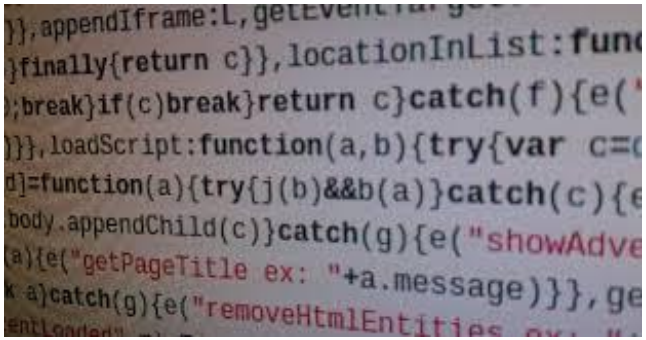


UT 10

CONTROL Y MANEJO DE EXCEPCIONES

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

Objetivos

La unidad permite al alumno conocer sobre los conceptos de las excepciones, su jerarquía y el manejo de estas, así como la creación de alguna excepción.

Contenidos

1.Concepto de excepciones.

2.Jerarquía de excepciones.

3.Manejo de excepciones.

Exception

Concepto de Excepciones
Tipos de errores

Errores en Java

Un error indica que existe un problema al interpretar su programa.

Errores de Sintaxis

Ejemplos:

- El olvido de un punto y coma al final de una sentencia Java se considera un error de sintaxis.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert ";" to complete Statement
    at ErrorIndicators.main(ErrorIndicators.java:8)
```

- El uso de = en lugar de == para comparar valores en una condición "si" es un error de sintaxis.

```
if(x=y){
  …
        }
```

Incorrecto

```
if(x==y){
  …
        }
```

Correcto

Estos errores producen fallos en la compilación

Errores de Lógica

Se producen cuando hemos aplicado mal la lógica en el planteamiento de un programa, proporcionando un resultado que no es el esperado.

Estos errores no producen errores de compilación, ni en tiempo de ejecución.

Ejemplo: un bucle se ejecuta demasiada cantidad de veces o el programa produce un resultado incorrecto.

```
for(int i = 0; i < 5; i++);
        System.out.println(i);
```

Esta sentencia solo se ejecutará una vez. ¿Por qué?

Estos errores NO producen fallos en la compilación

Errores de Lógica

```java
//Ejemplo 1
String nombre;
System.out.println(nombre);

//Ejemplo 2
String nombre1= null;
System.out.println (nombre1);
System.out.println (nombre1.length());

//Ejemplo 3.
int a =  7 ;
int b =  0 ;
System.out.println (a / b); // Error de división entre 0

//Ejemplo 4
int matriz[] = new int[5];

matriz[0]= 6;
matriz[1]= 2;
matriz[2]= 4;
matriz[3]= 10;
matriz[4]= 6;
matriz[5]= 16;
```

Ejemplo1_Errores

Errores en tiepo de ejecución: Exception

```java
package Ejemplos;

import javax.swing.JOptionPane;

public class Ejemplo2_ErroresNoPrevisibles {
    public static void main(String[] args) {

        String nombre = JOptionPane.showInputDialog("Introduce tu nombre");

        int edad = Integer.parseInt(JOptionPane.showInputDialog("Introduce tu edad"));

        System.out.println("Hola "+ nombre+" tu edad es de "+edad+" años");
    }

}
```

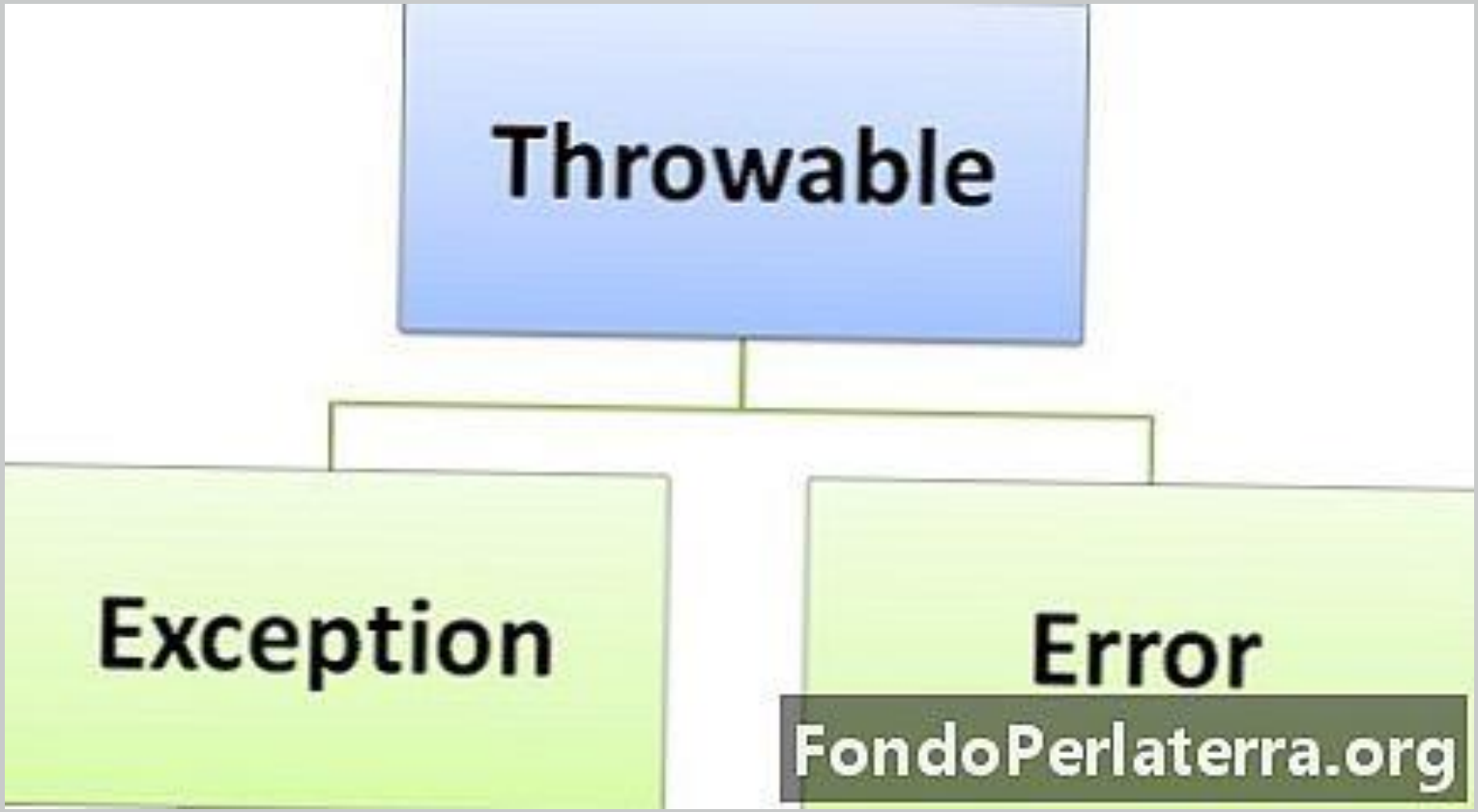
Ejemplo2_ErroresNoPrevisibles

Errores en tiempo de ejecución: Exception

- Una excepción es un error producido en tiempo de ejecución.
- Java aporta un sistema de control de errores limpio y flexible.
- Los errores se pueden producir al no introducir correctamente los datos, dividir por cero, exceder el tamaño de un array, abrir un archivo que no existe, etc…
- El **objetivo** es que el programa no termine de forma abrupta.

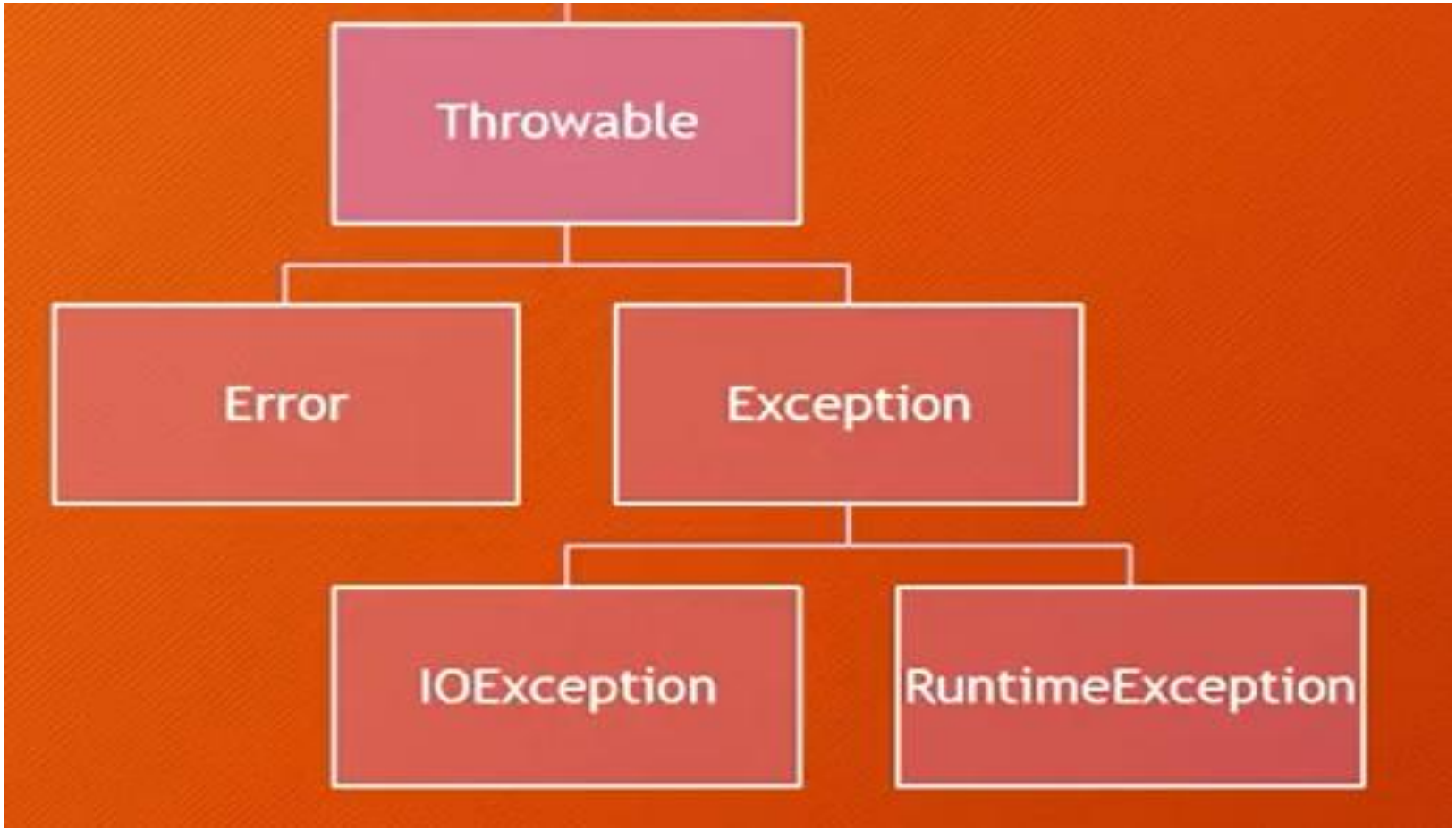
Errores en tiempo de ejecución: Exception

- Cuando se produce un error de este tipo Java genera un objeto de tipo Exception.
- Este objeto pertenecerá a una de las clases previstas en Java y podremos manejarlo utilizando sus métodos.



Jerarquía de Excepciones

Excepción: Throwable



Excepción: Throwable

- En Java todas las excepciones se representan por medio de clases.
- Todas las clases de excepción se derivan en **Throwable.**
- Cuando se produce una excepción, se genera un objeto de un tipo de clase de excepción.

Excepción. Throwable

Throwable tiene dos subclases directas:

- Exception
- Error

Error: errores producidos por la máquina virtual de Java, no en el programa.

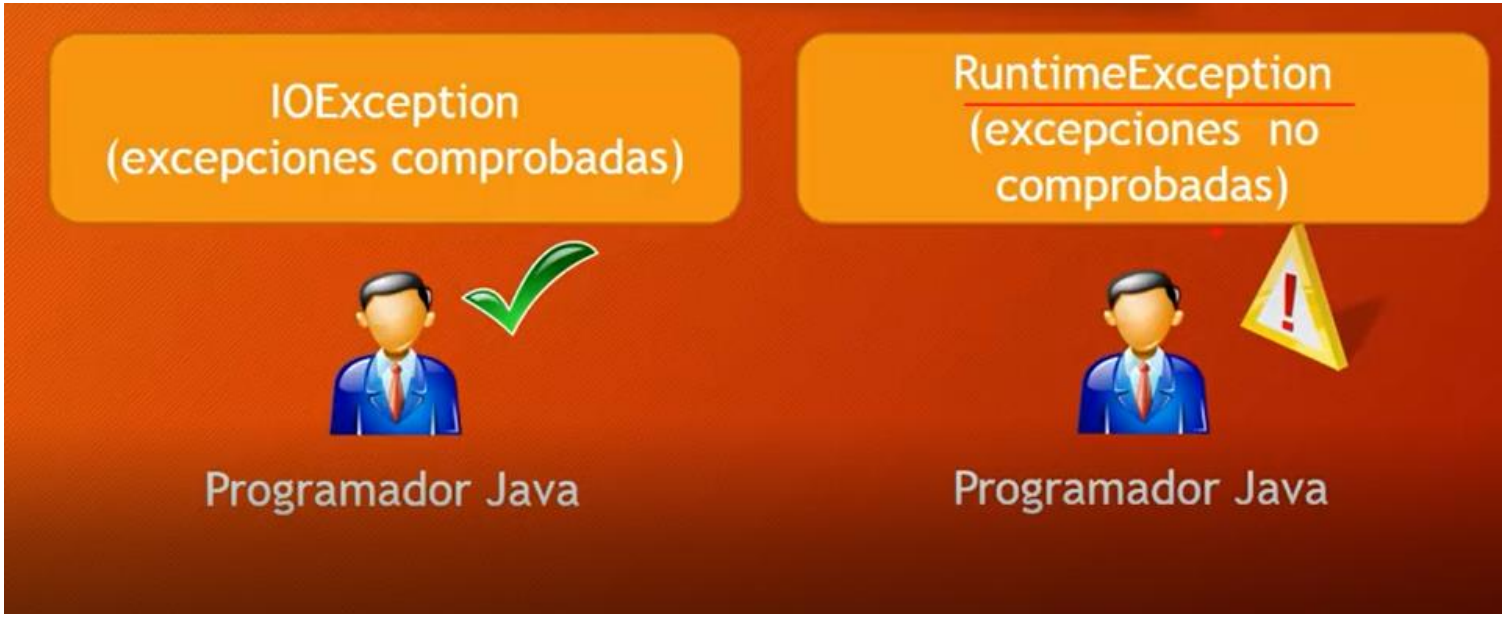
Muy frecuentemente de hardware.

No los podemos controlar.

Ejemplos: error de memoria, poco espacio en disco,…..

Exception: errores de la actividad del programa.

Throwable. Excepción



IOException

- No son responsabilidad del programador.
- No las puede resolver
- El compilador obliga a controlarlas
- Ejemplo: El programa utiliza una imagen que carga desde una carpeta y accidentalmente la imagen no se encuentra en la carpeta.

Ejemplo: ***Ejemplo3_CopiarFicheroIOExceptionsincontrolar.java***

Runtime Exception

- Son responsabilidad del programador.
- Debería haberlas previsto.
- El compilador no obliga a controlarlas
- Ejemplo: El programa recorre un vector excediendo su límite.

Ejemplo: ***Ejemplo2ErroresNoPrevisibles.java***

Runtime Exception

```
public static void main(String[] args) {
    int a = 4, b=0;
    System.out.println(a/b);
}
```

provoca:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
public static void main(String[] args) {
    int [] array = new int [5];
    array[5] = 1;
}
```

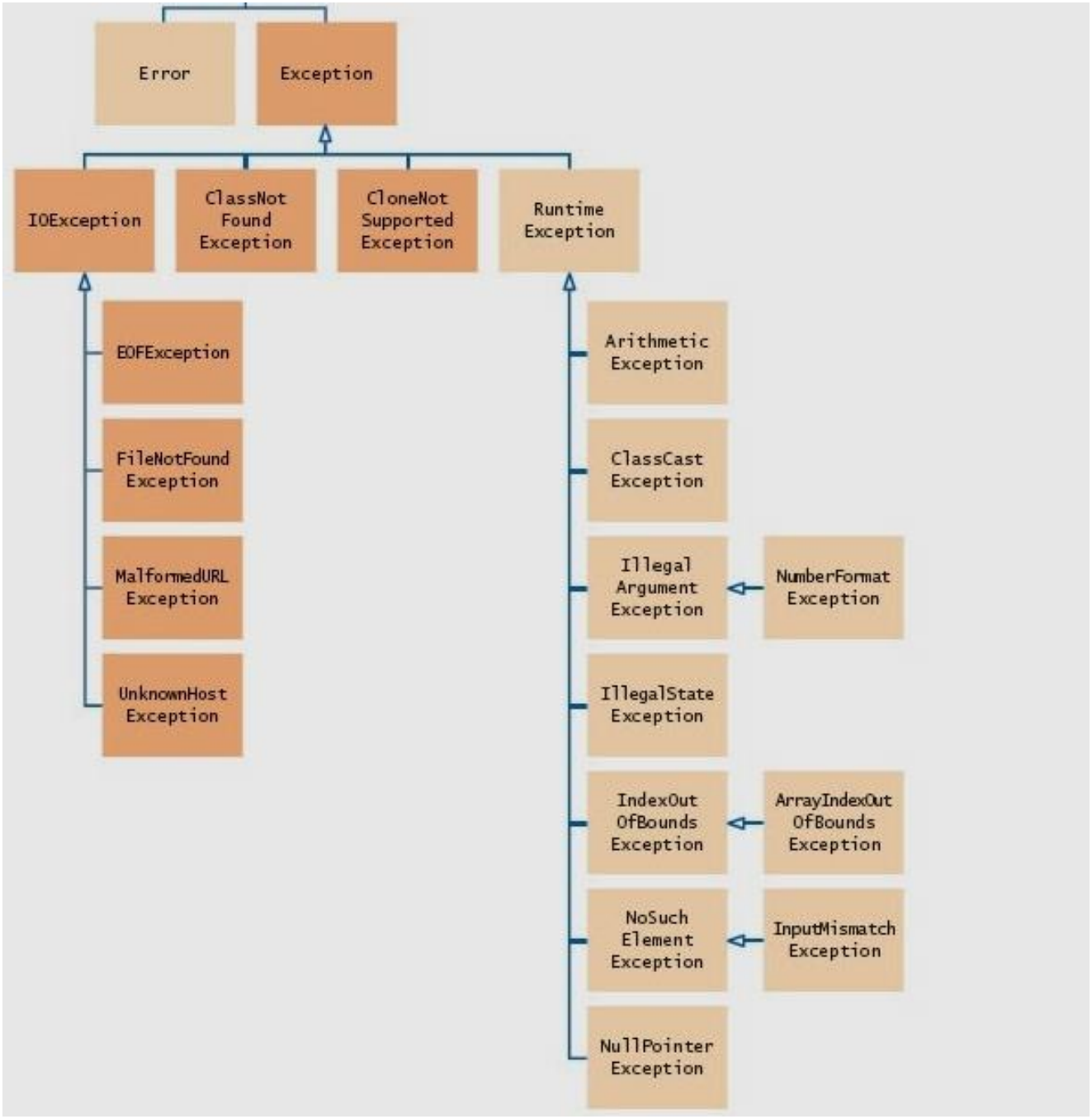
provoca:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Introduce un número entero: ");
    int n = sc.nextInt();
    System.out.print("Número introducido: " + n);
}
```

Se espera un int. Si por ejemplo se lee 4,5 provoca:

```
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:909)
    at java.util.Scanner.next(Scanner.java:1530)
    at java.util.Scanner.nextInt(Scanner.java:2160)
    at java.util.Scanner.nextInt(Scanner.java:2119)
    at excepciones1.Excepciones1.main(Excepciones1.java:7)
```

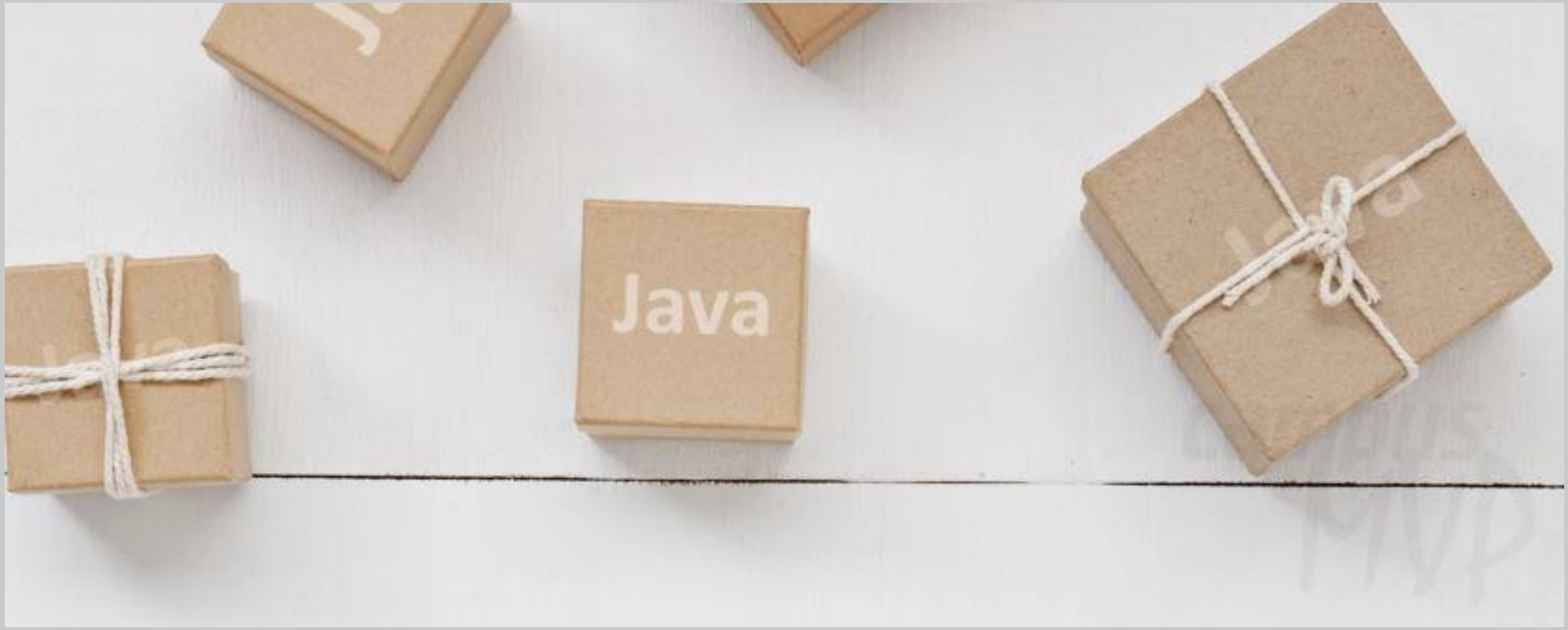

Error
Exception
IOException
ClassNot Found Exception
CloneNot Supported Exception
Runtime Exception
EOFException
FileNotFound Exception
MalformedURL Exception
UnknownHost Exception
Arithmetic Exception
ClassCast Exception
Illegal Argument Exception
NumberFormat Exception
IllegalState Exception
IndexOut OfBounds Exception
ArrayIndexOut OfBounds Exception
NoSuch Element Exception
InputMismatch Exception
NullPointer Exception

Algunas clases de Excepcion

- **Exception**, es un error genérico.
- **NumberFormatException**, captura letras por números.
- **ArithmeticException,** realiza una división por cero.
- **InputMismatchException**, la captura de datos no es correcta.
- **ArrayIndexOutOfBoundsException**, si sobrepasamos el tamaño del array.
- **ClassNotFoundException**, al tratar de utilizar una clase.
- **IllegalAccessException**, se intenta acceder a una clase a la que no se tiene permiso. Como puede ser ejecutar un método privado de otra clase.
- **IOException**, excepciones producidas al realizar tareas de entrada/salida por el programa. Como pueden ser: EOFException, FileNotFoundException, MalformedURLexception, SocketException, UnknownHostException, UTFDataFormatException.
- **NoSuchFieldException**, no se encuentra un determinado atributo.
- **NoSuchMethodException,** no se encuentra un determinado método.
- **RuntimeException**, producidos en tiempo de ejecución. Por ejemplo:

  AritmeticException, ClassCastException, IndexOutOfBoundsException, NegativArraysSizeException, NullPointerException.

- Etc…



Manejo de Excepciones

Gestión de Excepciones

```
try {
      instrucciones;
} catch (Exception e) {
      instruccinoes;
} finally {
      instrucciones
}
```

Código propio de la aplicación.

Código que trata la situación excepcional.

Código que se ejecuta tanto si se han finalizado las instrucciones de try como si ha habido una Excepción.

Gestión de Excepciones

Las excepciones se pueden gestionar con palabras reservadas como:

- **try:** incluye las instrucciones a monitorizar.
- **catch:** instrucciones para gestionar la excepción.
- **finally :** código que debe ejecutarse al salir del bloque try
- **throw** y **throws** **:** para lanzar excepciones desde un método o generar excepciones propias.

Excepción

- Encerramos el código que puede dar el error entre las llaves del **try**

```
Try {
    Sentencias;
}
```

- En el **catch** indicamos que hacemos si se produce el error.

```
catch1(tipoExcepcion 1  nombreObjeto )  {
    sentencias para la excepción;
}
 catch2(……..)
```

Ejemplo: Ejemplo4_ExcepcionControlada.java
Ejemplo5_VariosCatch

Excepción

- Podemos tener más de un **try** dentro de nuestro programa ya sea en el programa principal o en los métodos desarrollados, pero cada try debe tener a continuación su **catch**
- Se puede escribir código fuera de los bloques pero al final de los catch o finally, nunca entre try y catch.

Excepción

- Se pueden escribir tantos catch como errores distintos puedan aparecer.
- Si tenemos un caso genérico y otros particulares deberíamos poner el genérico el último.
- El nombre que le damos al error puede ser el mismo para todos los casos, lo mejor es que se identifiquen de alguna forma.
- El operador | nos permite tratar más de un tipo de excepción en el mismo catch si queremos que tenga el mismo tratamiento.

Excepción

- El bloque **finally** es opcional.
- Se ejecuta siempre (si venimos de try o de catch)
- Se suele utilizar como código para asegurar el cierre de recursos abiertos (ficheros, bases de datos,….)

Algunos Métodos

Métodos más utilizados definidos por Throwable	
String getMessage()	Devuelve una descripción de la excepción
Void printStackTrace()	Muestra el rastreo de pila
String toString()	Devuelve un objeto String que contiene una descripción completa de la excepción. Este método se invoca por println() al representar un objeto Throwable

Excepción. Práctica 10_1

Haz un programa que pida por teclado las dimensiones de un array bidimensional (validar que la dimensión esté entre 2 y 4) y lo rellene con los datos que se introduzcan por teclado con valores numéricos.

Posteriormente ordena cada fila de mayor a menor de forma independiente.

Realiza los procesos citados programando los siguientes métodos:

- Pedir filas
- Pedir columnas
- Cargar array
- Mostrar array
- Ordenar array

Programa los try y catch necesarios para controlar los errores que se puedan producir en el programa.

Anidar bloques try.

Un bloque try puede incluir otro conjunto de bloques try/catch, es decir los bloques try se pueden anidar.

Ver ejemplos:

- ExcepcionAnidada
- TryAnidados

Anidar bloques try. Ejemplo

Cuatro.java | CuadradosEnteros.java | Excepcion.java | TryAnidados.java

```
class TryAnidados{
    public static void main (String args[]){
        int numerador[] = {4,8,16,32,64,128,256,512};
        int denominador[] = {2,0,4,4,0,8};
        try {
            for(int i=0; i<numerador.length; i++){
                try {
                    System.out.println(numerador[i]+"/"+denominador[i]
                    +" = "+numerador[i]/denominador[i]);
                }
                catch (ArithmeticException exc){
                    System.out.println("No puede dividir por 0");
                }
             }
        }
        catch (ArrayIndexOutOfBoundsException exc){
            System.out.println("Elemento no encontrado en vector denominador");
            System.out.println("PROGRAMA TERMINADO");
        }
    }
}
```

Anidar bloques try

SALIDA DEL PROGRAMA

```
H:\PROGRAMACION\UT_8 CONTROL DE EXCEPCIONES\EJEMPLOS>javac T

H:\PROGRAMACION\UT_8 CONTROL DE EXCEPCIONES\EJEMPLOS>java T
4/2 = 2
No puede dividir por 0
16/4 = 4
32/4 = 8
No puede dividir por 0
128/8 = 16
Elemento no encontrado en vector denominador
PROGRAMA TERMINADO

H:\PROGRAMACION\UT_8 CONTROL DE EXCEPCIONES\EJEMPLOS>
```

throw y throws

throws : para lanzar excepciones desde un método.

Un método cuyo código pueda producir excepciones puede capturarlas y tratarlas o relanzarlas al método que lo invocó para que se ocupe.

Ejemplo:Practica_10_1_conThrows .java

throw: sirve para lanzar o generar excepciones propias en cualquier lugar del código.

Ejemplos:PruebaPersonaE2.java

PruebaPersonaE.java

finally

Bloque de código que se ejecuta siempre, es decir, se produzca o no un error.

Los bloques finally son opcionales y se pueden utilizar para cerrar un fichero u otro recurso, guardar un dato, ……..

```
Try {
    Sentencias;
 }
catch1(tipoExcepcion 1  nombreObjeto ) {
            sentencias para la excepción;
        }
        ……..
 finally{
 }
```

Ejemplo: cerrar objeto Scanner en PruebaPersonaE2.java