JERARQUÍA DE EXCEPCIONES

qué son las excepciones?

Cuando un programa Java viola las restricciones semánticas del lenguaje (se produce un error) la máquina virutal Java comunica este hecho al programa **mediante una excepción**.

Muchas tipos de errores pueden provocar una excepción: un desbordamiento de memoria, un disco duro estropeado, un intento de dividir por cero o intentar acceder a un vector fuera de sus límites.

Cuando esto ocurre, la máquina virtual Java **crea un objeto de clase Exception**.

qué son las excepciones?

Ejemplo 1:

```
public class PruebaExcepciones {

   public static void main(String[] args) {
       int numero1 = 5, numero2 = 0;

       int resultado = numero1 / numero2;

       System.out.println("El resultado es: " +  resultado);
   }
}
```

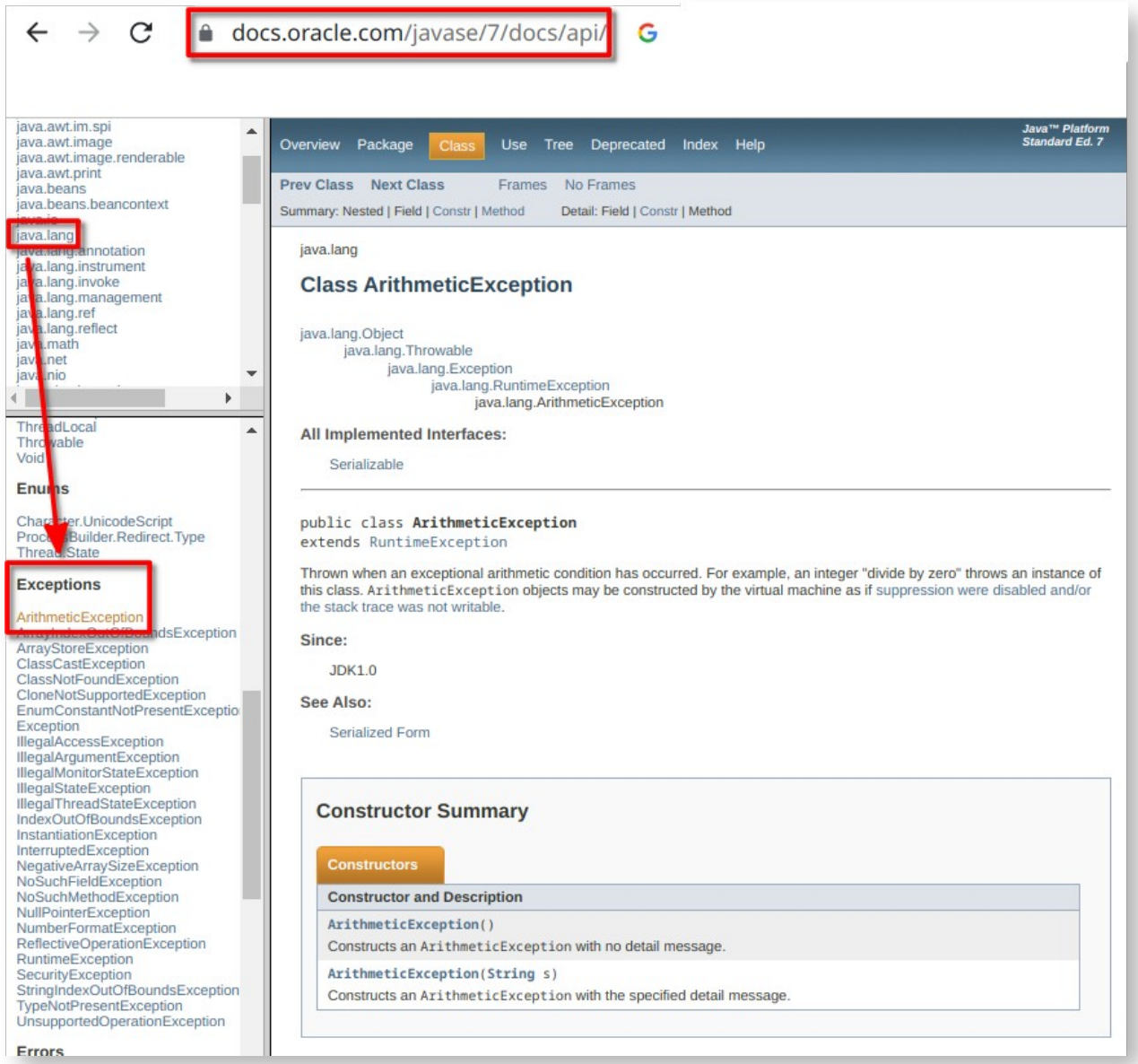
Java no detecta un error (aunque sabemos que no se podría realizar esta operación)

Java lanza esta excepción (del paquete **lang**)

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at PruebaExcepciones.main(PruebaExcepciones.java:6)
```

qué son las excepciones?

Ejemplo 1:



qué son las excepciones?

Ejemplo 1:

```
public class PruebaExcepciones {

   public static void main(String[] args) {
       int numero1 = 5, numero2 = 0;

       int resultado = numero1 / numero2;

       System.out.println("El resultado es: " +  resultado);

       System.out.println("Adiós");
   }
}
```

qué ocurre si añadimos más lineas de código?
Se ejecutarán

Muestra el mismo texto. En este ejemplo, la ejecución se queda en la línea 6.

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at PruebaExcepciones.main(PruebaExcepciones.java:6)
```

qué son las excepciones?

Ejemplo 2:

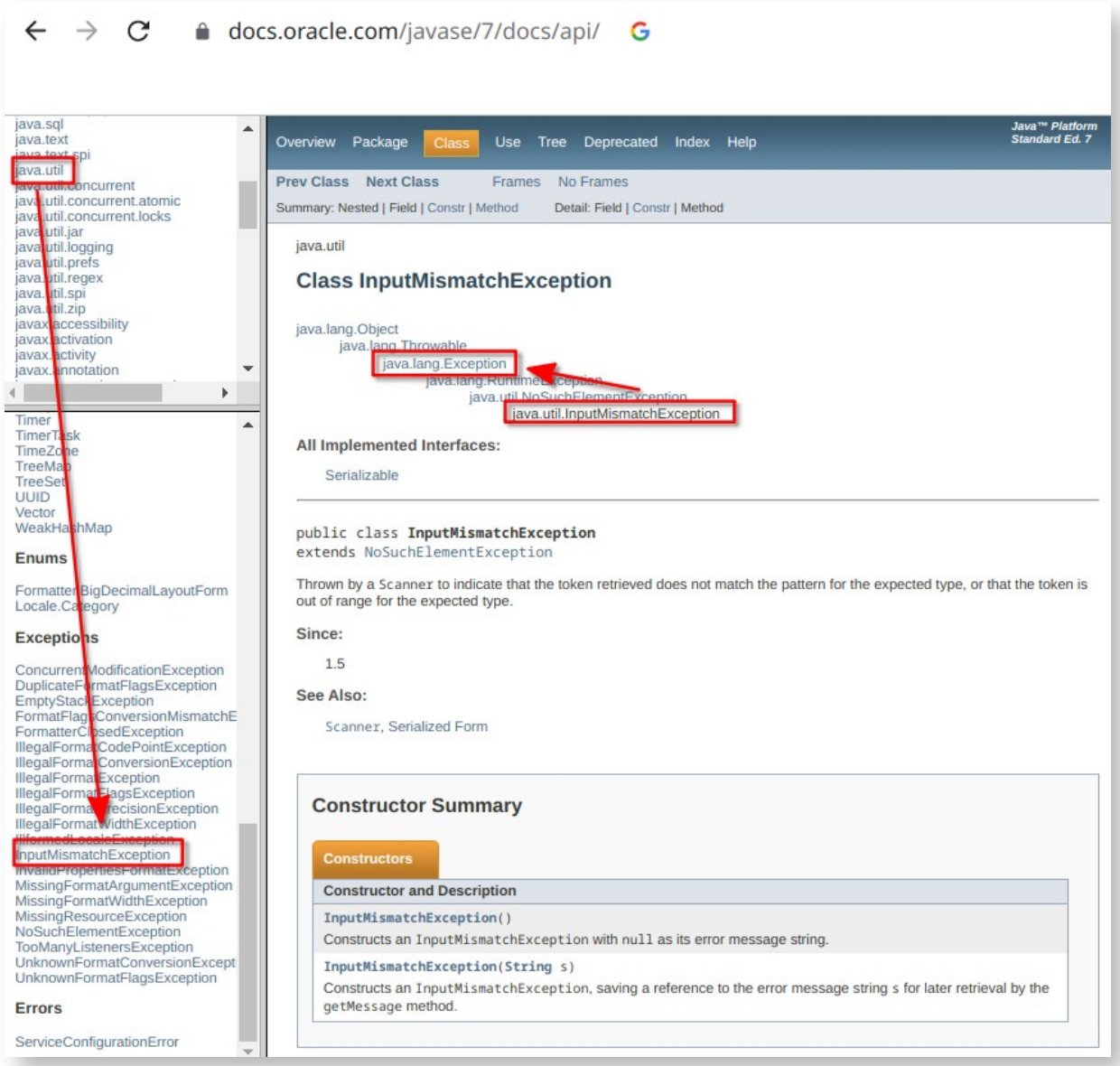
```
import java.util.Scanner;

public class PruebaExcepciones {

   public static void main(String[] args) {
       Scanner entrada = new Scanner(System.in);

       System.out.print("Introduce un número entero: ");
       int numero = entrada.nextInt();

       System.out.println(numero);
   }
}
```

qué ocurre si no introducimos un número entero?

```
Introduce un número entero: quince
Exception in thread "main" java.util.InputMismatchException
        …
                at PruebaExcepciones.main(PruebaExcepciones.java:9)
```

qué son las excepciones?

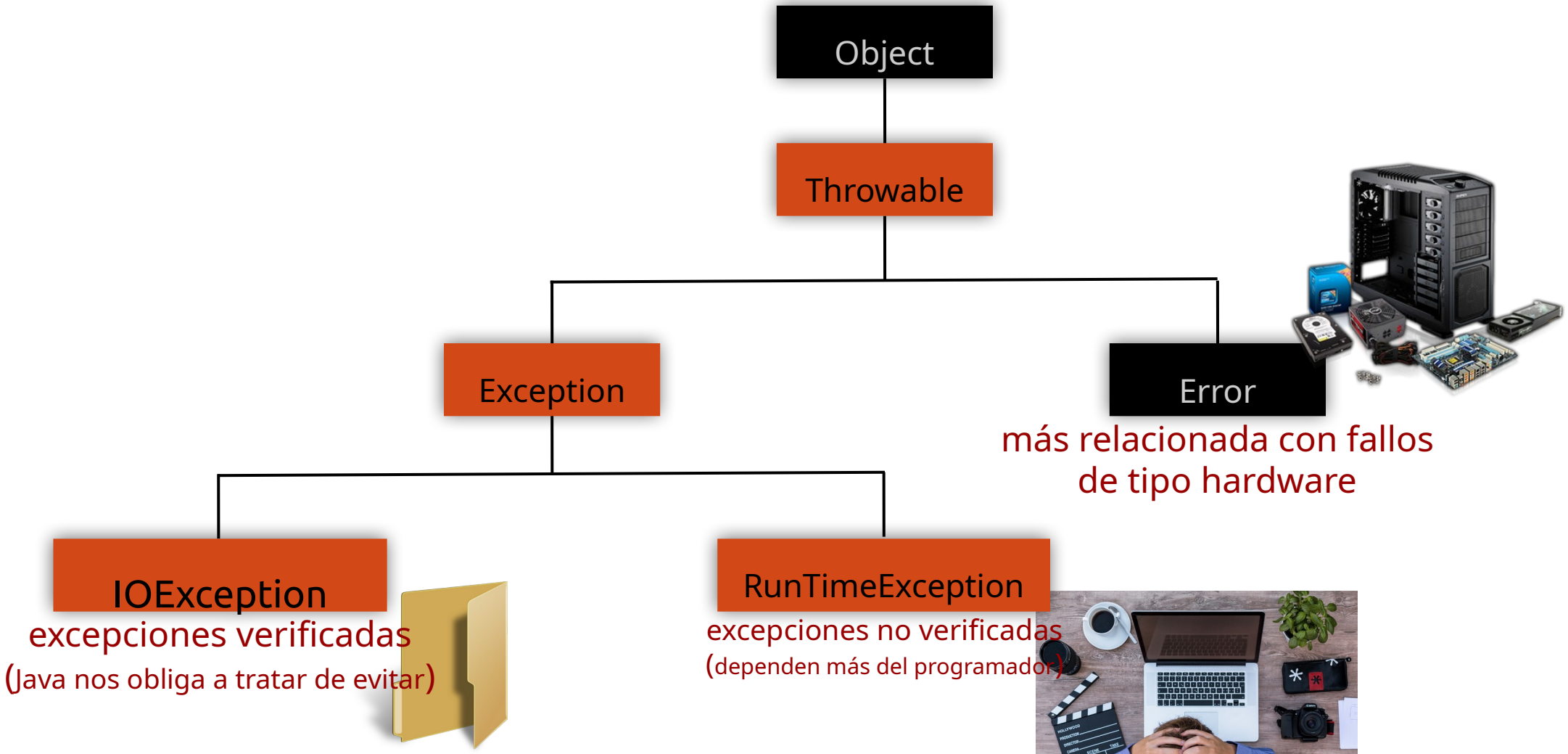
Ejemplo 2:



qué son las excepciones?

El manejo de excepciones va a permitir que el programa no se “frene”, evadiendo los diferentes errores que encuentre para que, el código que existe después de la línea de error, se pueda ejecutar sin problemas.

jerarquía de excepciones



más relacionada con fallos de tipo hardware

excepciones verificadas
(Java nos obliga a tratar de evitar)

excepciones no verificadas
(dependen más del programador)

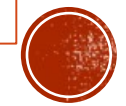
jerarquía de excepciones

Ejemplo 3 - excepción verificada:

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class PruebaExcepciones {

   public static void main(String[] args) throws FileNotFoundException, IOException {

      // Excepciones verificadas (IOException)
      BufferedReader bf = new BufferedReader(new FileReader("/home/abc/texto.txt"));
      String linea;
      while ((linea = bf.readLine()) != null){
       System.out.println(linea);
      }
   }
}
```

si el archivo existe, muestra el resultado correcto

```
esto es una prueba de excepciones no verificadas.
```



jerarquía de excepciones

Ejemplo 3 - excepción verificada:

```
...

public class PruebaExcepciones {

   public static void main(String[] args) throws FileNotFoundException, IOException {

      // Excepciones verificadas (IOException)
      BufferedReader bf = new BufferedReader(new FileReader("/home/abc/texto.txt"));
      String linea;
      while ((linea = bf.readLine()) != null){
       System.out.println(linea);
      }
   }
}
```

qué ocurre si el fichero ha sido eliminado?

```
Exception in thread "main" java.io.FileNotFoundException: /home/abc/texto.txt (No such file or
directory)
        ...
        at PruebaExcepciones.main(PruebaExcepciones.java:17)
```

jerarquía de excepciones

Ejemplo 3 - excepción verificada:

Cuando hay una excepción verificada, tenemos dos opciones:

1. Declarar la excepción que se puede dar en el método.

```
public static void main(String[] args) throws FileNotFoundException, IOException {
```

2. Capturarla con un try-catch.

```
…
    try {
          while ((linea = bf.readLine()) != null){
              System.out.println(linea);
          }
        } catch (IOException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
    }
…
```

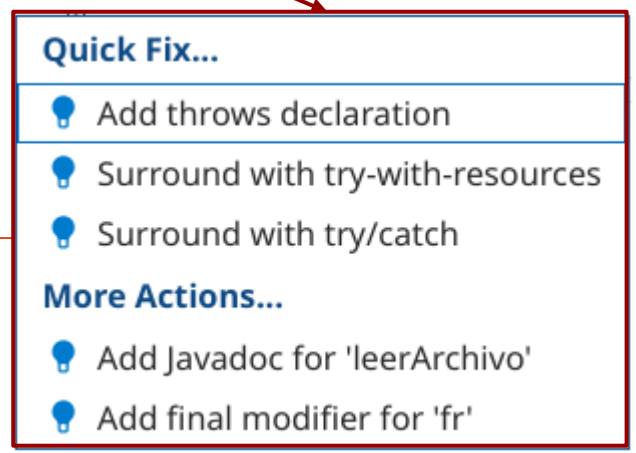
declaración de excepciones de un método

Ejemplo 4 - declaración:

```
import java.io.File;
import java.io.FileReader;

public class PruebaExcepciones {

    public void leerArchivo(){
        File archivo = new File("/home/abc/texto.txt");
        FileReader fr = new FileReader(archivo);
    }
    public static void main(String[] args) {

    }
}
```

Unhandled exception type FileNotFoundException

el siguiente error nos obliga a elegir una de estas dos opciones:
- Add throws declaration → añadir declaración
- Surround with try/catch → capturar excepción con este bloque



declaración de excepciones de un método

Ejemplo 4 - declaración:

```
import java.io.File;
import java.io.FileReader;
import java.io.FileNotFoundException;

public class PruebaExcepciones {

    public void leerArchivo()throws FileNotFoundException{
        File archivo = new File("/home/abc/texto.txt");
        FileReader fr = new FileReader(archivo);
    }
    public static void main(String[] args) {

    }
}
```

desaparece el error que aparecía en este línea

declaración de excepciones de un método

Ejemplo 4 - declaración:

Recuerda: Utilizaremos la declaración de excepciones cuando en dicho método no se quiere capturar el error, sino que dicho método se va a utilizar en otro método (en este otro ya se capturaría el error).

```
...
public class PruebaExcepciones {
   public void leerArchivo()throws FileNotFoundException{
       File archivo = new File("/home/abc/texto.txt");
       FileReader fr = new FileReader(archivo);
   }
   public void leerArchivo2(){
       leerArchivo();
   }
   public static void main(String[] args) {
   }
}
```

try catch para excepciones verificadas

Ejemplo 5 - try/catch:

```
...
public class PruebaExcepciones {
   public static void leerArchivo()throws FileNotFoundException{
       File archivo = new File("/home/abc/texto.txt");
       FileReader fr = new FileReader(archivo);
   }
   public static void leerArchivo2(){
          try {
               leerArchivo();
          } catch (FileNotFoundException ex){
               JOptionPane.showMessageDialog(null, "archivo no encontrado");
          }
   }
   public static void main(String[] args) {
          leerArchivo2();
   }
}
```

lanza un objecto (vamos a poner ex) de la clase FileNotFoundException

try catch para excepciones NO verificadas

Ejemplo 6 - try/catch excepciones NO verificadas:

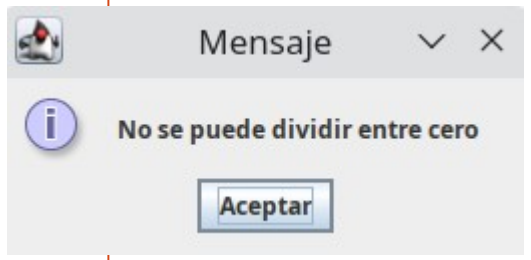
```
...
public class PruebaExcepciones {
   public void operaciones(){
       int numero1=4, numero2=0;
       int resultado = numero1/numero2;
       System.out.println(resultado);
   }
   public void operaciones2(){
       operaciones();
       System.out.println("programa terminado");
   }
   public static void main(String[] args) {
       PruebaExcepciones prueba = new PruebaExcepciones();
       prueba.operaciones2();
   }
}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at PruebaExcepciones.main(PruebaExcepciones.java:13)
```

try catch para excepciones NO verificadas

Ejemplo 6 - try/catch excepciones NO verificadas:

```
import javax.swing.JOptionPane;
public class PruebaExcepciones {
   public void operaciones(){
       int numero1=4, numero2=0;
       int resultado = numero1/numero2;
       System.out.println(resultado);
   }
   public void operaciones2(){
       try{
           operaciones();
       } catch (ArithmeticException ex){
           JOptionPane.showMessageDialog(null, "No se puede dividir entre cero");
       }
       System.out.println("programa terminado");
   }
   public static void main(String[] args) {
       PruebaExcepciones prueba = new PruebaExcepciones();
       prueba.operaciones2();
   }
}
```



throws en excepciones personalizadas

Ejemplo 7

Primero, crearemos una clase Excepcion0 que, como todas las excepciones, debe derivar de la clase Exception:

```
public class Excepcion0 extends Exception{
   public Excepcion0(){
       super("Se ha introducido el número 0");
   }
}
```

throws en excepciones personalizadas

Ejemplo 7 mediante declaración

```
import java.util.Scanner;
public class PruebaExcepciones {
   private int numero;
   private Scanner entrada;

   public void introducirNumeros() throws Excepcion0{
       entrada = new Scanner(System.in);
       do{
           System.out.println("introduce un número: ");
           numero = entrada.nextInt();
           if(numero == 0){  //excepción personalizada
               throw new Excepcion0();
           }
       }while(numero != 0);
   }
   public static void main(String[] args) throws Excepcion0 {
       PruebaExcepciones prueba = new PruebaExcepciones();
       prueba.introducirNumeros();
   }
}
```

tendríamos dos opciones:

1) si la **declaramos**, al ejecutar el programa mostraría el error de excepción personalidado:

```
Exception in thread "main" Excepcion0: Se ha introducido el número 0
        at PruebaExcepciones.introducirNumeros(PruebaExcepciones.java:14)
        at PruebaExcepciones.main(PruebaExcepciones.java:20)
```

throws en excepciones personalizadas

Ejemplo 7 mediante captura try/catch

```
import java.util.Scanner;
public class PruebaExcepciones {
   private int numero;
   private Scanner entrada;

   public void introducirNumeros() throws Excepcion0{
       entrada = new Scanner(System.in);
       do{
           System.out.println("introduce un número: ");
           numero = entrada.nextInt();
           if(numero == 0){  //excepción personalizada
               throw new Excepcion0();
           }
       }while(numero != 0);
   }
   public static void main(String[] args) {
       PruebaExcepciones prueba = new PruebaExcepciones();
       try {
           prueba.introducirNumeros();
       } catch (Excepcion0 e) {
           System.out.println("has introducido el número cero");
       }
       }
    }
```

tendríamos dos opciones:

2) si la **capturamos**, mejor opción en este caso, debemos introducir el bloque try/catch: