ΑΝΤΙΚΕΙΜΕΝΟΣΤΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Παπαδάκης Νικόλαος

Γενικά

- Ιστοσελίδα μαθήματος

  https://eclass.hmu.gr/courses/ECE136/

Στοιχεία Διδάσκοντα

Καθηγητής Παπαδάκης Νικόλαος

email: npapadak@cs.hmu.gr

Γραφείο Ισόγειο Σχολής Μηχανικών απέναντι από την Γραμματεία τμήματος ΗΜΜΥ

Ώρες γραφείου Τρίτη 11-12 Τετάρτη 12-11

Γενικά

- Ωρες Διδασκαλίας Τρίτη 9-11 Τετάρτη 10-12
- Εργαστήριο: 11 δίωρα εργαστήρια

- Ασκήσεις (Ατομικές)
  - 3 σειρές προγραματιστικών ασκήσεων 30%
- Ομαδικό Project 40%
- Τελική εξέταση 30%
- Βαθμολογία

  Αν τελικό >=4 τότε

  30%ασκήσεις+30%Project+40%Τελικό

  Αλλοίως 100%Τελικό

Εισαγωγή στον αντικειμενοστραφή προγραμματισμό

- Κλάσεις και αντικείμενα
- Κληρονομικότητα (inheritance)
- Υπερκάλυψη (overriding)
- Υπερφόρτωση (overloading)
- Κατασκευαστές κλάσεων (constructors)
- Προσδιοριστές πρόσβασης (access specifiers)
- Τροποποιητές (modifiers)
- Διασυνδέσεις (interfaces)
- Πακέτα (packages)

Κλάσεις

- Σε αντικειμενοστραφείς γλώσσες προγραμματισμού δίνεται η δυνατότητα σχεδιάσης νέων και σύνθετων τύπων δεδομένων
- Κλάση: σύνθετη δομή δεδομένων σχεδιασμένη από τον προγραμματιστή
- Μια κλάση περιέχει ως μέλη της μεταβλητές-πεδία (member variables) και συναρτήσεις-μέθοδοι (member functions)
- Ορισμός κλάσης:

```
class myClass
{
  Δήλωση μεθόδων και μεταβλητών . . .
}
```

Κλάσεις και Αντικείμενα

- Μια κλάση είναι το αντίστοιχο των τύπων δεδομένων για τις μεταβλητές (στον δομημένο προγραμματισμό).
- Μια η κλάση είναι απλώς ένα πρότυπο για την δημιουργία αντικειμένων.
- Το αντικείμενο αυτό καθ' εαυτό είναι μια οντότητα στη μνήμη η οποία περιέχει δεδομένα καθώς και μεθόδους.
- Μέσω των μεθόδων μπορούμε να επικοινωνήσουμε με το αντικείμενο και να αλλάξουμε τα δεδομένα του.
- Οι γνωστοί τύποι δεδομένων **int, float, double, char** θεωρούνται πρωτογενείς τύποι δεδομένων.

Ένα παράδειγμα κλάσης (μητρώο εργαζομένων)

Πρότυπο δημιουργίας μητρώου εργαζομένων:

Ονοματεπώνυμο
Διεύθυνση
Μηνιαίες αποδοχές

Το πρότυπο ***(κλάση***) καθορίζει τη δομή του μητρώου κάθε εργαζομένου (***αντικείμενο)***

```
class EmployeeRecord {
      String name;
String address;
int  salary;

    public void setSalary (int employeeSalary ) {
      salary = employeeSalary;
}

    public int getSalary ( ) {
      return salary;
}
}
```

Πρόσβαση σε δεδομένα και μεθόδους αντικειμένων

- Πρόσβαση σε public μεταβλητές μέλη:
  - **<object>.<member variable>**
  - Π.χ. someEmployeeRecord.salary = 1500;

  (ανάθεση της τιμής 1500 στη μεταβλητή salary)

- Κλήση μεθόδων:
  - **<object>.<member function>**
  - Π.χ.
    1. income=someEmployeeRecord.getSalary( );

    (Η μεταβλητή income λαμβάνει την τιμή που επιστρέφει η getSalary ())
    2. someEmployeeRecord.setSalary(1500);

    (εκτέλεση της μεθόδου setSalary περνώντας ως όρισμα την τιμή 1500)

Παράδειγμα

```
class Employee {
        int salary;
        int bonus;
        public void setSalary (int employeeSalary) {
                salary = employeeSalary
        }
        int getSalary ( ) {
                return salary;
   }
}

Employee  John;

John =new Employee ( ); //Sos αγνοείτε πάντα τις εντολές  με κόκκινο !!!!!!!!!
                        //Είναι απαραίτητες  αλλά εξηγούνται σε επόμενες διαφάνειες!!!

John.setSalary (1200);
John.bonus = 500;
int someSalary = John.getSalary ( );
```

Γλώσσες Προγραμματισμού – Αντικειμενοστρεφής Προγραμματισμός

- C++
- Python
- Java
- Etc.

url για την java

- http://www.java.sun.com
- http://www.jcreator.com/download.htm

Λειτουργία της JAVA

- Η Java είναι σχεδιασμένη για το Internet και γενικά για δικτυακές εφαρμογές
- Στην java το compiling(μεταγλώττιση) χωρίζεται σε δύο μέρη.

  - Στο πρώτο μέρος a) γίνεται έλεγχος συντακτικών λαθών και b)παράγεται ένας Byte code ο οποίος είναι κοινός ανεξαρτήτως σε ποίο μηχάνημα ή λειτουργικό έγινε το compiling.

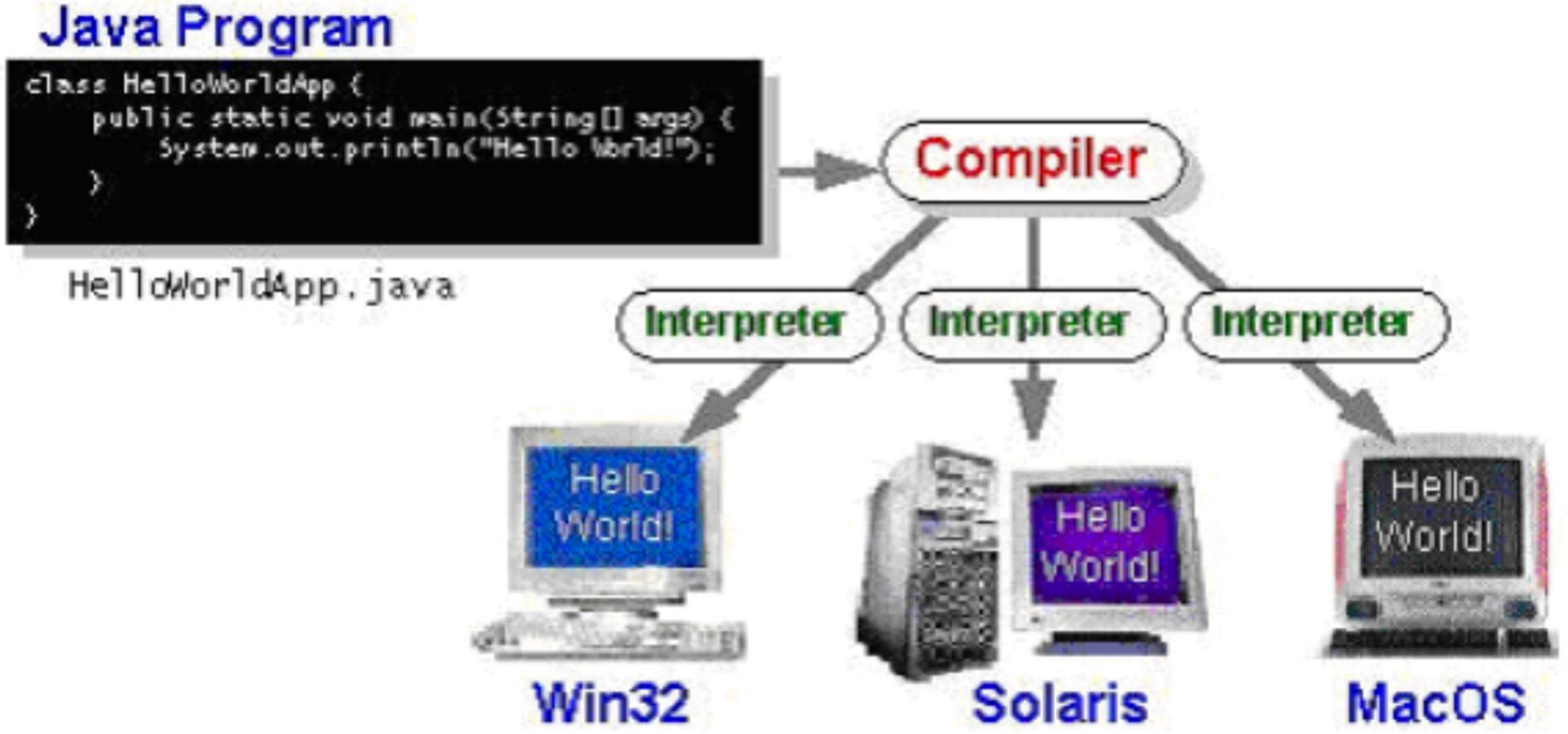
  -Στο δεύτερο μέρος γίνεται η μετάφραση του πιο πάνω Byte code σε εκτελέσιμο κώδικα μηχανής.

  Αυτό εξαρτάται από το είδος της μηχανής και του λειτουργικού που υπάρχει.

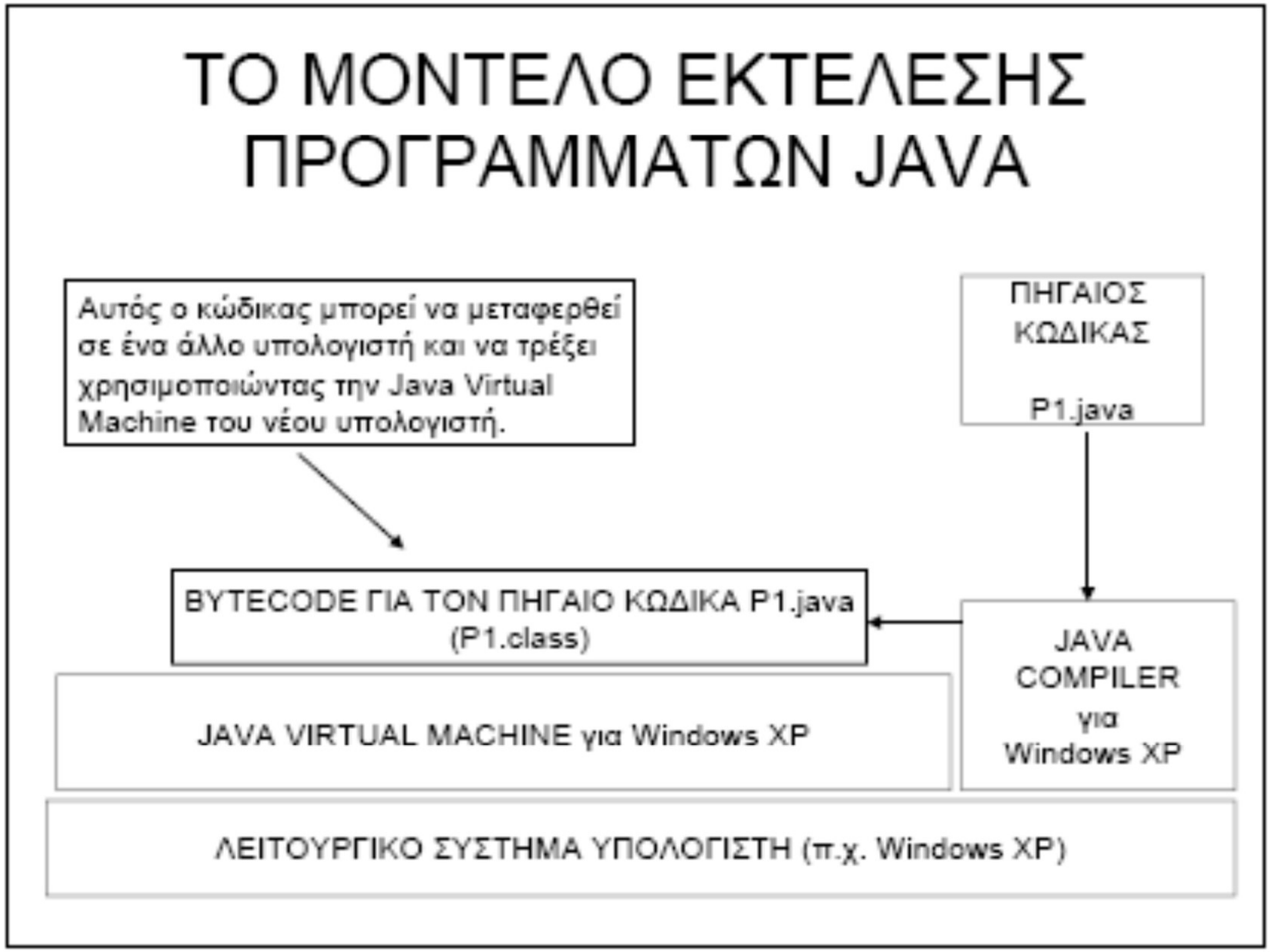
➔ Για τον λόγο αυτό όταν γίνεται εγκατάσταση της java σε ένα υπολογιστή εγκαθίσταται μαζί με τον Compiler και μια java virtual machine(JVM). Αυτή παίρνει τον Byte code βλέπει τι είδους μηχανή υπάρχει από κάτω και παράγει τον αντίστοιχο εκτελέσιμο κώδικα.

Λειτουργία της JVM

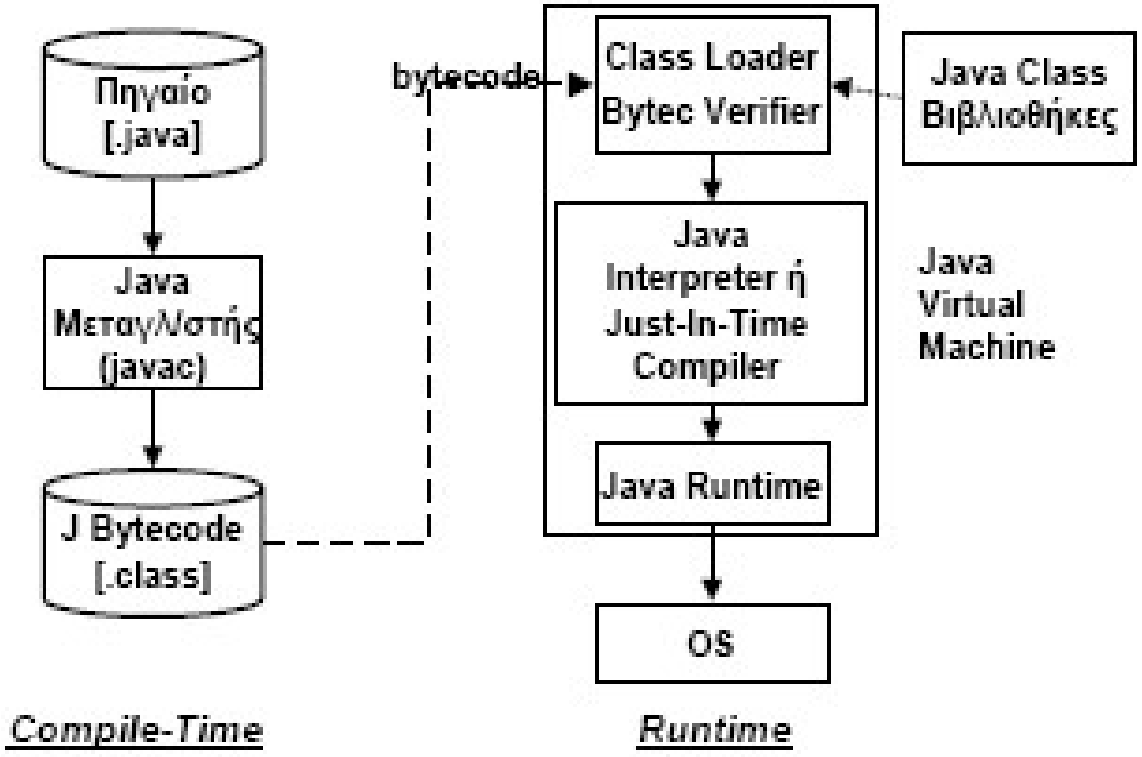
- Άρα μπορούμε να κάνουμε μεταγλώττιση σε ένα μηχάνημα π.χ. Solaris και να τρέχουμε τον byte code σε ένα μηχάνημα windows. Για τον λόγο αυτό είναι κατάλληλη για windows



ΤΟ ΜΟΝΤΕΛΟ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΩΝ JAVA



ΣΧΗΜΑΤΙΚΑ


Πηγαίο
[.java]
Java
Μεταγλωττής
(javac)
J Bytecode
[.class]
bytecode
Class Loader
Bytec Verifier
Java Class
Βιβλιοθήκες
Java
Interpreter ή
Just-In-Time
Compiler
Java Runtime
OS
Java
Virtual
Machine
Compile-Time
Runtime

ΠΑΡΑΤΗΡΗΣΕΙΣ

- Στο μοντέλο εκτέλεσης της Java:
  - Η Java Virtual Machine (JVM) εξαρτάται από το λειτουργικό σύστημα του Η/Υ στον οποίο έχει εγκατασταθεί.
  - Ο συμβολομεταφραστής της Java (javac) εξαρτάται και αυτός από το λειτουργικό σύστημα του Η/Υ στον οποίο έχει εγκατασταθεί.
  - Όμως όλοι οι μεταγλωττιστές (ανεξάρτητα του λειτουργικού συστήματος) παράγουν τον ίδιο bytecode για το ίδιο πηγαίο πρόγραμμα. Επίσης όλες οι JVM (ανεξάρτητα του λειτουργικού συστήματος στο οποίο τρέχουν) εκτελούν με τον ίδιο τρόπο ένα πρόγραμμα σε Java bytecode.
  - Οπότε εάν μεταφέρουμε μέσω Διαδικτύου (για παράδειγμα) bytecode σε διαφορετικούς υπολογιστές που έχουν εγκαταστημένη κάποια JVM τότε αυτό το πρόγραμμα θα εκτελεστεί επιτυχώς ανεξάρτητα από το λειτουργικό σύστημα
  - Το παραπάνω κέρδος έχει το τίμημα ότι στη Java χρειάζεται μια επιπλέον λειτουργία μετατροπής (από το bytecode στη γλώσσα μηχανής του συστήματος) σε σχέση με τις C/C++, οπότε η ταχύτητα εκτέλεσης του προγράμματος είναι μικρότερη (αυτό βέβαια σε πολλές περιπτώσεις στην πράξη δεν αποτελεί πρόβλημα)

Πρώτο πρόγραμμα σε java

- Για να δημιουργήσομε ένα πρόγραμμα java πρέπει να έχει κατάληξη .java
- Κάθε πρόγραμμα java έχει μια κλάση με το ίδιο όνομα με το αρχείο. Πχ.το

HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Πρώτο πρόγραμμα σε java

- Όπως και στην C χρειαζόμαστε μια συνάρτηση main η οποία θα αποτελεί το κυρίως πρόγραμμα.
- Η System.out.println είναι η εντολή για εκτύπωση και δημιουργία νέα γραμμής

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

JAVA vs. C

- Η Java όσον αφορά τη σύνταξη δεν διαφέρει κατά πολύ από την C.
- Μια πολύ σημαντική διαφορά είναι ότι η Java υποστηρίζει το ιδίωμα του ***οντοκεντρικού*** ή ***αντικειμενοστρεφούς προγραμματισμού*** (***object-oriented programming***).
- Πριν αναφερθούμε στα αντικειμενοστρεφή χαρακτηριστικά της Java, θα δούμε τις διαφορές με τη C στα βασικά στοιχεία των δύο γλωσσών (τύποι δεδομένων, μεταβλητές, τελεστές, βασικές εντολές)

```
class HelloWorld
{
     public static void main(String args[])
     {
          // print message
          System.out.println("Hello world!");
      }
}
```

Λέξεις σε ΚΌΚΚΙΝΟ: δεσμευμένες λέξεις

Ορίζει την κλάση

Όνομα της κλάσης

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Τα άγκιστρα { … } ορίζουν ένα λογικό block του κώδικα

- Αυτό μπορεί να είναι μία κλάση, μία συνάρτηση, ένα if statement
- Οι μεταβλητές που ορίζουμε μέσα σε ένα λογικό block, έχουν εμβέλεια μέσα στο block
- Αντίστοιχο των tabs στην Python, εδώ δεν χρειάζονται αλλά είναι καλό να τα βάζουμε για να διαβάζεται ο κώδικας πιο εύκολα.

Ορισμός της συνάρτησης main

```
class HelloWorld
{
      public static void main(String args[])
       {
              // print message
              System.out.println("Hello world!");
        }
}
```

Ορισμός της συνάρτησης main

```
class HelloWorld
{
      public static void main(String args[])
       {
              // print message
              System.out.println("Hello world!");
        }
}
```

public, static: θα τα εξηγήσουμε αργότερα

Ορισμός της συνάρτησης main

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Το τι επιστρέφει η μέθοδος
void: Η μέθοδος δεν επιστρέφει τίποτα.

Ορισμός της συνάρτησης main

```
class HelloWorld
{
      public static void main(String args[])
       {
              // print message
              System.out.println("Hello world!");
        }
}
```

Το όνομα της μεθόδου

- main: ειδική περίπτωση που σηματοδοτεί το σημείο εκκίνησης του προγράμματος.

Ορισμός της συνάρτησης main

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Ορίσματα της μεθόδου

- Ένας πίνακας από Strings που αντιστοιχούν στις παραμέτρους με τις οποίες τρέχουμε το πρόγραμμα.

Η κλάση String

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

- String: κλάση που χειρίζεται τα αλφαριθμητικά.
- Στη Java χρειάζεται να ορίσουμε τον τύπο της κάθε μεταβλητής
- Strongly typed language

Σχόλια!

```
/**
 * A class that prints a message "hello world"
 **/
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
     }
}
```

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Κάθε εντολή στη Java πρέπει να τερματίζει με το ;

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Αντικείμενο `System.out`
Ορίζει το ρεύμα εξόδου

Μέθοδος `println`:
Τυπώνει το String αντικείμενο που δίνεται ως όρισμα και αλλάζει γραμμή

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Το “Hello World” είναι ένα αντικείμενο της κλάσης String

Programming Style

Το όνομα της κλάσης ξεκινάει με κεφαλαίο και χρησιμοποιούμε την CamelCase σύμβαση

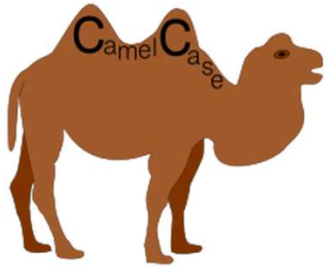
```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Στοίχιση του κώδικα:

- Οι εντολές μέσα σε ένα block του κώδικα ξεκινάνε ένα tab πιο μπροστά από το προηγούμενο.
- Όλες οι εντολές σε ένα block είναι στοιχισμένες
- Τα άγκιστρα είναι στοιχισμένα με την εντολή που ορίζει το block

Programming Style: Ονόματα

- Τα ονόματα των κλάσεων ξεκινάνε με κεφαλαίο, τα ονόματα των πεδίων, μεθόδων και αντικειμένων με μικρό.
  - Π.χ., HelloWorld
- Χρησιμοποιούμε ολόκληρες λέξεις (και συνδυασμούς τους) για τα ονόματα
  - Δεν πειράζει αν βγαίνουν μεγάλα ονόματα
- Χρησιμοποιούμε το CamelCase Style
  - Όταν για ένα όνομα έχουμε πάνω από μία λέξη, τις συνενώνουμε και στο σημείο συνένωσης κάνουμε το πρώτο γράμμα της λέξης κεφαλαίο
  - printName όχι print_name
- Χρησιμοποιούμε κεφαλαία και '_' για τις σταθερές.
  - Π.χ., PI_NUMBER



Παράδειγμα 2

- Φτιάξτε ένα πρόγραμμα που τυπώνει το λόγο δύο ακεραίων.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

- Ορισμός μεταβλητών
- Η Java είναι strongly typed γλώσσα: κάθε μεταβλητή θα πρέπει να έχει ένα τύπο.
- Οι τύποι int και double είναι πρωταρχικοί (βασικοί) τύποι (primitive types)
- Εκτός από τους βασικούς τύπους, όλοι οι άλλοι τύποι είναι κλάσεις

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής αντιστοιχίζεται με αυτό το χώρο στη μνήμη.

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής αντιστοιχίζεται με αυτό το χώρο στη μνήμη.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ανάθεση: αποτίμηση της τιμής της έκφρασης στο δεξιό μέλος του "=" και μετά ανάθεση της τιμής στην μεταβλητή στο αριστερό μέλος

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Μετατροπή τύπου (type casting): **(double)denominator** μετατρέπει την τιμή της μεταβλητής **denominator** σε **double.**
Αν δεν γίνει η μετατροπή, η διαίρεση μεταξύ ακεραίων μας δίνει πάντα ακέραιο.

Αναθέσεις

- Στην ανάθεση κατά κανόνα, η τιμή του δεξιού μέρους θα πρέπει να είναι ίδιου τύπου με την μεταβλητή του αριστερού μέρους.
- Υπάρχουν εξαιρέσεις όταν υπάρχει συμβατότητα μεταξύ τύπων

- byte → short → int → long → float → double
  - Μια τιμή τύπου T μπορούμε να την αναθέσουμε σε μια μεταβλητή τύπου που εμφανίζεται δεξιά του T.

- (Σε αντίθεση με την C) ο τύπος boolean δεν είναι συμβατός με τους ακέραιους.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ο τελεστής “+” μεταξύ αντικείμενων της κλάσης String συνενώνει (concatenates) τα δύο String.
Μεταξύ ενός String και ενός βασικού τύπου, ο βασικός τύπος μετατρέπεται σε String και γίνεται η συνένωση

HelloWorld.java

```
class HelloWorld
{
    public static void main(String args[])
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

- `javac HelloWorld.java`
- `java HelloWorld`

Χωρίς κανένα επίθεμα!

Παράδειγμα 2

- Φτιάξτε ένα πρόγραμμα που τυπώνει το λόγο δύο ακεραίων.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

- Ορισμός μεταβλητών
- Η Java είναι strongly typed γλώσσα: κάθε μεταβλητή θα πρέπει να έχει ένα τύπο.
- Οι τύποι int και double είναι πρωταρχικοί (βασικοί) τύποι (primitive types)
- Εκτός από τους βασικούς τύπους, όλοι οι άλλοι τύποι είναι κλάσεις

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής αντιστοιχίζεται με αυτό το χώρο στη μνήμη.

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής αντιστοιχίζεται με αυτό το χώρο στη μνήμη.

Η μνήμη του υπολογιστή

- Η κύρια μνήμη (main memory) του υπολογιστή κρατάει τα δεδομένα (και τις εντολές) για την εκτέλεση των προγραμμάτων.
  - Η μνήμη είναι προσωρινή, τα δεδομένα χάνονται όταν ολοκληρωθεί το πρόγραμμα.
- Η μνήμη είναι χωρισμένη σε bytes (8 bits)
  - Ο χώρος που χρειάζεται για ένα χαρακτήρα ASCII.
- Το κάθε byte έχει μια διεύθυνση, με την οποία μπορούμε να προσπελάσουμε τη συγκεκριμένη θέση μνήμης
  - Random Access Memory (RAM)
  - Σε 32-bit συστήματα μια διεύθυνση είναι 32 bits, σε 64-bit συστήματα μια διεύθυνση είναι 64 bits.

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	‘a’
0001	‘b’
0010	‘c’
0011	‘d’
0100	‘e’
0101	‘f’
0110	‘g’
0111	‘h’

Αποθήκευση μεταβλητών

- Η κύρια μνήμη (main memory) του υπολογιστή κρατάει τις μεταβλητές ενός προγράμματος
- Μια μεταβλητή μπορεί να απαιτεί χώρο περισσότερο από 1 byte.
  - Π.χ., οι μεταβλητές τύπου double χρειάζονται 8 bytes.
  - Η μεταβλητή τότε αποθηκεύεται σε συνεχόμενα bytes στη μνήμη.
- Η θέση μνήμης (διεύθυνση) της μεταβλητής θεωρείται το πρώτο byte από το οποίο ξεκινάει η αποθήκευση του της μεταβλητής.
  - Στο παράδειγμα μας η μεταβλητή βρίσκεται στη θέση 0000
  - Αν ξέρουμε την αρχή και το μέγεθος της μεταβλητής μπορούμε να τη διαβάσουμε.
- Άρα μία θέση μνήμης αποτελείται από μία διεύθυνση και το μέγεθος.

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0000	8.5
0001	
0010	
0011	
0100	
0101	
0110	
0111	

Αποθήκευση μεταβλητών πρωταρχικού τύπου

- Για τις μεταβλητές πρωταρχικού τύπου (char, int, double,…) ξέρουμε εκ των προτέρων το μέγεθος της μνήμης που χρειαζόμαστε.
- Όταν ο μεταγλωττιστής δει τη δήλωση μιας μεταβλητής πρωταρχικού τύπου δεσμεύει μια θέση μνήμης αντίστοιχου μεγέθους
  - Η δήλωση μιας μεταβλητής ουσιαστικά δίνει ένα όνομα σε μία θέση μνήμης
  - Συχνά λέμε η θέση μνήμης x για τη μεταβλητή x.

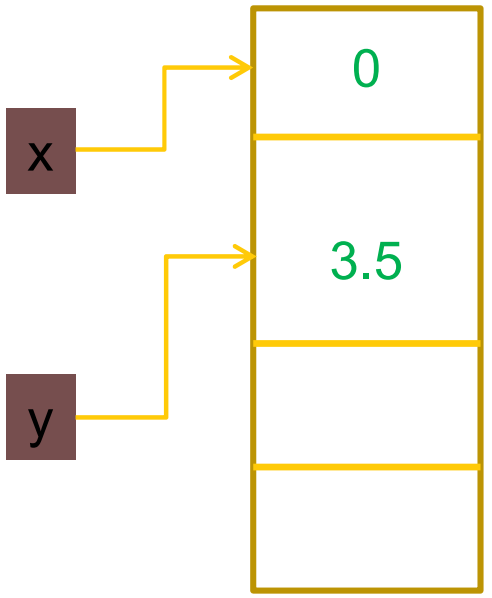
```
int x = 5;
int y = 3;
```

	Διεύθυνση μνήμης	Περιεχόμενο μνήμης
x	0000	5
	0001	
	0010	
	0011	
y	0100	3
	0101	
	0110	
	0111	

Αποθήκευση μεταβλητών

- Μπορούμε να σκεφτόμαστε την μνήμη του υπολογιστή σαν μια σειρά από «κουτάκια» διαφόρων μεγεθών στα οποία μπορούμε να αποθηκεύουμε δεδομένα
  - Το κάθε κουτάκι έχει διεύθυνση και περιεχόμενα
- Όταν ορίζουμε μια μεταβλητή πρωταρχικού τύπου (π.χ., `int x`) αυτό που γίνεται είναι ότι:
  - Δεσμεύουμε ένα κουτάκι κατάλληλου μεγέθους
    - 4 bytes για ένα int
  - Δίνουμε στο κουτάκι το όνομα της μεταβλητής
    - Το κουτάκι `x` αντί για το κουτάκι `0110`
  - Γι αυτό και η μεταβλητή ορίζεται μόνο μια φορά.
- Με την ανάθεση αλλάζουμε τα περιεχόμενα του κουτιού
- Αν δεν αρχικοποιήσουμε μια μεταβλητή η Java την αρχικοποιεί στο μηδέν (ή το αντίστοιχο του μηδέν για μη αριθμητικές μεταβλητές)

```
int x;
double y = 3.5;
```



Division.java

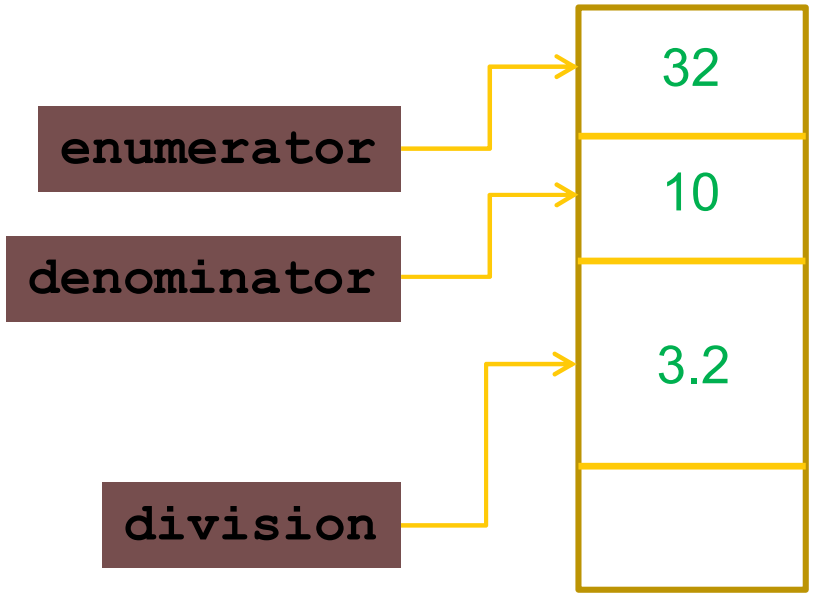
```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ανάθεση: αποτίμηση της τιμής της έκφρασης στο δεξιό μέλος του "=" και μετά ανάθεση της τιμής στην μεταβλητή στο αριστερό μέλος

Σε μια ανάθεση το αριστερό μέλος πρέπει να είναι μια μεταβλητή

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division =
        enumerator/(double)denominator;
    System.out.println(
                  "Result = " + division);
  }
}
```



Ανάθεση: Διαβάζουμε τα περιεχόμενα των μεταβλητών **enumerator** και **denominator** κάνουμε τον υπολογισμό και αλλάζουμε τα περιεχόμενα της μεταβλητής **division** αποθηκεύοντας το αποτέλεσμα της διαίρεσης.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Μετατροπή τύπου (type casting): **(double)denominator** μετατρέπει την τιμή της μεταβλητής **denominator** σε **double.**
Αν δεν γίνει η μετατροπή, η διαίρεση μεταξύ ακεραίων μας δίνει πάντα ακέραιο.

Αναθέσεις

- Στην ανάθεση κατά κανόνα, η τιμή του δεξιού μέρους θα πρέπει να είναι ίδιου τύπου με την μεταβλητή του αριστερού μέρους.
- Υπάρχουν εξαιρέσεις όταν υπάρχει συμβατότητα μεταξύ τύπων

- byte → short → int → long → float → double
  - Μια τιμή τύπου T μπορούμε να την αναθέσουμε σε μια μεταβλητή τύπου που εμφανίζεται δεξιά του T.

- (Σε αντίθεση με την C) ο τύπος boolean δεν είναι συμβατός με τους ακέραιους.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ο τελεστής “+” μεταξύ αντικείμενων της κλάσης String συνενώνει (concatenates) τα δύο String.
Μεταξύ ενός String και ενός βασικού τύπου, ο βασικός τύπος μετατρέπεται σε String και γίνεται η συνένωση

Strings

- Η κλάση String είναι προκαθορισμένη κλάση της Java που μας επιτρέπει να χειριζόμαστε αλφαριθμητικά.
- Ο τελεστής "+" μας επιτρέπει την συνένωση
- Υπάρχουν πολλές χρήσιμες μέθοδοι της κλάσης String. Η πιο χρήσιμη αυτή τη στιγμή είναι:
  - equals(String x): ελέγχει για ισότητα του αντικειμένου που κάλεσε την μέθοδο και του ορίσματος x.
- Άλλες μέθοδοι:
  - length(): μήκος του String
  - trim(): αφαιρεί κενά στην αρχή και το τέλος του string.
  - split(char delim): σπάει το string σε πίνακα από strings με βάσει το χαρακτήρα delim.
  - Επίσης, μέθοδοι για να βρεθεί ένα υπο-string μέσα σε ένα string, κλπ.

Escape sequences

- Για να τυπώσουμε κάποιους ειδικούς χαρακτήρες (π.χ., τον χαρακτήρα “) χρησιμοποιούμε τον χαρακτήρα \ και μετά τον χαρακτήρα που θέλουμε να τυπώσουμε
  - Π.χ., ακολουθία \”
- Αυτό ισχύει γενικά για ειδικούς χαρακτήρες.

\b	Backspace
\t	Tab
\n	New line
\f	Form feed
\r	Return (ENTER)
\”	Double quote
\’	Single quote
\\	Backslash
\ddd	Octal code
\uXXXX	Hex-decimal code

Ρεύματα εισόδου/εξόδου

- Τι είναι ένα ρεύμα? Μια αφαίρεση που αναπαριστά μια πηγή (για την είσοδο), ή ένα προορισμό (για την έξοδο) χαρακτήρων
  - Αυτό μπορεί να είναι ένα αρχείο, το πληκτρολόγιο, η οθόνη.
  - Όταν δημιουργούμε το ρεύμα το συνδέουμε με την ανάλογη πηγή, ή προορισμό.

Είσοδος & Έξοδος

- Τα βασικά ρεύματα εισόδου/εξόδου είναι έτοιμα αντικείμενα τα οποία ορίζονται σαν πεδία (στατικά) της κλάσης System
  - System.out
  - System.in
  - System.err
- Μέσω αυτών και άλλων βοηθητικών αντικειμένων γίνεται η είσοδος και έξοδος δεδομένων ενός προγράμματος.
- Μια εντολή εισόδου/εξόδου έχει αποτέλεσμα το λειτουργικό να πάρει ή να στείλει χαρακτήρες από/προς την αντίστοιχη πηγή/προορισμό.

Έξοδος

- Μπορούμε να καλέσουμε τις μεθόδους του System.out:
  - println(String s): για να τυπώσουμε ένα αλφαριθμητικό s και τον χαρακτήρα '\n' (αλλαγή γραμμής)
  - print(String s): τυπώνει το s αλλά δεν αλλάζει γραμμή
  - printf: Formatted output
    - printf("%d",myInt); // τυπώνει ένα ακέραιο
    - printf("%f",myDouble); // τυπώνει ένα πραγματικό
    - printf("%.2f",myDouble); // τυπώνει ένα πραγματικό με δύο δεκαδικά

Είσοδος

- Χρησιμοποιούμε την κλάση Scanner της Java
  - import java.util.Scanner;
- Αρχικοποιείται με το ρεύμα εισόδου:
  - Scanner in = new Scanner(System.in);
- Μπορούμε να καλέσουμε μεθόδους της Scanner για να διαβάσουμε κάτι από την είσοδο
  - nextLine(): διαβάζει μέχρι να βρει τον χαρακτήρα '\n'
  - next(): διαβάζει το επόμενο String μέχρι να βρει λευκό χαρακτήρα
  - nextInt(): διαβάζει τον επόμενο int
  - nextDouble(): διαβάζει τον επόμενο double.
  - nextBoolean(): διαβάζει τον επόμενο boolean.

Παράδειγμα

Με την εντολή αυτή φέρνουμε την κλάση Scanner μέσα στο πρόγραμμα μας ώστε να μπορούμε να φτιάξουμε αντικείμενα τύπου Scanner

```
import java.util.Scanner;

class TestIO
{
	public static void main(String args[])
	{
		System.out.println("Say Something:");
		Scanner input = new Scanner(System.in);
		String line = input.nextLine();
		System.out.println(line);
	}
}
```

new: δημιουργεί ένα αντικείμενο τύπου **Scanner** (μία μεταβλητή) με το οποίο μπορούμε πλέον να διαβάζουμε από την είσοδο.

- Το αντικείμενο αυτό αναπαριστά το πληκτρολόγιο στο πρόγραμμα μας. Ένα αντικείμενο φτάνει για να διαβάσουμε πολλαπλές τιμές.

Παράδειγμα

```
import java.util.Scanner;

class TestIO2
{
   public static void main(String args[])
   {
        Scanner input = new Scanner(System.in);
        double d= input.nextDouble();
        System.out.println("division by 4 = " + d/4);
        System.out.println("1+ (division by 4) = " +1+d/4);
        System.out.printf("1+ (division of %.2f by 4) = %.2f",d, 1+d/4);
   }
}
```

Το + λειτουργεί ως concatenation τελεστής μεταξύ Strings, άρα μετατρέπει τους αριθμούς σε Strings

Τι θα τυπώσει αυτό το πρόγραμμα?

Λογικοί τελεστές

- Λογικοί τελεστές για λογικές εκφράσεις
  - Άρνηση: `!B`
  - ΚΑΙ: `(A && B)`
  - Ή: `(A || B)`

- Έλεγχος για βασικούς τύπους A,B:
  - Ισότητας: `(A == B)`
  - Ανισότητας: `(A != B)` ή `(!(A == B))`
  - Μεγαλύτερο/Μικρότερο ή ίσο: `(A <= B)`, `(A >= B)`

- Έλεγχος για μεταβλητές (αντικείμενα) οποιουδήποτε άλλου τύπου γίνεται με την μέθοδο equals (πρέπει να έχει οριστεί):
  - Ισότητας: `(A.equals(B))`
  - Ανισότητας: `(!A.equals(B))`

- Λογικές σταθερές:
  - true: αληθές
  - false: ψευδές

Έλεγχος ισότητας για Strings

- Αν έχουμε δύο μεταβλητές String για να ελέγξουμε αν έχουν την ίδια τιμή πρέπει να χρησιμοποιήσουμε την μέθοδο equals.
- Παράδειγμα:

```
String firstString = "abc";
String secondString = "ABC";
boolean test1 = firstString.equals(secondString);
boolean test2 = firstString.equals("abc");
```

- Η παρακάτω εντολή δεν είναι σωστή

```
boolean test3 = (firstString == secondString);
```

- Περνάει από τον compiler και σε κάποιες περιπτώσεις θα δουλέψει αλλά δεν κάνει αυτό που θέλουμε.

ΕΙΣΑΓΩΓΗ ΣΤΗΝ JAVA II

Είσοδος – Έξοδος

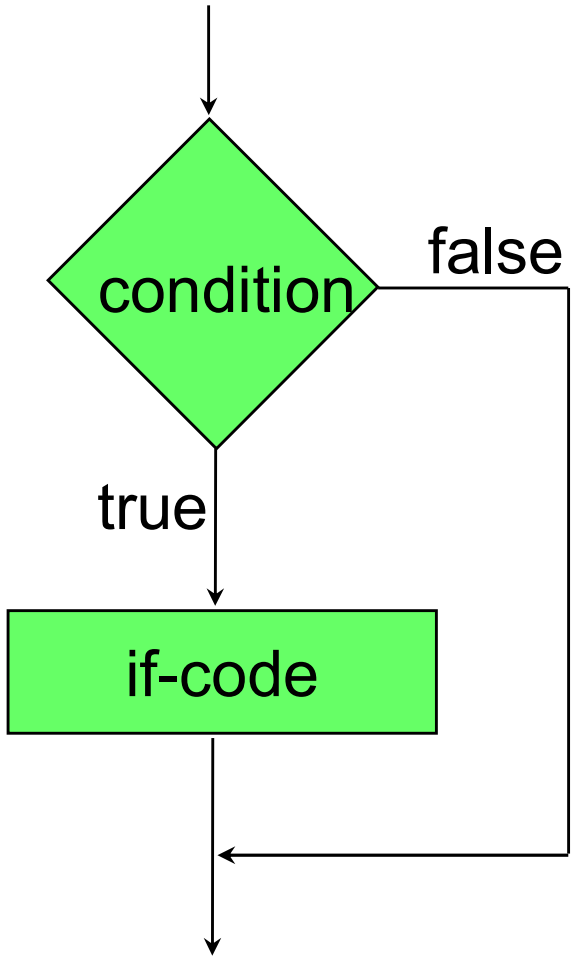
Έλεγχος ροής

Βρόγχοι – Το if-then Statement

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

```
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 IfTest1
{
    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");
        }
    }
}
```

```
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 == true){
            System.out.println(inputInt +
                                " is positive");
        }
    }
}
```

```
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 ισότητα

Programming Style: Λογικές μεταβλητές

- Συνηθίζεται όταν ορίζουμε λογικές μεταβλητές το όνομα τους να είναι αυτό που εκφράζει 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 γινεται true όταν έχουμε ισότητα και η μέθοδος isEmpty που είναι true όταν έχουμε άδειο String.

Βρόγχοι – Το if-then-else Statement

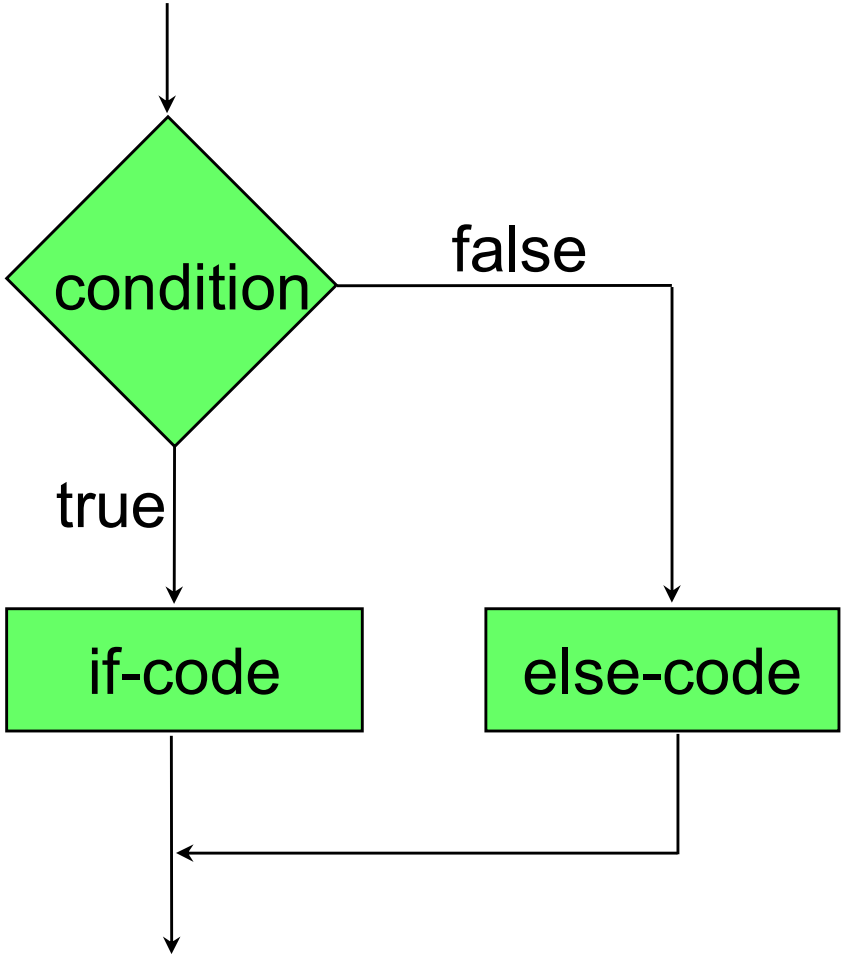
- Στην Java το if-then-else statement έχει το εξής συντακτικό

```
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.



```
import java.util.Scanner;

class IfTest2
{
    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{
              System.out.println(inputInt +
                                   " is not positive");
        }
    }
}
```

```
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 not positive");
        }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.