

EXCEPCIONES MÁS COMUNES

CLASE DE EXCEPCIÓN	USO
ArithmeticException	Errores en operaciones aritméticas
ArrayIndexOutOfBoundsException	Índice de array fuera de los límites
ClassCastException	Intento de convertir a una clase incorrecta
IllegalArgumentException	Argumento ilegal en la llamada a un método
IndexOutOfBoundsException	Índice fuera de colección
NegativeArraySizeException	Tamaño de array negativo
NullPointerException	Uso de referencia nula
NumberFormatException	Formato de número incorrecto
StringIndexOutOfBoundsException	Índice usado en String fuera de los límites

CLASES DE EXCEPCIÓN

- ▶ Heredan métodos de *Throwable*.
- ▶ Constructores que incluyen la posibilidad de paso de un mensaje.
- ▶ Métodos para obtener información de la excepción.
- ▶ ***printStackTrace()*** es el método que se invoca cuando no tratamos una excepción.

<https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html>

LANZAMIENTO DE EXCEPCIONES

- ▶ Cualquier código puede lanzar excepciones (hecho por java, por nosotros o de terceros).
- ▶ Si no vamos a tratar las excepciones en un método, tenemos que indicar que se relanzarán hacia arriba (**throws**).

Si no queremos darle tratamiento a una excepción, sino que queremos delegar el tratamiento, en alguno de los métodos que nos ha invocado, lo hacemos usando throws. Throws nos permite que un método cuyo código es posible que pueda tener una excepción excepcional no lo trate, sino que lo relance hacia otro método, esto es común en clases de servicios (aplicación de escritorio o aplicación web), y la excepción la trata la clase de la que venimos, por ejemplo, una interfaz gráfica, y no en la consola.

USO DE THROWS

- ▶ Un método cuyo código puede producir excepciones puede capturarlas y tratarlas, o relanzarlas para que sea otro quien las trate.
- ▶ **throws**. Lista separada por comas de tipos de excepción.

```
public static void writeList() throws IOException {  
}
```

Podemos crear nuestras propias excepciones, cuando no dependa de java, por ejemplo, en el ejercicio que hicimos de las cuentas bancarias, si una cuenta se queda con saldo negativo, podemos crear una excepción. Extendemos de la clase exception.

EXCEPCIONES PROPIAS

- ▶ Podemos crear nuestros propios tipos, extendiendo a **Exception**.
- ▶ Nos permiten manejar nuestras propias situaciones.

```
public class SaldoNegativoException extends Exception {  
  
    public SaldoNegativoException(double saldo) {  
        super("La cuenta ha quedado en descubierto (" +  
            Double.toString(saldo) + ")");  
    }  
  
}
```

Llamamos al constructor de la clase base, con super y ponemos el mensaje que nos interese.

También podemos lanzar una instancia de una excepción con throw, sin s:

USO DE THROW

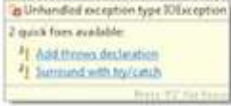
- ▶ Nos permite lanzar una excepción en un momento determinado.
- ▶ También se puede usar en el bloque catch, para tratar una excepción, pero aun así relanzarla.

```
public void sacarDinero(double cantidad) throws SaldoNegativoException {  
    saldo -= cantidad;  
    if (saldo < 0) {  
        throw new SaldoNegativoException(saldo);  
    }  
}
```

Relanzamos la excepción. Podríamos hacerlo también en un bloque catch.

Ejemplos:

```
public static void writeList() {  
    PrintWriter out = new PrintWriter(new FileWriter("OutFile.txt"));  
  
    for(int i = 0; i < 10; i++) {  
        out.println("Mensaje nº " + i);  
    }  
  
    out.close();  
}
```




Por ejemplo, vamos a escribir frases en un fichero de texto, ya eclipse nos avisa de que esto puede provocar una excepción. Nos da la opción de añadir throws, o try catch.

```
public static void writelist() throws IOException {  
    PrintWriter out = new PrintWriter(new FileWriter("OutFile.txt"));  
  
    for(int i = 0; i < 10; i++) {  
        out.println("Mensaje nº " + i);  
    }  
  
    out.close();  
}
```

Nos añade el código eclipse, y delegamos en el método que ha llamado a este.

```
*/  
public static void main(String[] args) {  
  
    writeList();  
    println("Fichero escrito correctamente");  
}  
  
public static void writeList() throws IOException {
```



Nos obliga eclipse, cuando llamamos al método que tratemos esa excepción, nos da la opción de añadir ese código y nos genera el bloque try- catch

```
try {  
    writelist();  
} catch (IOException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
}
```

Y ya podemos darle el tratamiento que queramos, por ejemplo:

```
public static void main(String[] args) {  
  
    try {  
        writelist();  
        System.out.println("Fichero escrito correctamente");  
    } catch (IOException e) {  
        System.out.println("Error al intentar abrir un fichero de texto");  
    }  
}
```

Si no lo hacemos con throws, el tratamiento de la excepción lo hacemos dentro el propio método writelist:

```
public static void writelist() {  
    PrintWriter out = null;  
    try {  
        out = new PrintWriter(new FileWriter("OutFile.txt"));  
  
        for (int i = 0; i < 10; i++) {  
            out.println("Mensaje nº " + i);  
        }  
    } catch (IOException ex) {  
        System.err.println("Error al abrir o escribir en el fichero");  
    } finally {  
        out.close();  
    }  
}
```

La ejecución sería la misma, en los dos casos, son dos formas distintas de programar la el tratamiento de la excepción.

Ejemplo de crear la propia excepción:

La cuenta bancaria, si el saldo es negativo:

```
24
25 public double getSaldo() {
26     return saldo;
27 }
28
29 public void ingresarDinero(double cantidad) {
30     saldo += cantidad;
31 }
32
33 public void sacarDinero(double cantidad) throws SaldoNegativoException {
34     saldo -= cantidad;
35     if (saldo < 0) {
36         throw new SaldoNegativoException(saldo);
37     }
38 }
39
40
```

La excepción, la hemos creado, y en este caso se imprime el mensaje de error con el saldo negativo:

```
3 public class SaldoNegativoException extends Exception {
4
5     public SaldoNegativoException(double saldo) {
6         super("La cuenta ha quedado en descubierto (" + Double.toString(saldo) + ")");
7     }
8
9 }
10
```