ!

```
if( i == j ){
    if ( j == k ){
        System.out.print(
            "i equals k");
    }
}
else {
    System.out.print(
        "i is not equal to j");
}
```

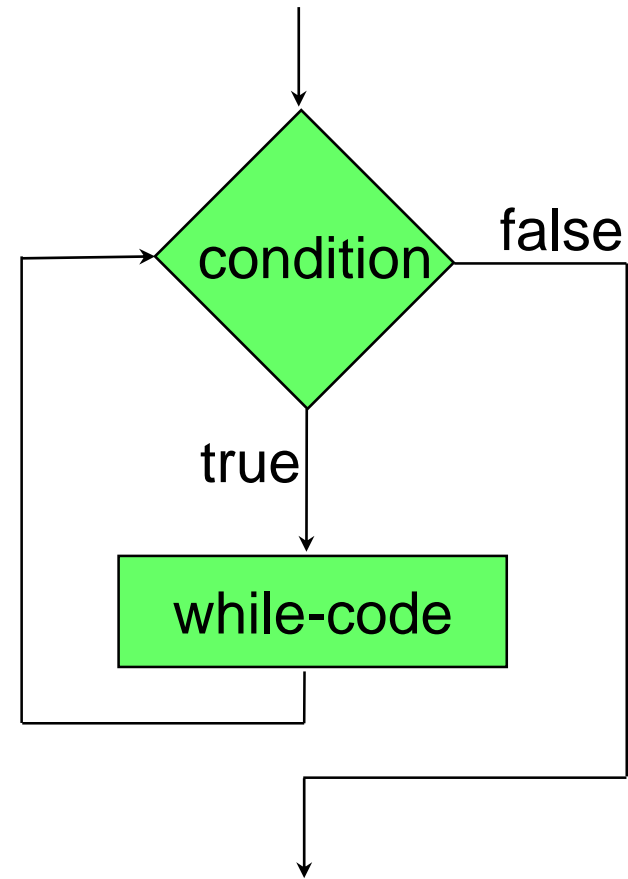
**Πάντα** να βάζετε `{ }` στο σώμα των if-then-else statements.  
Πάντα να στοιχίζετε σωστά τον κώδικα.

# 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

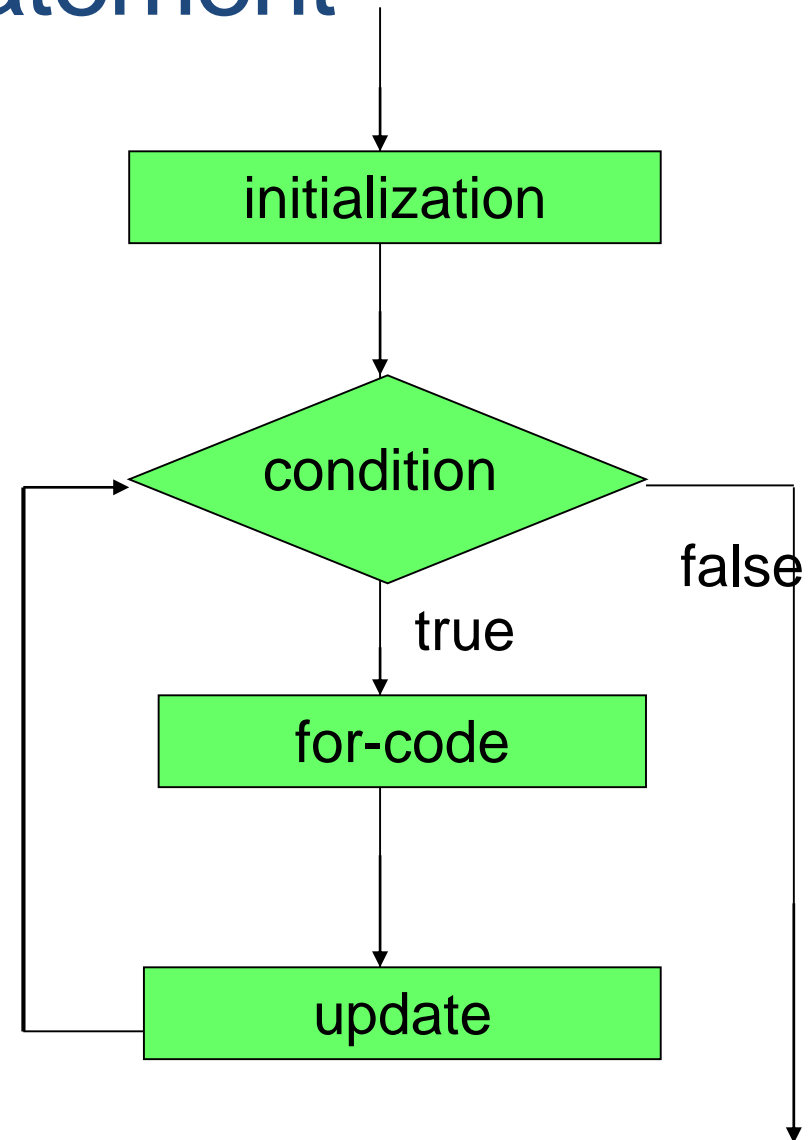
```
Scanner inputReader = new Scanner(System.in);  
String input = inputReader.next();  
  
while (input.equals("Yes"))  
{  
    System.out.println("Do you want to continue?");  
    input = inputReader.next();  
}
```

# Επαναλήψεις – for statement

- Στην Java το **for statement** έχει το εξής ΣΥΝΤΑΚΤΙΚΟ

```
for (initialization;  
    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;
}
```

Η μεταβλητή *i* είναι τοπική για το for-loop

# 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 ++;  
}
```

# 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;

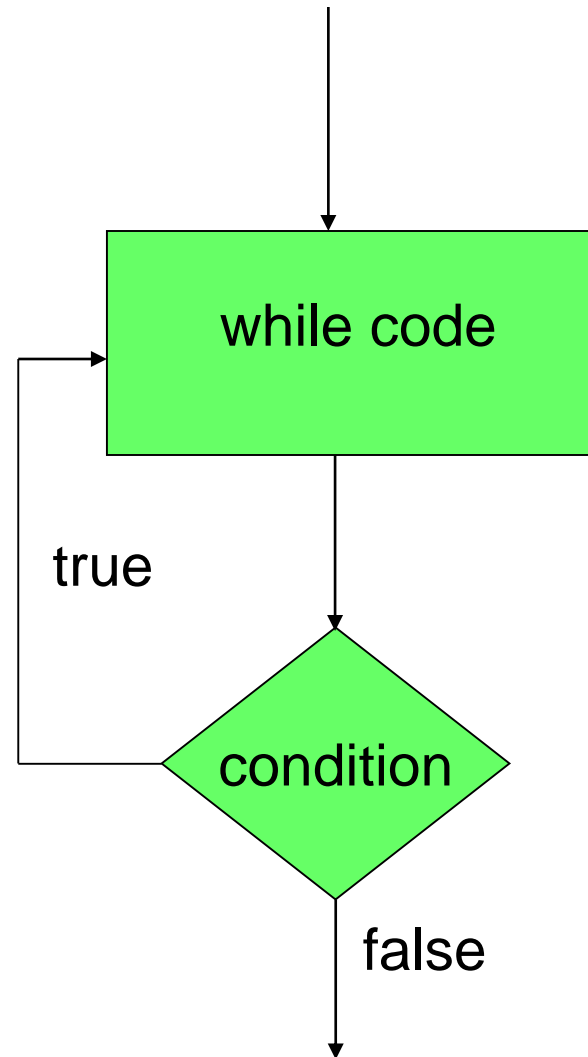
class Countdown
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        while (inputInt != 0)
        {
            if (inputInt < 0 ){
                for (int i = inputInt; i < 0; i ++){
                  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

# To 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.
- Οι μεταβλητές στο **condition** **δεν** μπορεί να είναι **τοπικές** μεταβλητές του while code



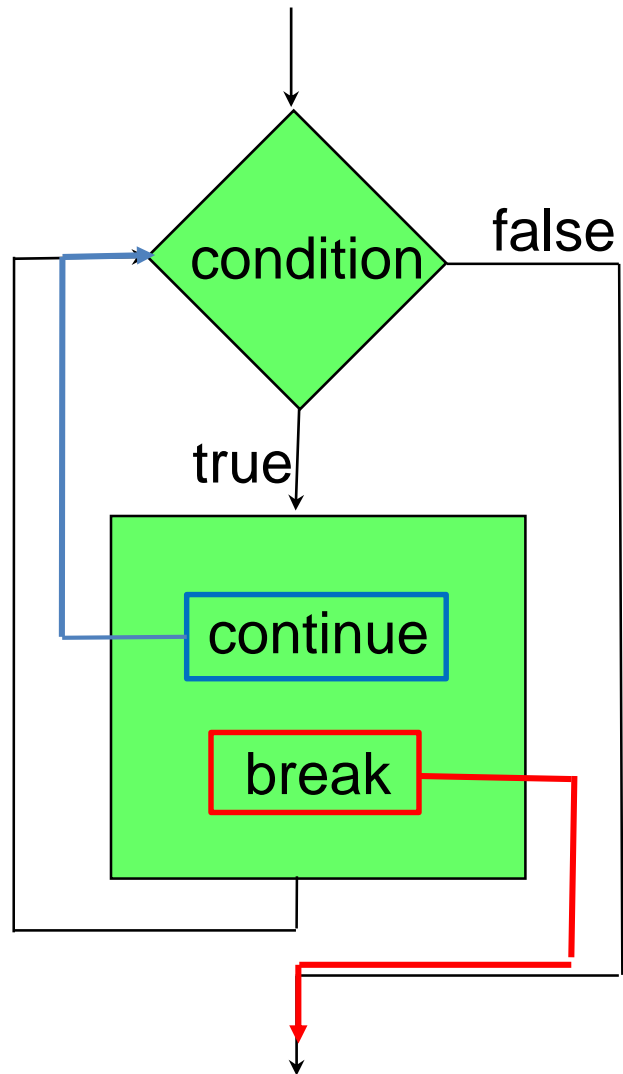
```
import java.util.Scanner;

class CountdownWithDo
{
    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 πρόωρη 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();
        }
    }
}
```

Ο τελεστής % υπολογίζει  
το υπόλοιπο διαίρεσης

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` πρέπει 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);
    }
    int z = 0;
    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

Δεν είναι λάθος  
γιατί ο πρώτος  
ορισμός είναι εκτός  
εμβέλειας

```
public static void main(String[] args)
{
    ... ..
    {
        ... ..
        {
            int y = 1;
            ... ..
            {
                ... ..
            }
            ... ..
        }
        ... ..
    }
    ... ..
}
```

Η διαφορά του κόκκινου από το μπλε είναι ο χώρος **εκτός** της εμβέλειας του **y**

Έξω από το μπλε **δεν** μπορούμε να χρησιμοποιήσουμε τη μεταβλητή **y**, αλλά μπορούμε να **ορίσουμε** νέα μεταβλητή **y**

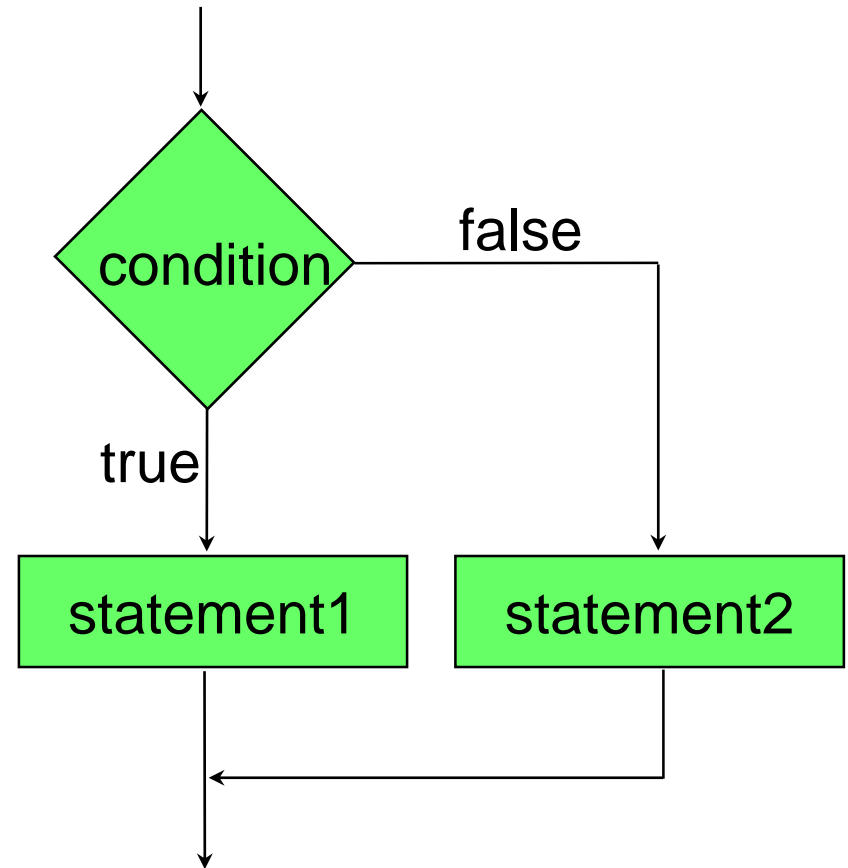
Η εμβέλεια του **y**

Μέσα στο μπλε μπορούμε να χρησιμοποιήσουμε την μεταβλητή **y**, αλλά **δεν** μπορούμε να **ορίσουμε** άλλη μεταβλητή με το όνομα **y**

Κάθε block έχει το δικό του χώρο μνήμης. Σε ένα χώρο μνήμης μια μεταβλητή μπορεί να οριστεί μόνο μία φορά. Ο χώρος μνήμης ενός block περιλαμβάνει και τα φωλιασμένα blocks.

# To 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?



# Παράδειγμα

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

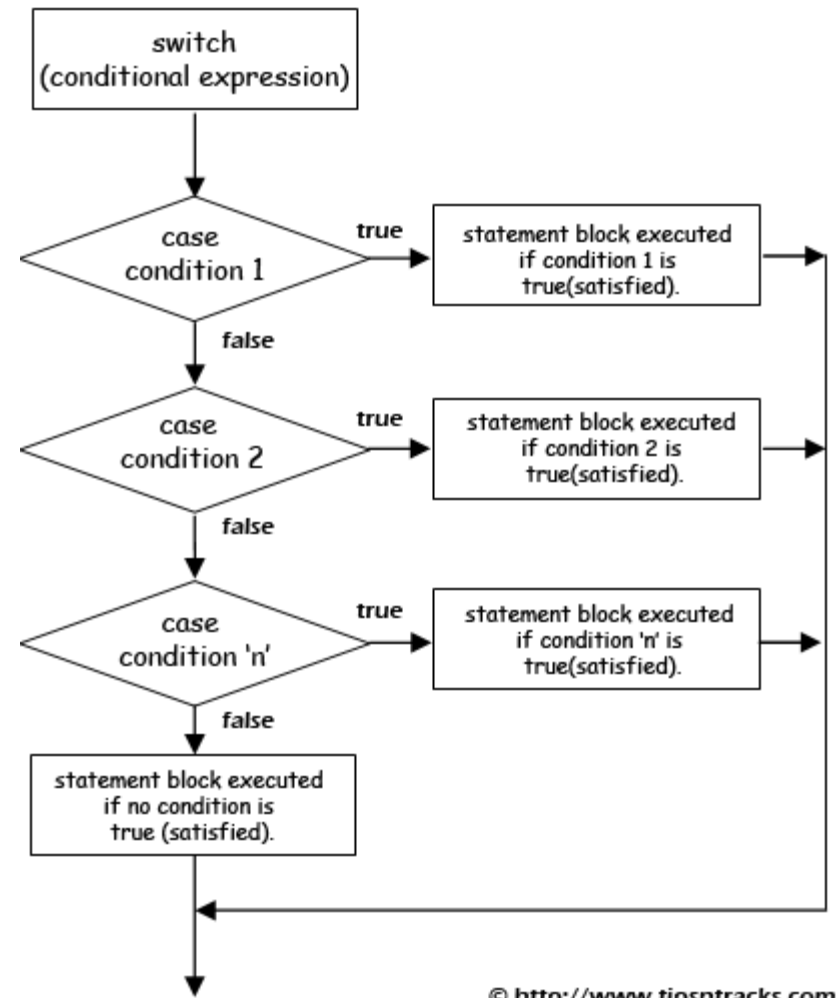
```
import java.util.Scanner;

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

        if (option.equals("GR"))
        {
            System.out.println("kalimera");
        }else if (option.equals("EN")){
            System.out.println("good morning");
        }else if (option.equals("FR")){
            System.out.println("bonjour");
        }else{
            System.out.println("I dont speak this language");
        }
    }
}
```

# Switch statement

```
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;  
}
```



© <http://www.tipsntracks.com>

- **case**: οι διάφορες περιπτώσεις/τιμές που θέλουμε να ελέγξουμε
- Ο έλεγχος ροής γίνεται με τα **break**. Αν δεν υπάρχει το **break** τότε εκτελείται όλος ο κώδικας που ακολουθεί το **case**.
- **default**: Κώδικας για την περίπτωση που κανένα **case** δεν ικανοποιείται

```
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");
        }
    }
}
```

# Παράδειγμα

- Ένα πρόγραμμα που να διαλέγεις μια κουρτίνα και να σου δείχνει τι υπάρχει από πίσω

```
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)
        {
            case 1:
                System.out.println("You selected curtain 1. Zong!");
                break;
            case 2:
                System.out.println(
                    "You selected curtain 2. Congratulations!");
                break;
            case 3:
                System.out.println("You selected curtain 3. Zong!");
                break;
            default:
                System.out.println("Zong!");
        }
    }
}
```