

4 Ejercicios java

4.29 Ejecuta una división por 0 metiendo excepciones

En un programa java puede existir algún tipo de problema o error durante la ejecución del mismo. Cuando esto sucede, se lanza una excepción. Puede ser una división entre 0, disco duro lleno, una posición de vector que no existe, hacer un cast inválido....

Cuando un programa lanza una excepción la ejecución del programa no continúa.

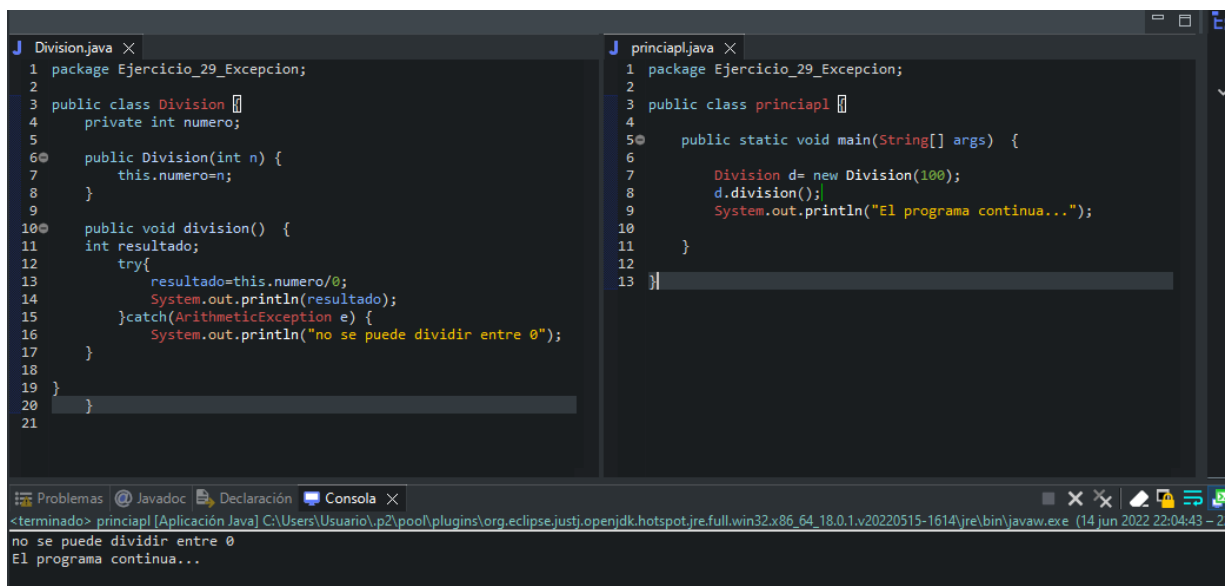
Para controlar una excepción y continuar con la ejecución:

Try-catch-finally

Para indicar que se puede lanzar una excepción:

Throws

Las excepciones se pueden extender para crear excepciones propias y lanzarlas.

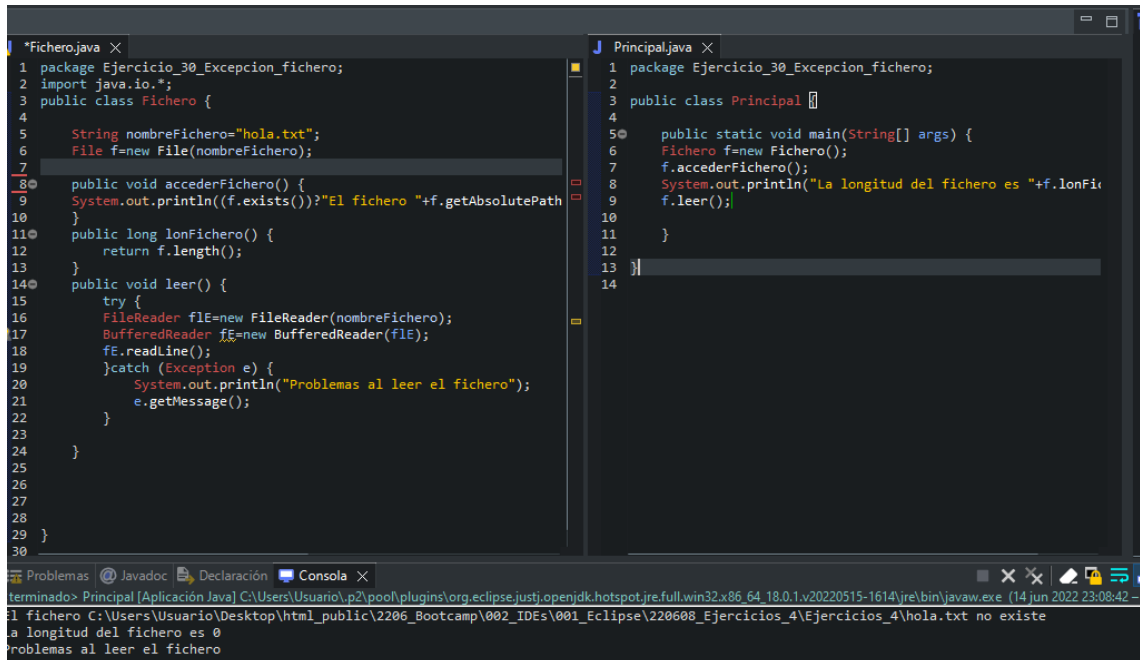


```
Division.java
1 package Ejercicio_29_Excepcion;
2
3 public class Division {
4     private int numero;
5
6     public Division(int n) {
7         this.numero=n;
8     }
9
10    public void division() {
11        int resultado;
12        try{
13            resultado=this.numero/0;
14            System.out.println(resultado);
15        }catch(ArithmeticException e) {
16            System.out.println("no se puede dividir entre 0");
17        }
18    }
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
principi1.java
1 package Ejercicio_29_Excepcion;
2
3 public class principi1 {
4
5     public static void main(String[] args) {
6
7         Division d= new Division(100);
8         d.division();
9         System.out.println("El programa continúa...");
10    }
11
12
13 }
```

```
<terminado> principi1 [Aplicación Java] C:\Users\Usuario\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\javaw.exe (14 jun 2022 22:04:43 - 2
no se puede dividir entre 0
El programa continúa...
```

4.30 Crea una clase que contenga un método que lea de un fichero que no existe. Haz distintas pruebas a llamar a ese método.



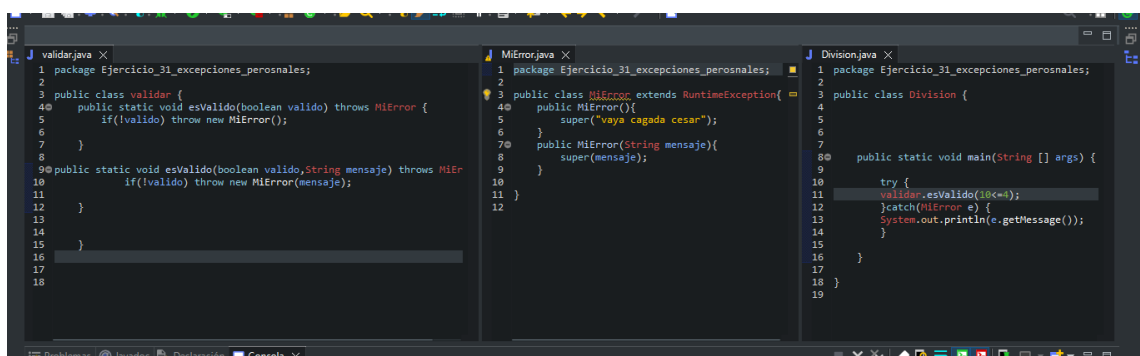
```
1 package Ejercicio_30_Excepcion_fichero;
2 import java.io.*;
3 public class Fichero {
4
5     String nombreFichero="hola.txt";
6     File f=new File(nombreFichero);
7
8     public void accederFichero() {
9         System.out.println((f.exists())?"El fichero "+f.getAbsolutePath()
10     }
11     public long lonFichero() {
12         return f.length();
13     }
14     public void leer() {
15         try {
16             FileReader file=new FileReader(nombreFichero);
17             BufferedReader fr=new BufferedReader(file);
18             fr.readLine();
19         }catch (Exception e) {
20             System.out.println("Problemas al leer el fichero");
21             e.getMessage();
22         }
23     }
24 }
25
26
27
28
29 }
30
```

```
1 package Ejercicio_30_Excepcion_fichero;
2
3 public class Principal {
4
5     public static void main(String[] args) {
6         Fichero f=new Fichero();
7         f.accederFichero();
8         System.out.println("La longitud del fichero es "+f.lonFichero());
9         f.leer();
10    }
11
12 }
13
14
```

Problemas al leer el fichero

4.31 Crea una excepción propia, que extienda de una existente (puede ser solo crear los constructores) y pruebas de lanzar ese tipo de excepción sobre la clase del ejercicio 30 (podéis duplicar la clase si no queréis tocar la del 30)

El programador java puede crear sus propias clase de excepciones



```
1 package Ejercicio_31_excepciones_perosnales;
2
3 public class validar {
4
5     public static void esValido(boolean valido) throws MiError {
6         if(!valido) throw new MiError();
7     }
8
9     public static void esValido(boolean valido,String mensaje) throws MiError {
10         if(!valido) throw new MiError(mensaje);
11     }
12 }
13
14
15
16
17
18
```

```
1 package Ejercicio_31_excepciones_perosnales;
2
3 public class MiError extends RuntimeException {
4     public MiError() {
5         super("vaya cagada cesar");
6     }
7     public MiError(String mensaje){
8         super(mensaje);
9     }
10 }
11
12
```

```
1 package Ejercicio_31_excepciones_perosnales;
2
3 public class Division {
4
5     public static void main(String [] args) {
6
7         try {
8             validar.esValido(10<=4);
9         }catch (MiError e) {
10             System.out.println(e.getMessage());
11         }
12     }
13 }
14
15
16
17
18
19
```

4.32 Crea un método que lea de un fichero y muestra su contenido por pantalla

La vocación del lenguaje java de ser independiente del sistema operativo supone una dificultad añadida en los procesos de escritura y lectura de ficheros , sobre todo en aplicaciones que acceden a ficheros a través de su URL.

La escritura y lectura de ficheros se basa en flujos de datos, sobre estos flujos se puede leer y escribir datos, los cuales son el canal de comunicación entre el programa en la memoria del ordenador y el fichero de soporte (disco) de almacenamiento. Los flujos se pueden clasificar en flujos de bytes (clases `InputStream` y `OutputStream`) y flujos de caracteres (clases `Reader` y `Writer`). Existen otras clases que sin conectar directamente con el origen de datos, por ejemplo un fichero, permiten transformar los datos por las clases generadoras de flujos. A estas clases se les denomina filtros.

El método general de lectura en ficheros consiste en enlazar un flujo y un filtro, creando un canal completo de transferencia de datos,

LAS CLASES READ Y WRITER

Son las clases abstractas de las que derivan las clases que permiten crear flujos de caracteres `Unicode(16bits)` de lectura y escritura en ficheros, las siguientes son las clases más usadas para la lectura y escritura derivadas de `Reader` y `Writer`:

`FileReader` permite crear flujos de bytes procedente de un fichero de bytes

`FileWriter` permite crear flujos de bytes para enviar a un fichero

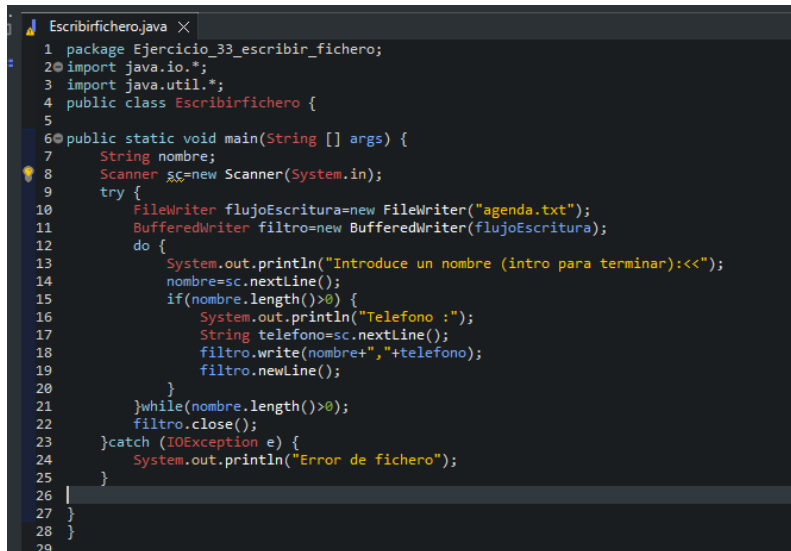
`BufferedReader` Permite crear un flujo de entrada de datos con área de almacenamiento (buffer) con lo que permite la lectura de datos mayores de un byte;

`BufferedWriter` Permite crear un flujo de envío de datos con área de almacenamiento (buffer);

```
LeerFicheroexto.java X
1 package Ejercicio_32_LeerFichero;
2 import java.io.*;
3 public class LeerFicheroexto {
4
5
6
7 public static void main(String[] args) {
8     String contenido="";
9
10    try {
11        FileReader flujo =new FileReader("Agenda.txt");
12        BufferedReader filtro=new BufferedReader(flujo);
13
14        while(contenido!= null) {
15            contenido=filtro.readLine();
16            if(contenido!=null) {
17                int pos=contenido.indexOf(",");
18                String nombre=contenido.substring(0, pos);
19                String telefono=contenido.substring(pos+1);
20                System.out.print("nombre :"+nombre);
21                System.out.println(" Telefono: "+telefono);
22            }
23        }
24        filtro.close();
25    }catch (IOException e) {
26        System.out.println("error de fichero");
27    }
28
29
30
31
32
```

4.33 Crea un método que escriba en un fichero.

Para escribir un fichero línea a línea en Java se recomienda utilizar un objeto flujo derivado de clase `FileWriter` y un filtro `bufferedWriter`, sobre este ultimo se aplica el método `newLine` que inserta los caracters de final de línea en un fichero de texto.



```
1 package Ejercicio_33_escribir_fichero;
2 import java.io.*;
3 import java.util.*;
4 public class Escribirfichero {
5
6 public static void main(String [] args) {
7     String nombre;
8     Scanner sc=new Scanner(System.in);
9     try {
10         FileWriter flujoEscritura=new FileWriter("agenda.txt");
11         BufferedWriter filtro=new BufferedWriter(flujoEscritura);
12         do {
13             System.out.println("Introduce un nombre (intro para terminar):<<");
14             nombre=sc.nextLine();
15             if(nombre.length()>0) {
16                 System.out.println("Telefono :");
17                 String telefono=sc.nextLine();
18                 filtro.write(nombre+","+telefono);
19                 filtro.newLine();
20             }
21         }while(nombre.length()>0);
22         filtro.close();
23     }catch (IOException e) {
24         System.out.println("Error de fichero");
25     }
26 }
27 }
28 }
29 }
```

4.34 Crea un método que borre un fichero dado.

Mencion especial se merece la clase `File` de utilidades de identificación de ficheros y directorios.El objeto de la clase `File` los podremos llamar fuente identifica al fichero.

Tiene los siguientes constructores:

`File(String directorio)`

`File(String fichero)`

`File(String directorio,String Fichero)`

`File(File directorio,String fichero)`

Si el objeto es un fichero:

`Boolean isFile()`

`Boolean exists()`

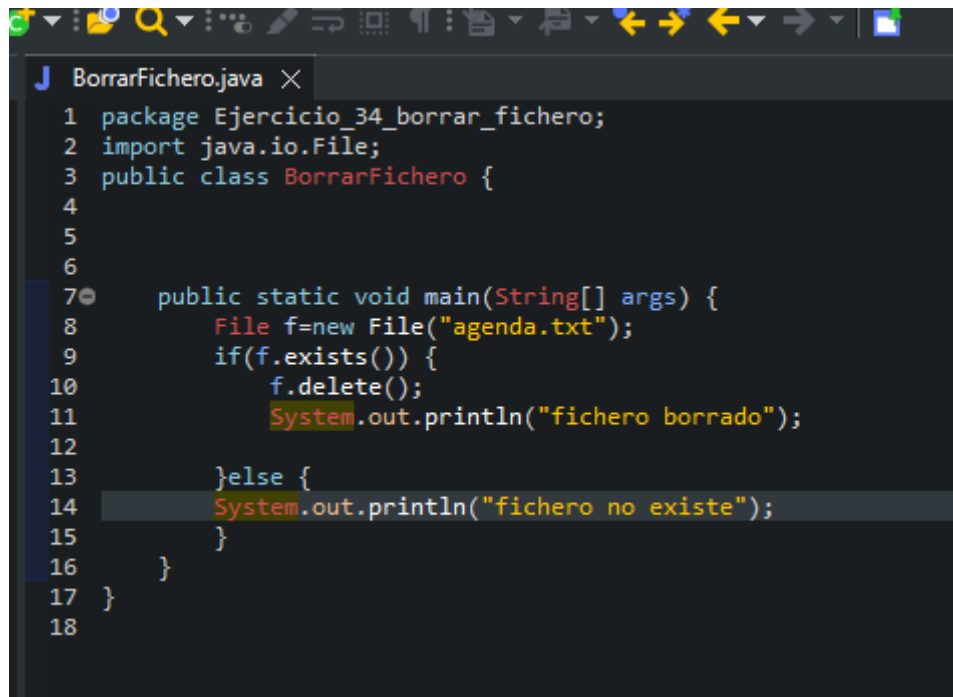
`Boolean canRead();`

`Boolean delete()`

`Long length()`

`Long lastModified();`

`Boolean renameTo()`



```
1 package Ejercicio_34_borrar_fichero;
2 import java.io.File;
3 public class Borrarfichero {
4
5
6
7     public static void main(String[] args) {
8         File f=new File("agenda.txt");
9         if(f.exists()) {
10             f.delete();
11             System.out.println("fichero borrado");
12
13         }else {
14             System.out.println("fichero no existe");
15         }
16     }
17 }
18
```