

ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Εισαγωγή στη Java

Ιστορία

- Ο Patrick Naughton απειλεί την Sun ότι θα φύγει.
- Τον βάζουν σε μία ομάδα αποτελούμενη από τους James Gosling και Mike Sheridan για να σχεδιάσουν τον προγραμματισμό των έξυπνων συσκευών της επόμενης γενιάς.
 - The Green project.
- Ο Gosling συνειδητοποιεί ότι η C++ δεν είναι αρκετά αξιόπιστη για να δουλεύει σε συσκευές περιορισμένων δυνατοτήτων και με διάφορες αρχιτεκτονικές.
 - Δημιουργεί τη γλώσσα Oak
- Το 1992 η ομάδα κάνει ένα demo μιας συσκευής PDA, *7 (star 7)
 - Δημιουργείται η θυγατρική εταιρία FirstPerson Inc
- Η δημιουργία των έξυπνων συσκευών αποτυγχάνει και η ομάδα (μαζί με τον Eric Schmidt) επικεντρώνεται στην εφαρμογή της πλατφόρμας στο Internet.
 - Ο Naughton φτιάχνει τον WebRunner browser (μετα HotJava)
 - Η γλώσσα μετονομάζεται σε Java και το ενδιαφέρον επικεντρώνεται σε εφαρμογές που τρέχουν μέσα στον browser.
- Ο Marc Andersen ανακοινώνει ότι ο Netscape browser θα υποστηρίζει Java μικροεφαρμογές (applets)

Ιστορία

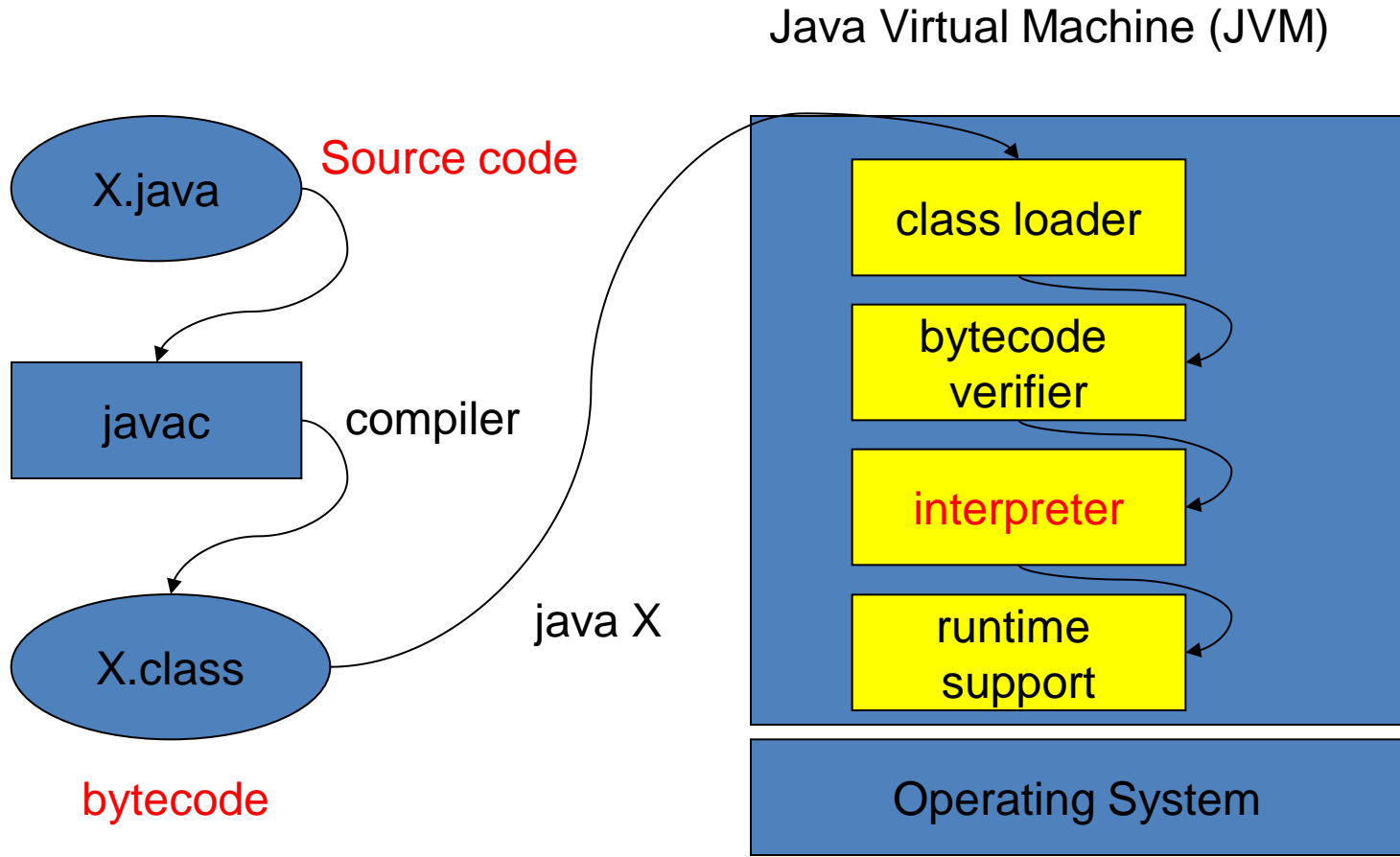
- Η Java είχε τους εξής στόχους:
 - "simple, object-oriented and familiar"
 - "robust and secure"
 - "architecture-neutral and portable"
 - "high performance"
 - "interpreted, threaded, and dynamic"

Ιστορία

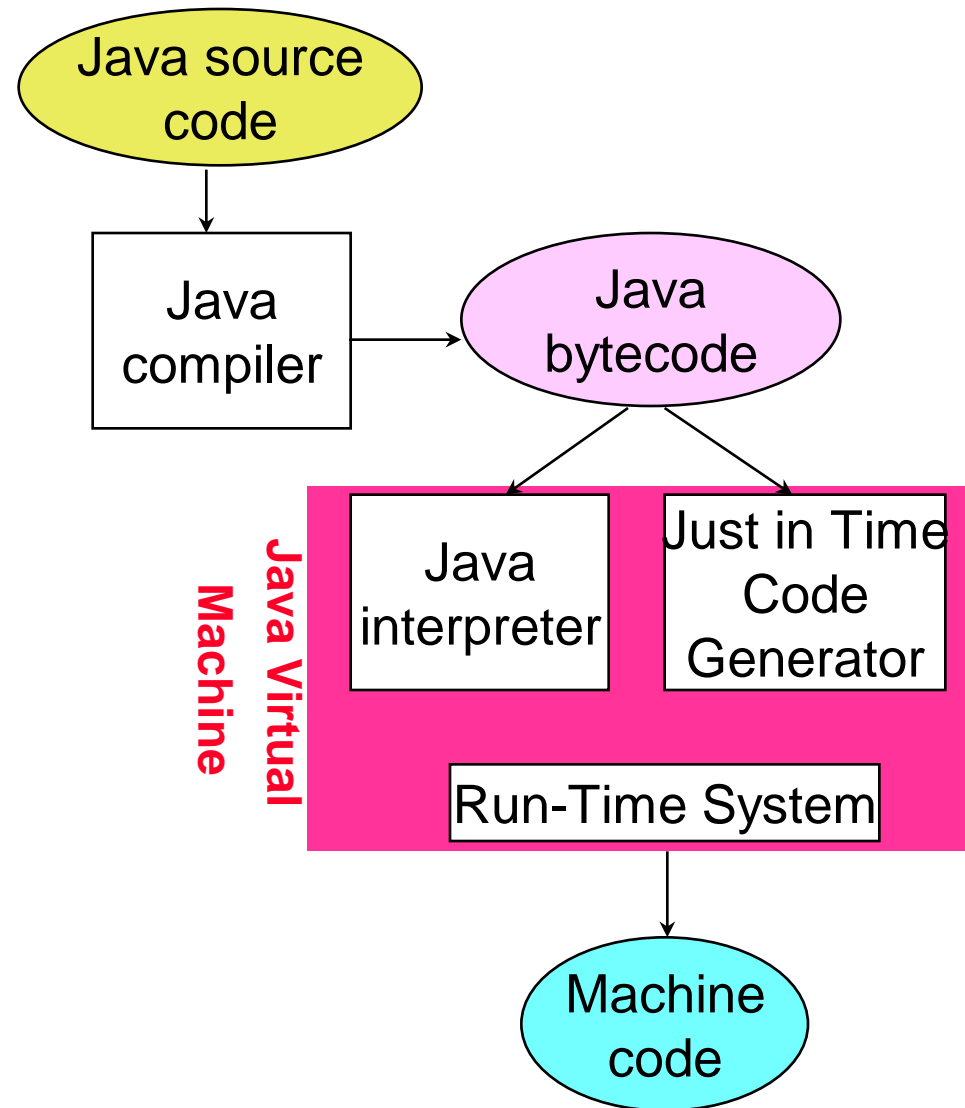
- Η Java είχε τους εξής στόχους:
 - "simple, object-oriented and familiar"
 - "robust and secure"
 - "architecture-neutral and portable"
 - "high performance"
 - "interpreted, threaded, and dynamic"

“architecture-neutral and portable”

- Το μεγαλύτερο πλεονέκτημα της Java είναι η **μεταφερισιμότητα (portability)**: ο κώδικας μπορεί να τρέξει πάνω σε οποιαδήποτε πλατφόρμα.
 - **Write-Once-Run-Anywhere** μοντέλο, σε αντίθεση με το σύνηθες **Write-Once-Compile-Anywhere** μοντέλο.
- Αυτό επιτυγχάνεται δημιουργώντας ένα **ενδιάμεσο κώδικα (bytecode)** ο οποίος μετά τρέχει πάνω σε μια **εικονική μηχανή (Java Virtual Machine)** η οποία το μεταφράζει σε γλώσσα μηχανής.
 - Οι προγραμματιστές πλέον γράφουν κώδικα για την εικονική μηχανή, η οποία δημιουργείται **για οποιαδήποτε πλατφόρμα**.



- **Just in Time (JIT) code generator** βελτιώνει την απόδοση των Java Applications μεταφράζοντας (compiling) bytecode σε machine code πριν την εκτέλεση ή κατά τη διάρκεια της εκτέλεσης



Java και το Internet

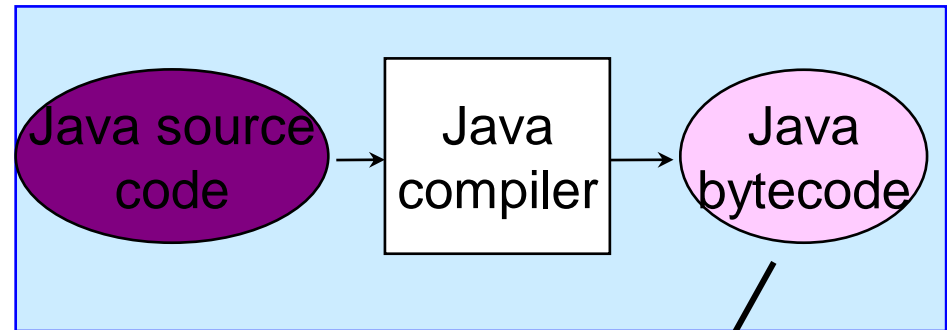
- Η προσέγγιση της Java είχε μεγάλη επιτυχία για Web εφαρμογές, όπου έχουμε ένα τεράστιο κατακευματισμένο **client-server** μοντέλο με πολλές διαφορετικές αρχιτεκτονικές
 - **Client-side programming**: Αντί να κάνει όλη τη δουλειά ο server για την δημιουργία της σελίδας κάποια από την επεξεργασία των δεδομένων γίνεται στη μηχανή του client.
 - **Web Applets**: κώδικας ο οποίος κατεβαίνει μαζί με τη Web σελίδα και τρέχει στη μηχανή του client. Είναι πολύ σημαντικό στην περίπτωση αυτή ο κώδικας να είναι portable.
 - **Server-side programming**: μία web σελίδα μπορεί να είναι το αποτέλεσμα ενός προγράμματος που συνδυάζει δυναμικά δεδομένα και είσοδο του χρήστη.
 - **Java Service Pages (JSPs)**: Η λύση της Java. Γίνεται compiled σε **servlets** και τρέχει στη μεριά του server.

"simple, object-oriented and familiar"

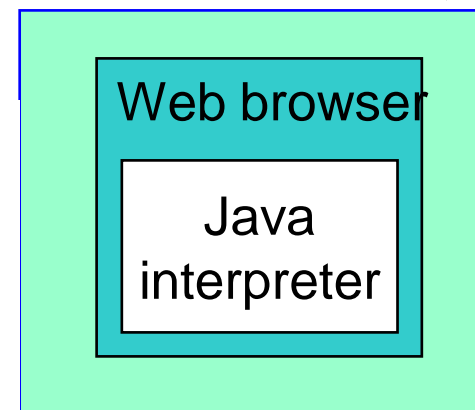
- **Familiar:** Η Java είχε ως έμπνευση της την C++, και δανείζεται αρκετά από τα χαρακτηριστικά της.
- **Object-oriented:** Η Java είναι «πιο αντικειμενοστραφής» από την C++ η οποία προσπαθεί να μείνει συμβατή με την C
 - Στην Java τα πάντα είναι αντικείμενα
- **Simple:** Η Java δίνει λιγότερο έλεγχο στο χρήστη, αλλά κάνει τη ζωή του πιο εύκολη. Η διαχείριση της μνήμης γίνεται αυτόματα.
 - Η γλώσσα φροντίζει να κάνει πιο γρήγορο και πιο robust τον προγραμματισμό παρότι αυτό μπορεί να έχει αποτέλεσμα τα προγράμματα να γίνονται πιο αργά.

Java Applets

- Το Internet Browser software περιλαμβάνει ένα **JVM**
 - ◆ **Φορτώνει** τον java byte code από τον remote υπολογιστή
 - ◆ **Τρέχει** τοπικά το Java Program μέσα στο παράθυρο του Browser



Remote computer



Local computer



HELLO WORLD

Το πρώτο μας πρόγραμμα σε Java

Δομή ενός απλού Java προγράμματος

- Το **όνομα** του αρχείου που κρατάει το πρόγραμμα είναι **X.java** (οπου X το όνομα του προγράμματος)
 - Στο παράδειγμα μας ονομάζουμε το πρόγραμμα μας: HelloWorld.java
- Μέσα στο πρόγραμμα μας πρέπει να έχουμε μια **κλάση** με το όνομα **X**.
 - `class X`
- Η κλάση **X** θα πρέπει να περιέχει μια **μέθοδο** `main` η οποία είναι το σημείο εκκίνησης του προγράμματος μας
 - `public static void main(String[] args)`

File HelloWorld.java

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

- javac HelloWorld.java
- java HelloWorld

```
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 strings.
- 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

Μέθοδος println:
Τυπώνει το String αντικείμενο που
δίνεται ως όρισμα

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

Το "Hello World" είναι ένα αντικείμενο της κλάσης String

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 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: `(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.
 - `length()`: μήκος του String
 - `equals(String x)`: τσεκάρει για ισότητα
 - `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.
 - 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

Ρευματα 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 συναρτήσεις:
 - `println(String s)`: για να τυπώσουμε ένα αλφαριθμητικό `s` και τον χαρακτήρα `'\n'`
 - `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 για να διαβάσουμε κάτι από την είσοδο
 - `nextLine()`: διαβάζει μέχρι να βρει τον χαρακτήρα '\n'
 - `next()`: διαβάζει το επόμενο String
 - `nextInt()`: διαβάζει τον επόμενο int
 - `nextDouble()`: διαβάζει τον επόμενο double.

Παράδειγμα

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

```
class TestIO
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        Scanner input = new Scanner(System.in);
```

```
        String line = input.nextLine();
```

```
        System.out.println(line);
```

```
    }
```

```
}
```

Λογικοί τελεστές

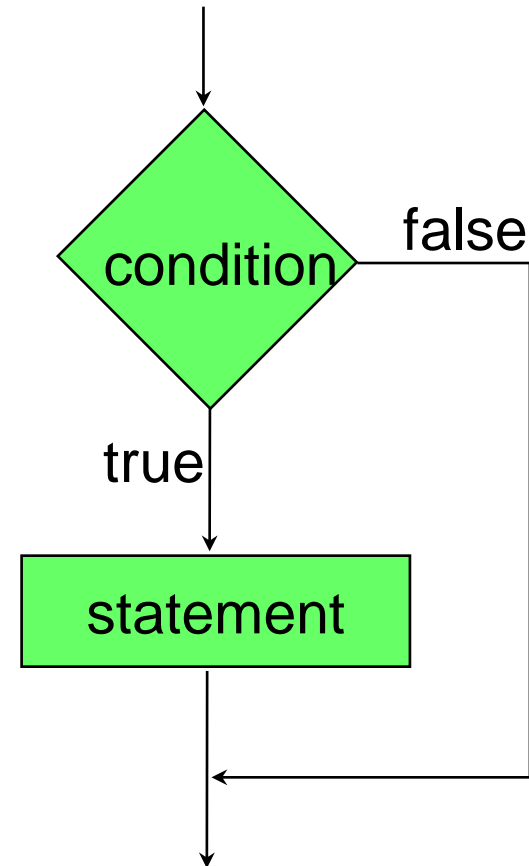
- Έλεγχος για βασικούς τύπους A,B:
 - Ισότητας: (A == B)
 - Ανισότητας: (A != B)
 - Μεγαλύτερο/Μικρότερο ή ίσο: (A <= B), (A >= B)
- Λογικοί τελεστές για λογικές εκφράσεις
 - Άρνηση: !B
 - ΚΑΙ: (A && B)
 - Ή: (A || B)

Βρόγχοι – Το if Statement

- Στη Java το **if** statement έχει το ακόλουθο ΣΥΝΤΑΚΤΙΚΟ:

```
if (condition) {  
    statement(s);  
}
```

- ◆ Αν η **λογική συνθήκη** είναι αληθής το **statement** εκτελείτε; Αν όχι, το **statement** προσπερνάται



Βρόγχοι – Το if-else statement

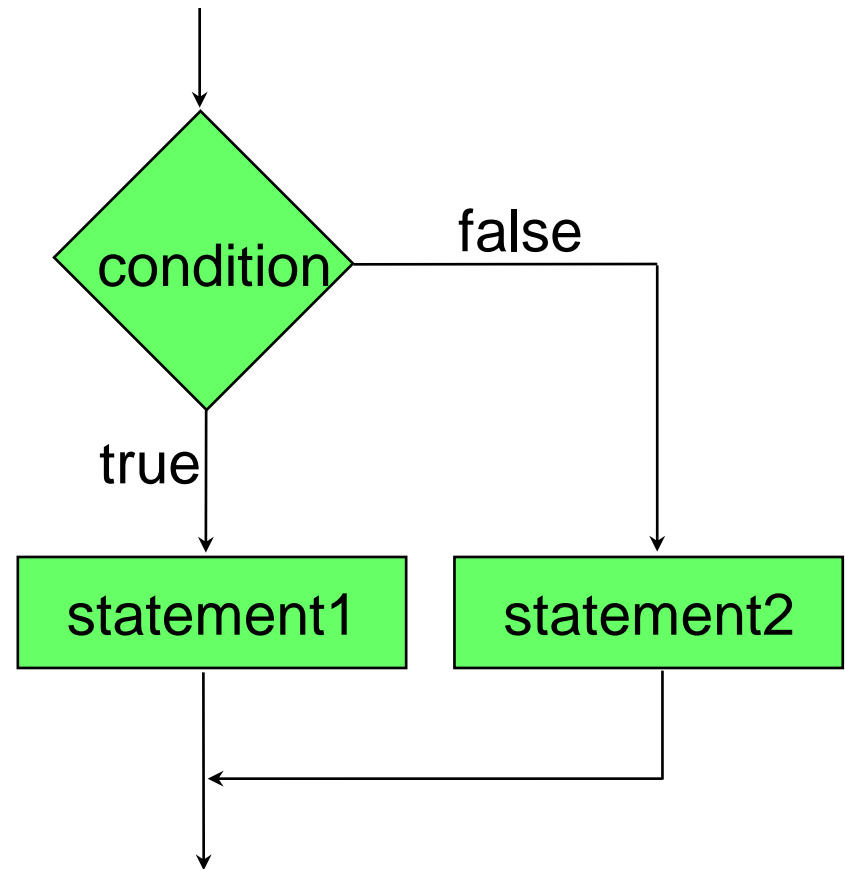
- Προσθέτοντας ένα **else** clause στο **if** statement παίρνουμε το **if-else** statement:

```
if (condition) {  
    statement(s)1;  
else {  
    statement(s)2;  
}
```

- Αν η συνθήκη είναι αληθής τότε το **statement1** εκτελείται; Αν η συνθήκη είναι ψευδής τότε το **statement2** εκτελείται

- Το σώμα του **if** statement ή του **else** statement μπορεί να είναι ένα άλλο **if** statement (**φωλιασμένο (nested) if statement**)

- Προσοχή: ένα **else** clause ταιριάζεται με το τελευταίο ελεύθερο **if** (no matter what the indentation implies)



Προσοχή!

WRONG!

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

CORRECT!

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

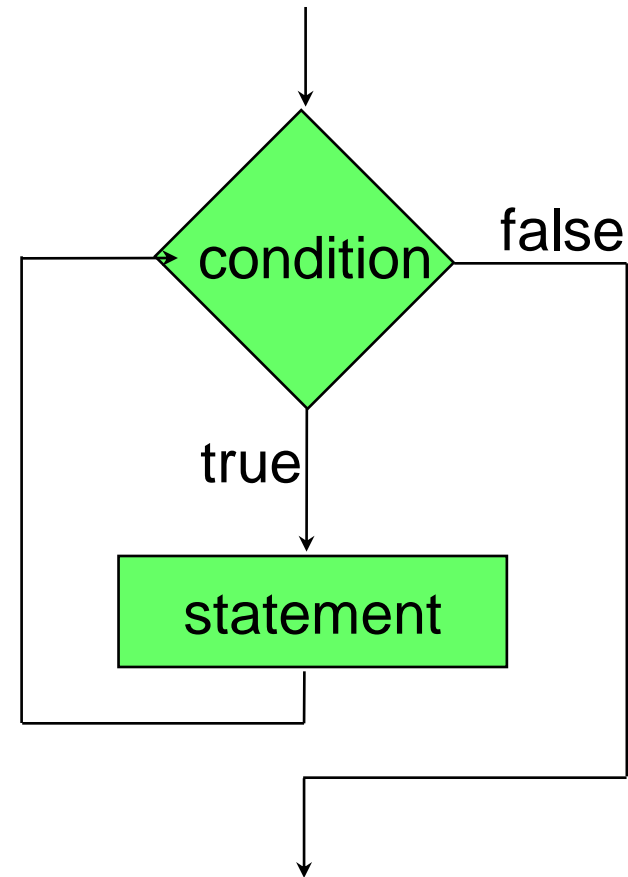
Πάντα να βάζετε `{ }` στο σώμα των if-else statements.

Επαναλήψεις - While statement

- Ένα **while** statement έχει το εξής συντακτικό:

```
Initialize the conditions  
while (condition) {  
    statement(s);  
}
```

- Αν η συνθήκη είναι αληθής, το *statement* εκτελείται; Μετά η συνθήκη αξιολογείται εκ νέου
- Τα Statements υλοποιούν τις επαναλήψεις και αλλάζουν την συνθήκη.
- Το *statement* εκτελείται μέχρι η συνθήκη να γίνει ψευδής



Επανάληψη – for statement

- Ένα **for** statement έχει το ακόλουθο συντακτικό:

```
for (initialization;  
    condition;  
    update)  
    {statement(s)};
```

- 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.

