Υ325 - Αντικειμενοστρεφής Τεχνολογία

2021

Τμ. ΗΜ&ΤΥ - Πανεπιστήμιο Πατρών

Υ325 Αντικειμενοστρεφής Τεχνολογία Φροντ. 2021-22

3ο Φροντιστήριο - 4/11/2021

Άσκηση από το 2ο φροντιστήριο

- Να γραφτεί μια κλάση της οποίας να μπορούμε να κατασκευάσουμε μόνο ένα στιγμιότυπο.

Λύση

- Να γραφτεί μια κλάση της οποίας να μπορούμε να κατασκευάσουμε μόνο ένα στιγμιότυπο.

```
class MySpecialClass {
    private static MySpecialClass instance;

    private MySpecialClass() {
    }

    public static MySpecialClass getInstance() {
      if (instance==null) {
        instance = new MySpecialClass();
      }
      return instance;
    }
}
```

Τύποι δεδομένων

Στη Java έχουμε δύο είδη τύπων:

1. Primitive data types (φαίνονται στον πίνακα παρακάτω)
2. Non-primitive data types (όλες οι κλάσεις, οι πίνακες (arrays) και τα interface - θα τα δούμε στο μέλλον)

Primitive τύποι

Primitive τύπος	προκαθορισμένη τιμή	μέγεθος
`boolean`	`false`	1 bit
`char`	`'\u0000'`	2 byte
`byte`	`0`	1 byte
`short`	`0`	2 byte
`int`	`0`	4 byte
`long`	`0L`	8 byte
`float`	`0.0f`	4 byte
`double`	`0.0d`	8 byte

Non-primitive τύποι

Οι κλάσεις (class), οι πίνακες (array) και τα interface, είναι non-primitive τύποι.

Για παράδειγμα, ο τύπος `String` που χρησιμοποιούμε για τα αλφαριθμητικά είναι μια *κλάση* (class) αντικειμένων. Στη Java μπορούμε να μιλήσουμε για την "*κλάση* `String`" και για "*τον τύπο* `String`" και θα εννοούμε ακριβώς το ίδιο, γιατί όλες οι κλάσεις είναι και τύποι.

Χρήσιμες συναρτήσεις

```
//class Integer:
//Διαβάζει το string s σαν δεκαδικό ακέραιο.
public static int parseInt(String s)

//class Character:
//Συγκρίνει την αριθμητική αξία αυτού του χαρακτήρα με αυτή του
    anotherCharacter.
public int compareTo(Character anotherCharacter)
//Αποφαίνεται αν ο χαρακτήρας ch είναι ψηφίο.
public static boolean isDigit(char ch)
//Επιστρέφει ένα στιγμιότυπο της Character που αντιπροσωπεύει τον
    αντίστοιχο χαρακτήρα c.
public static Character valueOf(char c)

//class String:
//Επιστρέφει τον χαρακτήρα στη θέση index.
public char charAt(int index)
//Μετατρέπει αυτό το string σε έναν πίνακα χαρακτήρων.
public char[] toCharArray()
```

Προσοχή

Οι μετατροπές αυτές γίνονται εντελώς διάφανα - δεν χρειάζεται να κάνουμε κάτι ιδιαίτερο για να πετύχουν.

Υπάρχει μια εξαίρεση: Όταν χρησιμοποιούμε τους τελεστές `==` και `!=` για μεταβλητές **non-primitive** τύπων, τότε αυτό που συγκρίνουμε είναι οι *αναφορές* των δύο μεταβλητών, όχι τα ίδια τα περιεχόμενα των αντικειμένων (δηλ. συγκρίνουμε τη θέση μνήμης στην οποία δείχνουν οι μεταβλητές, όχι το περιεχόμενο αυτών των θέσεων).

Π.χ. η συνθήκη ελέγχου `if (numbers.get(i)== numbers.get(j))` για τον πίνακα `numbers` δεν εξετάζει αν οι αριθμοί στη θέση i και j είναι ίδιοι. Για τη σύγκριση μπορούμε να χρησιμοποιήσουμε τη μέθοδο `equals`:

```
    if (numbers.get(i).equals(numbers.get(j)))
```

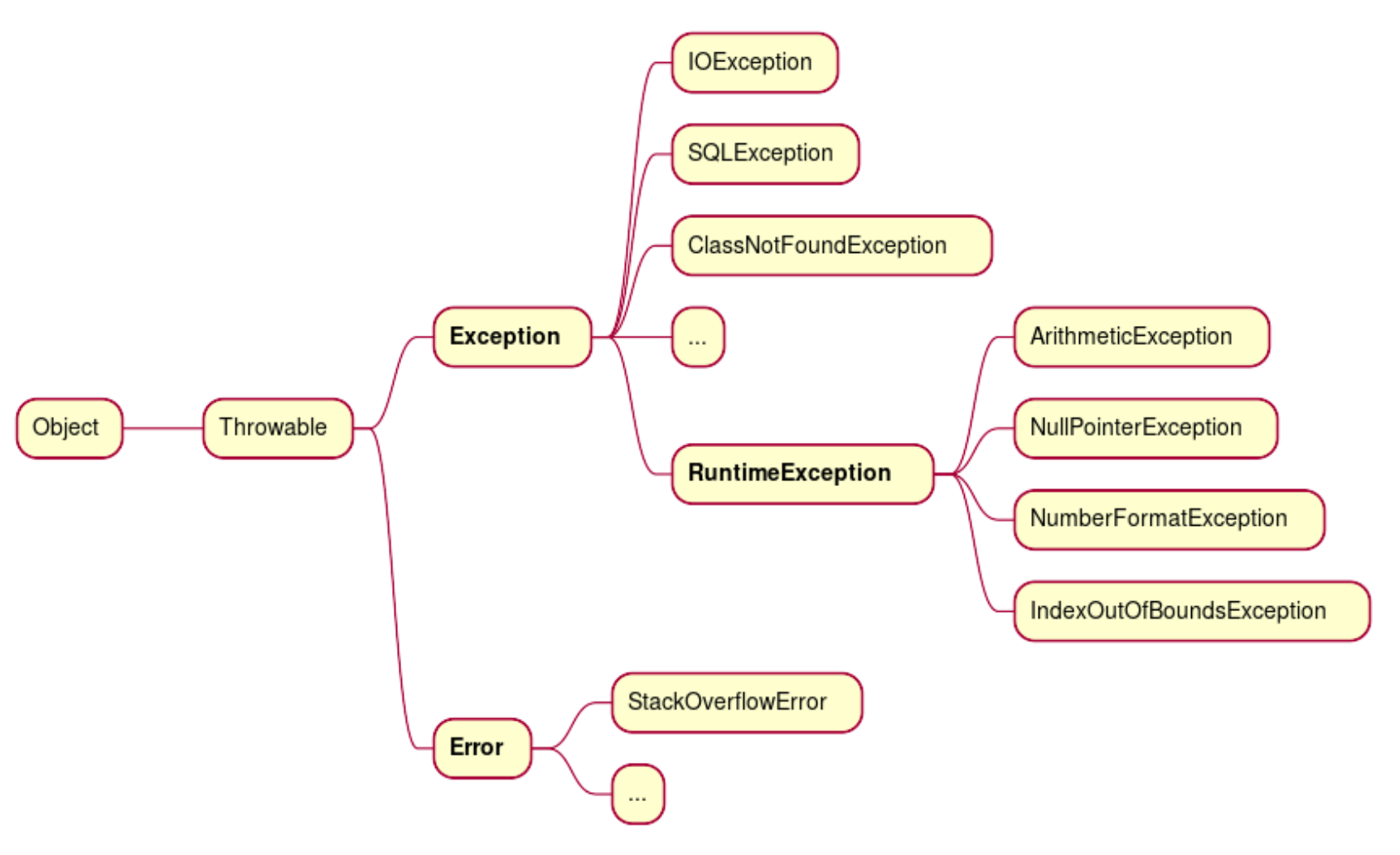
Χειρισμός εξαιρέσεων - Exception Handling

Χειρισμός εξαιρέσεων - Exception Handling

Ένας ισχυρός μηχανισμός για το χειρισμό των σφαλμάτων στο χρόνο εκτέλεσης, έτσι ώστε να μπορεί να συνεχιστεί η εκτέλεση της εφαρμογής.

Εξαίρεση είναι ένα γεγονός που διακόπτει την κανονική ροή του προγράμματος. **Εξαίρεση** είναι ένα **αντικείμενο** που *ρίχνουμε* ή *εγείρουμε* (thow, raise) σε χρόνο εκτέλεσης.

Ιεραρχία των εξαιρέσεων στη Java



Μέθοδοι της κλάσης Throwable

- `public String getMessage()` Returns a detailed message about the exception that has occurred.
- `public String toString()` Returns the name of the class concatenated with the result of `getMessage()`.
- `public void printStackTrace()` Prints the result of `toString()` along with the stack trace to System.err.

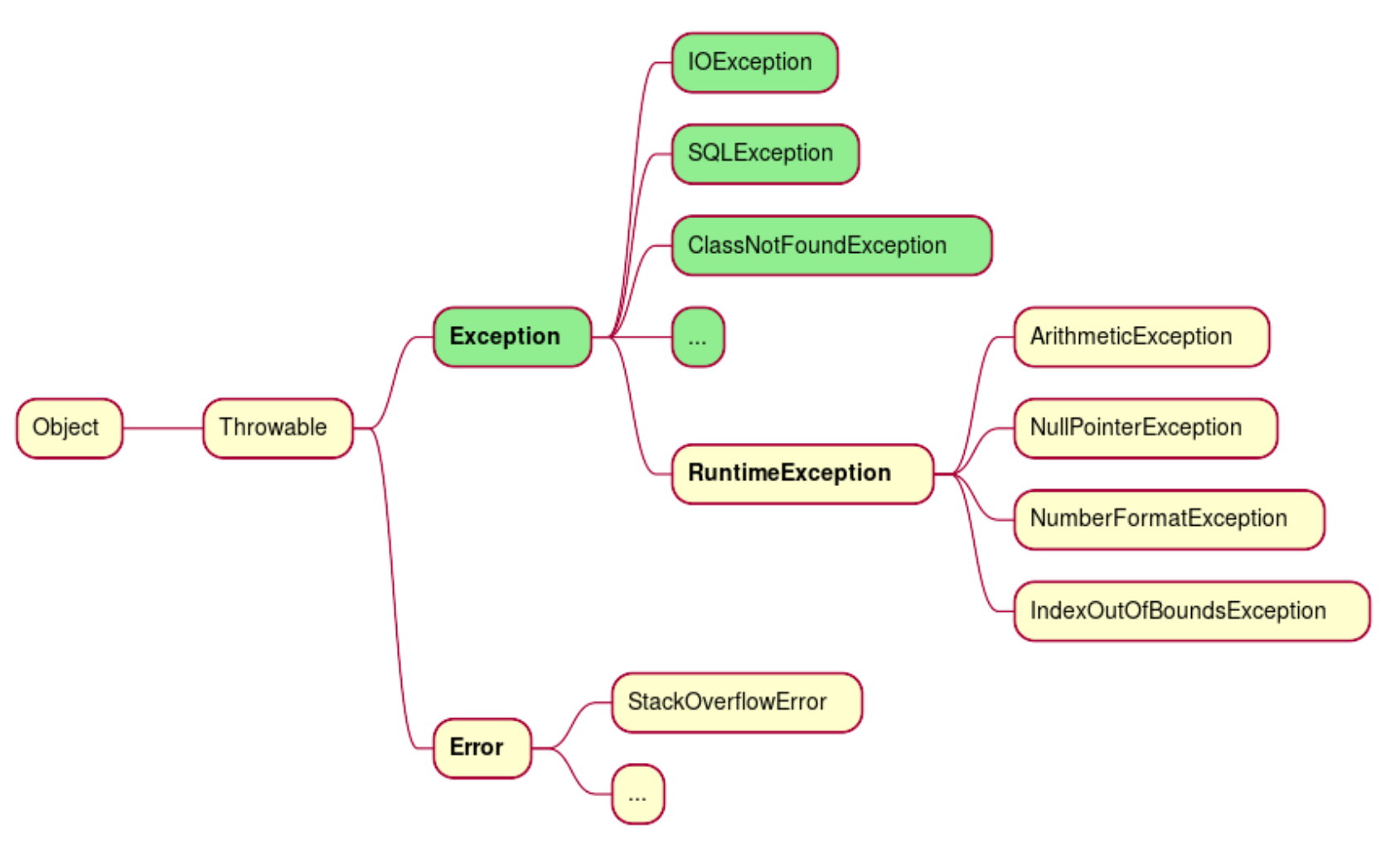
Λέξεις-κλειδιά

Λέξη	Περιγραφή
μπλοκ `try`	τοποθετούμε τον κώδικα που μπορεί να προκαλέσει εξαίρεση
μπλοκ `catch (Exception e)`	κώδικας που χειρίζεται την εξαίρεση
`finally`	κώδικας που εκτελείται πάντα, είτε γίνει εξαίρεση είτε όχι
`throw`	εγείρει μια εξαίρεση
`throws`	μέρος της υπογραφής της συνάρτησης δηλώνει πως η συνάρτηση μπορεί να προκαλέσει εξαίρεση

Τρεις κατηγορίες εξαιρέσεων

1. **Checked Exception**
2. **Unchecked Exception**
3. **Error**

Checked Exception



Checked Exception (συνέχεια)

Κληρονομούν την `Exception` (π.χ. `IOException`, `SQLException` ...) και **ελέγχονται σε χρόνο μεταγλώττισης**. Δηλ. ο προγραμματιστής είναι αναγκασμένος να γράψει κώδικα για να χειριστεί τις εξαιρέσεις, αλλιώς το πρόγραμμα δεν μεταγλωττίζεται.

```
    public static void main(String[] args)
    {
        FileReader file = new FileReader("somefile.txt");
    }
```

Δείτε την τεκμηρίωση του API για την FileReader.

Checked Exception (συνέχεια)

Οι Checked exceptions χρησιμοποιούνται για σφάλματα που έχουν να κάνουν με το χειρισμό πόρων (αρχεία, δίκτυο κλπ), δηλ. **πραγμάτων που δεν είναι στον άμεσο έλεγχο του προγράμματος**.

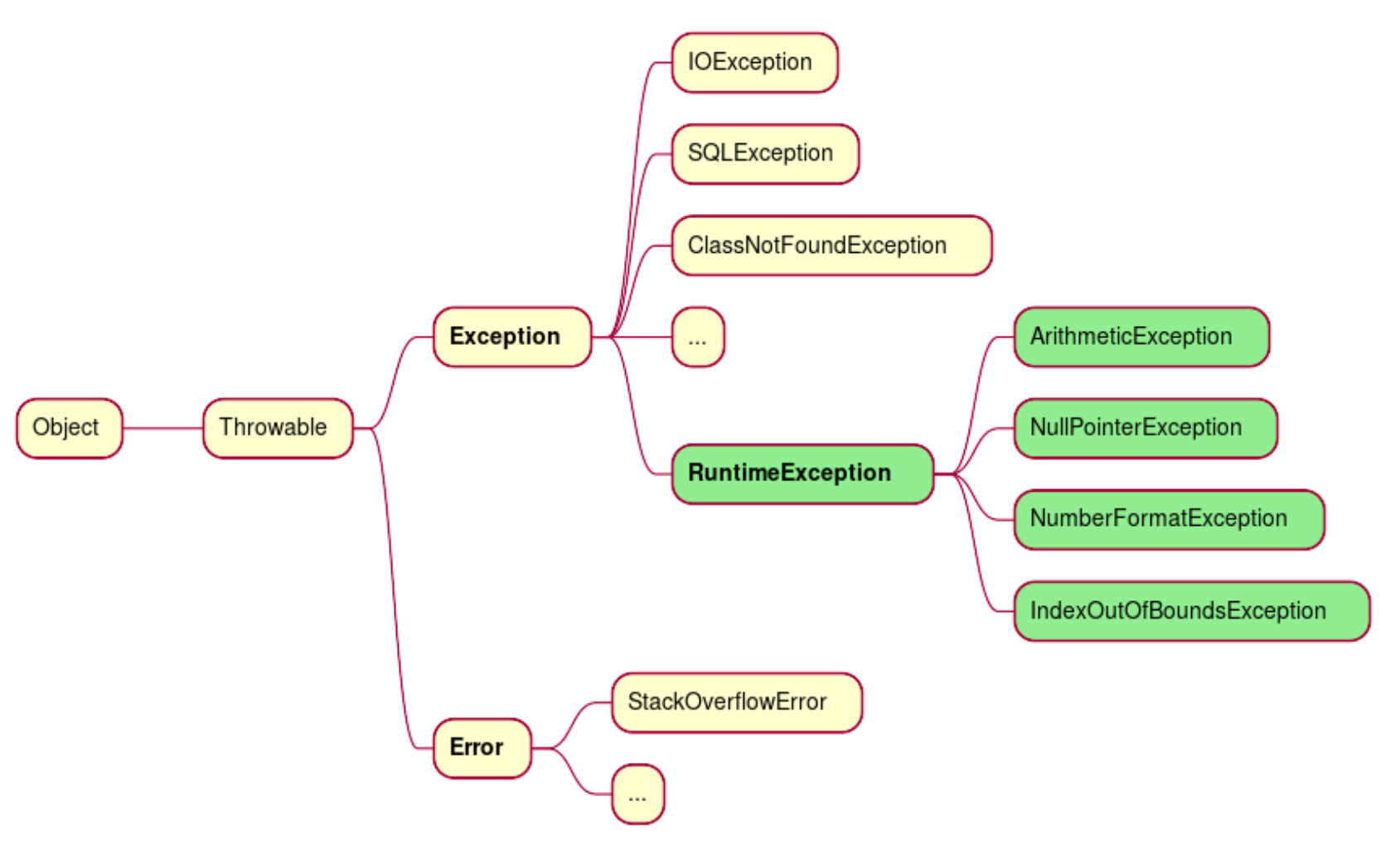
Ο προγραμματιστής δεν μπορεί να ξέρει αν την ώρα που εκτελείται το πρόγραμμα έχει πέσει το δίκτυο, ή αν το αρχείο που πάμε να ανοίξουμε υπάρχει στο δίσκο.

Υποχρεωτικά αυτές τις περιπτώσεις τις χειριζόμαστε στο χρόνο μεταγλώττισης. **Πρέπει** δηλαδή να γράψουμε κώδικα που να τις χειρίζεται.

Μερικές Checked Exceptions της Java

Εξαίρεση	Περιγραφή
`ClassNotFoundException`	Class not found.
`CloneNotSupportedException`	Attempt to clone an object that does not implement the Cloneable interface.
`IllegalAccessException`	Access to a class is denied.
`InstantiationException`	Attempt to create an object of an abstract class or interface.
`InterruptedException`	One thread has been interrupted by another thread.
`NoSuchFieldException`	A requested field does not exist.
`NoSuchMethodException`	A requested method does not exist.

Unchecked Exceptions



Unchecked Exceptionw (2)

Οι κλάσεις που κληρονομούν την `RuntimeException` ελέγχονται σε χρόνο εκτέλεσης, δηλ. τα RuntimeExceptions δεν απαιτείται να τα χειριστούμε στην μεταγλώττιση. Δεν υπάρχει εξαναγκασμός να χειριστούμε καμία από τις unchecked excpetions στη μεταγλώττιση.

Unchecked Exceptionw (3)

Με unchecked exception διακόπτεται η ροή του προγράμματος:

```java
public class Example1 {
    public static void main(String[] args) {
        // Θα προκύψει RunTimeException λόγω διαίρεσης με το 0
        // https://docs.oracle.com/en/java/javase/17/docs/api/
        //java.base/java/lang/ArithmeticException.html
        int a = 0;
        int b = 3;
        int c = b / a;

        System.out.println("Αποτέλεσμα: " + c);
        // Null pointer exception
        String s = null;
        System.out.println(s.length());
    }
}
```

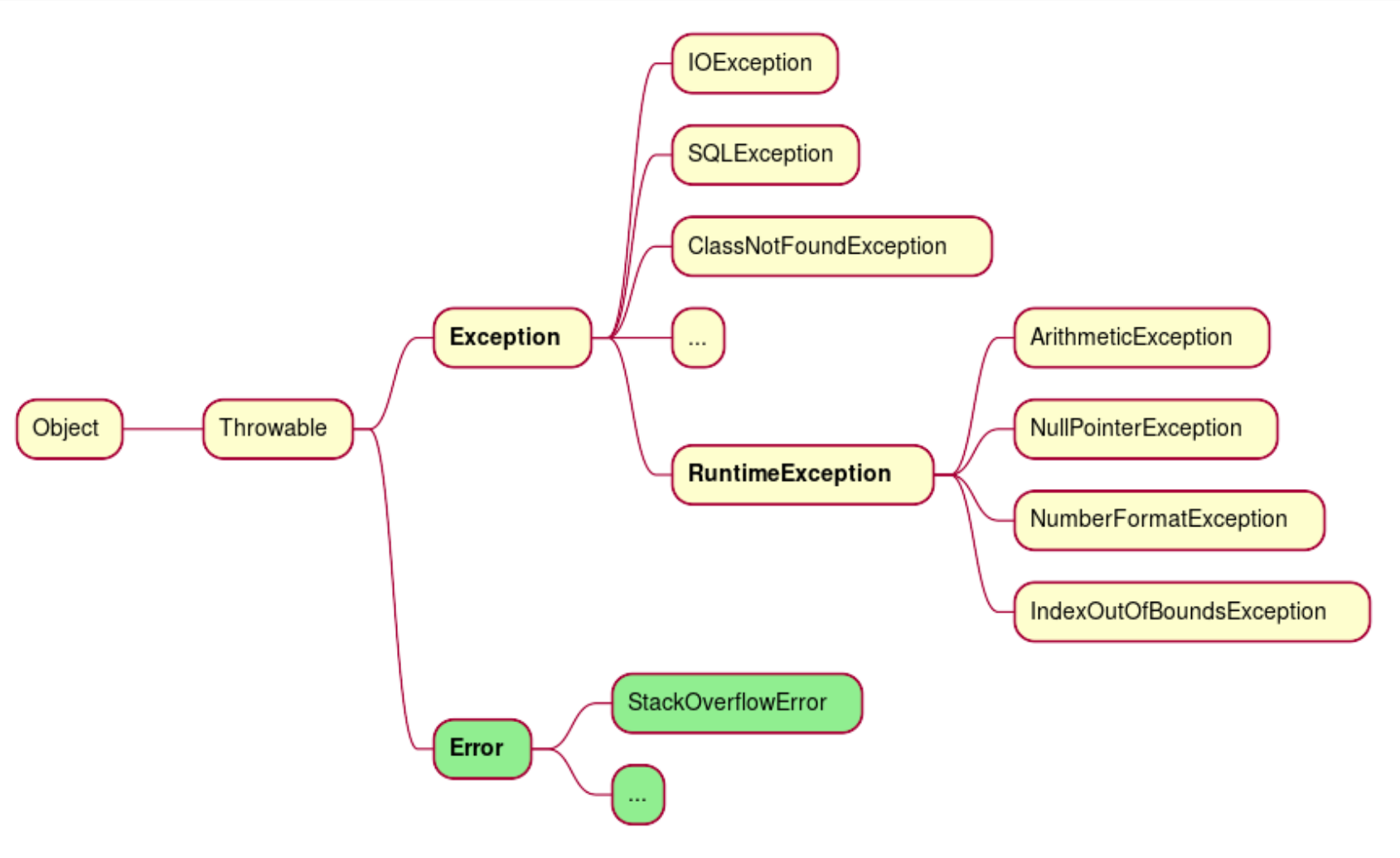
Μερικές Unchecked Exceptions της Java (1)

Εξαίρεση	Περιγραφή
`ArithmeticException`	Arithmetic error, such as divide-by-zero.
`ArrayIndexOutOfBoundsException`	Array index is out-of-bounds.
`ArrayStoreException`	Assignment to an array element of an incompatible type.
`ClassCastException`	Invalid cast.
`IllegalArgumentException`	Illegal argument used to invoke a method.
`IllegalMonitorStateException`	Illegal monitor operation, such as waiting on an unlocked thread.
`IllegalStateException`	Environment or application is in incorrect state.
`IllegalThreadStateException`	Requested operation not compatible with the current thread state.

Μερικές Unchecked Exceptions της Java (2)

Εξαίρεση	Περιγραφή
`IndexOutOfBoundsException`	Some type of index is out-of-bounds.
`NegativeArraySizeException`	Array created with a negative size.
`NullPointerException`	Invalid use of a null reference.
`NumberFormatException`	Invalid conversion of a string to a numeric format.
`SecurityException`	Attempt to violate security.
`StringIndexOutOfBounds`	Attempt to index outside the bounds of a string.
`UnsupportedOperationException`	An unsupported operation was encountered.

Error



`Error` σημαίνει ανεπανόρθωτο σφάλμα, π.χ. `OutOfMemoryError`, `VirtualMachineError`, `AssertionError` ...

Χειρισμός εξαίρεσης

Πως *χειριζόμαστε* τις εξαιρέσεις:

```
public class Example2 {
    public static void main(String[] args) {
        int a = 0; // Δοκιμάστε με a=1
        int b = 3;

        int c = 0;
        try {
            c = b / a;
            System.out.println(c);
            System.out.println("Δε συνέβεικαμίαεξαίρεση  ");
        } catch (Exception e) {
            System.out.println("Εξαίρεση: " + e.toString());
        } finally {
            System.out.println("Θα εκτελεστείοπωσδήποτε .");
        }
    }
}
```

Παραδείγματα

Example3.java

Example4.java

Όταν συμβεί εξαίρεση ...

... η JVM ελέγχει αν υπάρχει κώδικας που τη χειρίζεται.

- Αν δεν υπάρχει κώδικας για το χειρισμό της, τότε:
  1. τυπώνεται περιγραφή της εξαίρεσης,
  2. τυπώνεται το stack trace,
  3. το πρόγραμμα **τερματίζει**.
- Αν υπάρχει κώδικας χειρισμού της, το πρόγραμμα συνεχίζει να εκτελείται.

Διάδοση εξαίρεσης - Exception propagation

- Η εξαίρεση συμβαίνει στην κορυφή της στοίβας.
- Αν δεν τη χειριστούμε, θα περάσει στην μέθοδο που βρίσκεται από κάτω.
- Θα συνεχίσει να διαδίδεται προς τα κάτω μέχρι είτε να τη χειριστούμε ή να φτάσει στο τέλος της στοίβας και να τερματίσει το πρόγραμμα.

Σημαντικά σημεία

Η Java θα χειριστεί τις εξαιρέσεις μία κάθε φορά.

Τα μπλόκ `catch` πρέπει να ορίζονται από το πιο ειδικό στο ποιο γενικό, π.χ. να πιάνουμε πρώτα μια `IndexOutOfBoundsException` και έπειτα τη γενικότερη `Exception`. Αν δεν είναι σαφές το γιατί, κατασκευάστε ένα παράδειγμα.

Λέξη-κλειδί throw

Χρησιμοποιείται για να προκληθεί μια εξαίρεση (είτε της βιβλιοθήκης είτε μια που γράψαμε εμείς).

Λέξη-κλειδί throws

Μας πληροφορεί πως μπορεί να συμβεί μια εξαίρεση και άρα να γράψουμε κώδικα για να τη χειριστούμε.

- η λέξη `throws` μπορεί να υπάρξει μόνο στη δήλωση μιας μεθόδου,
- πληροφορεί τον compiler πως η μέθοδος μπορεί να προκαλέσει εξαίρεση,
- ο compiler ελέγχει αν υπάρχει κώδικας που να χειρίζεται την εξαίρεση.

Εξαιρέσεις που έχει ορίσει ο χρήστης

Όλες οι εξαιρέσεις πρέπει να είναι απόγονοι της `Throwable`.

- Αν θέλουμε να γράψουμε checked exception, τότε επεκτείνουμε την κλάση `Exception`.
- Αν θέλουμε να γράψουμε runtime exception, τότε επεκτείνουμε την κλάση `RuntimeException`.

Το μπλοκ try

- περιέχει τον κώδικα που μπορεί να προκαλέσει εξαίρεση,
- η εκτελείται μέχρι το σημείο που θα υπάρξει (αν υπάρξει) εξαίρεση,
- πρέπει να ακολουθείται είτε από μπλοκ `catch` είτε από μπλοκ `finally`, τα οποία θα χειριστούν την εξαίρεση,
- μπορούν να είναι εμφωλευμένα

```
try {
    //κώδικας που θα προκαλέσει εξάιρεση
} catch (Exception_class_Name name) {}
```

Το μπλοκ catch

- περιέχει τον κώδικα που χειρίζεται την εξαίρεση,
- εκτελείται μόνο αν υπάρξει εξαίρεση,
- μπορεί να ακολουθείται από άλλο `catch` ή ένα `finally`,
- αν υπάρχουν πολλά `catch` block, κάθε ένα εκτελείται με τη σειρά
  - ανά πάσα στιγμή συμβαίνει μόνο μια εξαίρεση και την χειρίζεται μόνο ένα `catch` block
  - τα catch block πρέπει να είναι από το πιο ειδικό στο πιο γενικό

Το μπλοκ finally

- εκτελείται πάντα, ανεξάρτητα από το αν υπάρχει ή όχι εξαίρεση,
- είναι πάντα τελευταίο