ΕΙΣΑΓΩΓΗ ΣΤΗ JAVA

Reference

- *Thinking in Java*, Bruce Eckel, http://www.mindview.net/Books/TIJ/

Ιστορία

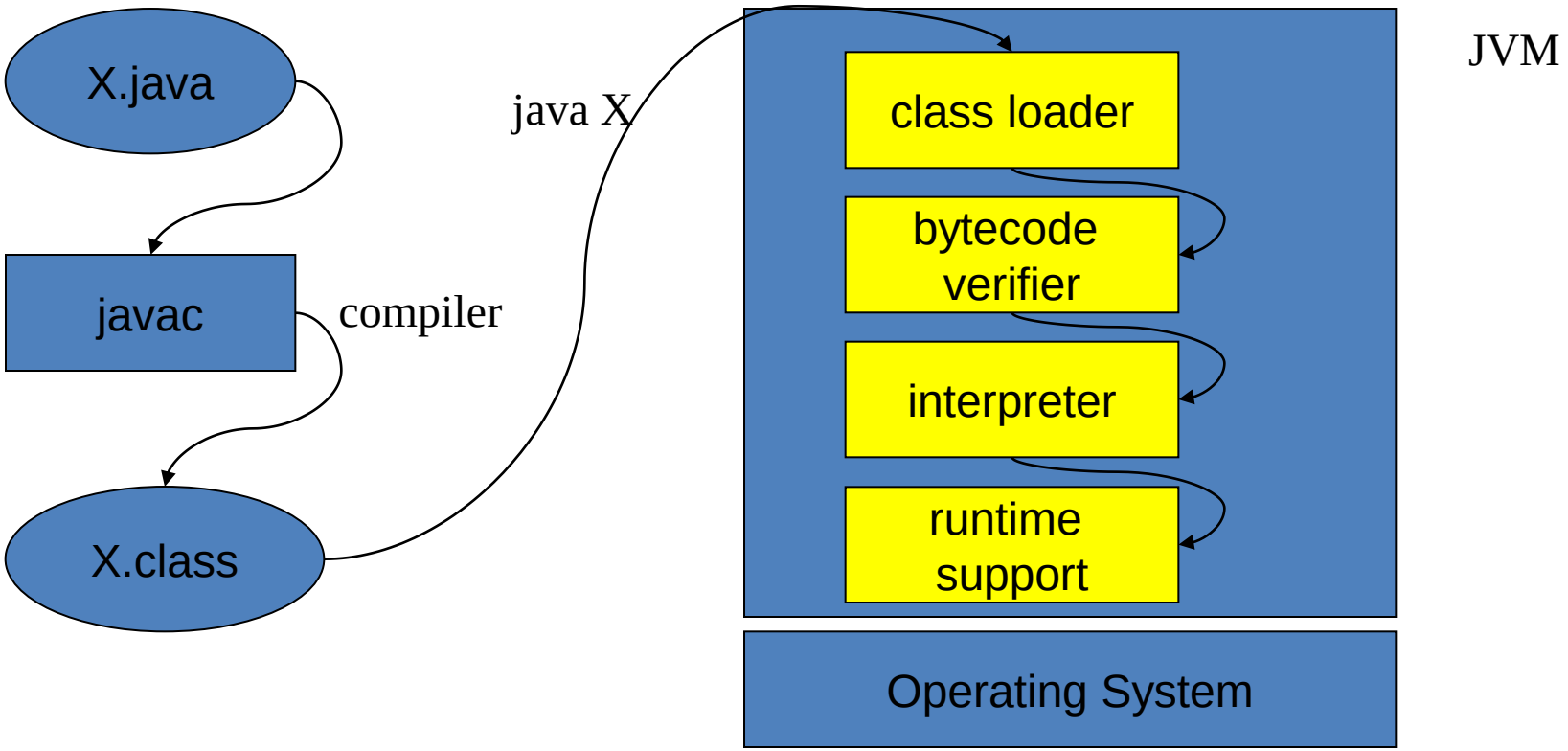
- Η Java δημιουργήθηκε από τον James Gosling στην Sun Microsystems το 1995. Η γλώσσα είχε τους εξής στόχους:
  - "simple, object-oriented and familiar"
  - "robust and secure"
  - "architecture-neutral and portable"
  - "high performance"
  - "interpreted, threaded, and dynamic"

Ιστορία

- Η Java δημιουργήθηκε από τον James Gosling στην Sun Microsystems το 1995. Η γλώσσα 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 μεταφερσιμότητα: 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.


X.java
javac
compiler
X.class
java X
class loader
bytecode verifier
interpreter
runtime support
Operating System
JVM

Java και το Internet

- Η προσέγγιση της Java είχε μεγάλη επιτυχία για Web εφαρμογές, όπου έχουμε ένα τεράστιο κατανεμημένο 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 αντικειμενοστρεφής» 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. Δεν υπάρχουν πλέον δείκτες και η διαχείριση της μνήμης γίνεται αυτόματα.
  - Η γλώσσα φροντίζει να κάνει πιο γρήγορο και πιο robust το προγραμματισμό παρότι αυτό μπορεί να έχει αποτέλεσμα τα προγράμματα να γίνονται πιο αργά.

Δομή προγράμματος

- Γενικά ένα απλό πρόγραμμα υλοποιείται σε ένα αρχείο X.java αποτελείται από
  - Ένα βασικό δομικό στοιχείο – κλάση - που ορίζεται ως public class X και περιλαμβάνει τον κώδικα της main του προγράμματος.
    - όνομα κλάσης == όνομα αρχειου !!…
    - πρέπει η main να δηλωθεί ως static void.
    - δέχεται σαν παραμέτρους από τη γραμμή εντολών ένα πίνακα από String.
    - το αρχείο μπορεί να περιέχει και άλλες μεθόδους
  - άλλα επιπλέον δομικά στοιχεία – κλάσεις - που ορίζονται ως non public κλάσεις που υλοποιούν τον υπόλοιπο κώδικα του προγράμματος.
- Τα πάντα υλοποιούνται μέσα σε κλάσεις.

Παράδειγμα

- Δημιουργήστε ένα πρόγραμμα που παίρνει το ύψος και το βάρος ενός ατόμου και τυπώνει το λόγο τους.

Αρχείο TestPerson.java

```
import java.lang.*;

class Person {
   private int weight;
   private int height;

   public Person(int w, int h) { // Constructor
          weight = w;
          height = h;
    }

    public void statistics(){ // Computes Ratio
          System.out.println(
              "Statistics for person : " + height/weight);
    }
}

public class TestPerson{
    // some classes that do some things
    public static void doSomething(){}
    public static void doSomethingElse(){}
    public static void doMore(){}

    public static void main(String args[]){
      int w = Integer.parseInt(args[0]); // Float.parseFloat(), ….
      int h = Integer.parseInt(args[1]);
      Person x = new Person(w,h); // create a person
      x.statistics(); // compute ratio

      w = Integer.parseInt(args[2]);
      h = Integer.parseInt(args[3]);
      Person y = new Person(w,h); // create another person
      y.statistics();
     }
}
```

Class Person

```
class Person {
  private int weight;
   private int height;

   public Person(int w, int h) { // Constructor
        weight = w;
        height = h;
     }

     public void statistics(){ // Computes Ratio
        System.out.println(
            "Statistics for person : " + height/weight);
     }
}
```

Class TestPerson

```
public class TestPerson{
   // some classes that do some things
   public static void doSomething(){}
   public static void doSomethingElse(){}
   public static void doMore(){}

   public static void main(String args[]){
    int w = Integer.parseInt(args[0]); //Float.parseFloat()…
    int h = Integer.parseInt(args[1]);
    Person x = new Person(w,h); // create a person
    x.statistics(); // compute ratio

    w = Integer.parseInt(args[2]);
    h = Integer.parseInt(args[3]);
    Person y = new Person(w,h); // create another person
    y.statistics();
     }
}
```

Compilation and Execution

- Compilation
  - javac TestPerson.java
  - Παράγει το μεταγλωττισμένο αρχείο TestPerson.class
- Execution
  - java TestPerson 70 170 80 160
  - Εκτελεί το πρόγραμμα με παραμέτρους εισόδου 70 170 80 160

Δομή Προγραμμάτων

- Εκδοχή 2η - πρόγραμμα που υλοποιείται σε περισσότερα από ένα αρχεία .java που βρίσκονται σε ένα κατάλογο.
  - Ένα από αυτά έχει τη δομή της εκδοχής 1
    - public class με main και 1 ή περισσότερες άλλες non public classes.
  - Καθένα από τα υπόλοιπα αρχεία περιέχει
    - Μια public κλάση (χωρίς μέθοδο main()) με όνομα το ίδιο όπως αυτό του αρχείου.
    - Άλλες non public κλάσεις που υλοποιούν τον υπόλοιπο κώδικα του προγράμματος.
    - Αντικείμενα της public κλάσης μπορούν να χρησιμοποιηθούν στον κώδικα που βρίσκεται στα υπόλοιπα αρχεία.
    - Αντικείμενα των non-public κλάσεων μπορούν να χρησιμοποιηθούν μόνο στον κώδικα του αρχείου στο οποίο βρίσκονται αυτές οι κλάσεις.

Δομή Προγραμμάτων

- Εκδοχή 3η - Γενικά ένα πρόγραμμα μπορεί να χωριστεί σε επιμέρους πακέτα - packages.
  - Κάθε package αντιστοιχεί σε ένα κατάλογο...
  - Για να χρησιμοποιήσουμε τις κλάσεις ενός πακέτου πρέπει να τις κάνουμε import
    - Π.χ.
      - `import java.lang.*;`
      - `import java.lang.Math;`

JAR

- JAR (Java Archive) είναι ένα archive file format (σαν το TAR) για Java αρχεία.
- Το πλεονέκτημα του jar format είναι ότι μπορούμε να χρησιμοποιήσουμε ένα jar αρχείο για να συμπιέσουμε ένα πρόγραμμα με πολλά αρχεία και να τρέξουμε το πρόγραμμα μας απ αυτό
  - java -jar TestPerson.jar 70 170 80 160
- Κάποια IDEs όπως το NetBeans δημιουργούν κατευθείαν ένα jar αρχείο.

Άλλα είδη προγραμμάτων

- Εφαρμογές Web – ο Java κώδικας καλείται μέσα από Web σελίδες
  - Εκτελείται από τον Web browser (applets) ή
- Κατανεμημένες εφαρμογές που εκτελούνται σε ένα τοπικό δίκτυο.
  - Επικοινωνία μέσω Java RMI, CORBA….
- Κατανεμημένες εφαρμογές (Web services) που συνδέονται προγραμματιστικά μέσω του διαδικτύου.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΓΛΩΣΣΑΣ JAVA

Η ιεραρχία της Java

- Όλα στην Java είναι οργανωμένα σε μια μοναδική ιεραρχία η οποία έχει στη ρίζα της την κλάση Object.
  - Αυτό έχει το πλεονέκτημα ότι μπορούμε να εκμεταλλευτούμε τις διάφορες σχέσεις μεταξύ κλάσεων.

Ορισμός κλάσεων

- Ο ορισμός των μεθόδων και των πεδίων γίνεται μαζί με τη δήλωση.

```
class Person {
 private int weight = 4;
 private int height = 50;

  public Person(int w, int h) { // Constructor
        weight = w;
        height = h;
   }

   public void statistics(){ // Computes Ratio
        System.out.println(
            "Statistics for person : " + height/weight);
   }
}
```

Δεν χρειάζεται “;”

Σχόλια

- Σχόλια όπως στη C, C++

```
/* this is a comment */
// this is another one
```

Βασικοί Τύποι

`boolean`	`true or false`
`char`	`1byte`
`byte`	`1byte integer`
`short`	`2byte integer`
`int`	`4byte integer`
`long`	`8byte integer`
`float`	`4byte real`
`double`	`8byte real`

Μεταβλητές

- Εκτός από τις μεταβλητές βασικού τύπου δεδομένων, όλες οι υπόλοιπες μεταβλητές είναι
  - Αναφορές σε αντικείμενα
    - κλάσης που προσφέρεται από τη Java (πχ String)
    - κλάσης του προγράμματος που κατασκευάζουμε εμείς
  - Διεύθυνση πίνακα
- Όλες οι μεταβλητές αρχικοποιούνται αυτόματα εκτός και αν τις αρχικοποιήσουμε κατά τη δήλωση
  - Οι μεταβλητές βασικού τύπου σε μηδέν.
  - Οι μεταβλητές μη βασικού τύπου σε null.
- Αρχικοποιούνται σε οποιοδήποτε σημείο του κώδικα,
  - είτε με χρήση της new που δεσμεύει χώρο για ένα νέο αντικείμενο, πίνακα, string
  - είτε με εκχώρηση της διεύθυνσης που περιέχεται σε μια άλλη μεταβλητή του ίδιου τύπου.
- Εκτός από τις μεταβλητές βασικού τύπου οι υπόλοιπες μεταβλητές δεσμεύουν χώρο από το heap και όχι το stack της μνήμης.

Class TestPerson

```
public class TestPerson{
   // some classes that do some things
   public static void doSomething(){}
   public static void doSomethingElse(){}
   public static void doMore(){}

   public static void main(String args[]){
    int w = Integer.parseInt(args[0]); //Float.parseFloat()…
    int h = Integer.parseInt(args[1]);
    Person x = new Person(w,h); // create a person
    x.statistics(); // compute ratio

    w = Integer.parseInt(args[2]);
    h = Integer.parseInt(args[3]);
    Person y = new Person(w,h); // create another person
    y.statistics();
     }
}
```

Wrapper Classes

- Η Java ορίζει wrapper classes για τους βασικούς τύπους δεδομένων.
  - Integer για τον int
  - Float για το float
  - κλπ
- Οι κλάσεις αυτές μας δίνουν επιπλέον ευελιξία στον χειρισμό των βασικών τύπων.

```
public String ratio(){ // makes string out of Ratio
    Integer w = weight;
    Integer h = height;
    String s = h.toString() + "/" + w.toString();
    return s;
 }
```

Static

- Μία μέθοδος ή ένα πεδίο είναι static όταν δεν συνδέονται με κανένα αντικείμενο, αλλά ανήκουν στην κλάση
  - Έχουμε πρόσβαση σε αυτές ακόμη και αν δεν έχουμε δημιουργήσει κάποιο αντικείμενο.
  - Αν δημιουργήσουμε αντικείμενο οι μεταβλητές είναι κοινές μεταξύ των αντικειμένων (μπορούν να χρησιμοποιηθούν ως global parameters/methods).
- Οι static μέθοδοι δεν έχουν πρόσβαση σε μη static δεδομένα ή μεθόδους.
- Τα ίδια ισχύουν και για την C++

Class TestPerson

```
public class TestPerson{
   // some classes that do some things
   public static void doSomething(){}
   public static void doSomethingElse(){}
   public static void doMore(){}

   public static void main(String args[]){
    int w = Integer.parseInt(args[0]); //Float.parseFloat()…
    int h = Integer.parseInt(args[1]);
    Person x = new Person(w,h); // create a person
    x.statistics(); // compute ratio

    w = Integer.parseInt(args[2]);
    h = Integer.parseInt(args[3]);
    Person y = new Person(w,h); // create another person
    y.statistics();
     }
}
```

Εντολές Ροής

```
if (x > 0)
  ++x;
else
 --x;

switch (x) {
 case 1: ...break;
 case 2: ...break;
 default:...break;
}

while(x > 0) x++;

for(int x = 4; x < 7; x++) { ..... }

for (float x: A) { … }
// διατρέχει τις τιμές του array A
```

Πίνακες

```
public class TestArrays {
  public static void main(String [] args){
   int arr0[];
   // int[] arr0;
   int arr1 [] = {1, 2, 3, 4};
   int arr2[] = new int [10];
   int arr3[][] = {{1, 2, 3}, {3, 4, 5}};
   int arr4[][] = new int [10][20];
   arr0 = new int [100];
   arr3 = arr4;
   System.out.println(arr3.length + " " +
        arr3[0].length);
  }
}
```

String

- Όπως και στην C++ υπάρχει μία κλάση String η οποία διευκολύνει τον χειρισμό των Strings
  - Επιτρέπει εύκολη ανάθεση, ψάξιμο μέσα στο string, συνδυασμό από strings
  - Για τα περισσότερα πράγματα που χρειάζεστε υπάρχει συνήθως μία μέθοδος.
- Δύο χρήσιμες μέθοδοι:
  - Trim(): Αφαιρεί αρχικά και τελικά whitespaces. Χρήσιμο για την αφαίρεση του carriage return αλλά και γενικά.
  - Split(delimiter): Κάνει split το αρχικό string με βάσει το delimiter και επιστρέφει ένα πίνακα από Strings με τα πεδία.

Strings are immutable objects

- Τα strings είναι immutable objects, δηλαδή το μέγεθος τους δεν αλλάζει.
- Οι εντολές:

```
String x = "abcd";
x = x + "e";
x = x + "f";
x = x + "g";
x = x + "h";
```

έχουν ως αποτέλεσμα κάθε φορά να δημιουργείται ένα νέο string το οποίο ανατίθεται στο x.

  - Αυτό είναι ακριβό υπολογιστικά.

StringBuilder

- Μπορούμε να χρησιμοποιούμε το StringBuilder γι αυτή τη δουλειά.

```
StringBuilder b = new StringBuilder("abcd");
b.append("e");
b.append("f");
b.append("g");
b.append("h");
String x = b.toString();
```

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….

IO – Read lines

```
import java.lang.*;
import java.io.*;

public class IOReadlines {

 public static void main (String args[]){
  try{
      InputStreamReader ir = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(ir);
      String line;

      while ((line = br.readLine()) != null){
           System.out.println(line);
      }
  } catch(IOException e) {
        System.out.println("Input error");
        System.exit(1);
  }
   }
}
```

IO – Read lines - Files

```
import java.io.*;

public class Main {
    public static void main(String[] args) {
        try{
            FileReader fr = new FileReader("InputFiles/f1.txt");
            BufferedReader br = new BufferedReader(fr);
            FileWriter fw = new FileWriter("InputFiles/f2.txt");
            PrintWriter pw = new PrintWriter(fw);
            String line;

            while((line = br.readLine())!=null){
                pw.println(line);
                System.out.println(line);
            }
            pw.close();
        } catch(IOException ex){
            System.out.println("IO Error" + ex);
        }
    }
}
```

IO Read Formatted Input

```
import java.lang.*;
import java.io.*;
public class tstIO3 {
    public static void main (String args[]){
        try{
            InputStreamReader ir = new InputStreamReader(System.in);
            StreamTokenizer st = new StreamTokenizer(ir);
            int tag; double number; String s;
            while((tag = st.nextToken()) != StreamTokenizer.TT_EOF) {
                   if (tag == StreamTokenizer.TT_NUMBER) {
                       number = st.nval;
                       System.out.println("Num : " + number);
                   } else
                       if(tag == StreamTokenizer.TT_WORD) {
                            s = st.sval; System.out.println("Word : " + s);
                       }
            }
        } catch(IOException e) {….}
    }
}
```

Εξαιρέσεις

- Ένα ***exception*** είναι ένα αντικείμενο το οποίο δημιουργείται και επιστρέφεται με την εντολή throw (αντί της return) από την κλήση κάποιας μεθόδου όταν προκύπτει μια μη φυσιολογική κατάσταση στον κώδικα της μεθόδου η οποία επιβάλει να διακοπεί η εκτέλεση αυτής της μεθόδου.
  - Πολλές έτοιμες κλάσεις στη Java επιβάλλουν στον κώδικα που τις χρησιμοποιεί να διαχειρίζεται τις εξαιρέσεις που επιστρέφουν…

Εξαιρέσεις

```
import java.lang.*;

class A {
    //...
    int f(int x, int y) throws IllegalArgumentException {
      if (y != 0) return x/y;
      else
          throw new IllegalArgumentException("y is 0");
    }

    int g(int index) throws IndexOutOfBoundsException  {
      int[] A = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
      if (index < 10) return A[index];
      else
          throw new IndexOutOfBoundsException("index is greater that 10");
    }
}
```

Εξαιρέσεις

αν δε διαχειριστούμε μια εξαίρεση τότε το πρόγραμμα μας διακόπτεται και τυπώνεται σχετικό μήνυμα !!

```
public class tstException{
    public static void main(String args[]){
      A a = new A();
      int ret;
      try {
          ret = a.f(1, 2);
          ret = a.g(8);
         ret = a.f(2, 0);
          ret = a.g(100);
      } catch (IllegalArgumentException retEx1) {
          System.out.println(retEx1.getMessage());
      } catch (IndexOutOfBoundsException retEx2){
          System.out.println(retEx2.getMessage());
      }
    }
}
```

Αν κάτι πάει στραβά δημιουργείται η εξαίρεση retEx

Για κάθε διαφορετικό τύπο εξαιρέσεων που μπορεί να προκύψουν λόγω της κλήσης μεθόδων στο try block φτιάχνουμε ένα catch clock
Σαν παράμετρος είναι η εξαίρεση που μπορεί να προκύψει

Εξαιρέσεις

```
// κώδικας main()….ή άλλης μεθόδου
try {
// κώδικας που καλεί μεθόδους σε αντικείμενα
// οι οποίες επιστρέφουν 1 ή περισσότερες
// εξαιρέσεις διαφορετικού τύπου
  κλήση μεθόδου 1
  κλήση μεθόδου 2
  κλήση μεθόδου 3
  ……
} catch (ExType1 retEx1) {
      System.out.println(retEx1.getMessage());
} catch (ExType2 retEx2) {
      System.out.println(retEx2.getMessage());
      return;
} catch (ExType3 retEx3) {
      System.out.println(retEx3.getMessage());
      System.exit(1);
}
// Εντολές που ακολουθούν
…….
```

- αν κάτι δεν πάει καλά κατά την εκτέλεση του κώδικα στο try block κατά την κλήση μιας μεθόδου που επιστρέφει μια εξαίρεση τύπου ExType1
  - η διεύθυνση της εξαίρεσης που επιστρέφεται αποθηκεύεται στη μεταβλητή εξαίρεση1_που_επιστρέφεται
  - ο κώδικας του try block παύει να εκτελείται και ξεκινάει η εκτέλεση του κώδικα του αντίστοιχου catch block
  - κατόπιν του τέλους της εκτέλεσης του catch block (αν δεν γίνει έξοδος του προγράμματος) συνεχίζεται η εκτέλεση με τις εντολές που ακολουθούν **μετά το try-catch block**

Εξαιρέσεις

- Ένα ***exception*** είναι ένα αντικείμενο το οποίο δημιουργείται και επιστρέφεται με την εντολή throw (αντί της return) από την κλήση κάποιας μεθόδου όταν προκύπτει μια μη φυσιολογική κατάσταση στον κώδικα της μεθόδου η οποία επιβάλει να διακοπεί η φυσιολογική εκτέλεση αυτής της μεθόδου.
  - Πολλές έτοιμες κλάσεις στη Java επιβάλλουν στον κώδικα που τις χρησιμοποιεί να διαχειρίζεται τις εξαιρέσεις που επιστρέφουν…
  - … με αυτόν τον τρόπο θα έχει πάντα την προσοχή του ο προγραμματιστής στο γεγονός ότι πρέπει να διαχειρίζεται κάποιες περιπτώσεις μη κανονικής εκτέλεσης – **ο compiler θα του το υπενθυμίζει**

Εξαιρέσεις

```
public class tstException{
    public static void main(String args[]){
       A a = new A();
       int ret;
       try {
           ret = a.f(1, 2);
           ret = a.g(8);
          ret = a.f(2, 0);
           ret = a.g(100);
       } catch (IllegalArgumentException retEx1) {
        } catch (IndexOutOfBoundsException retEx2){
        }
    }
}
```

… τι πρόβλημα έχουμε σε αυτή την περίπτωση ???

Εξαιρέσεις

```
public class tstException{
    public static void main(String args[]){
      A a = new A();
      int ret;
      try {
          ret = a.f(1, 2);
          ret = a.g(8);
         ret = a.f(2, 0);
          ret = a.g(100);
      } catch (IllegalArgumentException retEx1) {
       } catch (IndexOutOfBoundsException retEx2){
       }
    }
}
```

… συχνά στην πράξη κάτι που ενοχλεί τον προγραμματιστή όταν κάνει μια δουλειά, προσπαθεί γρήγορα να το βγάλει από τη μέση και να προχωρήσει στην επίτευξη του βασικού του στόχου…

… με την προοπτική να επιστρέψει στο θέμα που τον ενοχλεί αργότερα για να δώσει μια καλύτερη λύση….

…**πολλές φορές το αργότερα δεν έρχεται ποτέ**

Στην περίπτωση των εξαιρέσεων αυτό σημαίνει ότι μπορεί να έχουμε «αόρατες» εξαιρέσεις για τις οποίες δεν γίνεται κατάλληλη διαχείριση
Αν δεν υπήρχε το τετριμμένο (επιβεβλημένο από τον compiler) catch για αυτές, τουλάχιστον θα ήταν «ορατές»

Constructors

- Όπως και στην C++ αν δεν προσδιορίσουμε τον Constructor, τότε χρησιμοποιείται ο default.
- Στην περίπτωση της Java, ο default constructor κάνει κάτι: αρχικοποιεί όλα τα πεδία σε «μηδενική» τιμή.
- Όπως και στην C++ αν ορίσουμε ένα constructor, τότε ο default παύει να υφίσταται.

Class Person

```
class Person {
  private int weight;
   private int height;

   public Person(int w, int h) { // Constructor
        weight = w;
        height = h;
     }

     public void statistics(){ // Computes Ratio
        System.out.println(
            "Statistics for person : " + height/weight);
     }
}
```

Keyword this

- Ένα αντικείμενο μπορεί να χρησιμοποιήσει το keyword this για να αναφερθεί στον εαυτό του
  - Στην Java το this είναι αναφορά και όχι δείκτης
- Το this είναι χρήσιμο και για να καλέσουμε ένα constructor μέσα σε ένα άλλο constructor.

Class Person

```
class Person {
  private int weight;
   private int height;

    public Person(int w, int h) { // Constructor
        weight = w;
        height = h;
     }

    public Person(int h) { // Constructor
        this(h-100,w);
     }

     public void statistics(){ // Computes Ratio
        System.out.println(
            "Statistics for person : " + height/weight);
     }
}
```

Destructors?

- Στην Java δεν χρειαζόμαστε destructors!
- Η αποδέσμευση της μνήμης γίνεται αυτόματα από το garbage collector ο οποίος φροντίζει για τη διαγραφή των αντικειμένων που δεν χρησιμοποιούνται πλέον.
- Αν ο χρήστης θέλει να δώσει μια ένδειξη στον garbage collector μπορεί να χρησιμοποιήσει την εντολή finalize().

Garbage Collector

- Ένα πρόγραμμα που τρέχει στο background και φροντίζει για την αποδέσμευση μνήμης.
- Πως βρίσκει τι πρέπει να αποδεσμευτεί?
  - Reference Counting: Πόσοι χρησιμοποιούν το reference στο αντικείμενο. Έχει το πρόβλημα ότι είναι ακριβό να βρεις κυκλικές αναφορές μεταξύ «νεκρών» αντικειμένων.
  - Η τεχνική που χρησιμοποιείται είναι να βρίσκεις όλα τα «ζωντανά» αντικείμενα ξεκινώντας από την μνήμη στο stack. Όσα δεν βρεις είναι νεκρά.
- Πως τα ελευθερώνει?
  - Stop-and-Copy: βρίσκει όλα τα ζωντανά references και τα αντιγράφει σε ένα καινούριο χώρο στο heap.
  - Mark-and-Sweep: πρώτα μαρκάρουμε τα ζωντανά references και μετά με ένα δεύτερο πέρασμα σβήνουμε τα νεκρά. Δεν αντιγράφονται απαραίτητα.
  - JVM χρησιμοποιεί ένα συνδυασμό των δύο.

Garbage Collection

- Πλεονεκτήματα:
  - Ο προγραμματισμός γίνεται πολύ πιο εύκολος.
  - Αποφεύγονται λάθη με λανθασμένους δείκτες και memory leaks.
- Μειονεκτήματα:
  - Το garbage collection κάνει το πρόγραμμα πιο αργό.
  - Λιγότερος έλεγχος για τον προγραμματιστή, που μπορεί να περιορίζει την αποτελεσματικότητα του κώδικα.

Μέθοδοι

- Overloading επιτρέπεται όπως και στην C++
- Πέρασμα παραμέτρων:
  - Εφόσον οι μεταβλητές σε αντικείμενα είναι πάντα αναφορές, το πέρασμα παραμέτρων είναι πάντα δια αναφοράς.

Παράδειγμα 2

- Θα κατασκευάσουμε σε java το παράδειγμα από την διάλεξη 2:
  - Δύο οχήματα κινούνται πάνω σε μία ευθεία.
  - Σε κάθε βήμα διαλέγουν τυχαία, αν θα πάνε αριστερά, δεξιά, ή θα μείνουν στην ίδια θέση.
  - Η προσομοίωση τελειώνει όταν συγκρουστούν.

```
import java.lang.* ;
import java.util.* ;

class Car
{
        private int pos;
        public Car()
        {
                pos = 0;
        }
        public Car(int x)
        {
                pos = x;
        }
        public void move()
        {
                Random r = new Random();
                pos += r.nextInt(2) - 1;
        }
        public int GetPos()
        {
                return pos;
        }
}
```

```
public class CarGame{

        private static Car x = new Car();
        private static Car y = new Car();

        public static void main(String[] args)
        {
                int iter = 1;
                x.move();
                y.move();
                while (x.GetPos() != y.GetPos()){
                        x.move();
                        y.move();
                        iter ++;
                }
                System.out.println("Colision after "
                        + iter + "moves at position "
                        + x.GetPos());
        }
}
```

Η δήλωση της main πρέπει να είναι ακριβώς έτσι

Αρχείο CarGame.java

```
import java.lang.* ;
import java.util.* ;

class Car
{
  private int pos;

  public Car(){
    pos = 0;
  }
  public Car(int x){
    pos = x;
  }
  public void move(){
    Random r = new Random();
    pos += r.nextInt(2) - 1;
  }
  public int GetPos(){
    return pos;
  }
}

public class CarGame{

      private static Car x = new Car();
      private static Car y = new Car();

      public static void main(String[] args){
           int iter = 1;
           x.move(); y.move();
           while (x.GetPos() != y.GetPos()){
             x.move(); y.move();
             iter ++;
           }
           System.out.println("Colision after " + iter + "moves at position " + x.GetPos());
      }
}
```

Κληρονομικότητα

- Η κληρονομικότητα ορίζεται με το keyword extends
  - `class Employee extends Person { … }`
- Ο πολυμορφισμός γίνεται πολύ πιο φυσικά γιατί το late binding είναι η default συμπεριφορά στην εκτέλεση των προγραμμάτων.

Παράδειγμα

```
import java.lang.*;
import java.io.*;

class Person{
    private  String fname;
    private String lname;

    public Person(String fn, String ln){
      fname = fn;
      lname = ln;
     }

   public String getPersonalDetails(){
      return fname + lname;
     }
}
```

Παράδειγμα

```
class Employee extends Person {
    private int basicSalary;

    public Employee( String fn, String ln, int sal){
      super(fn, ln);
      basicSalary = sal;
    }

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

Παράδειγμα

```
class Customer extends Person {
    private  int credit;
    private  String creditType;

    public Customer( String fn, String ln, String ct){
      super(fn, ln);
      creditType = ct;
      credit = 0;
    }
    public void chargeCredit(int amount){
      credit -= amount;
    }
    public int getCredit(){
      return credit;
    }
}
```

Παράδειγμα

```
public class ShopTest2{

    public static void main(String args[]){
     InputStreamReader stdin = new InputStreamReader(System.in);
     BufferedReader console = new BufferedReader(stdin);

     try{
         String line = console.readLine();
         String lineParts[] = line.split(" ");
         Employee emp = new Employee(lineParts[0], lineParts[1],
                                     Integer.parseInt(lineParts[2]));

         line = console.readLine();
         lineParts = line.split(" ");
         Customer cust = new Customer (lineParts[0], lineParts[1],
                                     lineParts[2]);


         ……..
     } catch (IOException ex){
     ……..
     }
     }
}
```

Παράδειγμα

```
public class ShopTest2{

    public static void main(String args[]){
     InputStreamReader stdin = new InputStreamReader(System.in);
     BufferedReader console = new BufferedReader(stdin);

     try{
         …….
         System.out.println("Employee details : " +
                   emp.getPersonalDetails() +
                   " Salary : " + emp.getSalary());

         cust.chargeCredit(250);
         System.out.println("Customer details : " +
                             cust.getPersonalDetails() +
                             " Credit : " + cust.getCredit());
     } catch (IOException ex){
         System.out.println("Error in reading input" + ex.getMessage());
     } catch (NumberFormatException ex) {
         System.out.println("Error in parsing numeric input" +
                                        ex.getMessage());
     }
     }
}
```

Παράδειγμα

```
import java.lang.*;
import java.io.*;

class Person{
    protected  String fname;
    protected String lname;

    public Person(String fn, String ln){
      fname = fn;
      lname = ln;
    }

    public String getPersonalDetails(){
      return "First Name : " + fname + " Last Name : " +
   lname;
    }
}
```

Παράδειγμα

```
class Employee extends Person {
    private int basicSalary;

    public Employee(String fn, String ln, int sal){
       super(fn, ln);
       basicSalary = sal;
    }

    public int getSalary(){
       return basicSalary;
    }

    public String getPersonalDetails(){
       return super.getPersonalDetails() +
          " Basic Salary : " + String.valueOf(basicSalary);
    }
}
```

Παράδειγμα

```
class Customer extends Person {
    private  int credit;
    private  String creditType;


    public Customer(String fn, String ln, String ct){
      super(fn, ln);
      creditType = ct;
      credit = 0;
    }

    public void chargeCredit(int amount){
      credit -= amount;
    }

    public int getCredit(){
      return credit;
    }

    public String getPersonalDetails(){
      return super.getPersonalDetails() +
             "Credit Card Type : " + creditType +
             " Credit Amount : " + String.valueOf(credit);
    }

}
```

Παράδειγμα

το ποια μέθοδος θα εκτελεστεί εξαρτάται από την κλάση του αντικειμένου στο οποίο αναφέρεται το x και όχι από την κλάση του x → μηχανισμός LATE BINDING

```
public class ShopTest4Polymorfism{
    private static void helperMethodHTMLFormattedText( Person x) {
      System.out.println("<HTML>" + x.getPersonalDetails() + "</HTML>");
    }

    public static void main(String args[]){
      InputStreamReader stdin = new InputStreamReader(System.in);
      BufferedReader console = new BufferedReader(stdin);

      try{
        String line = console.readLine();
        String lineParts[] = line.split(" ");
        Employee emp = new Employee(lineParts[0], lineParts[1],
                          Integer.parseInt(lineParts[2]));

        line = console.readLine();
        lineParts = line.split(" ");
        Customer cust = new Customer (lineParts[0], lineParts[1],
                                      lineParts[2]);
        ……..
      }
    }
}
```

Παράδειγμα

το ποια μέθοδος θα εκτελεστεί εξαρτάται από την κλάση του αντικειμένου στο οποίο αναφέρεται το x και όχι από την κλάση του x → μηχανισμός LATE BINDING

```
public class ShopTest4Polymorfism{

    private static void helperMethodHTMLFormattedText( Person x) {
       System.out.println("<HTML>" + x.getPersonalDetails() + "</HTML>");
    }

    public static void main(String args[]){
       InputStreamReader stdin = new InputStreamReader(System.in);
       BufferedReader console = new BufferedReader(stdin);

       try{
          ……..
          helperMethodHTMLFormattedText( emp );
          helperMethodHTMLFormattedText( cust );
       } catch (IOException ex){
           System.out.println("Error in reading input" + ex.getMessage());
       } catch (NumberFormatException ex) {
           System.out.println("Error in parsing numeric input" +
                                    ex.getMessage());
       }
    }
}
```