Programación Frontend y Backend

BLOQUE JAVA

Excepciones

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Excepciones

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

EXCEPCIONES

- Excepciones o Problemas, a la hora de programar siempre se producen error, más o menos graves como ya hemos comentado alguna vez. Estas excepciones pueden producirse por un error de código o incluso algo externo a nuestro código, como no poder conectarse a una BBDD o no recibir la señal de una torre de control.

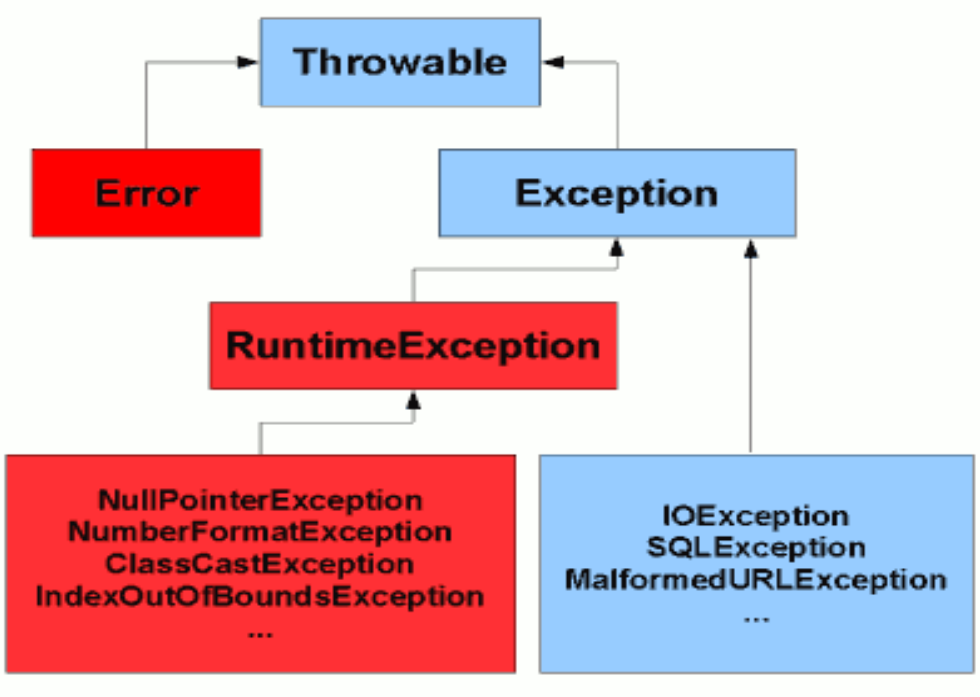
- Estos errores se pueden gestionar y debemos hacerlo correctamente. En java disponemos de un mecanismo consistente en el uso de bloques **try/catch/finally**.

EXCEPCIONES

- La técnica básica consiste en colocar las instrucciones que podrían provocar problemas dentro de un bloque try, y colocar a continuación uno o más bloques catch, de tal forma que si se provoca un error de un determinado tipo, lo que haremos será saltar al bloque catch capaz de gestionar ese tipo de error específico.
- El bloque catch contendrá el código necesario para gestionar ese tipo específico de error. Suponiendo que no se hubiesen provocado errores en el bloque try, nunca se ejecutarían los bloques catch.

```
try {

} catch (Exception e) {

}
```

EXCEPCIONES



EXCEPCIONES

- Cuando se produce o se lanza una excepción el programa se interrumpe.
- La clase **Throwable** tiene las siguientes características:
  - Almacena un mensaje que permite detallar el error que se produjo.
  - Puede contener la causa del error que provoco el actual error y también de tipo **Throwable.**

EXCEPCIONES

```
int a = 4;
int b = 2;
System.out.println("A / B = " + (a/b));
```

```
<terminated> Excepciones [Java Application] C:\_ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nov. 2020 19:30:48)
A / B = 2
```

EXCEPCIONES

```
int a = 4;
int b = 0;
System.out.println("A / B = " + (a/b));
```

```
<terminated> Excepciones [Java Application] C:\_ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nov. 2020 19:32:16)
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at es.eoi.ejercicios.resueltos.Excepciones.main(Excepciones.java:9)
```

EXCEPCIONES

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (Exception e) {
    System.out.println("Se ha producido un error al dividir a/b.");
}
```

```
<terminated> Excepciones [Java Application] C:\_ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nov. 2020 19:35:41)
Se ha producido un error al dividir a/b.
```

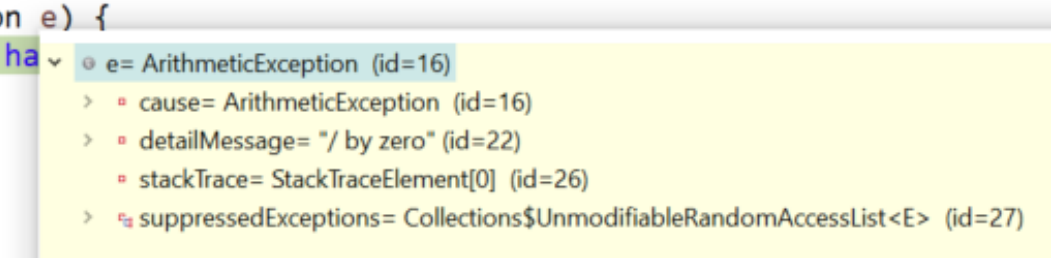
EXCEPCIONES

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (ArithmeticException e) {
    System.out.println("Se ha producido un error aritmético al dividir a/b.");
}
```

```
<terminated> Excepciones [Java Application] C:\_ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nov. 2020 19:39:49)
Se ha producido un error aritmético al dividir a/b.
```

EXCEPCIONES

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (ArithmeticException e) {
    System.out.println("Se ha
}
```



EXCEPCIONES

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (ArithmeticException e) {
    System.out.println("Se ha producido un error aritmético al dividir a/b.");
} catch (Exception e) {
    System.out.println("Se ha producido un error al dividir a/b.");
}
```

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

EXCEPCIONES (FINALLY)

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (ArithmeticException e) {
    System.out.println("Se ha producido un error aritmético al dividir a/b.");
} catch (Exception e) {
    System.out.println("Se ha producido un error al dividir a/b.");
} finally {
    System.out.println("Siempre me ejecuto tanto si ha funcionado como si no.");
}
```

EXCEPCIONES (FINALLY)

```
int a = 4;
int b = 0;

try {
    System.out.println("A / B = " + (a/b));
} catch (ArithmeticException e) {
    System.out.println("Se ha producido un error aritmético al dividir a/b " + e.getLocalizedMessage());
} catch (Exception e) {
    System.out.println("Se ha producido un error al dividir a/b.");
} finally {
    System.out.println("Siempre me ejecuto tanto si ha funcionado como si no.");
}
```

```
<terminated> Excepciones [Java Application] C:\_ej_ico_0\Java\jdk1.8.0_144\bin\javaw.exe (14 nov. 2020 19:49:58)
Se ha producido un error aritmético al dividir a/b / by zero
Siempre me ejecuto tanto si ha funcionado como si no.
```

EXCEPCIONES

Error: Es una subclase de **Throwable** que indica problemas graves que una aplicación no debería intentar resolver: "*Memoria Agotada*" o "*Error interno de la JVM*".

Exception: Clase que indican situaciones que una aplicación debería tratar de forma razonable. Los dos tipos principales son:

- **RuntimeException**: Se usan en caso de errores del programador del tipo acceso a una posición de array no correcta o división por 0 de un numero.
- **IOException**: Se usan en caso de errores que no pueden ser evitados por el programador y van muchas veces relacionados con el manejo de ficheros.

EXCEPCIONES

Objeto Exception

Cuando lanzamos una excepción (es decir, cuando delegamos el problema al código que nos ha invocado), se crea un objeto que contiene toda la información de lo que ha ocurrido.

Con la referencia a este objeto, se pone en marcha el mecanismo de gestión de excepciones, que intenta localizar el trozo de código que va a tratar el error.

A este trozo de código se le llama *exception handler*.

EXCEPCIONES

Lanzar Excepción

Imaginemos que nuestro método recibe como argumento una referencia a un objeto (***t***).
Cuando ***t*** nos llega *null*, **¡¡no sabemos qué hacer!!**

Así que decidimos abdicar de esa responsabilidad de tratar el error lanzando una excepción:

```
if(t == null)  {
        throw new NullPointerException();
}
```

Cuando se ejecuta un ***throw***, se crea un objeto que representa la condición del error.
Es muy importante el nombre de la clase retornada (en el ejemplo *NullPointerException*) que nos dice el tipo del error.

EXCEPCIONES

Lanzar Excepción

Los objetos Exception se crean como todos los objetos en Java, utilizando **new**. Además del constructor por defecto, se le puede pasar una cadena de caracteres con información sobre la excepción.

```
if(t == null)  {
        throw new NullPointerException("t = null");
}
```

EXCEPCIONES

Exception Handler

Después de lanzar la excepción, el sistema intenta encontrar “alguien” (*exception* handler) que la trate, siguiendo por la lista de invocaciones de métodos que se realizaron hasta el método que lanzó la excepción, en orden inverso. Si no se encuentra ninguno, se llega hasta la JVM, que trata el error y el programa termina.

Los *exception handlers* pueden estar especializados en tratar tipos de excepciones, por tanto, lo que se intenta localizar en realidad es un manipulador de excepción que trate el tipo de excepción que se ha lanzado.

EXCEPCIONES

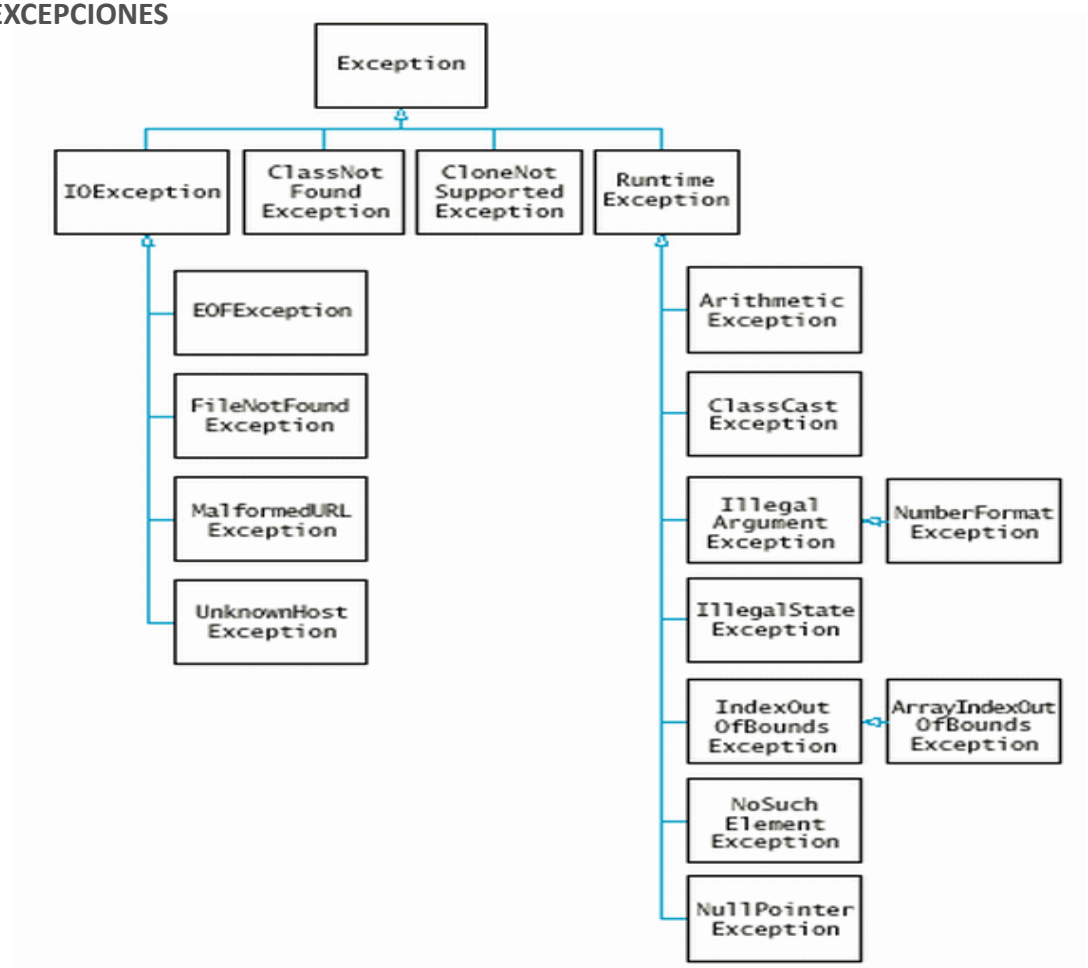
TRATAMIENTO DE EXCEPCIONES

En Java, cada método debe informar al cliente de las excepciones que el mismo puede lanzar durante su ejecución. De este modo los invocadores de este método, saben a qué atenerse y podrán capturar las excepciones adecuadamente o simplemente pasarlas al método que a su vez les invocó.

```
public static void lanzaExcepcion(String msg) {
    throw new Exception(msg);
}
```

```
public static void lanzaExcepcion(String msg) throws Exception {
    throw new Exception(msg);
}
```

EXCEPCIONES



EXCEPCIONES

Métodos útiles del objeto Exception

getMessage( ): Retorna un String que es una breve descripción de la excepción.

printStackTrace( ): Pinta por error estandar la pila de ejecución del momento en que ocurrió la excepción.

getClass( ): Retorna un objeto que representa la clase de excepción.