Εξαιρέσεις (Exceptions)

- Γεγονότα λάθους εκτέλεσης (Run Time Errors)
- Στέλνονται (κυρίως) από τήν JVM στο πρόγραμμα.
- Συνήθως είναι αντικείμενα των κλάσεων:

  java.lang.Throwable,

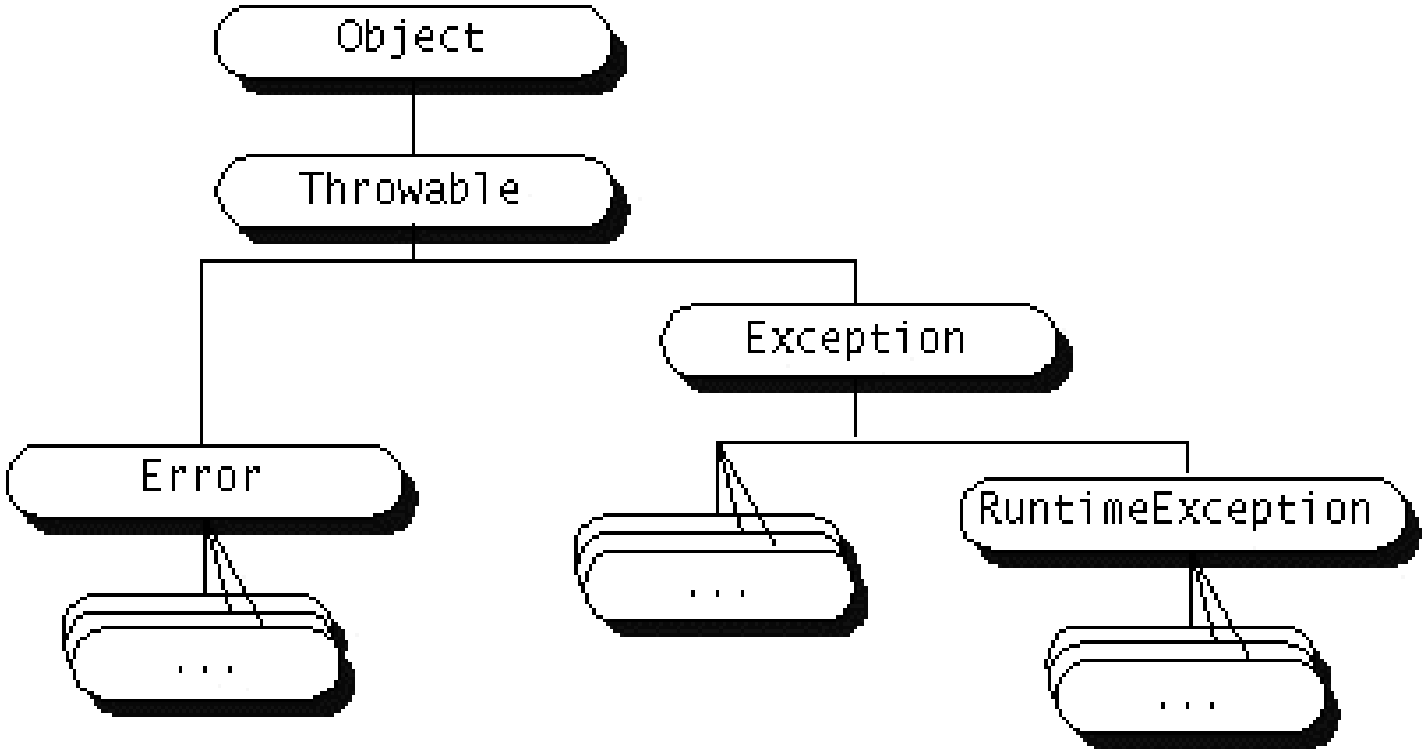
  java.lang.Error,

  java.lang.Exception,
  java.lang.RuntimeException.

Παραδείγματα Εξαιρέσεων

- Διαίρεση με 0 (ArithmeticException)
- Προσπέλαση στοιχείου πίνακα που δεν υπάρχει (ArrayIndexOutOf BoundsException)
- Προσπέλαση αρχείου που δεν υπάρχει (FileNotFoundException)

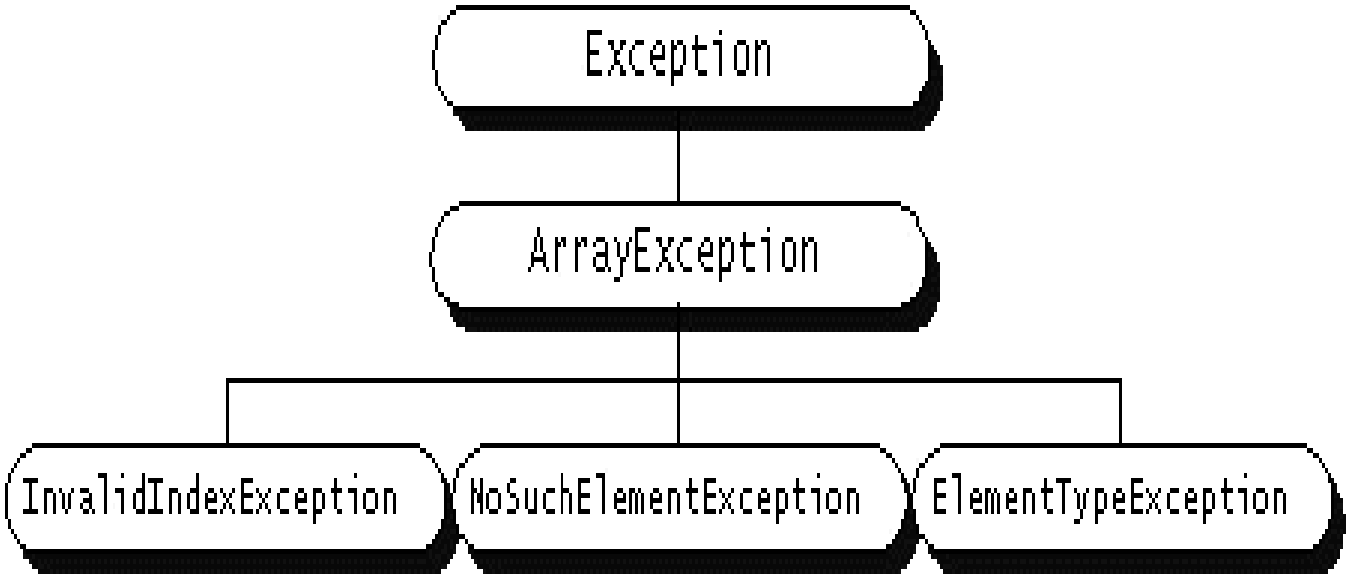
Throwable και Exception



Πλεονεκτήματα Εξαιρέσεων

- Διαχωρισμός λειτουργικού κώδικα από κώδικα χειρισμού λαθών (Καλύτερη οργάνωση κώδικα).
- Προώθηση σφαλμάτων για χειρισμό σε διαφορετικό επίπεδο της στοίβας κλήσεως μεθόδων (call stack).
- Οργάνωση λαθών σε ομάδες (Ιεράρχηση Εξαιρέσεων.

Παράδειγμα Ομαδοποίησης



Ελεγχόμενες (Checked) και μη Ελεγχόμενες Εξαιρέσεις

- Μια ομάδα εξαιρέσεων (Runtime Exceptions) δεν απαιτείται να χειριστούν από το σύστημα (π.χ. Αριθμητικά Λάθη, Δείκτες Πινάκων κλπ).
- Οι υπόλοιπες εξαιρέσεις (π.χ. IOExceptions, Εξαιρέσεις που δημιουργούνται από τον χρήστη) πρέπει να χειριστούν ή να δηλωθούν από το πρόγραμμα.

Παράδειγμα Μη Ελεγχόμενης Εξαίρεσης

```
public class Echo {
  public static void main (String[] arguments) {
    System.out.println (arguments[0]);
  }
}
```

```
>javac  Echo.java
>java  Echo  Hello
Hello

>java Echo
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException
        at Echo.main(Echo.java:7)
```

Παράδειγμα Ελεγχόμενης Εξαίρεσης

```
import java.io.*;
public class Show {
  public static void main (String[] arguments) {
    BufferedReader br = new BufferedReader(new FileReader(arguments[0]));
    while (br.ready()) {
        System.out.println (br.readLine());
    }
  }
}
```

```
>javac Show.java
Show.java:4: unreported exception java.io.FileNotFoundException; must be caught
 or declared to be thrown
     BufferedReader br = new BufferedReader(new FileReader(arguments[0]));
                                                          ^
Show.java:5: unreported exception java.io.IOException; must be caught or declar
ed to be thrown
     while (br.ready()) {
                    ^
Show.java:6: unreported exception java.io.IOException; must be caught or declar
ed to be thrown
       System.out.println (br.readLine());
```

Χειρισμός Εξαιρέσεων
με χρήση try {} catch {}

```
import java.io.*;

public class Show {

  public static void main (String[] arguments) {

    try {

      BufferedReader br = new BufferedReader(new FileReader(arguments[0]));

      while (br.ready()) {
        System.out.println (br.readLine());
      }

    } catch (IOException e) {

      System.out.println (e.getMessage());
    }
  }
}
```

Χειρισμός Εξαιρέσεων

```
try {
    // κώδικας που μπορεί να προκαλέσει εξαίρεση
} catch (<exception_1>  <var1>) {
    // κώδικας χειρισμού exception_1
} catch (<excpetion_2> <var2>) {
    // κώδικας χειρισμού exception_2
} finally {
    // κώδικας ανεξερέτου εξαίρεσης
}
```

Διάδοση εξαιρέσεων από Μεθόδους

- Μια μέθοδος μπορεί στην δήλωσή της να πληροφορεί ότι μπορεί να δημιουργήσει (throw) κάποια εξαίρεση. π.χ
- `public void writeList() throws IOException ArrayIndexOutOfBoundsException {`
- Οι εξαιρέσεις αυτές δεν είναι υποχρεωτικό να χειριστούν μέσα στην μέθοδο.

Χειρισμός Εξαίρεσης
με δήλωση throws

```
import java.io.*;

public class Show {

  public static void main (String[] arguments) throws IOException {

      BufferedReader br = new BufferedReader(new FileReader(arguments[0]));

      while (br.ready()) {
        System.out.println (br.readLine());
      }
  }
}
```

Χρήση της finally

```
public void writeList() {
    PrintWriter out = null;

    try {
        System.out.println("Entering try statement");
        out = new PrintWriter(
                  new FileWriter("OutFile.txt"));

        for (int i = 0; i < size; i++)
            out.println("Value at: " + i + " = " + victor.elementAt(i));
    } catch (ArrayIndexOutOfBoundsException e) {
        System.err.println("Caught ArrayIndexOutOfBoundsException: " +
             e.getMessage());
    } catch (IOException e) {
        System.err.println("Caught IOException: " + e.getMessage());
    } finally {
        if (out != null) {
            System.out.println("Closing PrintWriter");
            out.close();
        } else {
            System.out.println("PrintWriter not open");
        }
    }
}
```

Δημιουργία Εξαιρέσεων

- Η εντολή throw χρησιμοποιειται για την διάδοση μιας εξαίρεσης.

```
Exception myException = new Exception ("An error Occurred");

throw myException;
```

Νήματα Ελέγχου (Threads)

- Ανεξάρτητη σειριακή εκτέλεση εντολών. Ένα πρόγραμμα μπορεί να εκτελεί πολλά παράλληλα νήματα.
- Ένα τμήμα μιας εφαρμογής μπορεί να εκτελεί κάποια λειτουργία ενώ κάποια άλλη λειτουργία μπορεί να βρίσκεται σε εξέλιξη
- Περιβάλλον Εκτέλεσης (Execution Context)
  Διεργασία Ελαφρού Τύπου(Lightweight Process)

Java Threads

- java.lang.Thread είναι μια βασική κλάση που υποστηρίζει νήματα ελέγχου.
- Υπάρχουν δύο τρόποι για να οριστεί ένα νέο νήμα ελέγχου σε μια εφαρμογή:
- Δημιουγία μιας υπο-κλάσης που κληρονομεί την κλάση Thread με υπερφόρτωση της μεθόδου run()
- Δημιουργία μιας κλάσσης που υλοποιεί το interface Runnable

Πρώτο Παράδειγμα

```
public class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + getName());
    }
}
```

```
public class TwoThreadsDemo {
    public static void main (String[] args) {
        new SimpleThread("Jamaica").start();
        new SimpleThread("Fiji").start();
    }
}
```

```
0 Jamaica
0 Fiji
1 Fiji
1 Jamaica
2 Jamaica
2 Fiji
3 Fiji
3 Jamaica
4 Jamaica
4 Fiji
5 Jamaica
5 Fiji
6 Fiji
6 Jamaica
7 Jamaica
7 Fiji
8 Fiji
9 Fiji
8 Jamaica
DONE! Fiji
9 Jamaica
DONE! Jamaica
```

Thread Methods

- run() Ο κώδικας που θα εκτελεστεί στο νέο νήμα ελέγχου
- start() Οταν καλείτε για κάποιο αντικείμενο της κλάσσης Thread η java δημιουργει ένα νέο νήμα και εκτελεί την μέθοδο run(). Οταν η μέθοδος run() εκτελεστεί τότε το νήμα τερματίζει.

Δεύτερο Παράδειγμα

```
public class SimpleThread2 implements Runnable {

  public void run() {
    for (int i = 0; i < 10; i++) {
      System.out.println(i + " " + Thread.currentThread().getName());
            try {
              Thread.currentThread().sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + Thread.currentThread().getName());
    }
}
```

```
public class TwoThreadsDemo2 {
  public static void main (String[] args) {
    new Thread(new SimpleThread2(), "Jamaica" ).start();
    new Thread(new SimpleThread2(), "Fiji" ).start();
  }
}
```

Περισσότερες Μέθοδοι

- setPriority() Καθορίζει την προτεραιότητα του Νήματος. Νήματα με χαμηλότερη προτεραιότητα εκτελούνται όταν δεν τρέχουν νήματα με υψηλότερη προτεραιότητα
- yield() Προκαλεί τον τρέχον νήμα να σταματήσει προσωρινά για να εκτελεστεί κάποιο άλλο νήμα. (static)
- sleep(int milisec) Σταματά την εκτέλεση του νήματος για κάποιο χρονικό διάστημα. (static)
- join() Χρησιμοποιείται από τον δημιουργό ενός νήματος για να περιμένει τον τερματισμό εκτέλεσης του νήματος.

Συγχρονισμός

- Οταν πολλά νήματα χρησιμοποιούν τα ίδια δεδομένα μπορεί να δημιουργηθούν προβλήματα.
- Μια λύση είναι ορισμένες μέθοδοι αντικειμένων να δηλώνονται σαν synchronised.
- Αν ένα νήμα καλέσει καποια synchronised μέθοδο ενός αντικειμένου τότε όλες οι κλήσεις άλλων synchronized μεθόδων θα περιμένουν.