

Excepciones

Introducción

Un **fichero** o **archivo** es un conjunto de bits almacenados en un dispositivo como por ejemplo un disco duro. La ventaja de utilizar ficheros es que los datos que guardamos permanecen en el dispositivo aun cuando apaguemos el ordenador, es decir, no son volátiles. Los ficheros tienen un nombre y se ubican en directorios o carpetas, el nombre debe ser único en ese directorio; es decir, no puede haber dos ficheros con el mismo nombre en el mismo directorio. Por convención suelen tener diferentes extensiones por lo general de 3 letras (PDF, TXT, DOC, GIF, JAVA, DOCS, XLSX, ...) que nos permiten saber el tipo de fichero del que se trata.

Los dispositivos de soporte magnético se utilizan para guardar información de forma persistente. (Programas, datos, etc.).

La información dentro de un fichero de datos se guarda:

- Por **campos**: Son cada una de las variables miembro de un objeto.
- Por **registros**: Son cada uno de los objetos. Los registros son una agrupación de campos.

Por ejemplo, un fichero de empleados puede contener datos de los empleados de una empresa, un fichero de texto puede contener líneas de texto que se correspondan con las líneas de un papel.

La forma en que se agrupan los datos en el fichero depende completamente de la persona que los diseña.

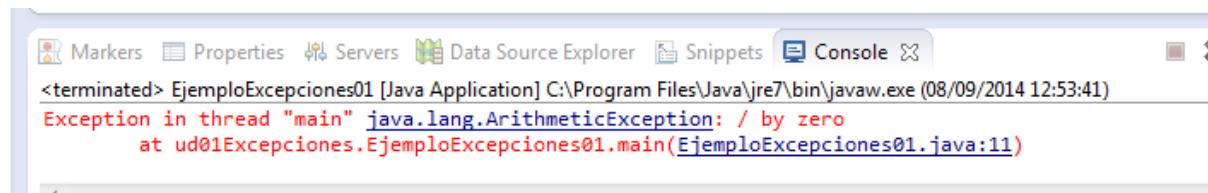
EXCEPCIONES: DETECCIÓN Y TRATAMIENTO

Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias. Cuando no es capturada por el programa, se captura por el gestor de excepciones por defecto que retorna el mensaje y detiene el programa.

Ejemplo que produce una excepción y visualiza un mensaje indicando el error:

```
package ejemplo01Excepciones;
```

```
public class EjemploExcepciones01 {  
    public static void main(final String[] args) {  
        int numerador = 10, denominador = 0, cociente;  
        cociente = numerador/denominador;  
        System.out.println("Resultado = "+cociente);  
    }  
}
```

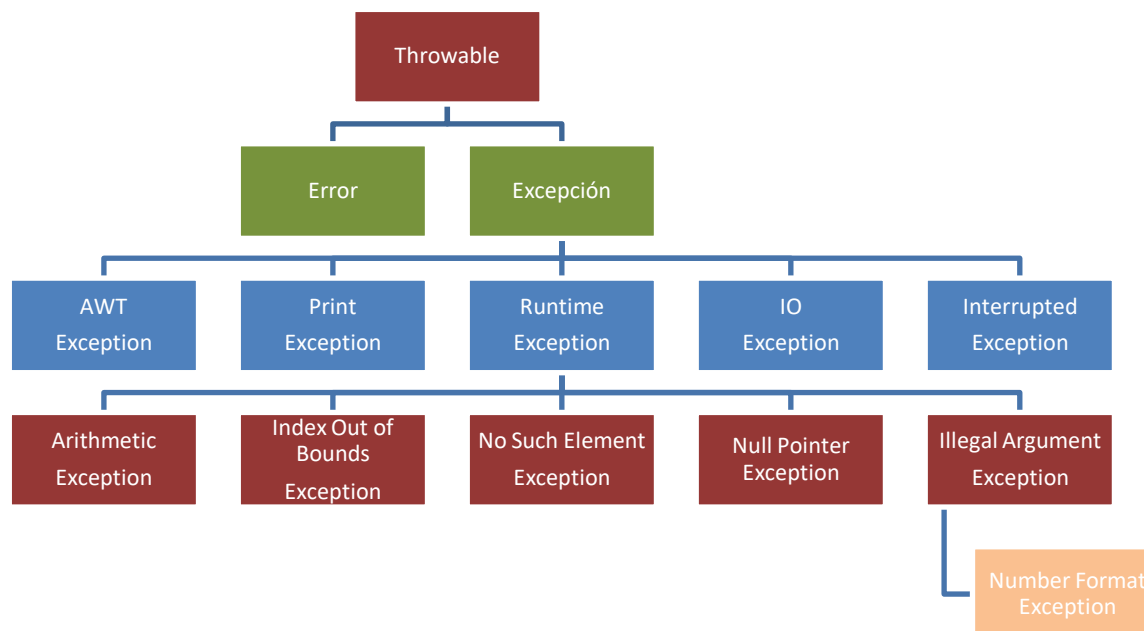


Cuando dicho error ocurre dentro de un método Java, el método crea un objeto **Exception** y lo maneja fuera, en el sistema de ejecución. El manejo de excepciones Java está diseñado

Unidad didáctica 1 MANEJO DE FICHEROS

pensando en situaciones en las que el método que detecta un error no es capaz de manejarlo, un método así lanzará una *excepción*.

Las excepciones en Java son objetos de clases derivadas de la clase base **Exception** que a su vez es una clase derivada de la clase base **Throwable**.



Captura de Excepciones

Para capturar una excepción se utiliza el bloque **try-catch**. Se encierra en un bloque **try** el código que puede generar una excepción, este bloque va seguido por uno o más bloques **catch**. Cada bloque **catch** especifica el tipo de excepción que puede atrapar y contiene un manejador de excepciones. Después del último bloque **catch** puede aparecer un bloque **finally** (opcional) que siempre se ejecuta haya ocurrido o no la excepción: se utiliza el bloque **finally** para cerrar ficheros o liberar otros recursos del sistema después de que ocurra una excepción:

```
try{
    //código que genera la excepción
}catch (Excepcion1 e1){
    //manejo de la excepción 1
}catch (Excepcion2 e2){
    //manejo de la excepción 1
}

//etc.....
finally{
    se ejecuta después de try o catch
}
```

El siguiente ejemplo captura el error de dividir por cero y muestra dos mensajes uno del programador y el otro propio de Java

Unidad didáctica 1 MANEJO DE FICHEROS

```
package ejemplo01Excepciones;
public class EjemploExcepciones01B {
    public static void main(final String[] args) {
        int numerador = 10, denominador = 0, cociente;
        try {
            cociente = numerador/denominador;
            System.out.println("Resultado = "+cociente);
        } catch (ArithmeticException ae) {
            System.out.println("Error de ejecución");
            ae.printStackTrace();
        }
    }
}
```

El siguiente ejemplo utiliza la cláusula **finally** que se ejecuta siempre, es decir, si el programa ejecuta el código **try** y no lanza la excepción pasa a **finally**, en el caso de que se produzca la excepción también salta a **finally**

```
package ejemplo01Excepciones;
/*
 * Ejemplo que lanza una excepción al intentar dividir por 0
 * Añadimos las cláusulas catch y finally
 */
public class EjemploExcepciones01C {
    public static void main(final String[] args) {
        int numerador = 10, denominador = 0, cociente;
        try {
            cociente = numerador/denominador;
            System.out.println("Resultado = "+cociente);
        } catch (ArithmeticException ae) {
            System.out.println("Error de ejecución");
            ae.printStackTrace();
        } finally {
            System.out.println("Fin del programa");
        }
    }
}
```

El siguiente ejemplo captura 3 excepciones que se pueden producir. Cuando se encuentra el primer error se produce un salto al bloque **catch** que maneja dicho error; en este caso al encontrar la sentencia de asignación `arraynum[10] = 20;` se lanza la excepción **ArrayIndexOutOfBoundsException** (ya que el array está definido para 4 elementos y se da valor al elemento de la posición 10) donde se ejecutan las instrucciones indicadas en el bloque, las sentencias situadas debajo de la que causó el error dentro del bloque **try** no se ejecutarán:

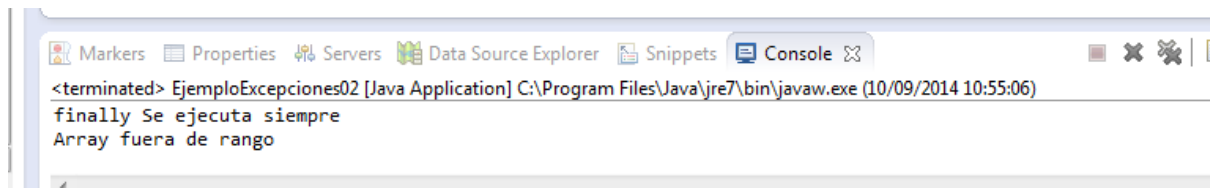
Unidad didáctica 1 MANEJO DE FICHEROS

```
package ejemplo01Excepciones;
public class Ejemplo02Excepciones {
    public static void main(String[] args) {
        /*
         * El ejemplo captura 3 excepciones que se pueden producir.
         Cuando se encuentra
         * el primer error se produce un salto al bloque catch que
         maneja dicho error;
         * en este caso al encontrar la sentencia de asignación
         arraynum[10] = 20;
         * se lanza la excepción ArrayIndexOutOfBoundsException (ya
         que el array
         * está definido para 4 elementos y se da valor al elemento
         de la posición 10)
         */
        String cad1 = "20", cad2 = "0", mensaje;
        int numerador, denominador, cociente;
        int[] arraynum = new int[4];

        try{
            //codigo que puede producir errores
            arraynum[10] = 20; //sentencia que produce la
            excepcion

            numerador = Integer.parseInt(cad1); //no se ejecuta
            denominador = Integer.parseInt(cad2); //no se ejecuta
            cociente = numerador/denominador; //no se ejecuta
            mensaje = String.valueOf(cociente); //no se ejecuta
        }catch(NumberFormatException nfe){
            mensaje = "Caracteres no numéricos";
        }catch (ArithmeticException ae) {
            mensaje = "División por cero";
        }catch (ArrayIndexOutOfBoundsException aio) {
            mensaje = "Array fuera de rango";
        }finally{
            System.out.println("finally Se ejecuta siempre");
        }
        System.out.println(mensaje); //si se ejecuta
    } // fin main
} // fin clase
```

El programa muestra la siguiente pantalla:



Unidad didáctica 1 MANEJO DE FICHEROS

Para capturar una excepción genérica utilizamos la clase **Exception**. Si se usa habrá que ponerla al final de la lista de manejadores para evitar que los manejadores que vienen después se ignoren, por ejemplo, el siguiente código maneja excepciones, si se produce alguna para la que no se haya definido manejador será capturada por Exception:

```
try{
    //codigo que puede producir errores
}catch (Exception e) {
    //tratamiento si se produce cualquier otra excepción
}finally{
    //se ejecuta haya o no excepción
}
```

Ejemplo que utiliza la clase Exception para capturar todo tipo de errores

```
package ejemplo01Excepciones;
```

```
public class EjemploExcepciones03 {
    public static void main(String[] args) {
        /*
         * El ejemplo captura 3 excepciones que se pueden producir.
         * Si utilizamos la clase Exception recoge todo tipo de
Excepciones
         */
        String cad1 = "20", cad2 = "0", mensaje;
        int numerador, denominador, cociente;
        int[] arraynum = new int[4];

        try{
            //codigo que puede producir errores
            arraynum[10] = 20; //sentencia que produce la
excepcion

            numerador = Integer.parseInt(cad1); //no se ejecuta
            denominador = Integer.parseInt(cad2); //no se ejecuta
            cociente = numerador/denominador; //no se ejecuta
            mensaje = String.valueOf(cociente); //no se ejecuta
        }catch(Exception e){
            mensaje = "Error de ejecucion";
            e.printStackTrace();

        }finally{
            System.out.println("finally Se ejecuta siempre");
        }
        System.out.println(mensaje); //si se ejecuta
    } // fin main
}
```

Unidad didáctica 1 MANEJO DE FICHEROS

Las clases para captar excepciones tienen una jerarquía, la clase más genérica deberá ir en el último **catch**

```
try{
    //codigo que puede producir errores
}catch(NumberFormatException nfe){
    //tratamiento de la excepción
}catch (ArithmeticException ae) {
    //tratamiento de la excepción
}catch (ArrayIndexOutOfBoundsException aio) {
    //tratamiento de la excepción
}catch (Exception e) {
    //tratamiento si se produce cualquier otra excepción
}finally{
    //se ejecuta haya o no excepción
}
```

Ejemplo de la jerarquía de excepciones

```
package ejemplo01Excepciones;
```

```
public class EjemploExcepciones04 {
    public static void main(String[] args) {
        /*
         * El ejemplo Jerarquia de excepciones
         */
        String cad1 = "20", cad2 = "0", mensaje = null;
        int numerador, denominador, cociente;
        int[] arraynum = new int[4];

        try{
            //codigo que puede producir errores
            arraynum[10] = 20; //sentencia que produce la
            //excepcion

            numerador = Integer.parseInt(cad1); //no se ejecuta
            denominador = Integer.parseInt(cad2); //no se ejecuta
            cociente = numerador/denominador; //no se ejecuta
            mensaje = String.valueOf(cociente); //no se ejecuta
        }catch(NumberFormatException nfe){
            mensaje = "Caracteres no numéricos";
        }catch (ArithmeticException ae) {
            mensaje = "División por cero";
        }catch (ArrayIndexOutOfBoundsException aio) {
            mensaje = "Array fuera de rango";
        }catch(Exception e) {
            e.printStackTrace();
        }finally{
            System.out.println("finally Se ejecuta siempre");
        }
    }
}
```

Unidad didáctica 1 MANEJO DE FICHEROS

```
        System.out.println(mensaje); //si se ejecuta
    } // fin main
}
```

Para obtener más información sobre la excepción se puede llamar a los métodos de la clase base Throwable, algunos son:

Método	Función
String getMessage()	Devuelve la cadena de error del objeto
String getLocalizedMessage()	Crea una descripción local de este objeto
String toString()	Devuelve una breve descripción del objeto
void printStackTrace() o, printStackTrace(PrintStream) o printStackTrace(PrintWriter)	Imprime el objeto y la traza de la pila de llamadas lanzada.

El siguiente ejemplo utiliza dichos métodos:

```
package ejemplo01Excepciones;
```

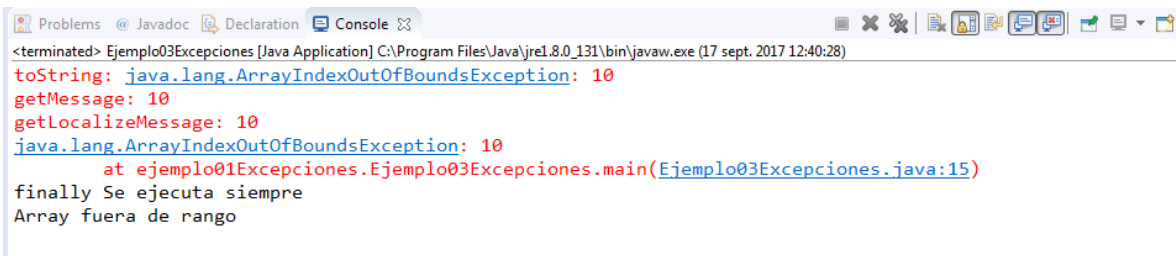
```
public class EjemploExcepciones05 {
    /*
     * El ejemplo utiliza los métodos de la clase Throwable
     */

    public static void main(final String[] args) {
        String cad1 = "2a0", cad2 = "0", mensaje;
        int numerador, denominador, cociente;
        int[] arraynum = new int[4];
        try{
            //codigo que puede producir errores

            arraynum[10] = 20; //sentencia que produce la
excepcion

        }catch (ArrayIndexOutOfBoundsException aio) {
            mensaje = "Array fuera de rango";
            System.err.println("toString: " +aio.toString());
            System.err.println("getMessage: " +aio.getMessage());
            System.err.println("getLocalizeMessage: "
+aio.getLocalizedMessage());
            aio.printStackTrace();
        }finally{
            System.out.println("finally Se ejecuta siempre");
        }
    }
}
```

Muestra la siguiente salida al ejecutarse:



```
<terminated> Ejemplo03Excepciones [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (17 sept. 2017 12:40:28)
toString: java.lang.ArrayIndexOutOfBoundsException: 10
getMessage: 10
getLocalizedMessage: 10
java.lang.ArrayIndexOutOfBoundsException: 10
    at ejemplo01Excepciones.Ejemplo03Excepciones.main(Ejemplo03Excepciones.java:15)
finally Se ejecuta siempre
Array fuera de rango
```

Una sentencia **try** puede estar dentro de otra sentencia **try**. Si la sentencia **try** interna no tiene un manejador **catch**, se busca el manejador en las sentencias **try** externas.

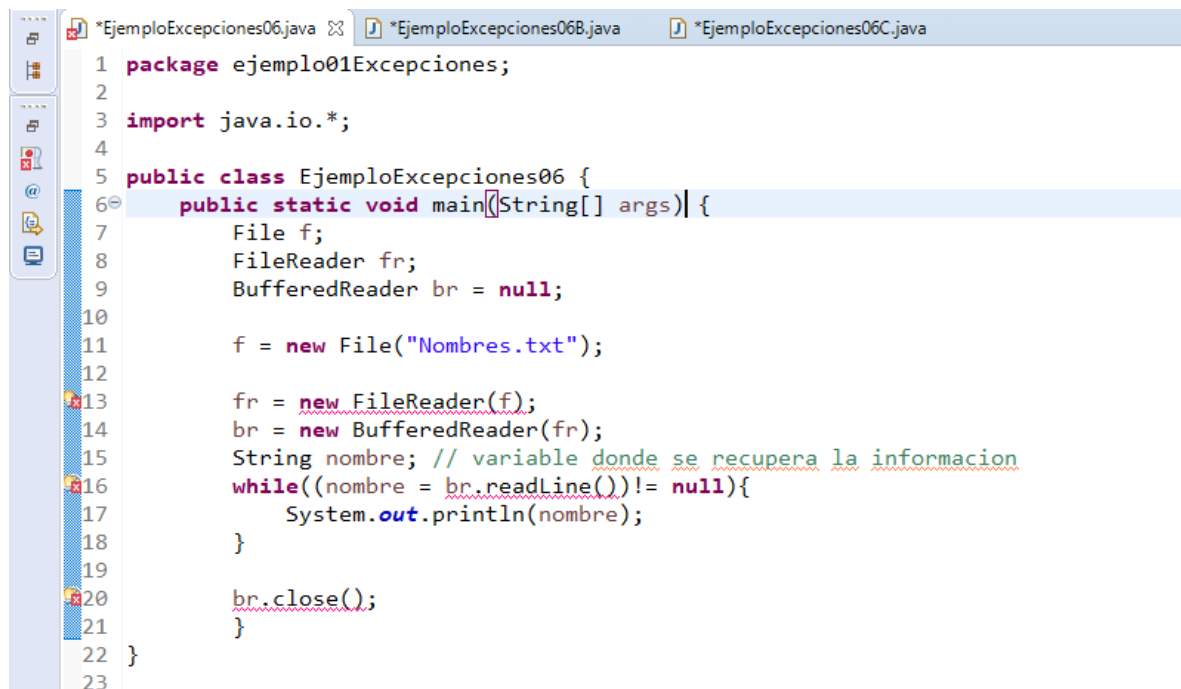
Especificar Excepciones

Para especificar excepciones utilizamos la palabra **throws**, seguida de la lista de todos los tipos de excepciones potenciales, si un método decide no gestionar una excepción (mediante **try-catch**), debe especificar que puede lanzar una excepción. El siguiente ejemplo indica que el método **main()** puede lanzar las excepciones **IOException** y **ClassNotFoundException**:

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

Los métodos que pueden lanzar excepciones, deben saber cuáles son esas excepciones en su declaración. Una forma típica de saberlo es compilando el programa. Por ejemplo, si al fichero **EjemploLecturaObjetos02.java** le quitamos la cláusula **throws** del método **main()** y el bloque **try-catch** al compilar aparecerán errores:

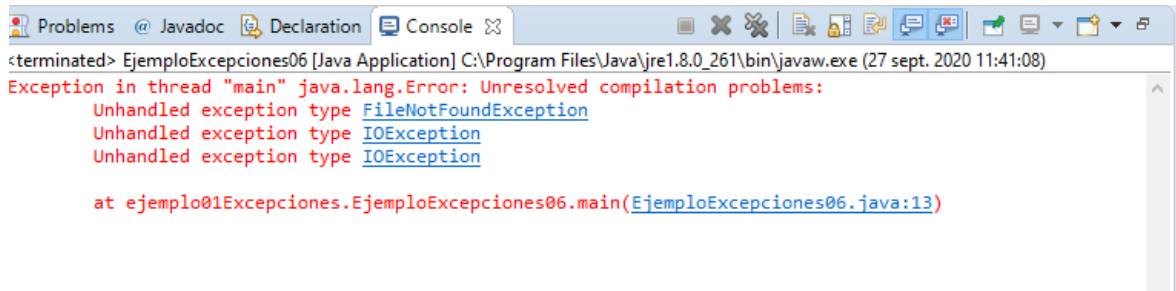
Se producen errores de excepciones no declaradas.



```
1 package ejemplo01Excepciones;
2
3 import java.io.*;
4
5 public class EjemploExcepciones06 {
6     public static void main(String[] args) {
7         File f;
8         FileReader fr;
9         BufferedReader br = null;
10
11         f = new File("Nombres.txt");
12
13         fr = new FileReader(f);
14         br = new BufferedReader(fr);
15         String nombre; // variable donde se recupera la informacion
16         while((nombre = br.readLine()) != null){
17             System.out.println(nombre);
18         }
19
20         br.close();
21     }
22 }
23
```

Si lo ejecutamos nos muestra la siguiente pantalla

Unidad didáctica 1 MANEJO DE FICHEROS



Para solucionarlo podemos añadirle al main el **throws** correspondiente:

```
package ejemplo01Excepciones;

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

public class EjemploExcepciones06B {
    public static void main(String[] args) throws IOException {
        File f;
        FileReader fr;
        BufferedReader br = null;

        f = new File("Nombres.txt");
        fr = new FileReader(f);
        br = new BufferedReader(fr);
        String nombre; // variable donde se recupera la
        informacion

        while((nombre = br.readLine()) != null){
            System.out.println(nombre);
        }

    }
}
```

O bien manejando excepciones dentro del bloque try-catch quedaría:

```
package ejemplo01Excepciones;

import java.io.*;

public class EjemploExcepciones06C {
    public static void main(String[] args) {
        File f;
        FileReader fr;
        BufferedReader br = null;
        try{
            f = new File("Nombres.txt");
```

Unidad didáctica 1 MANEJO DE FICHEROS

informacion

```
fr = new FileReader(f);
br = new BufferedReader(fr);
String nombre; // variable donde se recupera la

while((nombre = br.readLine())!= null){
    System.out.println(nombre);
}

}catch(FileNotFoundException fn){
    System.out.println("No se encuentra el fichero");
}catch(IOException ioe){
    System.out.println("Error de L/E");
}finally{
    try{
        System.out.println("Fin del programa");
        br.close();
    }catch(IOException ioe){
        System.out.println("Error al cerrar el fichero");
    }
}
}
```

Se puede consultar la API de Java <http://docs.oracle.com/javase/8/docs/api/> para ver las excepciones que se pueden producir en los diferentes paquetes.

