ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Οντοκεντρικός Προγραμματισμός

Ενότητα 3: JAVA: ΕΞΑΙΡΕΣΕΙΣ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ, ΝΗΜΑΤΑ

Εξαιρέσεις

ΔΙΔΑΣΚΟΝΤΕΣ: Ιωάννης Χατζηλυγερούδης, Χρήστος Μακρής

Πολυτεχνική Σχολή

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

ΕΞΑΙΡΕΣΕΙΣ

ΕΞΑΙΡΕΣΕΙΣ

Εξαίρεση (Exception): Ένα συμβάν κατά τον χρόνο εκτέλεσης (π.χ. αδυναμία ανοίγματος αρχείου ή ανάγνωσης πληροφορίας, μη ύπαρξη αντικειμένου στη στοίβα) που απαιτεί ειδικό χειρισμό. Ονομάζεται και σφάλμα (error).

Χειρισμός εξαιρέσεων (Exception handling): Αναγνώριση και αντιμετώπιση του συμβάντος (σφάλματος).

Εξαιρέσεις προκαλούνται → `από το σύστημα`

Εξαιρέσεις προκαλούνται → `από το πρόγραμμα`

ΓΙΑΤΙ ΕΙΔΙΚΟΣ ΧΕΙΡΙΣΜΟΣ ΕΞΑΙΡΕΣΕΩΝ;

Η κλασσική διαχείριση λαθών περιπλέκει τον κώδικα, ώστε να χάνεται η «διαύγειά» του, διότι δεν υπάρχει διάκριση μεταξύ του κώδικα που αφορά στην κυρίως (ή «κανονική») λειτουργία και αυτού που αφορά στον χειρισμό των λαθών-εξαιρέσεων.

Ο ξεχωριστός μηχανισμός χειρισμού εξαιρέσεων επιτρέπει τη συγγραφή καθαρού, εύρωστου και ανεκτικού σε λάθη κώδικα. Διατηρεί μια ισορροπία μεταξύ αξιοπιστίας και διαύγειας.

Γλώσσες που διαθέτουν τέτοιο μηχανισμό: PL/1, Ada, Java

ΜΗΧΑΝΙΣΜΟΣ ΕΞΑΙΡΕΣΕΩΝ ΣΤΗ JAVA

Στόχοι

- Έγερση (ή πρόκληση) εξαιρέσεων
- Σύλληψη και διευθέτηση εξαιρέσεων

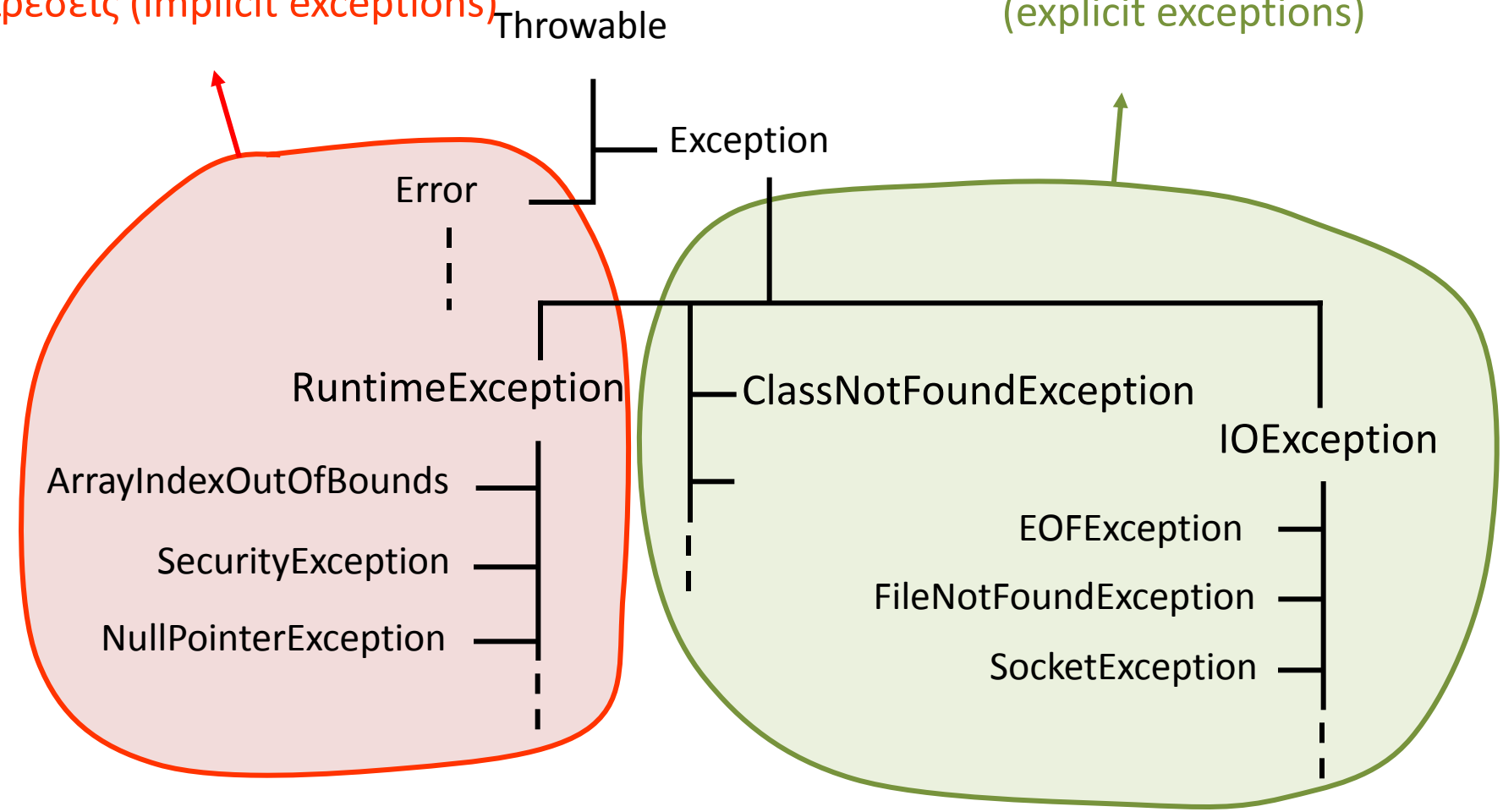
Χαρακτηριστικά

- Οι εξαιρέσεις στη Java είναι αντικείμενα (στιγμιότυπα)
- Υπάρχει ιδιαίτερη ιεραρχία κλάσεων εξαιρέσεων

ΙΕΡΑΡΧΙΑ ΕΞΑΙΡΕΣΕΩΝ

Αφανείς ή εσωτερικές εξαιρέσεις (implicit exceptions)

Εμφανείς (ή σαφείς) εξαιρέσεις (explicit exceptions)



Throwable, Exception, RuntimeException → **Java.lang**
IOException → **Java.io**

ΔΗΛΩΣΗ ΕΞΑΙΡΕΣΕΩΝ (1)

Σύνταξη

```
<προσδ.> <τύπος> <όνομα-μεθ.> (<παραμ.>) throws
<όν. εξαιρ.-1> [, <όν. εξαίρ.-2>, ...]
```

(εμφανείς εξαιρέσεις)

π.χ.

```
public ComplexNumber divide (double d) throws IOException
```

Δηλώνουμε δηλ. ότι η μέθοδος divide έχει κάποιο κώδικα στο σώμα της, που μπορεί να προκαλέσει εξαίρεση του τύπου IOException, αν κάτι πάει στραβά, π.χ. αν ζητηθεί από τη μέθοδο να υπολογίσει κάτι που δεν επιτρέπεται (ή δεν προβλέπεται) από τον τρόπο υλοποίησής της, κάτι δηλ. που προκαλεί σφάλμα/λάθος.

ΔΗΛΩΣΗ ΕΞΑΙΡΕΣΕΩΝ (2)

Αυτό σημαίνει ότι:

(α) η μέθοδος προκαλεί η ίδια, άμεσα, εξαίρεση (χρησιμοποιώντας την εντολή **throw** στο σώμα της).

(άμεση πρόκληση/έγερση)

ή

(β) χρησιμοποιεί στο σώμα της μέθοδο που προκαλεί εξαίρεση, την οποία όμως δεν χειρίζεται η ίδια, αλλά περνά τον χειρισμό της στην καλούσα μέθοδο.

(έμμεση πρόκληση/έγερση)

(Οι κλάσεις εξαιρέσεων συνήθως έχουν δύο δημιουργούς, έναν χωρίς ορίσματα και έναν με ένα όρισμα τύπου String, που αντιπροσωπεύει το μήνυμα στον χρήστη).

ΑΜΕΣΗ ΠΡΟΚΛΗΣΗ/ΕΓΕΡΣΗ ΕΞΑΙΡΕΣΗΣ

Δημιουργία νέου στιγμιοτύπου εξαίρεσης
(χρήση εντολής `throw`)

Π.χ.

```
Exception dwz = new Exception ("Διαίρεση με το μηδέν");
throw dwz;
```

ή

```
throw new Exception ("Διαίρεση με το μηδέν");
```

στο σώμα της μεθόδου

Π.χ.

```
public ComplexNumber divide (double d) throws Exception{
   if (d == 0)
     throw new Exception ("Διαίρεση με το μηδέν");
   return new ComplexNumber (mdReal/d, mdImag/d);
}
```

ΕΜΜΕΣΗ ΠΡΟΚΛΗΣΗ/ΕΓΕΡΣΗ– ΣΥΛΛΗΨΗ ΕΞΑΙΡΕΣΗΣ

Χρήση στο σώμα μιας μεθόδου Α άλλης μεθόδου Β, η οποία προκαλεί/εγείρει εξαίρεση

Π.χ.

```
public void readFile (String filename) throws IOException{
  while (numBytes <= myBuffer.length){
         myInputStream.read (myBuffer);
         myBytes++;
  }
}
```

Προκαλεί εξαίρεση τύπου

Δηλ. η readFile προκαλεί **<u>έμμεσα</u>** εξαίρεση.

Ο χειρισμός της εξαίρεσης περνά στη μέθοδο που καλεί την readFile (έμμεσος χειρισμός/σύλληψη)

ΔΙΑΧΕΙΡΙΣΗ ΕΞΑΙΡΕΣΕΩΝ

Πολλές μέθοδοι (της βιβλιοθήκης) της Java προκαλούν εξαιρέσεις (δηλ. περιέχουν την εντολή *throw* στο σώμα τους).

Αυτές τις εξαιρέσεις, οποιαδήποτε μέθοδος τις χρησιμοποιεί επιβάλλεται (από τον μεταγλωττιστή) να τις χειριστεί (δηλ. να προστατεύσει τον κώδικά της από αυτές τις εξαιρέσεις).

Π.χ. η " `System.in.read( );` " , αν δεν προστατευτεί, θα προκαλέσει το παρακάτω λάθος μεταγλώττισης:

```
"Test.java:23: unreported exception java.io.IOException
  must be caught or declared to be thrown
  System.in.read( );"
                ^
```

ΑΜΕΣΗ ΣΥΛΛΗΨΗ (ΔΙΑΧΕΙΡΙΣΗ) ΕΞΑΙΡΕΣΕΩΝ

Χρήση μπλοκ try-catch

Σύνταξη:

```
try {<κώδικας που μπορεί να εγείρει εξαίρεση>}
                    .....
catch (<κλάση εξαίρ.> <μεταβλ. εξαίρ.>)
      {<κώδικας χειρισμού>}
```

Ερμηνεία: Δοκίμασε (try) αυτό το κομμάτι κώδικα, που μπορεί να προκαλέσει εξαίρεση. Αν εκτελεστεί σωστά, προχώρησε με το υπόλοιπο πρόγραμμα. Αν όχι, τότε κάνε σύλληψη (catch) της εξαίρεσης και χειρίσου την.

Μπορεί να υπάρχουν περισσότερα του ενός catch μπλοκ για το ίδιο try μπλοκ.

ΔΙΑΔΙΚΑΣΙΑ ΑΜΕΣΗΣ ΣΥΛΛΗΨΗΣ-ΔΙΑΧΕΙΡΙΣΗΣ

1. Όταν εγερθεί εξαίρεση, από τον κώδικα ενός try μπλοκ, ο έλεγχος μεταφέρεται έξω από το μπλοκ και γίνεται αναζήτηση του κατάλληλου catch μπλοκ
2. Αν βρεθεί, ο έλεγχος μεταφέρεται στο catch μπλοκ και γίνεται ο χειρισμός της εξαίρεσης
3. Αν δεν εγερθεί εξαίρεση, τα catch μπλοκ παραλείπονται
4. Μετά τον χειρισμό, ο έλεγχος δεν επιστρέφει στο σημείο που ηγέρθη η εξαίρεση, αλλά συνεχίζει με τον κώδικα που υπάρχει μετά τα try και catch μπλοκ

ΠΑΡΑΔΕΙΓΜΑ (Χωρίς σύλληψη)

```
import java.io.*;
public class ExceptionTest {
 public static void main (String args[ ]) {
    int num;
    num= getNumber();
    System.out.println ("Το διπλάσιο: " + 2*num);
 }
 public static int getNumber () {
       String line;
       BufferedReader br= new BufferedReader (
                        new InputStreamReader (System.in));
       System.out.println ("Δώσε αριθμό: ");
       line = br.readLine();
       return Integer.parseInt(line);
 }
}
```

```
Exception java.io.IOException must be caught or it
must be declared in the throws clause of this method.
line = br.readLine ();
           ^
```

ΠΑΡΑΔΕΙΓΜΑ (Άμεση σύλληψη)

```
import java.io.*;
public class ExceptionTest {
  public static void main (String args[ ]) {
       int num;
       num= getNumber();
       System.out.println ("Το διπλάσιο: " + 2*num);
  }
  public static int getNumber () {
       String line;
       BufferedReader br= new BufferedReader(
                          new InputStreamReader (System.in));
       System.out.println ("Δώσε αριθμό: ");
       try {
            line = br.readLine();
            return Integer.parseInt(line);
       }
       catch (IOException e) {
           System.out.println (e);
       }
       catch (NumberFormatException e) {
           System.out.println (e);
       }
   }
}
```

ΠΑΡΑΔΕΙΓΜΑ (Έμμεση σύλληψη ή μετάδοση σύλληψης)

```
public class ExceptionTest {
 public static void main (String args[ ]) {
  try{int num;
       num= getNumber();
       System.out.println ("Το διπλάσιο: " + 2*num); }
  catch (IOException e) {System.out.println (e);}
  catch (NumberFormatException e) {System.out.println (e);}
 }

 public static int getNumber () throws Exception {
    String line;
    BufferedReader br= new BufferedReader
                        (new InputStreamReader (System.in));
    System.out.println ("Δώσε αριθμό: ");
    line = br.readLine();
    return Integer.parseInt(line);
  }
}
```

Σύλληψη εξαίρεσης

Καλούσα μέθοδος

Μέθοδος που περιέχει μέθοδο που προκαλεί εξαίρεση

FINALLY ΜΠΛΟΚ

Για την εκτέλεση ενεργειών που πρέπει να εκτελεστούν οπωσδήποτε, ανεξάρτητα από τον αν προκληθεί ή όχι εξαίρεση (π.χ. απόδοση πόρων του συστήματος), χρησιμοποιείται το **finally** μπλοκ.

Π.χ.

```
try {readFile ();}
        …..
catch (IOException e) { … }
finally {close Textfile4;}
```

ΔΗΜΙΟΥΡΓΙΑ ΝΕΩΝ ΤΥΠΩΝ ΕΞΑΙΡΕΣΕΩΝ (ΑΠΟ ΤΟΝ ΧΡΗΣΤΗ)

- Πρέπει να είναι υποκλάσεις της Exception ή κάποιας υποκλάσης της
- Πρέπει να έχει δύο δημιουργούς (ένα χωρίς όρισμα, ένα με όρισμα String)

Π.χ.

```
public class xxException extends Exception {
    public xxException () { }
    public xxException (String msg) {
       super(msg);
    }
}
```

ΠΑΡΑΔΕΙΓΜΑ

```
public class DivideByZeroException
                              extends ArithmeticException {
    public DivideByZeroException (){
        super ("Attempted to divide by zero");
    }
}
```

Μέθοδος modulo

```
double modulo (double num, double den)
                        throws DivideByZeroException {
    if (den == 0)
         throw new DivideByZeroException ();
    return (double) (num/den);
}
```

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων