Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programming)

Εξαιρέσεις *(Exceptions)* – Λάθη *(Errors)* Χειρισμός των Εξαιρέσεων *(Exception Handling)*

Assertions

Παναγιώτης Σφέτσος, PhD
http://aetos.it.teithe.gr/~sfetsos/
sfetsos@it.teithe.gr

Περιεχόμενα Μαθήματος

- Εξαιρέσεις στη Java
- Διαχείριση των Εξαιρέσεων
- Η εντολή try … catch… finally
- Ρίχνοντας / Εγείροντας’ Εξαιρέσεις
- Assertions

Εξαιρέσεις στη Java *(1/8)*

Μηχανισμός εύρεσης και χειρισμού των λαθών - Εξαιρέσεις

Λάθη:

- του προγραμματιστή (λογικά λάθη), τα συντακτικά ο μεταγλωττιστής
- του χρήστη (π.χ. λάθος είσοδος)
- σε συσκευές του υλικού ή του δικτύου, κλπ.

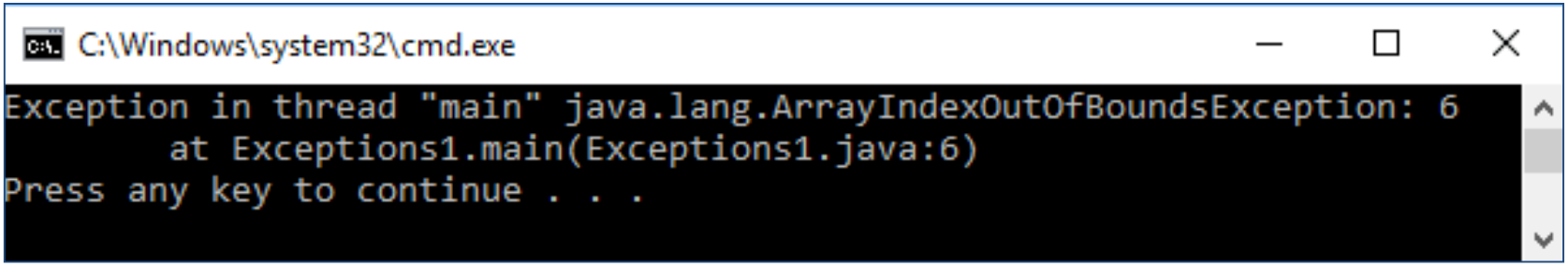
Η java διαθέτει ένα ενεργητικό μηχανισμό αντιμετώπισης λαθών, που καλούνται exceptions

Ερώτηση: Τι θα συμβεί αν υπάρξει **λογικό λάθος** στο πρόγραμμά μας;

Απάντηση: Runtime Error – απότομος τερματισμός του προγράμματος.

Εξαιρέσεις στη Java *(2/8)*

```
class Exceptions1 {
   public static void main(String[] args){
      double array1[] = new double[5];
      array1[6]=3.14; }}
```



Ερώτηση: Πως θα διαχειριστούμε το λάθος ώστε το πρόγραμμα να συνεχίσει την κανονική του ροη;

Απάντηση: Με τους διαχειριστές των εξαιρέσεων

Εξαιρέσεις στη Java *(3/8)*

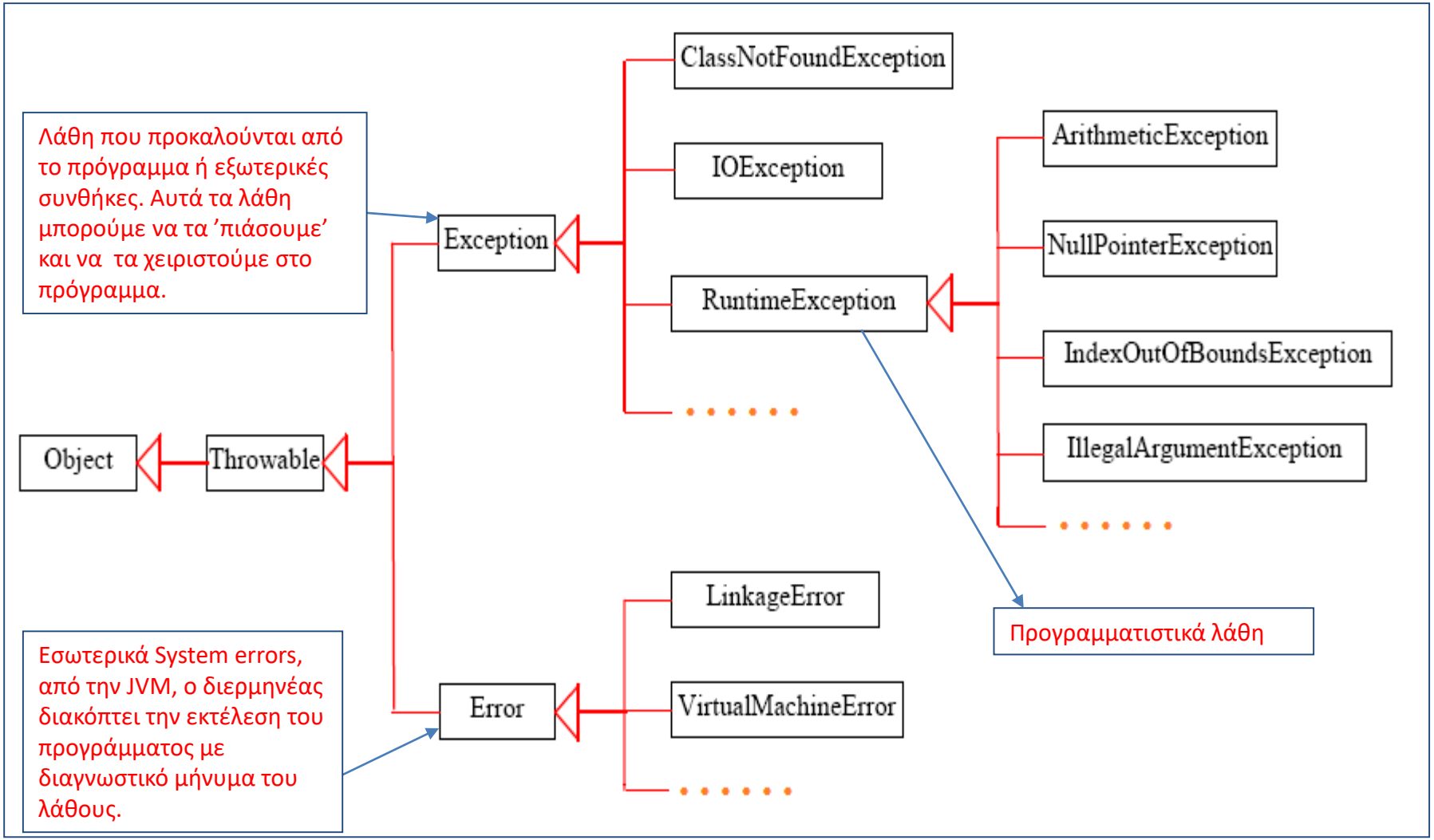
- **Εξαίρεση** είναι ένα συμβάν (***συνήθως λάθος***), κατά την εκτέλεση του προγράμματος που διακόπτει την ροή εκτέλεσης του.
- Όταν συμβεί κάποιο λάθος, τότε το πρόγραμμα **ρίχνει** *(throws)* μία **εξαίρεση** *(exception)* και το σύστημα ψάχνει να βρει τον κατάλληλο διαχειριστή εξαιρέσεων για να το διαχειριστεί.
- Ένας **διαχειριστής εξαιρέσεων** *(exception handler)* μπορεί να **πιάσει** *(catch)* κάποια εξαίρεση βάση του τύπου της εξαίρεσης και του ορισμού της catch.

Εξαιρέσεις στη Java *(4/8)*

- Αν δεν βρει κατάλληλο διαχειριστή εξαιρέσεων, τότε χρησιμοποιείται ο **default exception handler**, ο οποίος εμφανίζει μήνυμα λάθους και **τερματίζεται η εκτέλεση του προγράμματος**.
- Ωστόσο στα προγράμματα ορίζουμε τους κατάλληλους διαχειριστές εξαιρέσεων οι οποίοι ***θα πιάνουν την εξαίρεση*** και θα φροντίζουν ώστε το πρόγραμμα να ανανήψει από το λάθος.

Εξαιρέσεις στη Java *(5/8)*

- **Κλάσεις Εξαιρέσεων – Αντικείμενα εξαιρέσεων της Throwable**

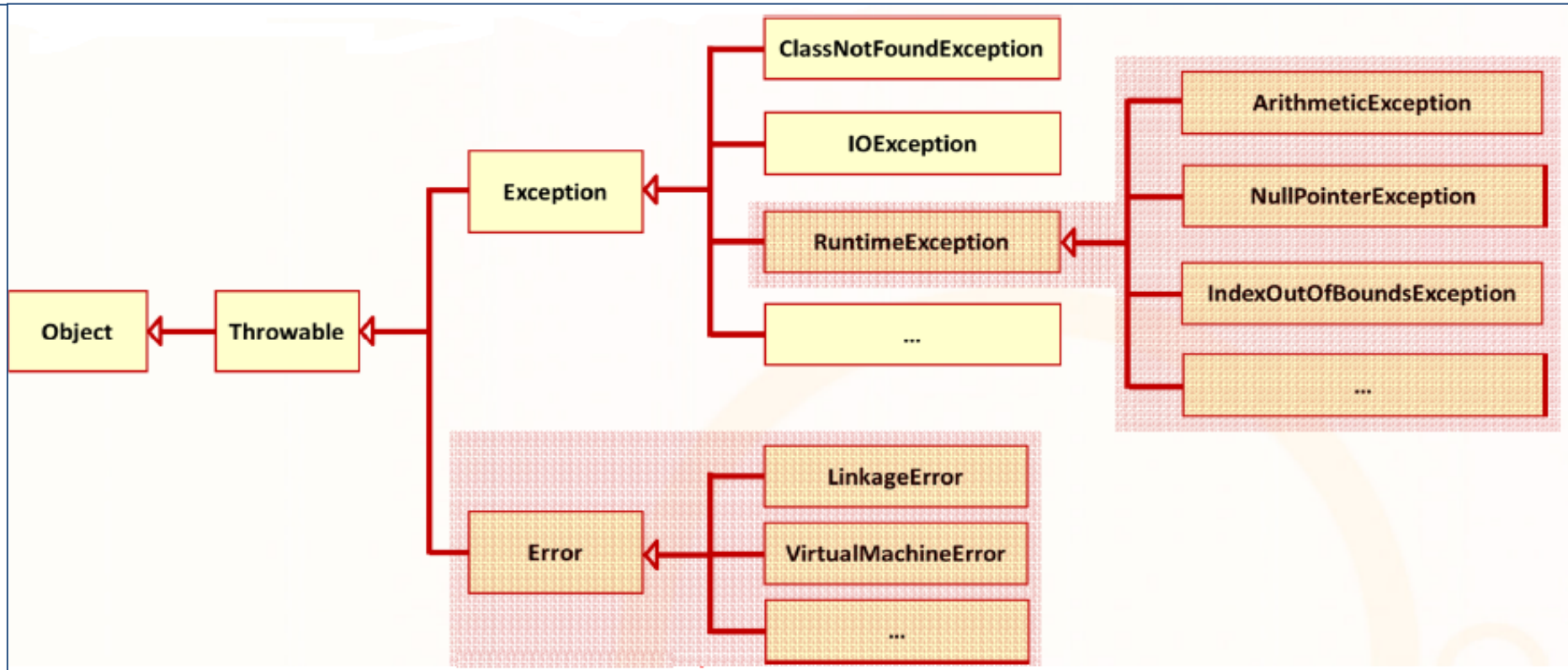


Εξαιρέσεις στη Java *(6/8)*

- **Checked Exceptions vs. Unchecked Exceptions**

Unchecked Exception: Μια κλάση που είναι υποκλάση της **RuntimeException** ή της **Error**. Συνήθως απρόβλεπτα λάθη που δύσκολα διαχειριζόμαστε (try…catch).

Checked Exception: Κάθε άλλη κλάση που δεν είναι υποκλάση της RuntimeException ή της Error. Ο προγραμματιστής μπορεί να τα προβλέψει και να τα διαχειριστεί.



Εξαιρέσεις στη Java *(7/8)*

- **Μερικά Exceptions στη Java**

Exception	Περιγραφή
ArithmeticException	Συνέβη αριθμητικό λάθος
ArrayIndexOutOfBoundsException	Εκτός ορίων του πίνακα
ClassNotFoundException	Η κλάση δε βρέθηκε
IllegalAccessException	Λάθος στην πρόσβαση περιεχομένου κλάσης
InstantiationException	Λάθος στη δημιουργία στιγμιοτύπου αφηρημένης κλάσης
NoSuchFieldException	Ανύπαρκτο πεδίο
NoSuchMethodException	Ανύπαρκτη μέθοδος
NullPointerException	Null αναφορά σε αντικείμενο
NumberFormatException	Λάθος στον αριθμητικό τύπο
StringIndexOutOfBoundsException	Εκτός ορίων του String
UnsupportedOperationException	Μη υποστηριζόμενη εντολή

Εξαιρέσεις στη Java *(8/8)*

- **Πως γράφουμε μια νέα κλάση Εξαίρεσης** *(exception class)*

(1) Πρέπει πρώτα να αποφασίσουμε αν θα είναι **checked** ή **unchecked** εξαίρεση για να την επεκτείνουμε (κληρονομήσουμε-αντίστοιχα): την **Exception** ή την **RuntimeException**.

(2) Η κλάση μπορεί να έχει δύο δομητές έναν που κληρονομεί (με την super() χωρίς παράμετρο) και έναν που δέχεται σαν παράμετρο το μήνυμα και το περνά με την super(String message) στην υπερκλάση όποια και αν είναι αυτή.

Παραδείγματα:

```
class MyException extends RuntimeException {
   private String message;
   public MyException(String message) {this.message = message;}
   public String getMessage() { return message;} }
```

```
class MyException extends Exception
   public MyException() {super();}
   public MyException(String message) {super(message); }  }
```

Διαχειριστές Εξαιρέσεων – Η εντολή try…catch…finally *(1/4)*

- Προστατεύουμε το κομμάτι του κώδικα που μπορεί να προκαλέσει λάθος στο μπλοκ εντολών της εντολής **try {}**.
- Αν προκληθεί στο συγκεκριμένο κομμάτι του κώδικα κάποιο σφάλμα η **catch() {}** το πιάνει και το χειρίζεται (διαχειριστής εξαιρέσεων).
- Ο διαχειριστής εξαιρέσεων είναι ένα κομμάτι κώδικα που χειρίζεται το ειδικό λάθος που προέκυψε.

```
try {
   εντολές
   :       :
} catch (όνομα του τύπου της Εξαίρεσης) {
   εντολές
   :       :
} catch (όνομα του τύπου της Εξαίρεσης) {
   εντολές
   :       :
} finally {
   εντολές
   :       :
}
```

```
try {
  statements;
}
catch (Exception1 exVar1) {
  handler for exception1;
}
catch (Exception2 exVar2) {
  handler for exception2;
}
...
catch (ExceptionN exVar3) {
  handler for exceptionN;
}
```

Διαχειριστές Εξαιρέσεων – Η εντολή try…catch…finally *(2/4)*

- Αν *δημιουργηθεί* ***εξαίρεση*** *στο μπλοκ της try* τότε ***εκτελείται*** το κατάλληλο catch μπλοκ διαφορετικά ο κώδικας δεν μπαίνει μέσα στη catch αλλά συνεχίζει σαν να μην υπήρχε η εντολή try…catch.

- Η catch είναι περίπου όπως μία μέθοδος δηλαδή ακολουθούν παρενθέσεις και το σώμα σε άγκιστρα. Στην παρένθεση μπαίνει το όνομα της κλάσης της εξαίρεσης που θα πιαστεί και ακολουθεί το όνομα μιας μεταβλητής.

Διαχειριστές Εξαιρέσεων – Η εντολή try…catch…finally *(3/4)*

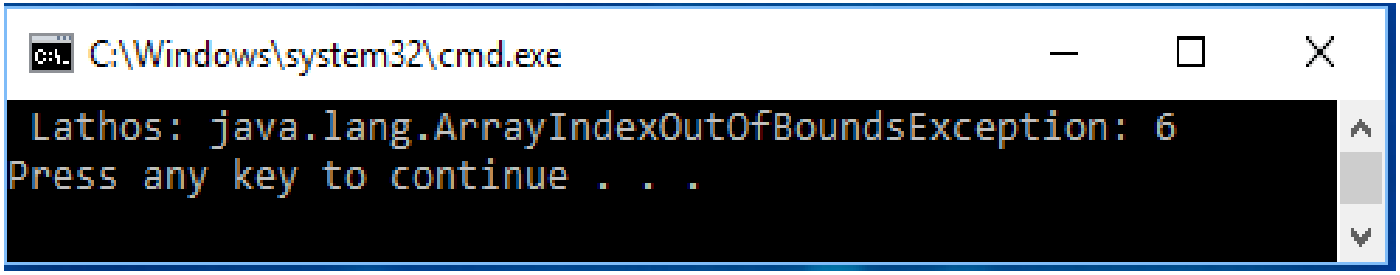
- Μέσα στην catch μπορούμε να αναφερθούμε στο αντικείμενο (εξαίρεση). Ο συνηθέστερος τρόπος χρήσης του αντικειμένου αυτού είναι να κληθεί η ***μέθοδος getMessage()***. Η μέθοδος αυτή υπάρχει σε όλες τις εξαιρέσεις και εμφανίζει ένα λεπτομερές μήνυμα σφάλματος.

- Μπορούμε να έχουμε **πολλές διαφορετικές catch** και προαιρετικά την **finally.** Η finally (όταν την συμπεριλάβουμε) περιέχει ένα κομμάτι κώδικα που εκτελείται είτε συμβεί λάθος είτε όχι μέσα στην try.

Διαχειριστές Εξαιρέσεων – Η εντολή try…catch…finally *(4/4)*

Παράδειγμα 1ο:

```
class Exceptions2 {
  public static void main(String[] args){
    double array1[] = new double[5];

    try {
       array1[6]=3.14;

    }catch (ArrayIndexOutOfBoundsException e) {
       System.out.println(" Lathos: " + e);
    }
}}
```



‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις*(1/10)*

Throwing Exceptions

- Σε οποιαδήποτε κλάση μπορούμε να **ορίσουμε** και να **εγείρουμε** δικά μας exceptions ή να χρησιμοποιήσουμε εξωτερικές κλάσεις που να ορίζουν δικά τους exceptions. Η Java απαιτεί να δηλώνουμε κάθε exception που πιθανόν να εμφανιστεί κατά την εκτέλεση μιας μεθόδου (ακόμη και της main()):
  public static void main(String args[]) throws java.io.IOException {
- Πρώτα **δημιουργεί** *(με new)* **ένα αντικείμενο εξαίρεσης** το οποίο θα πρέπει να είναι στιγμιότυπο της κλάσης Exception ή κάποιας υποκλάσης της.
- Έπειτα με την **εντολή *throw*** και το αντικείμενο εξαίρεσης προκαλείται exception. Με χρήση της throw μπορούμε να ***εγείρουμε οποιοδήποτε αντικείμενο εξαίρεσης της Throwable***.

‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(2/10)*

- Η εκτέλεση του κώδικα θα **διακοπεί** στην εντολή throw και ο υπόλοιπος **κώδικας που ακολουθεί δεν θα εκτελεστεί.** Επίσης η μέθοδος μέσα στην οποία συνέβη η εξαίρεση δεν θα επιστρέψει κάποια τιμή.

```
if (number2 == 0) throw new ArithmeticException();
if (number2 == 0) throw new ArithmeticException("O diairetis den
                                                 mporei na einai 0");
```

```
public void setRadius(double newRadius) throws IllegalArgumentException {
  if (newRadius >= 0)
   radius =  newRadius;
 else
   throw new IllegalArgumentException(“H aktina den mporei na einai arnitiki");
 }
```

‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(3/10)*

- Η εντολή **throw** διακόπτει την ροή εκτέλεσης, μεταφέροντας τον έλεγχο του προγράμματος σε κάποιον **διαχειριστή εξαιρέσεων** (σαν μία μορφή return).

- Με την χρήση της throw, και όταν εγερθεί εξαίρεση, **τότε είναι ξεκάθαρο ποιος διαχειριστής θα αναλάβει να την αντιμετωπίσει**.

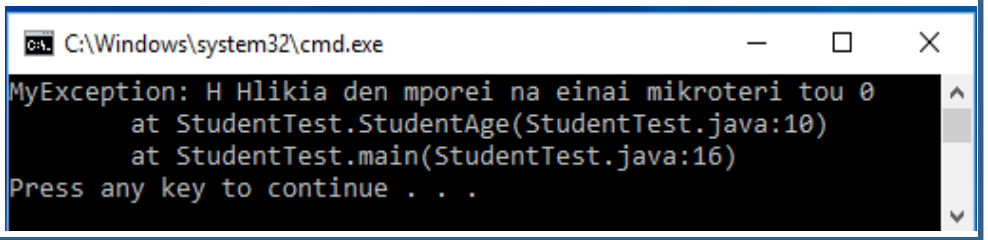
- Το αντικείμενο της εξαίρεσης, (δηλ. η εξαίρεση), θα δοθεί στον ***κατάλληλο exception handler***, απ' όπου και συνεχίζεται η εκτέλεση του προγράμματος.

- Χρήση της κατάλληλης **try…catch**.

‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(4/10)*

Παράδειγμα 2ο :

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

class StudentTest {
  static void  StudentAge(int age) throws MyException{
     if(age < 0)
        throw new MyException("H Hlikia den mporei na einai
                                  mikroteri tou 0");
     else
        System.out.println("Sosti eisodos Hlikias!!");}

  public static void main(String[] args) {
      try {
           StudentAge(-2);   //klisi tis methodou mesa se try..catch
     }catch (MyException e) {
           e.printStackTrace();
      }}}
```



‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(5/10)*

Παράδειγμα 3ο (α):

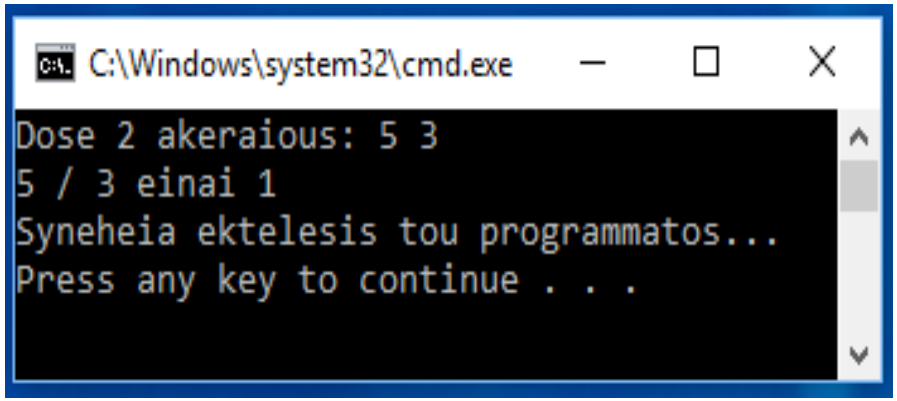
```
import java.util.Scanner;

class Exception3 {
 public static int quotient(int number1, int number2) {
  //προαιρετικά – θα μπορούσε να παραληφθεί
  if (number2 == 0)throw new ArithmeticException("O diairetis den
                                    mporei na einai 0");
 return number1 / number2;}

 public static void main(String[] args) {
  Scanner input = new Scanner(System.in);
  System.out.print("Dose 2 akeraious: ");
  int number1 = input.nextInt();
  int number2 = input.nextInt();
  try {
    int result = quotient(number1, number2);
      System.out.println(number1 + " / " + number2 + " einai " +
                                  result);
  }
```

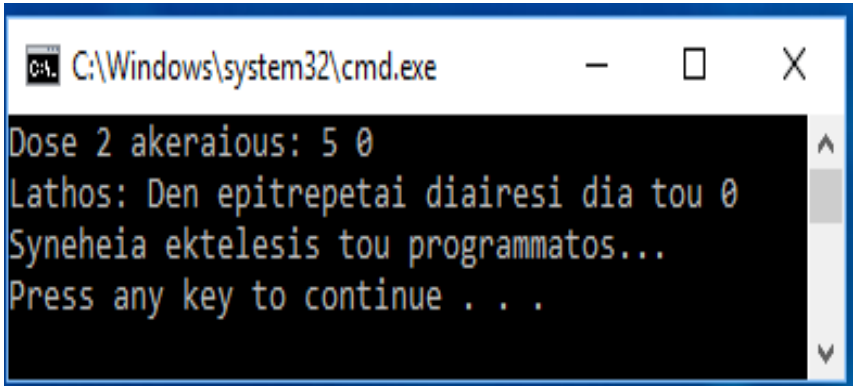
‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(6/10)*

Παράδειγμα 3ο (β):

```
catch (ArithmeticException ex) {
 System.out.println("Lathos: Den epitrepetai diairesi dia tou 0 ");
 }

  System.out.println("Syneheia ektelesis tou programmatos...");}
 }
```

(2) Run: (α) Σωστό, **(β) με λάθος στον διαιρέτη**





‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(7/10)*

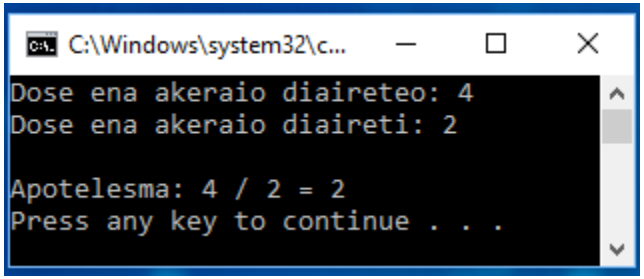
Παράδειγμα 4ο (παραλλαγή του 3 με πολλαπλές εξαιρέσεις) (α):

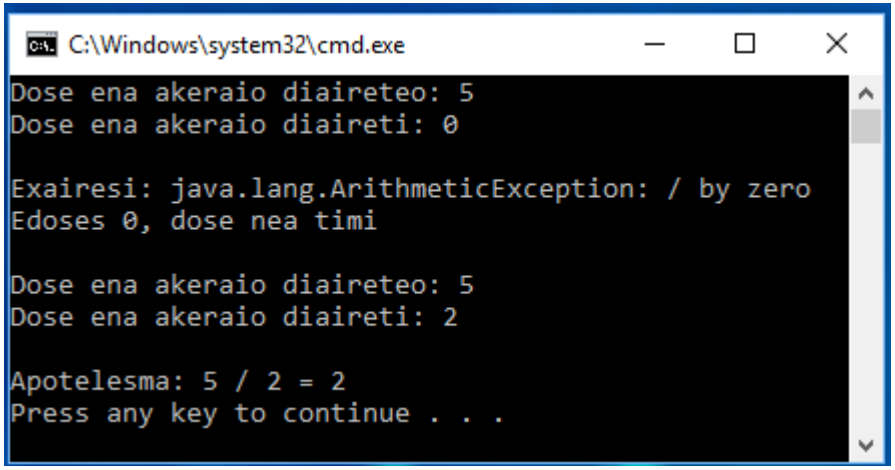
```
import java.util.*;
class Exception4 {
 public static int quotient( int numerator, int denominator) throws
                                        ArithmeticException {
  return numerator / denominator;}

 public static void main( String args[]) {
  Scanner scanner = new Scanner( System.in );
  boolean Loop = true;
  do {
   try {
    System.out.print( "Dose ena akeraio diaireteo: " );
    int numerator = scanner.nextInt();
    System.out.print( "Dose ena akeraio diaireti: " );
    int denominator = scanner.nextInt();
    int result = quotient( numerator, denominator );
    System.out.printf( "\nApotelesma: %d / %d = %d\n",
                        numerator,denominator, result );
    Loop = false; }
```

‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(8/10)*

Παράδειγμα 4ο (παραλλαγή του 3 με πολλαπλές εξαιρέσεις) (β):

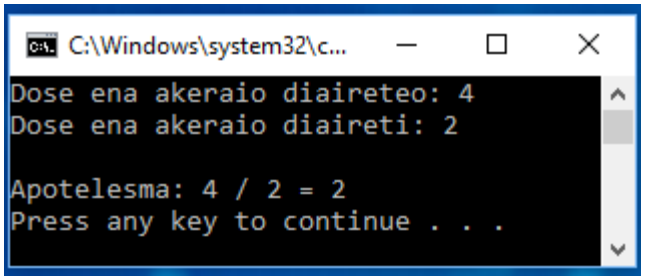
```
catch (InputMismatchException inputMismatchException)
 {
  System.err.printf( "\nExairesi: %s\n", inputMismatchException );
  scanner.nextLine(); //dose nea eisodo
  System.out.println("Lathos timi, dose akeraio\n" );
  }
  catch (ArithmeticException arithmeticException)
  {
   System.err.printf( "\nExairesi: %s\n", arithmeticException );
   System.out.println("Edoses 0, dose nea timi \n" );
  }
  } while (Loop); }}
```

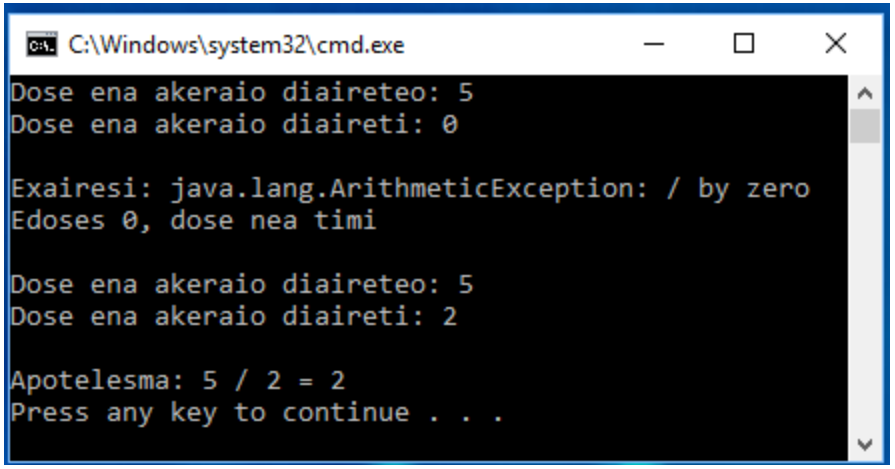


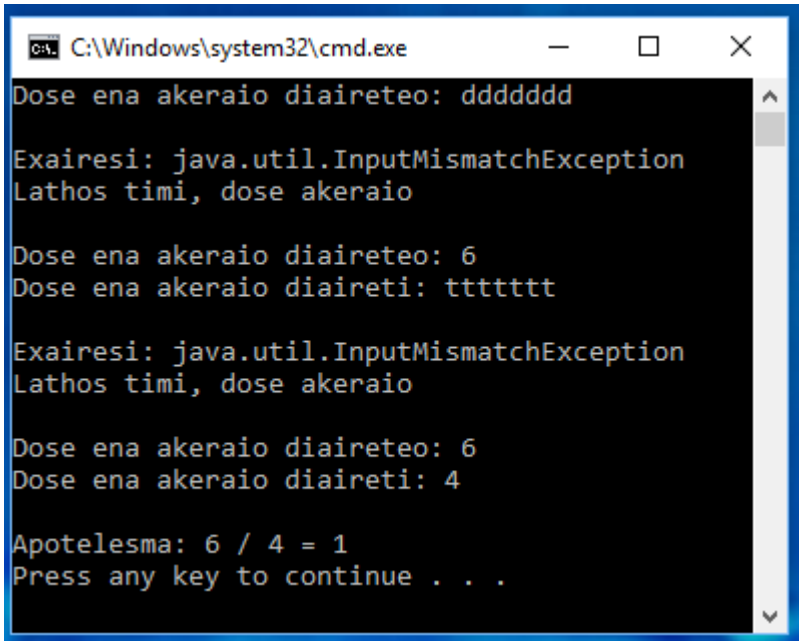


‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(9/10)*

Παράδειγμα 4ο (παραλλαγή του 3 με πολλαπλές εξαιρέσεις) (γ):

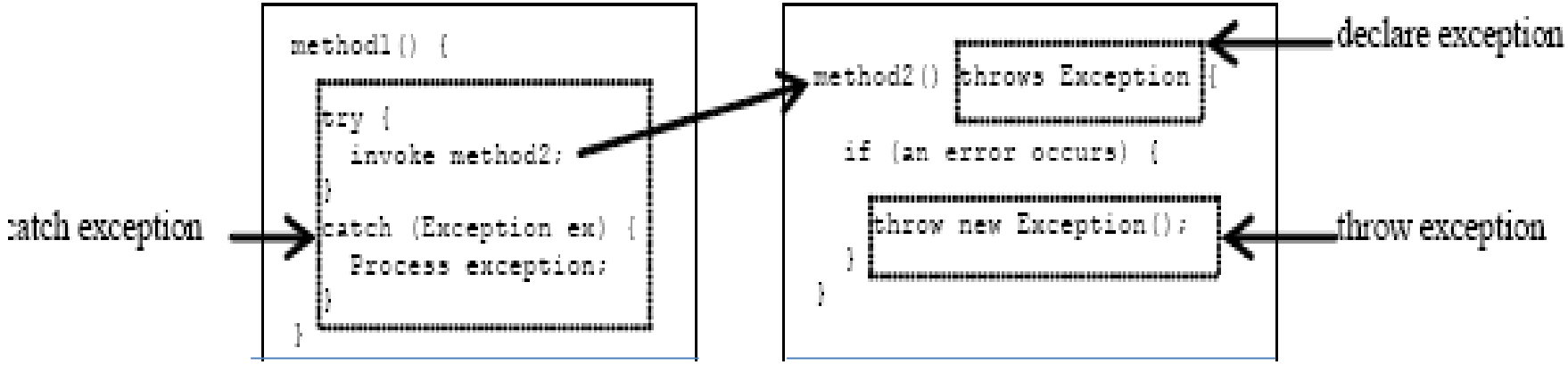


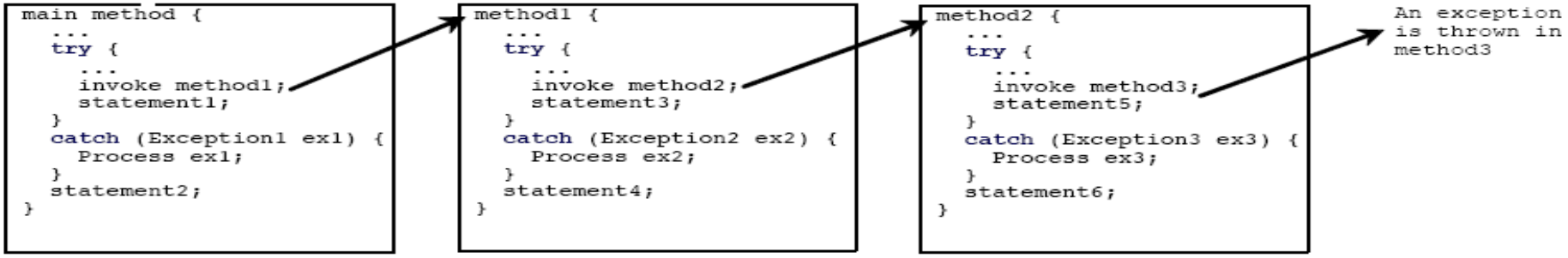


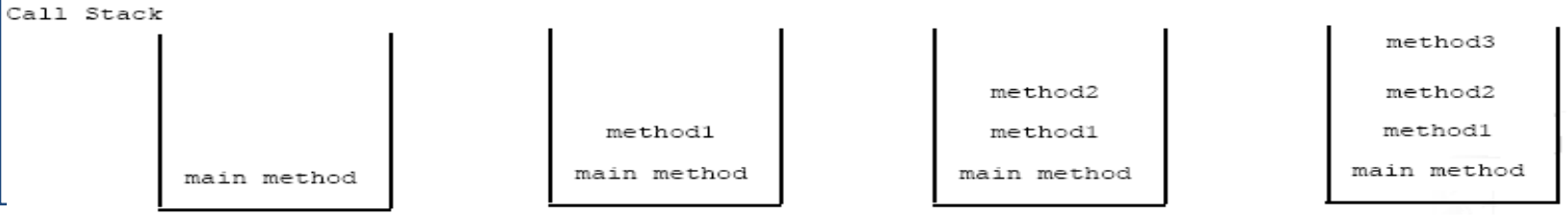


‘Ρίχνοντας / Εγείροντας’ Εξαιρέσεις *(10/10)*

Δήλωση και Διαχειρισμός Εξαιρέσεων κατά την κλήση μεθόδων







Παραδείγματα από το βιβλίο του Liang

Assertions *(1/4)*

- **Assertion:** Εντολή που επιτρέπει την **επιβεβαίωση κάποιων υποθέσεων** για το πρόγραμμα που εκτελείται.
- Ισχύουν από JAVA 1.4 και μετά.
- Περιλαμβάνουν boolean εκφράσεις που πρέπει να είναι true κατά τη διάρκεια εκτέλεσης.
- Δηλώνονται με τη λέξη assert.
  assert *<έκφραση>* π.χ., assert age==19;
  assert *<έκφραση>*: *<μήνυμα>*, assert (age==19): “Hlikia= ”+age;
- Όταν εκτελεστεί το assertion και είναι λάθος *(false),* τότε θα ‘ριχτεί’ εξαίρεση: **throw AssertionError**
- Η κλάση AssertionError class έχει ένα no-arg δομητή και 7 overloaded single-argument δομητές με τύπο *παραμέτρων int, long, float, double, boolean, char, and Object.*

Assertions *(2/4)*

- Τα assertions δεν είναι ενεργοποιημένα εξ αρχής. Τα ενεργοποιούμε κατά την ***εκτέλεση του προγράμματος*** με τον διακόπτη **–enableassertions**, ή τη συντομογραφία **–ea.**
  Π.χ.
  c > java –ea AssertionExample

- Μπορούν να ενεργοποιηθούν σε επίπεδο πακέτου ή κλάσης

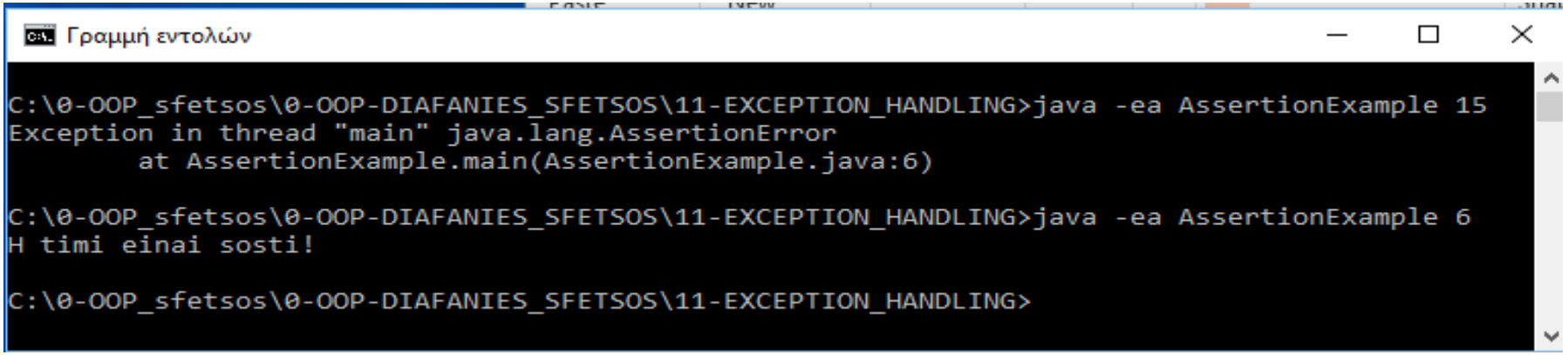
- Παράδειγμα: **java –ea:package1 –da:MyClass AssertDemo**
  (ενεργοποίηση των assertions στο πακέτο package1 και απενεργοποίηση στην κλάση MyClass).

Assertions *(3/4)*

Παράδειγμα:

```
class AssertionExample {
    public static void main(String[] args) {
        // get a number in the first argument
        int number = Integer.parseInt(args[0]);
         assert number <= 10; //Exception an o arithmos einai>10
         System.out.println("H timi einai sosti!"); } }
```

- **Προσέξτε μια μεταγλώττιση, δύο διαφορετικές εκτελέσεις:**



Assertions *(4/4)*

Assertions vs. Exceptions

- Τα exceptions διαχειρίζονται σφάλματα και εξασφαλίζουν την ευρωστία του προγράμματος.
- Τα assertions επιβεβαιώνουν την ορθότητα του προγράμματος.
- Άρα τα assertions δεν **πρέπει** και **δεν μπορούν** να αντικαταστήσουν τον χειρισμό των εξαιρέσεων.
- Τα assertions μπορούν να απενεργοποιηθούν, ενώ αυτό δεν συμβαίνει στον χειρισμό των εξαιρέσεων.