Excepcions en Java

Què són les excepcions?

Una **excepció** (**exception**) és un esdeveniment (***event***) que es produeix durant l'execució d'un programa. L'excepció constitueix un flux alternatiu al flux principal del programa, el qual tracta una situació que no és l'habitual.

La principal utilitat de les excepcions és el tractament dels errors. Quan es produeix un error en l'execució del programa, l'aplicació llança (**throws**) una excepció, la qual ha de ser capturada i tractada (**catch**) en algun lloc del codi tancada dintre d'un bloc **try**.

Els errors i les excepcions en Java es representen amb les classes **Error** i **Exception**, les quals deriven de la classe **Throwable**. La classe Error respon a errors de la màquina virtual de Java. Per als errors del codi Java, s'utilitza la classe Exception.

Captura d'excepcions

El mecanisme de programació amb excepcions és el següent:

El fragment de codi sospitós de generar una excepció es tanca dintre d'un bloc try {}, al qual segueixen un o més blocs catch () {}, un per cada una de les excepcions que es preveu capturar. Després es pot afegir un bloc finally {} amb el tractament comú a totes les situacions i que s'executa tant si s'ha llançat alguna excepció com si no.

```
try {
   //codi que pot generar una excepció
}
catch (tipus excepcio){
   //codi per tractar l'excepció
```

```
try {
   //codi que pot generar una excepció
}
catch (tipus excepcio){
   //codi per tractar l'excepció
}
finally {
   //codi a executar en qualsevol cas
   //aquest bloc pot no ser-hi
}
```

```
try {
    //codi que pot generar una excepció
}
catch (tipus1 excepcio1){
   //codi per tractar l'excepció de tipus 1
}
catch (tipus2 excepcio2){
   //codi per tractar l'excepció de tipus 2
}
finally {
   //codi a executar en qualsevol cas
   //aquest bloc pot no ser-hi
}
```

Dintre d'un segment try es poden llançar diverses excepcions, identificades pel seu tipus. Cada bloc catch en captura un tipus. El bloc finally, opcional, conté el codi que s'ha d'executar en tots els casos. Quan es llança una excepció en el codi contingut en el bloc try (o dintre d'alguna funció invocada a partir d'aquest bloc), l'execució passa al bloc catch corresponent. Un cop executat el bloc catch, l'execució continua després del bloc catch, no torna al punt del llançament de l'excepció.

Captura d'excepcions al mateix mètode que genera l'excepció

El programa Excepcio1.java il•lustra la captura d'una excepció ArithmeticException que es produeix en executar-se una divisió per zero. L'excepció es llança a la instrucció de l'interior del bloc try i es captura al bloc catch.

```
/**
 * Excepcio1.java
 * Exemple de captura d'excepció al mateix mòdul
 * @author Jose Moreno
 * @version
 */
public class Excepcio1 {
     public static void main(String[] args) {
       int x, y, z;

       x=10;
       y=0;
       try{
          z = x/y;  //aqui es produeix una divisió per zero
       }
       catch (ArithmeticException ae) {
          System.out.println("Error en dividir: " + ae.getMessage());
       }
    }
}
```

A més de capturar excepcions específiques, podem **capturar excepcions genèriques**. Només cal afegir (ha de ser l'últim) un bloc catch (Throwable e) que capturarà qualsevol excepció, atès que es tracta de la **superclasse** de totes les excepcions.

Els blocs try es poden enniuar, de manera que els blocs més interns capturen les excepcions més lleus, mentre que els externs tracten les més greus.

Captura d'excepcions a un mètode diferent del que genera l'excepció

En aquest cas, l'excepció es produeix (és llançada) en un mètode al qual s'ha arribat mitjançant invocació de mètodes a partir d'un punt del codi que es troba a l'interior del bloc try.

Quan a l'interior d'un mètode es genera una excepció, es pot optar per capturar-la allí mateix amb el mecanisme try-catch ja vist, o bé indicar a la definició del mètode que aquest mètode llança una excepció (**throws**). Això significa que el mètode no captura l'excepció sinó que la propaga cap al mètode que l'ha invocat. Aquest mecanisme de propagació cap a mètodes més generals es pot continuar fins al main(). Si en aquest no es captura i es torna a llançar, serà la màquina virtual de Java la que tractarà l'error, situació que cal evitar perquè s'interromp el programa de forma no controlada.

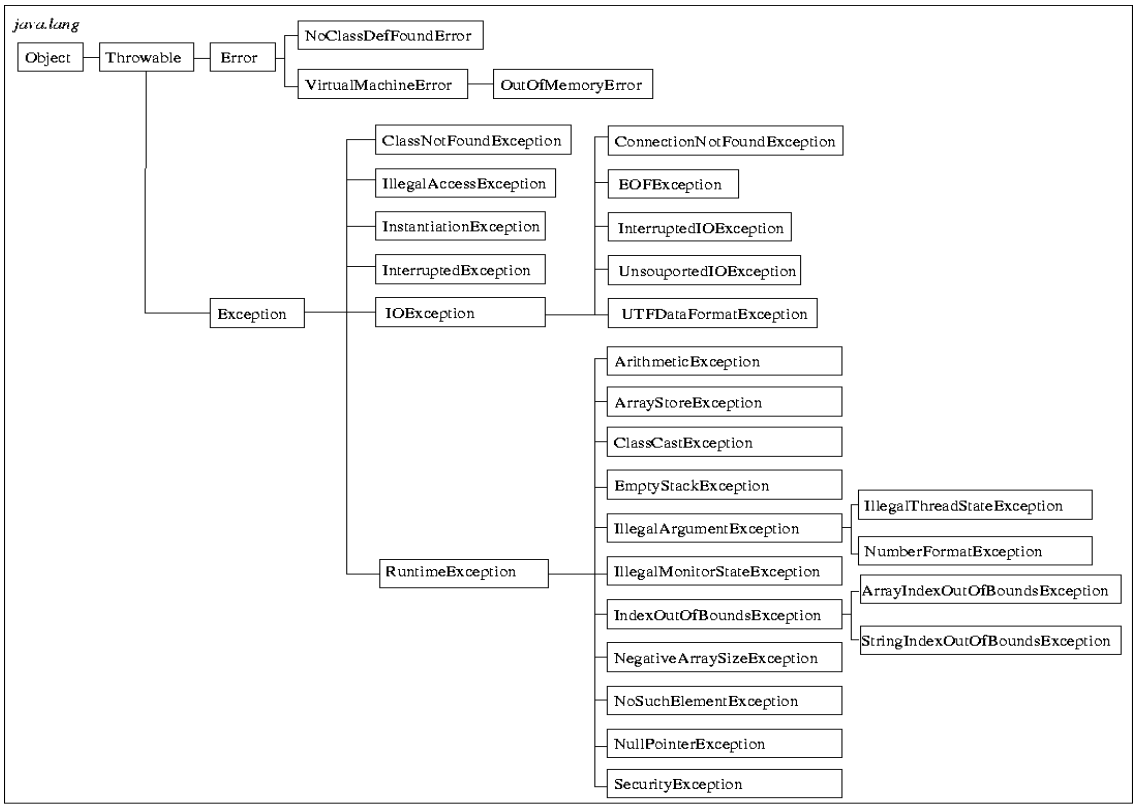
El programa Excepcio2.java il•lustra la captura d'una excepció ArithmeticException que es produeix en executar-se una divisió per zero. La excepció es llança a la instrucció de l'interior del bloc try i es captura al bloc catch.

```
/**
 * Excepcio2.java
 * Exemple de captura d'excepció a un altre mètode
 * @author Jose Moreno
 * @version
 */
public class Excepcio2 {
     public static void main(String[] args) {
       int x, y, z;

       x=10;
       y=0;
       try{
          z = f(x,y);  //aqui es produeix una divisió per zero
```

```
        }
        catch (ArithmeticException ae) {
           System.out.println("Error en dividir: " + ae.getMessage());
        }
    }
    public static int f(int a, int b) throws ArithmeticException {
     return a/b;
    }
}
```

La següent imatge representa esquemàticament la jerarquia de les classes Exception del paquet java.lang.



Les RuntimeException no necessiten ser verificades. Per tant, no és obligatori capturar-les ni tractar-les, tot i que, com a regla general, cal protegir l'execució de les nostres aplicacions contra tota possible incidència.

La resta de classes derivades de Exception han de ser verificades. Cas contrari, el compilador genera un error.

Definició d'excepcions

Derivant de la classe Exception, podem definir nous tipus d'excepció que responguin a les circumstàncies que es produeixen en el nostre codi.

El següent exemple calcula les solucions d'una equació de segon grau del tipus ax2+bx+c=0. L'entrada al programa són els coeficients a, b, c. Cal capturar l'error a=0 (l'equació no és de segon grau i no es pot aplicar la fórmula general) i l'error de discriminant negatiu (el discriminant és b2-4ac, valor que es troba dintre d'una arrel quadrada en la fórmula general).

S'ha simplificat la documentació de les classes i mètodes per estalviar espai i poder visualitzar més codi en una pantalla.

Al mètode main() es capturen les excepcions que es puguin produir en dos blocs try enniuats. Al bloc exterior es capturen els d'entrada de dades: IOException per a errors en lectura del stream d'entrada estàndard i NumberFormatException per a errors de format (l'entrada no és un nombre). Al bloc interior es capturen les excepcions específiques del funcionament del programa: PrimerGrauException per al cas que el coeficient de segon grau sigui zero i CapSolucioRealException per tractar el cas de discriminant negatiu, cas que no té cap solució real.

A la classe Eq2nGrau, el mètode solucionar() llança les excepcions PrimerGrauException i CapSolucioRealException. en cas que es produeixin. Cas contrari, calcula les solucions.

```
/**
 * Eq2nGrauTest.java
 * Programa per resoldre una equacio de segon grau
 * @author Jose Moreno
 * @version 2
 */
import java.io.*;
public class Eq2nGrauTest {
   public static void main(String[] args) {
      BufferedReader br =
         new BufferedReader(new InputStreamReader(System.in));
      //Entrada de coeficients
      System.out.println("Entra els coeficients de l'equació ax2+bx+c=0");
      try {
          System.out.print("a = ");
          double a = Double.valueOf(br.readLine()).doubleValue();
          System.out.print("b = ");
          double b = Double.valueOf(br.readLine()).doubleValue();
          System.out.print("c = ");
          double c = Double.valueOf(br.readLine()).doubleValue();
          try {
                Eq2nGrau eq2g = new Eq2nGrau(a, b, c);
                eq2g.solucionar();
                System.out.println(
                       "Solucions: "+eq2g.getSolucio(1)+", "+eq2g.getSolucio(2)
                );
          }
          catch (PrimerGrauException pg) {
                System.out.println(pg.getMessage());
          }
          catch (CapSolucioRealException csr) {
                System.out.println(csr.getMessage());
          }
      }
      catch (NumberFormatException nfe) {
         System.out.println("La dada entrada no es numerica. " +
nfe.getMessage());
      }
      catch (IOException ioe) {
         System.out.println("Error en lectura de coeficients. " + ioe.getMessage());
      }
   }
}
class Eq2nGrau {
    private double a, b, c; // coeficients de l'equacio ax2+bx+c=0
    private double [] x; // solucions de l'equacio
   public Eq2nGrau(double a, double b, double c) {
```

```
    this.a=a; this.b=b; this.c=c;
    x = new double[2];
   }
   public double getSolucio(int n) {
    return x[n-1];
   }
   public void solucionar() throws PrimerGrauException, CapSolucioRealException {
    if (a==0.0) throw new PrimerGrauException("a=0. L'equacio es de primer
grau");
    else { //l'equacio es de segon grau
         double discriminant=b*b-4*a*c;
         if (discriminant<0)
               throw new CapSolucioRealException("Discriminant negatiu. Cap
solució real");
         else { //l'equacio te solucio (discriminant no negatiu)
               double d = Math.sqrt(discriminant);
               // calcular les solucions
               x[0] = (-b+d)/(2*a);
               x[1] = (-b-d)/(2*a);
         }
    }
   }
}
class PrimerGrauException extends Exception {
    public PrimerGrauException(String s) {
         super(s);
    }
}
class CapSolucioRealException extends Exception {
    public CapSolucioRealException(String s) {
         super(s);
    }
}
```

Les excepcions es defineixen derivant de la classe Exception. Els constructors habituals de les excepcions accepten un String per mostrar el missatge corresponent a l'error, tot i que podem definir de manera diferent els constructors per acceptar altres paràmetres si ens convé. En el nostre cas, hem invocat al constructor de la classe base Exception amb **super()**.