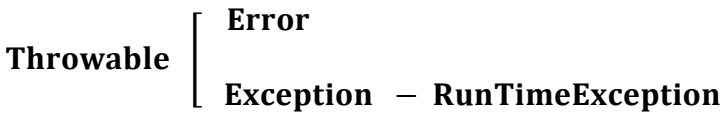
9 ΕΞΑΙΡΕΣΕΙΣ - EXCEPTIONS

Με τον όρο **Εξαιρέσεις** (**Exceptions**) στη Java χαρακτηρίζουμε τα **σφάλματα** που μπορεί να προκύψουν κατά την εκτέλεση ενός προγράμματος, όπως διαίρεση με το μηδέν, προσπάθεια πρόσβασης σε στοιχείο ενός πίνακα με τιμή δείκτη εκτός των ορίων που έχουν δηλωθεί κ.λ.π.. Σε όλες τις Γλώσσες Προγραμματισμού, σε περίπτωση σφάλματος διακόπτεται η εκτέλεση του προγράμματος και εμφανίζονται ένα ή περισσότερα μηνύματα με πληροφορίες για το είδος και το σημείο του κώδικα στο οποίο προέκυψε το σφάλμα. Η Java δίνει τη δυνατότητα στο χρήστη να διαχειριστεί το σφάλμα που μπορεί να προκύψει, έτσι ώστε το πρόγραμμα να συνεχίσει κανονικά την εκτέλεσή του. Αυτό γίνεται με το **Χειριστή Εξαιρέσεων**, ένα block εντολών, οι οποίες καλούνται όταν προκύψει κάποιο σφάλμα και το χειρίζονται ανάλογα. Η Java περιέχει μια βιβλιοθήκη χειρισμού των πιο συνηθισμένων σφαλμάτων, αλλά δίνει και στο χρήστη τη δυνατότητα να χειριστεί τα σφάλματα με όποιον τρόπο επιθυμεί. Οι Εξαιρέσεις είναι **αντικείμενα**, οπότε ανήκουν σε κάποια κλάση. Οι Εξαιρέσεις που ενδιαφέρουν τον προγραμματιστή (γιατί υπάρχουν και σφάλματα που αφορούν την εικονική μηχανή της Java, τα οποία ανήκουν στην κλάση **Error**) είναι τα σφάλματα που προγράμματος, τα οποία ανήκουν στην κλάση **Exception** και πιο πολύ αυτά που συμβαίνουν κατά το χρόνο εκτέλεσης του προγράμματος (**Run-Time Exceptions**), τα οποία ανήκουν στην κλάση **RunTimeException**. Όλες οι εξαιρέσεις ανήκουν στην κλάση **Throwable**., ενώ οι κλάσεις **Error** και **Exception** είναι υπο-κλάσεις της κλασης **Throwable** και η κλάση **RunTimeException** είναι υπο-κλάση της κλασης **Exception** όπως φαίνεται στο επόμενο σχήμα :



Ο **Χειριστής Εξαιρέσεων** περιλαμβάνει τις παρακάτω εντολές - λέξεις κλειδιά :

try : block εντολών, στο οποίο μπορεί να **προκύψει** μια εξαίρεση.

catch : block εντολών, οι οποίες θα **χειριστούν** την εξαίρεση, αν προκύψει.

finally : προαιρετικό block εντολών, οι οποίες θα **εκτελεστούν**, **οπωσδήποτε**, υπάρξει δεν υπάρξει εξαίρεση.

throws : εντολή, η οποία **προωθεί** μια εξαίρεση στην καλούσα μέθοδο να τη χειριστεί.

Ο χειρισμός των εξαιρέσεων έχει την παρακάτω μορφή :

```
try{
    block εντολών, στο οποίο μπορεί να προκύψει μια εξαίρεση.
}
catch ( <Τύπος_Εξαίρεσης_1> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
catch ( <Τύπος_Εξαίρεσης_2> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
…
catch ( <Τύπος_Εξαίρεσης_n> <Αντικείμενο_Εξαίρεσης> ) {
    block εντολών, οι οποίες θα χειριστούν την εξαίρεση, αν προκύψει.
}
finally{
    προαιρετικό block εντολών, οι οποίες θα εκτελεστούν, οπωσδήποτε.
}
```

9.1 Παράδειγμα Εξαίρεσης που προκαλείται σε εντολή της `main()`

Στο επόμενο πρόγραμμα προσπαθούμε να αλλάξουμε το περιεχόμενο της θέσης 5 ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις 0 μέχρι **a.length**−1 = 4 :

```
package exceptions1;
public class Exceptions1 {

    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};
        //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων
        a[5] = 99;
    }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions1.Exceptions1.main(Exceptions1.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

Με το τρέξιμο του προγράμματος προκύπτει η εξαίρεση `ArrayIndexOutOfBoundsException` στη γραμμή 7 της `main()`, η οποία ανήκει στην κλάση `Exceptions1`, η οποία ανήκει στο πακέτο `exceptions1` (`exceptions1.Exceptions1.main`).

9.2 Παράδειγμα Εξαίρεσης σε εντολή επανάληψης της `main()`

Στο επόμενο πρόγραμμα προσπαθούμε να εμφανίσουμε το περιεχόμενο των θέσεων 0 μέχρι 5 ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις 0 μέχρι **a.length**−1=4, οπότε προκαλείται η εξαίρεση **ArrayIndexOutOfBoundsException** :

```
package exceptions2;
public class Exceptions2 {
  public static void main(String[] args) {
    int a[] = {1,2,3,4,5};
     //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων κατά την Εμφάνιση
    for ( int i = 0; i <= a.length; i++ )
       System.out.println("a[" + i + "] = " + a[i]);
    }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions2.Exceptions2.main(Exceptions2.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

9.2.1 Χειρισμός της Εξαίρεσης σε εντολή επανάληψης της `main()`

Στο προηγούμενο παράδειγμα η προσπάθεια πρόσβασης στα στοιχεία στις θέσεις `i = a.length` = 5 και `i = a.length + 1` = 6 του πίνακα `a` εκτός των ορίων της δήλωσης των θέσεων του πίνακα (0 μέχρι **`a.length - 1`**) με την εντολή

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί 2 φορές την εξαίρεση `ArrayIndexOutOfBoundsException`. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε **την εντολή που θα προκαλέσει την εξαίρεση** στο `block try` και **τις εντολές χειρισμού της εξαίρεσης** στο `block catch` , στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα καθώς και το περιεχόμενο του αντικειμένου της εξαίρεσης :

```
package exceptions2a;
public class Exceptions2a {
 public static void main(String[] args) {
  int a[] = {1,2,3,4,5};
  int i;
  //Εμφάνιση στοιχείων πίνακα -
  //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων σε block try
     for ( i = 0; i <= a.length + 1; i++ )
       try {
          System.out.println("a[" + i + "] = " + a[i]);
       }
```

```
    // Χειρισμός Εξαίρεσης - Εμφάνιση αντίστοιχου μηνύματος
    catch (IndexOutOfBoundsException obj){
      System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων ");
      System.out.println("Είδος Εξαίρεσης : " + obj);
    }
  }
}
```

Έξοδος Προγράμματος

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων Δήλωσης
Είδος Εξαίρεσης : java.lang.ArrayIndexOutOfBoundsException: 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων Δήλωσης
Είδος Εξαίρεσης : java.lang.ArrayIndexOutOfBoundsException: 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

Παρατήρηση

Για να χειριστούμε **ΟΛΕΣ** τις εξαιρέσεις που μπορεί να συμβούν μέσα σε μια εντολή επανάληψης, θα πρέπει **η εντολή που πρόκειται να προκαλέσει την εξαίρεση** να ανήκει στο `block try` **και όχι η εντολή επανάληψης**, η οποία περιέχει την εντολή που πρόκειται να προκαλέσει την εξαίρεση.

9.3 Παράδειγμα Εξαίρεσης σε μέθοδο που καλείται από τη `main()`

Στο επόμενο πρόγραμμα προσπαθούμε **με την κλήση της μεθόδου `displayA()`** να εμφανίσουμε το περιεχόμενο των θέσεων `0` μέχρι `5` ενός πίνακα, ο οποίος όμως, σύμφωνα με τη δήλωσή του (και την αρχικοποίηση) περιέχει τις θέσεις `0` μέχρι `4`, οπότε προκαλείται η εξαίρεση `ArrayIndexOutOfBoundsException` :

```
package exceptions3Method;
public class Exceptions3Method {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα  -
  //Πρόσβαση σε στοιχείο του πίνακα εκτός ορίων - Μήνυμα Λάθους
  for ( int i = 0; i <= a.length; i++ )
       System.out.println("a[" + i + "] = " + a[i]);
  }

  public static void main(String[] args) {
  int a[] = {1,2,3,4,5};

  //Κλήση μεθόδου displayA()
  displayA(a);
  }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
    at exceptions3Method.Exceptions3Method.displayA(Exceptions3Method.java:7)
    at exceptions3Method.Exceptions3Method.main(Exceptions3Method.java:14)
Java Result: 1
BUILD SUCCESSFUL (total time: 1 second)
```

9.3.1 Χειρισμός της Εξαίρεσης Μέσα στη Μέθοδο που Προκαλείται

Και σε αυτό το παράδειγμα η προσπάθεια πρόσβασης στο στοιχείο στη θέση `i` = **`a.length`** = 5 του πίνακα `a` εκτός των ορίων της δήλωσης των θέσεων του πίνακα (`0` μέχρι **`a.length - 1`**) με την εντολή στη μέθοδο **`displayA()`**

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί την εξαίρεση `ArrayIndexOutOfBoundsException`. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε στη μέθοδο `displayA()` **την εντολή που θα προκαλέσει την εξαίρεση** στο `block try` και **τις εντολές χειρισμού της εξαίρεσης** στο `block catch`, στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα :

```
package exceptiions3MethodTry;
public class Exceptiions3MethodTry {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα  - Χειρισμός Εξαίρεσης στη μέθοδο
   for ( int i = 0; i <= a.length; i++ )
    try {
       System.out.println("a[" + i + "] = " + a[i]);
    }

    catch (IndexOutOfBoundsException obj){
      System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
    }
  }

  public static void main(String[] args) {
    int a[] = {1,2,3,4,5};

    //Κλήση μεθόδου displayA()
    displayA(a);
    }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.3.2 Χειρισμός της Εξαίρεσης της Μεθόδου στη `main()`

Και σε αυτό το παράδειγμα η προσπάθεια πρόσβασης σε στοιχείο (`i` = **`a.length`** = 5) του πίνακα `a` εκτός των ορίων της δήλωσης των θέσεων του πίνακα (`0` μέχρι **`a.length - 1`**) με την εντολή στη μέθοδο **`displayA()`**

```
System.out.println("a[" + i + "] = " + a[i]);
```

προκαλεί την εξαίρεση `ArrayIndexOutOfBoundsException`. Για να τη χειριστούμε και να μην προκληθεί η διακοπή της εκτέλεσης του προγράμματος, γράφουμε **την εντολή κλήσης της μεθόδου, η οποία περιέχει τις εντολές που θα προκαλέσουν την εξαίρεση** στο `block try` και **τις εντολές χειρισμού της εξαίρεσης** στο `block catch` , στο οποίο εμφανίζουμε ένα αντίστοιχο μήνυμα :

```
package exceptions3MethodTryMain;
public class Exceptions3MethodTryMain {
  static void displayA(int a[]){
  //Εμφάνιση στοιχείων πίνακα - Χειρισμός Εξαίρεσης στη main
  for ( int i = 0; i <= a.length; i++ )
     System.out.println("a[" + i + "] = " + a[i]);
  }

  public static void main(String[] args) {
   int a[] = {1,2,3,4,5};

   //Χειρισμός Εξαίρεσης στην Κλήση της μεθόδου displayA()
   try {
    displayA(a);
   }
   catch (IndexOutOfBoundsException obj){
     System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
        }
   }
}
```

Έξοδος Προγράμματος

```
run:
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.4 Παράδειγμα 2 Εξαιρέσεων σε εντολή επανάληψης της main()

Στο επόμενο πρόγραμμα προσπαθούμε να εμφανίσουμε το περιεχόμενο των θέσεων 0 μέχρι 5 ενός πίνακα δια του αντιστοίχου δείκτη i. Η προσπάθεια πρόσβασης στο στοιχείο στη θέση i = **a.length** = 5 του πίνακα a εκτός των ορίων της δήλωσης των θέσεων του πίνακα (0 - **a.length - 1**) προκαλεί την εξαίρεση ArrayIndexOutOfBoundsException, ενώ η διαίρεση του στοιχείου a[0] δια του i = 0 προκαλεί την εξαίρεση ArithmeticException, η οποία προκαλεί και τον τερματισμό του προγράμματος με την εκτέλεση της πρώτης εντολής **System.out.println()** στην εντολή **for** :

```
package exceptionstwoexceptions;
public class ExceptionsTwoExceptions {
    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};
        //Εμφάνιση στοιχείων πίνακα
        for ( int i = 0; i <= a.length; i++ )
           System.out.println("a[" + i + "]/" + i + " = " + (a[i]/i));
     }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at exceptionstwoexceptions.ExceptionsTwoExceptions.main(ExceptionsTwoExceptions.java:7)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

9.4.1 Χειρισμός των 2 Εξαιρέσεων στη main()

Για να χειριστούμε τις 2 πιθανές εξαιρέσεις (ArithmeticException και ArrayIndexOutOfBoundsException) που μπορεί να προκύψουν πρέπει να συμπεριλάβουμε στο πρόγραμμα 2 εντολές catch εμφανίζοντας και το αντίστοιχο μήνυμα :

```
package exeptions2Catch;
public class Exeptions2Catch {

    public static void main(String[] args) {
        int a[] = {1,2,3,4,5};

        //Εμφάνιση στοιχείων πίνακα  - Χειρισμός Εξαίρεσης
        for ( int i = 0; i <= a.length; i++ )

         try {
           System.out.println("a[" + i + "]/" + i + " = " + (a[i]/i));
         }


         catch (ArithmeticException obj){
            System.out.println("Διαίρεση με το μηδέν");
         }

         catch (ArrayIndexOutOfBoundsException obj){
            System.out.println("Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων");
         }
     }
}
```

Έξοδος Προγράμματος

```
run:
Διαίρεση με το μηδέν
a[1]/1 = 2
a[2]/2 = 1
a[3]/3 = 1
a[4]/4 = 1
Πρόσβαση σε Θέση Πίνακα Εκτός Ορίων
BUILD SUCCESSFUL (total time: 1 second)
```

9.5 Η λέξη-κλειδί throws

Αν μια μέθοδος μπορεί να προκαλέσει μια εξαίρεση, την οποία δεν μπορεί ή δεν θέλουμε να τη χειριστεί η ίδια η μέθοδος, θα πρέπει να **προωθήσει** την εξαίρεση στην καλούσα μέθοδο με τον όρο `throws`. Αυτό γίνεται στην υπογραφή της μεθόδου, όπου **μετά τη λίστα των παραμέτρων** γράφεται η λέξη `throws` με τα πιθανά είδη εξαιρέσεων. Ο γενικός τύπος της υπογραφής της μεθόδου σ' αυτή την περίπτωση είναι :

```
<Τύπος_Επιστροφής> <Όνομα_Μεθόδου>(Λίστα_Παραμέτρων)
                   throws Λίστα_Εξαιρέσεων
```

9.5.1 Προώθηση Εξαίρεσης μεθόδου στην καλούσα μέθοδο `main()`

Αν θέλουμε να διαβάσουμε ένα χαρακτήρα στη `main()` καλώντας τη μέθοδο `prompt()`, θα **πρέπει** να **προωθήσουμε** με το `throws` την εξαίρεση στην είσοδο δεδομένων που μπορεί να προκύψει στη μέθοδο `prompt()`, στην καλούσα μέθοδο, τη `main()` , όπως φαίνεται στο επόμενο πρόγραμμα.

```
package exceptionsthrows;
public class ExceptionsThrows {
    public static char prompt()
                        throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args) {
      char ch;
      ch = prompt();
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Exception in thread "main" java.lang.RuntimeException: Uncompilable source
code - unreported exception java.io.IOException; must be caught or declared
to be thrown
    at exceptionsthrows.ExceptionsThrows.main(ExceptionsThrows.java:11)
Java Result: 1
BUILD SUCCESSFUL (total time: 2 seconds)
```

Παρατήρηση

Το πρόγραμμα εμφανίζει το παραπάνω μήνυμα, γιατί ενώ υπάρχει πιθανότητα να προκληθεί στη μέθοδο `prompt()` μια εξαίρεση στην είσοδο δεδομένων (`java.io.IOException`), η μέθοδος `prompt()` την προωθεί στη `main()` με την εντολή **`throws java.io.IOException`**, αλλά η καλούσα μέθοδος **δεν τη χειρίζεται**. Ο χειρισμός μπορεί να γίνει με 2 τρόπους :

- Να την **προωθήσει** και η καλούσα μέθοδος.
- Να τη **χειριστεί** η καλούσα μέθοδος με τη χρήση `try – catch`.

9.5.2 Προώθηση Εξαίρεσης μεθόδου και από την καλούσα μέθοδο `main()`

Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrows1;
public class ExceptionThrows1 {
    public static char prompt()
                        throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args)
                         throws java.io.IOException {
      char ch;
      ch = prompt();

      // Εμφάνιση Χαρακτήρα
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 9 seconds)
```

9.5.3 Χειρισμός Εξαίρεσης μεθόδου από την καλούσα μέθοδο `main()`

Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrows2;
public class ExceptionThrows2 {
    public static char prompt()
                          throws java.io.IOException{
    System.out.print("Δώσε ένα χαρακτήρα : ");
    return (char) System.in.read();
    }

    public static void main(String[] args)
                           {
      char ch;
      // Έλεγχος για Πιθανή Εξαίρεση IOException
      try {
           ch = prompt();
        }

       catch (java.io.IOException obj){
            System.out.println("Εξαίρεση σε Είσοδο Δεδομένων");
            ch = 'a';
       }

       // Εμφάνιση Χαρακτήρα
       System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 3 seconds)
```

9.5.4 Χειρισμός Εξαίρεσης μεθόδου Μέσα την καλούσα μέθοδο

Σ' αυτή την περίπτωση δε χρειάζεται να προωθήσουμε την εξαίρεση στην καλούσα μέθοδο, γιατί **τη χειρίζεται η ίδια η μέθοδος**. Το πρόγραμμα και η έξοδος θα έχουν την παρακάτω μορφή :

Πρόγραμμα

```
package exceptionthrowstry;
public class ExceptionThrowsTry {
    public static char prompt(){
    char ch = 'a';

    // Έλεγχος για Πιθανή Εξαίρεση IOException
    try {
     System.out.print("Δώσε ένα χαρακτήρα : ");
     ch = (char) System.in.read();
    }
```

```
    catch (java.io.IOException obj){
            System.out.println("Εξαίρεση σε Είσοδο Δεδομένων");
    }
    return ch;
    }

    public static void main(String[] args)
                        {
      char ch;
      // Κλήση Μεθόδου για Διάβασμα Χαρακτήρα
      ch = prompt();

      // Εμφάνιση Χαρακτήρα
      System.out.println("Δώσατε το χαρακτήρα " + ch );
    }
}
```

Έξοδος Προγράμματος

```
run:
Δώσε ένα χαρακτήρα : q
Δώσατε το χαρακτήρα q
BUILD SUCCESSFUL (total time: 3 seconds)
```

9.5.5 Ελεγμένες και Μη Ελεγμένες Εξαιρέσεις και ο Όρος `throws`

Οι εξαιρέσεις που μπορεί να συμβούν κατά την εκτέλεση ενός προγράμματος δεν είναι απαραίτητο να συμπεριλαμβάνονται στη λίστα `throws`, γιατί η `Java` θεωρεί πως είναι φυσικό να συμβούν τέτοιου είδους εξαιρέσεις, οπότε δε χρειάζεται να ελεγχθεί, αν συμβεί μια τέτοια εξαίρεση σε μια μέθοδο, ή αν θα τη χειριστεί και λέγονται **μη ελεγμένες** εξαιρέσεις. Οι πιο συνηθισμένες εξαιρέσεις αυτού του είδους είναι :

Εξαίρεση	**Σημασία**
`ArithmeticException`	Αριθμητικό Σφάλμα, όπως Διαίρεση με μηδέν
`ArrayIndexOutOfBoundsException`	Δείκτης Πίνακα εκτός ορίων
`IllegalArgumentException`	Λάθος παράμετρος στην κλήση μεθόδου
`NegativeArraySizeException`	Αρνητικό Μέγεθος Πίνακα
`NumberFormatException`	Λάθος μορφή δεδομένων

Υπάρχουν όμως και οι **ελεγμένες** εξαιρέσεις, όπως η `java.io.IOException` που **πρέπει** να συμπεριλαμβάνονται στη λίστα `throws` μιας μεθόδου, **αν δεν τις χειρίζεται η μέθοδος**.