Excepciones

Contenido

Excepciones

- ¿Qué es una excepción?
- La clase Throwable.
- Manejo de excepciones.
- Tipos de excepciones.
- ¿Cómo lanzar excepciones?
- ¿Cómo interpretar el call stack?

¿Qué es una excepción?

Las excepciones son eventos anómalos que ocurren durante la ejecución de un programa y que *interrumpen* el flujo normal de ejecución.

Cuando ocurre una excepción, se *bloquea* el normal flujo del programa, lo que a menudo provoca que se finalice abruptamente.

Las excepciones pueden estar causadas por un usuario, un problema en la lógica o un error del sistema.

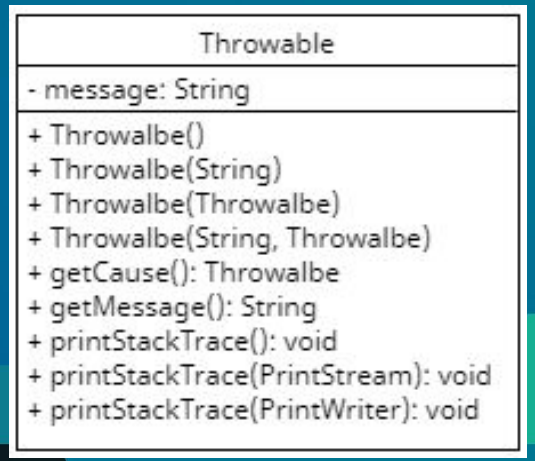
Cómo todo en Java, las excepciones son objetos.

La clase Throwable

La clase Throwable

Todos los errores y excepciones en Java, derivan de la clase **Throwable**.

Solo las instancias de dicha clase (o sus derivadas) pueden ser lanzadas por la *JVM* o 'manualmente' a través de la sentencia *throw*.



La clase Throwable

Constructor: Además del constructor por defecto, se tienen las siguientes sobrecargas:

- (String): Representa el mensaje detallado.
- (Throwable): Representa la *causa* asociada.
- (String, Throwable): Representan el mensaje detallado y la causa asociada.

getCause: Retorna el objeto Throwable que representa una causa (permite encadenamiento).

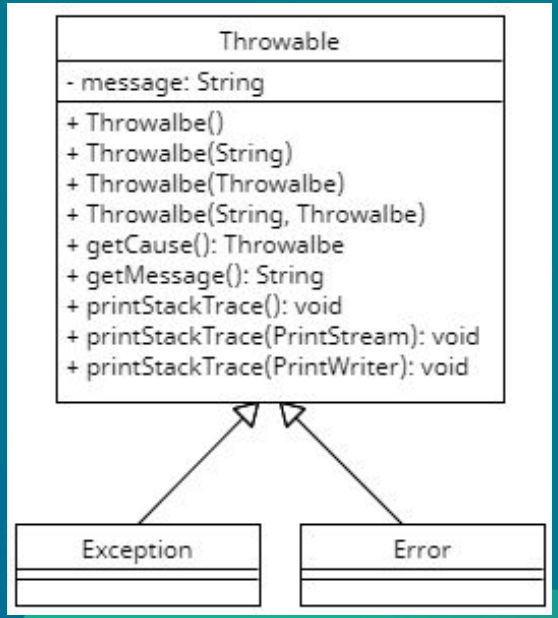
getMessage: Retorna el mensaje detallado.

printStackTrace: Permiten imprimir el call stack en la consola (sin parámetros) o en algún recurso externo (archivo de texto, por ejemplo).

La clase Throwable

Exception y **Error** son las clases derivadas de la clase **Throwable**.

- Una ***Exception*** indica problemas lógicos que una aplicación querría manejar. *Checked Exceptions*.
- Un ***Error*** indica problemas graves que una aplicación **no** debería tratar de detectar. La mayoría de estos se dan en condiciones anormales. *Unchecked Exceptions*.



Manejo de excepciones

Manejo de excepciones

Java provee las palabras reservadas ***try - catch - finally*** para implementar el manejo de excepciones.

El bloque **try** encapsula las instrucciones que podrían lanzar una excepción mientras que el bloque **catch** maneja la excepción si ocurre.

Si ninguna excepción ocurrió dentro del bloque **try**, el bloque **catch** nunca se ejecuta y el programa continua su flujo.

El bloque (opcional) **finally**, agrupa instrucciones que se ejecutarán **siempre**, independientemente si el bloque try generó o no una excepción.

Manejo de excepciones

Los bloques *try - catch - finally* son la solución que ofrece la orientación a objetos a los problemas de tratamiento de errores.

La idea consiste en separar físicamente las instrucciones básicas del programa para el flujo de control normal de las instrucciones para tratamiento de errores.

Así, las partes del código que podrían lanzar excepciones se colocan en un bloque ***try***, mientras que el código para tratamiento de excepciones en el bloque try se pone en un bloque ***catch*** aparte.

Manejo de excepciones

```
public static void main(String[] args) {

    try {

        // INSTRUCCIONES CON POSIBLES EXCEPCIONES
    }
    catch(Exception e) {

        // INSTRUCCIONES QUE SE EJECUTARÁN SI SE
        // PRODUCE UNA EXCEPCIÓN
    }
    finally {

        // BLOQUE OPCIONAL QUE SE EJECUTA SIEMPRE
    }
```

Manejo de excepciones

Un bloque try puede tener más de un bloque catch.

Se pueden utilizar múltiples bloques catch cuando se quiere diferenciar ciertos tipos de excepciones (para tomar distintas acciones).

Los bloques catch deben ir de lo particular a lo general.

```
public static void main(String[] args) {

    try {

        // INSTRUCCIONES CON POSIBLES EXCEPCIONES
    }
    catch(ArithmeticException e) {

        // CATCH DE EXCEPCIÓN PARTICULAR
    }
    catch(NullPointerException e){

        // CATCH DE EXCEPCIÓN PARTICULAR
    }
    catch(Exception e) {

        // CATCH DE UNA EXCEPCIÓN GENERAL
    }
    finally {

        // BLOQUE OPCIONAL QUE SE EJECUTA SIEMPRE
    }
```

Tipos de excepciones

Tipos de excepciones

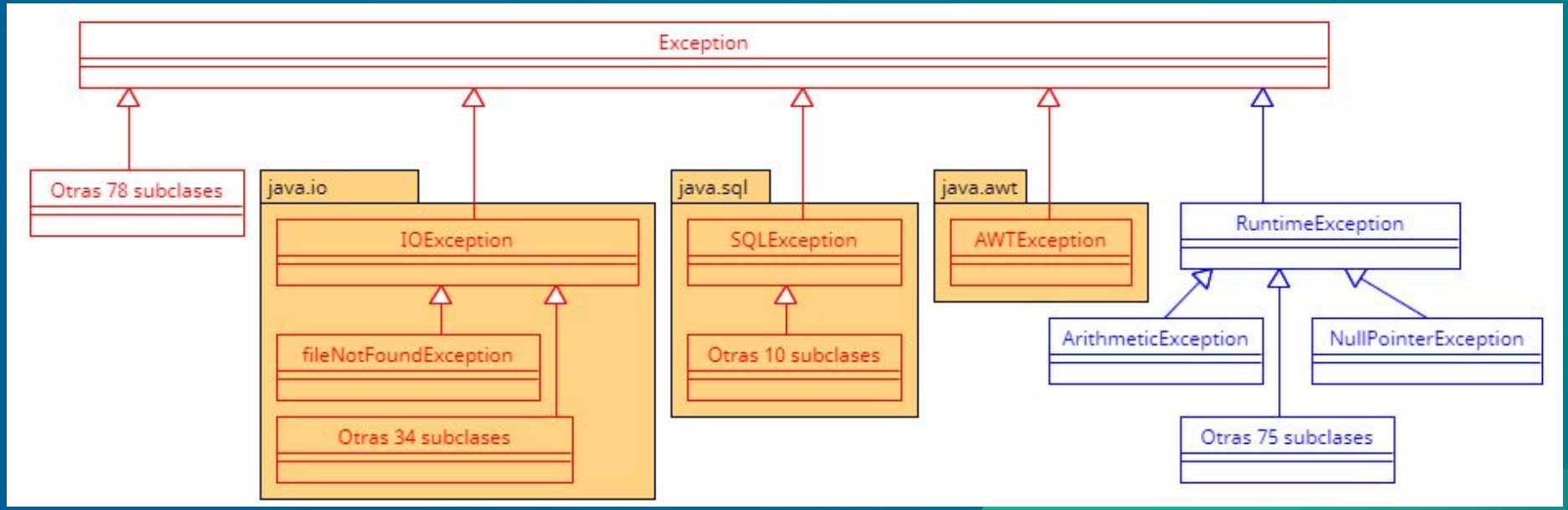
La clase ***Exception*** (y sus derivadas) son excepciones verificadas (checked exceptions):

- Son aquellas en la que el compilador **obliga** a manejar o lanzar.

La clase ***Error*** (y sus derivadas) y la clase ***RuntimeException*** (y sus derivadas) son excepciones no verificadas (unchecked exceptions):

- Son aquellas en la que el compilador **no** obliga a manejar o lanzar.

Tipos de excepciones



Tipos de excepciones

```java
public void metodoUncheckedException(){

    int a = 3 / 0;
    System.out.println(a);
}
```

```java
public void metodoUncheckedException(String s){

    System.out.println(s.length());
}
```

```java
public void metodoCheckedException()
{
    File archivo = new File("FakePath");

    // BLOQUE TRY CON RECURSOS
    try(BufferedReader br = new BufferedReader(new FileReader(archivo))){
        // CÓDIGO...
    }
    catch(Exception e)
    {
        System.out.println(e.getMessage());
    }
}
```

```java
public void metodoCheckedException() throws FileNotFoundException
{
    File archivo = new File("FakePath");

    // BLOQUE TRY CON RECURSOS
    BufferedReader br = new BufferedReader(new FileReader(archivo));
    // CÓDIGO...
}
```

¿Cómo lanzar excepciones?

Una excepción puede ser lanzada manualmente usando la palabra reservada ***throw***, toda excepción derivada de la clase base *Throwable* puede ser lanzada de esta forma.

Esto lo que hace es interrumpir la secuencia de ejecución del programa y transferir el control al primer bloque catch que pueda hacerse cargo de esta nueva excepción.

```java
public void metodoUncheckedException(String s){

    if(s == null){

        throw new NullPointerException("Se debe ingresar una cadena válida");
    }

    System.out.println(s.length());
}
```

¿Cómo interpretar el stack trace?

```java
public static void main(String[] args) {

    Persona p = new Persona(-1);  (1)
```

```
run:
Exception in thread "main" java.lang.IllegalArgumentException: El valor no puede ser negativo.
        at javaapplicationexcepciones.Persona.verificarNegativo(Persona.java:30)  (4)
        at javaapplicationexcepciones.Persona.setEdad(Persona.java:22)  (3)
        at javaapplicationexcepciones.Persona.<init>(Persona.java:17)  (2)
        at javaapplicationexcepciones.App.main(App.java:24)  (1)
```

```java
public class Persona {

    private int edad;

    public Persona(int edad){

        this.setEdad(edad);
        System.out.println(this.edad);  (2)
    }

    private void setEdad(int edad){

        this.verificarNegativo(edad);
        this.edad = edad;  (3)
    }

    private void verificarNegativo(int valor){

        if(valor < 0){  (4)

            throw new IllegalArgumentException("El valor no puede ser negativo.");
        }
    }
}
```

Ejercitación