Προγραμματισμός II (Java)

4. Διαχείριση εξαιρέσεων

Διαχείριση λαθών – Εξαιρέσεις

- Δημιουργία
- Ανίχνευση
- Διαχείριση
- Βασικοί τύποι εξαιρέσεων
- Δημιουργία τύπων εξαίρεσης

Αντιμετώπιση λαθών

- Υπάρχουν λάθη χρόνου εκτέλεσης (run time errors) τα οποία δεν ανιχνεύονται στη μεταγλώττιση.
  - Αναζήτηση στοιχείου έξω από τα όρια ενός πίνακα
  - Άνοιγμα αρχείου που δεν υπάρχει
  - Κλήση αναφοράς σε null
- Θα πρέπει να αντιμετωπίζονται όταν το πρόγραμμα εκτελείται
- Ο κώδικας που δημιουργεί κάποιο λάθος μεταδίδει πληροφορία στον κώδικα που καλείται να το αντιμετωπίσει. Αυτό γίνεται με τις ***Εξαιρέσεις – Exceptions***
- Σηματοδοτούν εμφάνιση συνθηκών ιδιαίτερης μεταχείρισης και διακόπτουν τη συνήθη ροή του κώδικα

Παραδείγματα

- Αναζήτηση στοιχείου έξω από τα όρια ενός πίνακα

```
ArrayList<String> messages = new ArrayList<String>();
messages.add(“Hello”);
messages.add(“How are you?”);
System.out.println(messages.get(0));
System.out.println(messages.get(2)); //ArrayIndexOutOfBoundsException
```

- Αναφορά σε null αντικείμενο

```
Human[] anthrwpoi = new Human[5];
anthrwpoi[0] = new Human(“Bill”,”Gates”,50);
anthrwpoi[2].setName(“George”); //NullPointerException
```

- Άνοιγμα αρχείου που δεν υπάρχει

```
FileReader fro = new FileReader( "myFile.txt" ); //FileNotFoundException
```

Οι εξαιρέσεις είναι αντικείμενα

- Αντί να ελέγχουμε στον κώδικα για κάποιο συγκεκριμένο λάθος σε όσα σημεία μπορεί να εμφανιστεί, μπορούμε να προσθέσουμε ***ένα μόνο*** μπλόκ κώδικα που αποκαλείται διαχειριστής εξαίρεσης (Exception Handler)
- Αν συμβεί εξαίρεση, η ροή του προγράμματος στο μπλοκ διακόπτεται και ο έλεγχος περνά στο διαχειριστή εξαίρεσης
- Αυτός ανιχνεύει τον τύπο του αντικειμένου εξαίρεσης
- Αν κάποια εξαίρεση δεν τη διαχειριστούμε παίρνουμε μήνυμα από το μεταγλωττιστή

Παράδειγμα

- Θέλουμε να δημιουργήσουμε μια μέθοδο που να αθροίζει όσους αριθμούς δώσει ο χρήστης. Να συνεχίζει μέχρι ο χρήστης να δώσει το σύμβολο '='.

```
public static void main(String[] args) {
        Main m=new Main();
        double z=m.sumNumbers();
        System.out.println("Sum is:"+z);
    }
public double sumNumbers(){
    double sum=0;
    while(????)
                sum+=readDouble();
    return sum;
}
public double readDouble(){
    Scanner in =new Scanner(System.in);
    return in.nextDouble();
}
```

- Τι θα βάλω στη συνθήκη του while;
- Τι θα γίνει αν ο χρήστης δώσει '=';
- Τι θα γίνει αν δώσει κάτι που δεν είναι αριθμός;

Αναλυτικά με τις εξαιρέσεις

- Όταν εμφανίζεται μία εξαίρεση σταματά η εκτέλεση της μεθόδου
- Δεν υπάρχει η απαραίτητη πληροφορία στο scope της μεθόδου
- Ανεβαίνουμε και αντιμετωπίζουμε το πρόβλημα σε κάποιο «υψηλότερο» scope. «***Ρίχνουμε***» μια εξαίρεση (throw)
  - Ένα αντικείμενο εξαίρεσης (Exception object) δημιουργείται στον σωρο (heap) με χρήση του τελεστή new όπως κάθε άλλο Java object
  - Διακόπτεται η εκτέλεση της μεθόδου και κρατιέται μια αναφορά στο Exception object σε κάποια περιοχή της μνήμης
  - Αναλαμβάνει ο μηχανισμός διαχείρισης της εξαίρεσης (Exception handling mechanism) που πρέπει να βρει το κατάλληλο μέρος για να συνεχίσει να εκτελεί κώδικα

Αντικείμενα εξαιρέσεων

- Το keyword throw προκαλεί τα ακόλουθα γεγονότα:
  - εκτελεί την έκφραση new και δημιουργεί ένα αντικείμενο που δεν θα υπήρχε κανονικά
  - Το αντικείμενο αυτό επιστρέφεται στην ουσία από την μέθοδο ή το block μέσα στο οποίο δημιουργήθηκε μολονότι η μέθοδος δεν έχει δηλωθεί να επιστρέφει παραμέτρους αυτού του τύπου και τα blocks δεν έχουν φυσικά παραμέτρους που να επιστρέφουν
  - Το πρόγραμμα βγαίνει από τη μέθοδο ή το block.
- Τα αντικείμενα exception που «ρίχνονται» μπορεί να είναι ***διάφορων*** τύπων, ανάλογα το είδος της εξαίρεσης
- Κάθε αντικείμενο κωδικοποιεί στα πεδία του όλη την πληροφορία που αφορά την εξαίρεση, ώστε ο μηχανισμός διαχείρισης εξαιρέσεων στα ψηλότερα επίπεδα να μπορεί να πάρει τις κατάλληλες αποφάσεις

Προδιαγραφές εξαιρέσεων

- Αν κάποιος πρόκειται να χρησιμοποιήσει τον κώδικά μας τότε θα πρέπει να ξέρει τις εξαιρέσεις που μπορεί να παράγονται
- Σε κάθε μέθοδο που ρίχνει εξαιρέσεις χρησιμοποιούμε τη δήλωση ***throws***

```
void f() throws tooBig, tooSmall, divZero {
   // κώδικας της μεθόδου
}
```

- Αν κάποιος καλέσει στον κώδικά του τη μέθοδό μας θα πρέπει να τη βάλει μέσα σε ***try*** block.
- Σε αντίθετη περίπτωση ο μεταγλωττιστής εκδίδει μηνύματα λάθους. Αυτό εξασφαλίζει διαχείριση εξαιρέσεων κατά τη μεταγλώττιση.
- Τέτοιες προδιαγραφές υπάρχουν ήδη σε πολλές μεθόδους κλάσεων της Java

Δημιουργία νέων τύπων εξαιρέσεων

- Κάθε τύπος εξαίρεσης που δημιουργούμε πρέπει να κληρονομεί από κάποιο υπάρχουσα τάξη.
- Συνήθως αυτή που είναι νοηματικά κοντά. Αλλιώς από την Exception

```
class MyException extends Exception {
    public myException() {
    // κενός κατασκευαστής
    }
    public myException(String ms) {
    // κατασκευαστής με όρισμα
    }
    // specific fields or methods may exist
}
```

Παράδειγμα – έλεγχος εισόδου

- Στη μέθοδο readDouble() ελέγχουμε τι δίνει ο χρήστης

```
public double readDouble() throws Exception{
      Scanner in =new Scanner(System.in);
       if(in.hasNextDouble()){
           return in.nextDouble();
       }
       else throw new Exception();
}
```

- Εναλλακτικά μπορούμε να περάσουμε ένα μήνυμα στην εξαίρεση

```
public double readDouble() throws Exception{
      Scanner in =new Scanner(System.in);
       if(in.hasNextDouble()){
           return in.nextDouble();
       }
       else throw new Exception("Not a number");
}
```

Παράδειγμα – έλεγχος εισόδου (2)

- Μπορούμε να χρησιμοποιήσουμε δική μας εξαίρεση;
- Στο αρχείο MySpecialException.java

```
public class MySpecialException extends Exception{

}
```

- Οπότε

```
public double readDouble() throws MySpecialException{
      Scanner in =new Scanner(System.in);
       if(in.hasNextDouble()){
           return in.nextDouble();
       }
       else throw new MySpecialException();
}
```

Σύλληψη εξαίρεσης

- Όταν ρίχνουμε μία εξαίρεση πρέπει στα υψηλότερα επίπεδα κάποιο μπλοκ κώδικα να την ανιχνεύσει και να την διαχειριστεί.
- ***Προστατευόμενη περιοχή*** είναι ένα κομμάτι κώδικα που μπορεί να ρίξει εξαιρέσεις και που ακολουθείται από κώδικα που διαχειρίζεται αυτές τις εξαιρέσεις.
- Μια προστατευόμενη περιοχή μπορεί να ανιχνεύσει ένα ή περισσότερους τύπους εξαιρέσεων και
  - να τους διαχειριστεί ή
  - να τους ρίξει ακόμη πιο πάνω (με νέο throw)

Παράδειγμα

- Μη διαχείριση σφάλματος εισόδου

```
public double sumNumbers() throws MySpecialException{
      double sum=0;
      sum+=readDouble();
      return sum;
}
 public static void main(String[] args) throws MySpecialException{
        Main m=new Main();
        double z=m.sumNumbers();
        System.out.println("Sum is:"+z);
    }
```

- Έτσι αποφεύγουμε να χειριστούμε την εξαίρεση

Προστατευόμενη περιοχή - ***try***

- Ένα τέτοιο μπλοκ ονομάζεται try block αφού εκεί δοκιμάζουμε να κάνουμε κλήσεις σε διάφορες «επικίνδυνες μεθόδους»
- Το try block είναι ένα κανονικό scoper που περικλείεται από την κωδική λέξη try

```
try {
   // κώδικας που παράγει εξαιρέσεις
}
```

- Προστασία από εξαιρέσεις: Βάζουμε όλον τον «επικίνδυνο» κώδικα σε ένα try block και πιάνουμε όλες τις εξαιρέσεις στο ίδιο μέρος.

Διαχειριστές εξαιρέσεων - ***catch***

- Κάθε εξαίρεση που ανιχνεύεται μέσα στο try καταλήγει στον αντίστοιχο κώδικα (***catch*** block) που θα την εξυπηρετήσει. Τα catch ελέγχονται διαδοχικά.
- Παράδειγμα:

```
try {
   // κώδικας που παράγει εξαιρέσεις
} catch (type1 id1) {
   // χειρισμός εξαιρέσεων τύπου 1
} catch (type2 id2) {
   // χειρισμός εξαιρέσεων τύπου 2
} catch (type3 id3) {
   // χειρισμός εξαιρέσεων τύπου 3
}
```

- Αν πολλές μέθοδοι στο try ρίχνουν τον ίδιο τύπο εξαίρεσης, τότε χρειαζόμαστε μόνο έναν exception handler για αυτόν τον τύπο.

Η τάξη Exception

- Ένας διαχειριστής εξαιρέσεων με όρισμα τύπου Exception συλλαμβάνει όλες τις εξαιρέσεις.

```
catch(Exception e) {
        System.out.println("caught an exception");
}
```

- Γι' αυτό μπαίνει στο τελευταίο catch στη σειρά για να πιάσει κάθε άλλη εξαίρεση.
- Η Exception και η Throwable προσφέρουν τις ακόλουθες μεθόδους:
  - String getMessage()
  - String toString() //τυπώνει το όνομα της εξαίρεσης
  - void printStackTrace() //τυπώνουν το σωρό
  - void printStackTrace(PrintStream) //μεθόδων που κλήθηκαν

Παράδειγμα

- Διαχείριση σφάλματος εισόδου

```
public double sumNumbers(){
      double sum=0;
       try{
                sum+=readDouble();
       }
       catch(MySpecialException ex){              //θα συμβεί αν ο χρήστης δεν δώσει αριθμό
            System.err.println(ex.toString());    //τυπώνει το όνομα της κλάσης εξαίρεσης
       }
       catch(Exception ex){                       //θα συμβεί σε οποιαδήποτε άλλη περίπτωση λάθους
           System.err.println(ex.getMessage());   //τυπώνει το μήνυμα της εξαίρεσης
       }
      return sum;
}
```

- Η MySpecialException είναι ειδικότερη από την Exception

Το block ***finally***

- Όταν υπάρχει κάποιο κομμάτι κώδικα που θα πρέπει να εκτελεστεί είτε συμβαίνει ένα Exception σε ένα try μπλοκ είτε όχι βάζουμε τον κώδικα σε ένα finally μπλοκ ώς εξής:

```
static Switch sw = new Switch();
try {
   sw.on();
   …//επικίνδυνος κώδικας
} catch (A a) {
   ...
} catch (B b) {
   ...
} finally {
   sw.off(); //κώδικας που τρέχει πάντοτε και κάνει reset
}
```

Τερματισμός ή ανάνηψη (συνέχιση)

- Τερματισμός: Θεωρούμε ότι τα λάθη είναι κρίσιμα και δε συνεχίζουμε την εκτέλεση του κώδικα από το σημείο που ρίχθηκε η εξαίρεση
- Ανάνηψη: Προσπαθούμε να συνεχίσουμε την εκτέλεση του προγράμματος από εκεί που διακόπηκε

```
while (true) {
   try {
            //κώδικας που παράγει εξαιρέσεις
   }
   catch (Type1 id1) { //διορθώνουμε την εξαίρεση
   continue;}
   ...
   catch (TypeN idN) { // διορθώνουμε την εξαίρεση
   continue;}
   // κώδικας που εκτελείται αν δεν υπάρχει εξαίρεση
   break;
}
```

Παράδειγμα

- Ας διαχωρίσουμε τις εξαιρέσεις

```
public double readDouble() throws MySpecialException, Exception {
        Scanner in = new Scanner(System.in);
        if (in.hasNextDouble()) {
            return in.nextDouble();
        } else if (in.hasNext() && in.next().equals("=")){
            throw new MySpecialException();    // αν ο χρήστης δώσει ‘=‘
        }
        else{
            throw new Exception("Not a number");     // αν ο χρήστης δώσει
    οτιδήποτε άλλο
        }
    }
```

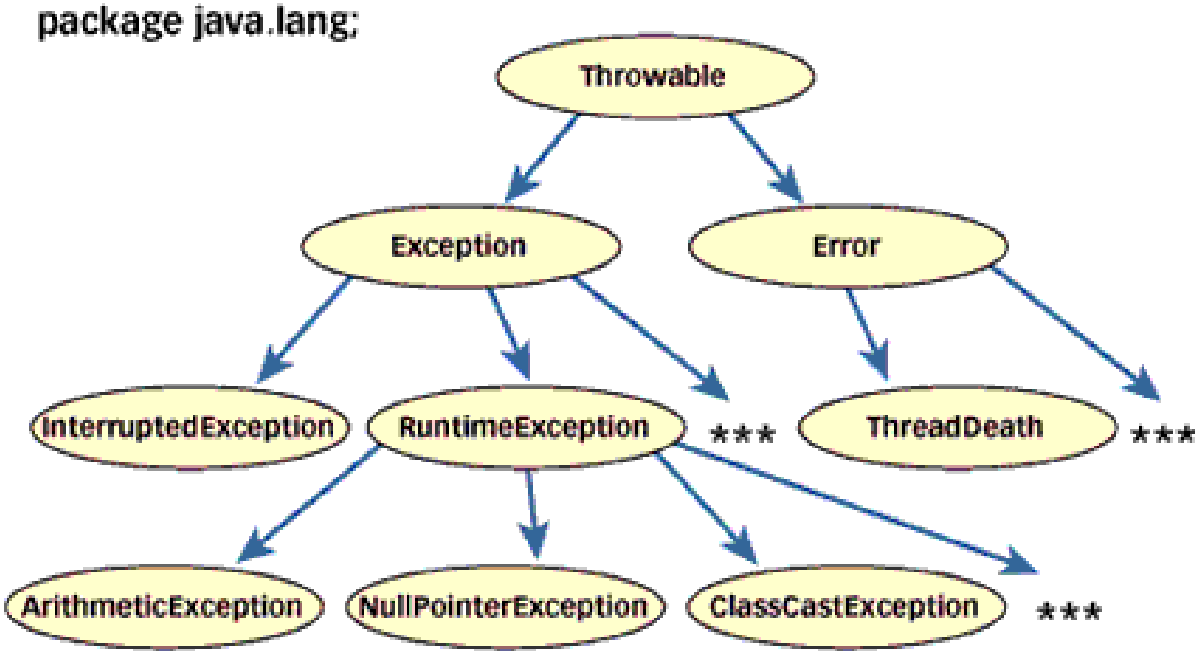
Παράδειγμα (2)

- Τερματισμός επανάληψης

```
public double sumNumbers() {
    double sum = 0;
    while (true) {
        try {
            sum += readDouble();
        } catch (MySpecialException ex) {
            break;
        } catch (Exception ex) {
            System.err.println(ex.getMessage());
        }
    }
    return sum;
}
```

Ιεραρχία εξαιρέσεων

- Η Java περιέχει την τάξη ***Throwable*** που περιγράφει οτιδήποτε μπορεί να ριχθεί σαν Exception
- Άλλες εξαιρέσεις ορίζονται στα πακέτα util, net και io.



Παράδειγμα

```
public class ExceptionMethods {
    public static void main(String args[]) {
        try {
            throw new Exception("Here's my Exception");
        } catch (Exception e) {
            System.out.println("Caught Exception");
            System.out.println( e.getMessage());
            System.out.println( e.toString());
            System.out.println( e.printStackTrace());
        }
    }
}
```

Η δύο τελευταίες κλήσεις τυπώνουν

java.lang.Exception: Here's my Exception

java.lang.Exception: Here's my Exception
at ExceptionMethods.main

Κληρονομικότητα και εξαιρέσεις

- Όταν σε μια απόγονη τάξη κάνουμε override μια μέθοδο της γονικής τάξης, τότε η μέθοδος δεν μπορεί να ρίχνει επιπλέον εξαιρέσεις.

```
class Test {
  String method (String s){
     if (s==null)
        throw new NullPointerException();
     else
        return s;}}

class SubTest extends Test{
String method (String s) throws Exception{
     if (s==null)
        throw new Exception();
     else
        return s;}}
```

overridden method does not throw java.lang.Exception

Παράδειγμα 1

- Ανάγνωση από πληκτρολόγιο
  - Δημιουργούμε ένα αντικείμενο με χρήση του ρεύματος εισόδου in (τάξη InputStream).

    BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
  - Και καλούμε τη μέθοδο ανάγνωσης γραμμής

```
try{
        line=stdin.readLine();
}
catch (IOException e){
        line=“”; //σε περίπτωση αποτυχίας γίνεται κενό
}
```

Παράδειγμα 2

- Μετατροπή String σε int

```
try {
   x= Integer.parseInt(line);
}
catch(NumberFormatException e) {
   x=0; //σε περίπτωση αποτυχίας γίνεται 0
}
```