2. ΕΙΣΑΓΩΓΗ ΣΤΗ JAVA

Η γλώσσα προγραμματισμού Java

Βασικό συντακτικό, ορισμός μεταβλητών, έλεγχος ροής

ΕΠΑΝΑΛΗΨΗ WHILE, FOR

Παράδειγμα

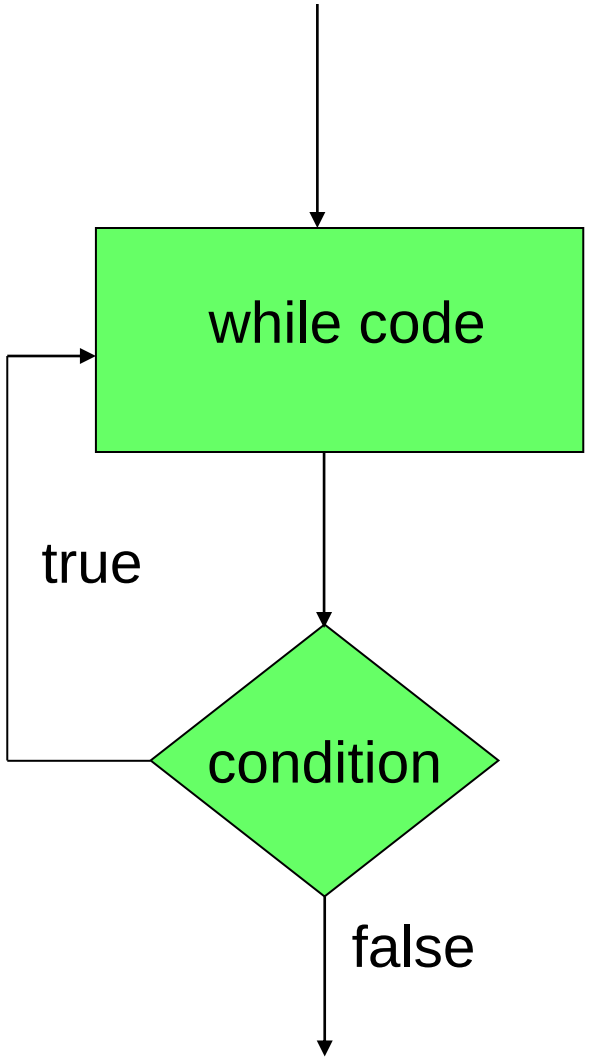
- Κάνετε πρόγραμμα που παίρνει σαν είσοδο ένα αριθμό και υλοποιεί μια αντίστροφη μέτρηση. Αν ο αριθμός είναι θετικός η αντίστροφη μέτρηση γίνεται προς τα κάτω μέχρι το μηδέν, αν είναι αρνητικός γίνεται προς τα πάνω μέχρι το μηδέν. Η διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να δώσει την τιμή μηδέν.

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

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

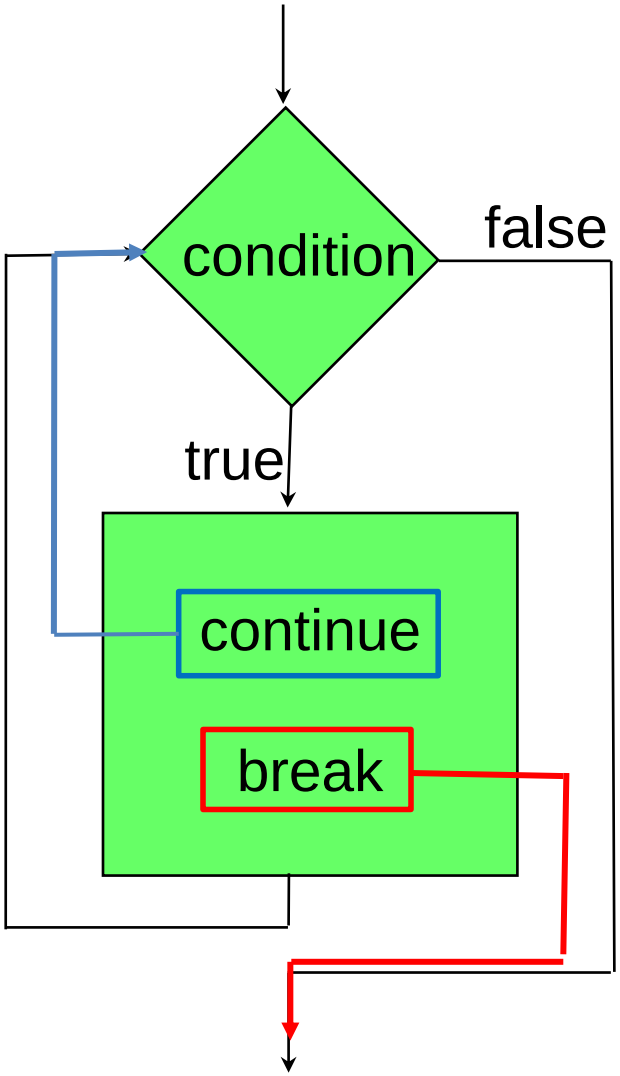
```
import java.util.Scanner;

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 ΕΝΤΟΛΕΣ BREAK ΚΑΙ CONTINUE

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 (everything is not ok){
      <some code>
      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 CountDownOdd
{
    public static void main(String[] args){
        Scanner reader = new Scanner(System.in);
        System.out.print("Give an odd number for the countdown:");
        int inputInt = reader.nextInt();
        while (inputInt != 0){
            if (inputInt%2 == 1){ //odd number
                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);
                    }
                }
            }
            System.out.print("Give an odd number for the countdown:");
            inputInt = reader.nextInt();
        }
    }
}
```

Η αντίστροφη μέτρηση εκτελείται μόνο για περιττούς αριθμούς

Ο τελεστής % υπολογίζει το υπόλοιπο διαίρεσης

Ο κώδικας στο while μπαίνει ολόκληρος μέσα σε ένα if

Η αντίστροφη μέτρηση εκτελείται μόνο για περιττούς αριθμούς

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

Ελέγχουμε την αντίθετη συνθήκη (για άρτιο αριθμό). Αν ισχύει τότε καλούμε την continue.

```
import java.util.Scanner;

class CountDownWithBreak
{
    public static void main(String[] args){
        Scanner reader = new Scanner(System.in);
        int inputInt = reader.nextInt();
        while (inputInt != 0){
            if (inputInt%2 == 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);
                }
            }
            inputInt = reader.nextInt();
        }
    }
}
```

Η αντίστροφη μέτρηση εκτελείται μόνο για περιττούς αριθμούς και αν ο χρήστης δώσει άρτια τιμή σταματάμε

Ελέγχουμε για άρτιο αριθμό και αν ισχύει τότε κάνουμε break.

```
import java.util.Scanner;

class CountDownWhileTrue
{
    public static void main(String[] args)
    {
        Scanner reader = new Scanner(System.in);
        while (true)
        {
            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);
                }
            }
        }
    }
}
```

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. Η έξοδος από το loop γίνεται με την κλήση της break.

Κάποιες φορές αυτή η κατασκευή είναι πιο βολική από το να ελέγχουμε την συνθήκη εξόδου στο while. Π.χ., όταν έχουμε πολλαπλές συνθήκες εξόδου.

Υλοποίηση με do-while

```
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)
    }
}
```

Χρήση break/continue σε for loops

- Μπορούμε να χρησιμοποιήσουμε τις εντολές break και continue και μέσα σε for loops
- Η εντολή break δουλεύει ακριβώς όπως και με το while, μας βγάζει αμέσως εκτός της επανάληψης.
- Η εντολή continue επιστρέφει στον έλεγχο της συνθήκης αφού πρώτα εκτελεστεί η εντολή ενημέρωσης.

Εκτύπωση όλων των περιττών αριθμών μεταξύ 0…9, που είναι μικρότεροι ή ίσοι από την τιμή της μεταβλητής limit

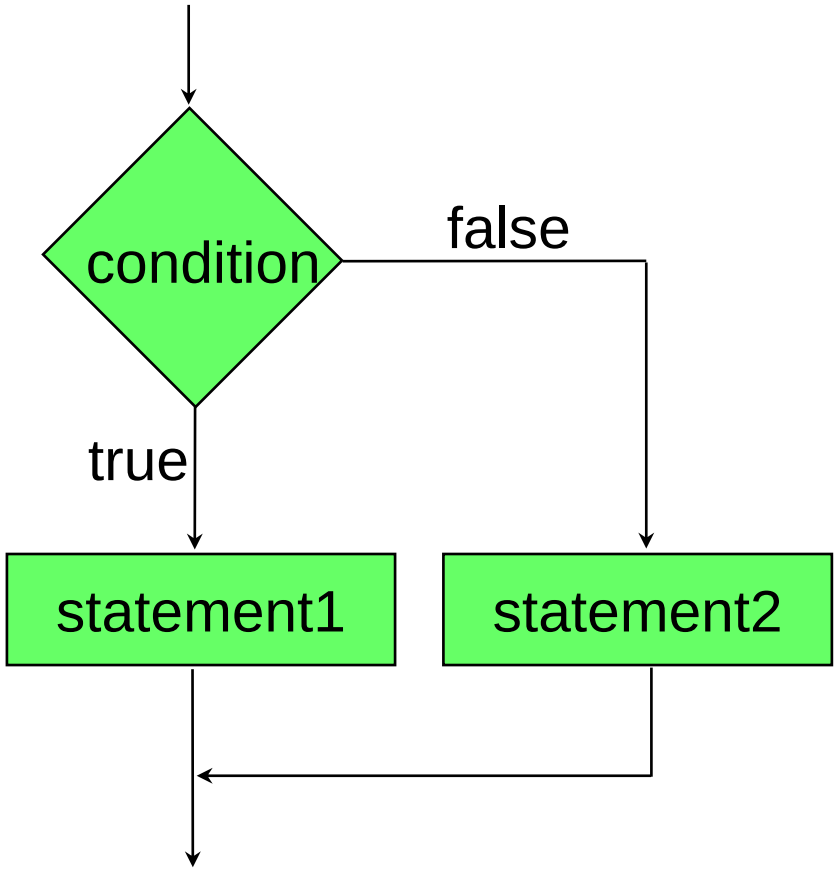
```
class ForBreakContinue
{
  public static void main(String[] args){
      int limit = 6;
      for (int i = 0; i < 10; i ++){
         if (i%2 == 0){
            continue;
         }
         if (i > limit){
            break;
         }
         System.out.println(i);
      }
   }
}
```

Αν το i είναι άρτιος επιστρέφουμε στην αρχή του for-loop αφού πρώτα εκτελεστεί η εντολή ενημέρωσης i++

Η ΕΝΤΟΛΗ SWITCH

else if

- 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?
- Χρησιμοποιούμε το else if



Παράδειγμα

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

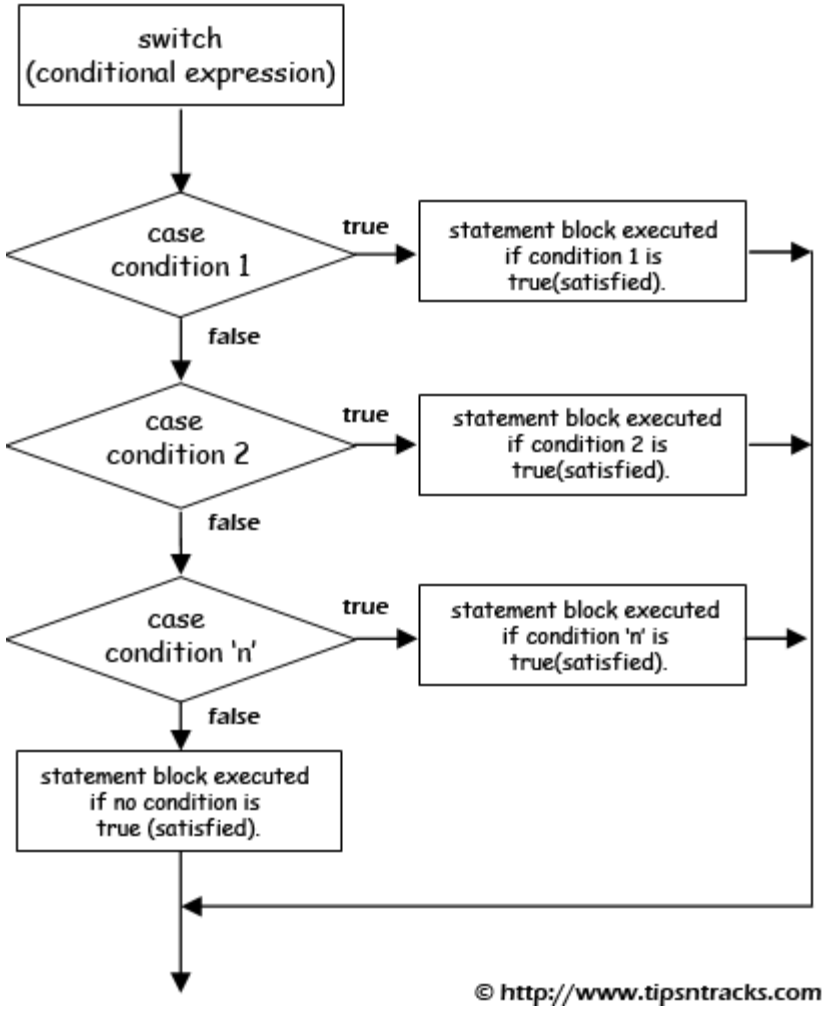
```
import java.util.Scanner;

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 don't speak this language");
       }
    }
}
```

Η εντολή switch

- Αν έχουμε πάρα πολλά ενδεχόμενα ο κώδικας μας δεν φαίνεται ωραίος.
- Μπορούμε να κάνουμε έλεγχο για πολλά ενδεχόμενα χρησιμοποιώντας την εντολή switch.

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;
}
```



- 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 SwitchTest{
  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");
    }
  }
}
```

ΤΟΠΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ ΕΜΒΕΛΕΙΑ ΜΕΤΑΒΛΗΤΩΝ

Τοπικές μεταβλητές

- Κάθε φορά που ανοίγουμε και κλείνουμε άγκιστρα δημιουργούμε ένα block κώδικα.
- Μέσα στο block μπορούμε να ορίσουμε νέες μεταβλητές. Οι μεταβλητές αυτές λέγονται τοπικές μεταβλητές για το block
- Η μεταβλητές αυτές υπάρχουν μόνο μέσα στο block. Όταν βγούμε από το block χάνονται.

Παράδειγματα

```
class LocalVariables
{
    public static void main(String[] args){
        int small = 2;
        int large = 1;
        if (small > large){
            int temp = small;
            small = large;
            large = temp;
        }
    }
}
```

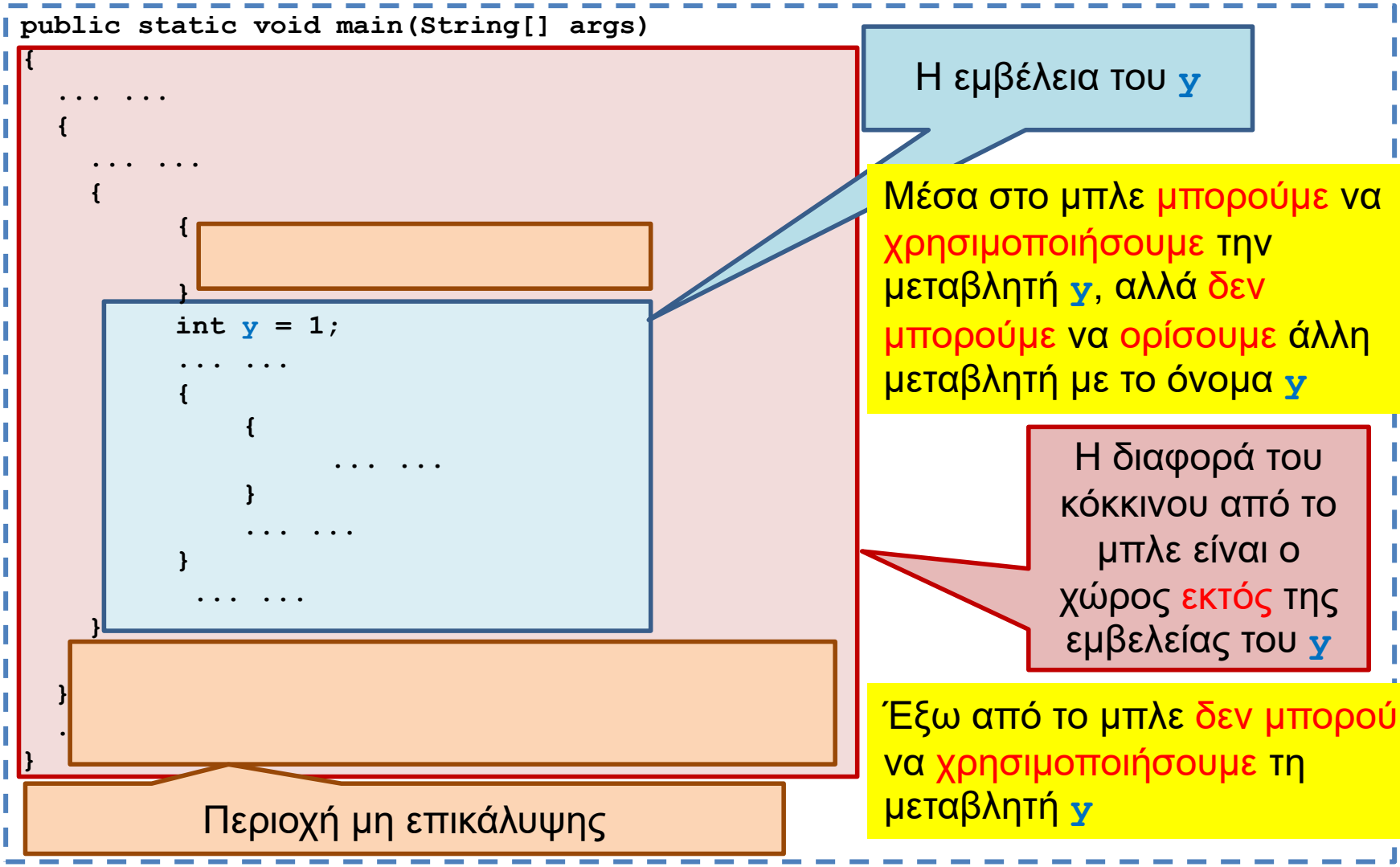
Η μεταβλητή int temp είναι τοπική μεταβλητή στο if-code block

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

Η μεταβλητή int i είναι τοπική μεταβλητή στο block που ορίζεται για το if

Εμβέλεια (scope) μεταβλητών

- Οι τοπικές μεταβλητές λέγονται τοπικές γιατί υπάρχουν μόνο μέσα στο block στο οποίο ορίζονται, και συνήθως εξυπηρετούν κάποιο ανάγκη «τοπική» για το 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.
- Εμβέλεια: Λέμε ότι η τοπική μεταβλητή έχει εμβέλεια (scope) μέσα στο block το οποίο ορίζεται.
  - Μπορείτε να το σκέφτεστε ότι αυτή είναι η περιοχή που η μεταβλητή υπάρχει, ή έχει ισχύ.
- 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 στην περιοχή που έχει εμβέλεια η μεταβλητή.



Στην πορτοκαλί περιοχή μπορούμε να ορίσουμε μια άλλη μεταβλητή με το όνομα **y** ώστε η εμβέλεια της να μην επικαλύπτεται με αυτή της μπλε μεταβλητής y

Παράδειγματα

```
class LocalVariables
{
    public static void main(String[] args){
        int small = 2;
        int large = 1;
        if (small > large){
            int temp = small;
            small = large;
            large = small;
        }
        System.out.println(small + " " + large);
        System.out.println(temp);
    }
}
```

Η μεταβλητή int temp είναι τοπική μεταβλητή στο if-code block

Μπορούμε να τυπώσουμε τα small, large γιατί έχουν εμβέλεια σε όλη την main

ΔΕΝ μπορούμε να χρησιμοποιήσουμε την μεταβλητή temp γιατί είμαστε εκτός της εμβέλειας του if-block. Σε αυτό το σημείο η μεταβλητή δεν υπάρχει για τον κώδικα μας, και ο compiler χτυπάει λάθος

```
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 παρένθεση με τη συνθήκη του while είναι εκτός της εμβέλειας του while-code block
Άρα η **inputInt** πρεπει να οριστεί έξω από το while-loop

Η μεταβλητή **int i** πρέπει να οριστεί για κάθε for ξεχωριστά

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;
            double 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 δύο σημεία

Δεν είναι λάθος γιατί η εμβέλεια της μεταβλητής int z είναι όλες οι εντολές που ακολουθούν, και η εμβέλεια της μεταβλητής double z, είναι μόνο μέσα στο block της for.