Programación

Unidad 7: Excepciones

Unidad 7

- Gestión de errores en Java
- Excepciones verificadas (checked) vs. no verificadas (unchecked)
- Manejo de excepciones
- Lanzamiento de excepciones
- Algunos métodos útiles
- Algunas excepciones y errores

Gestión de errores en Java

Unidad 7: Excepciones

Gestión de errores en Java

Utiliza la clase de Excepciones Java. Una **excepción** es una condición anormal que se produce en una porción de código durante su ejecución.

La existencia de las excepciones permite:

- **Encapsular** en clases los errores.
- **Separar el flujo de ejecución normal** del tratamiento de errores.

Las excepciones se pueden: o bien **tratar** o bien **relanzar** para que sean tratadas por otro método del **stack**.

Unidad 7 - Excepciones

Gestión de errores en Java: Jerarquía de clases

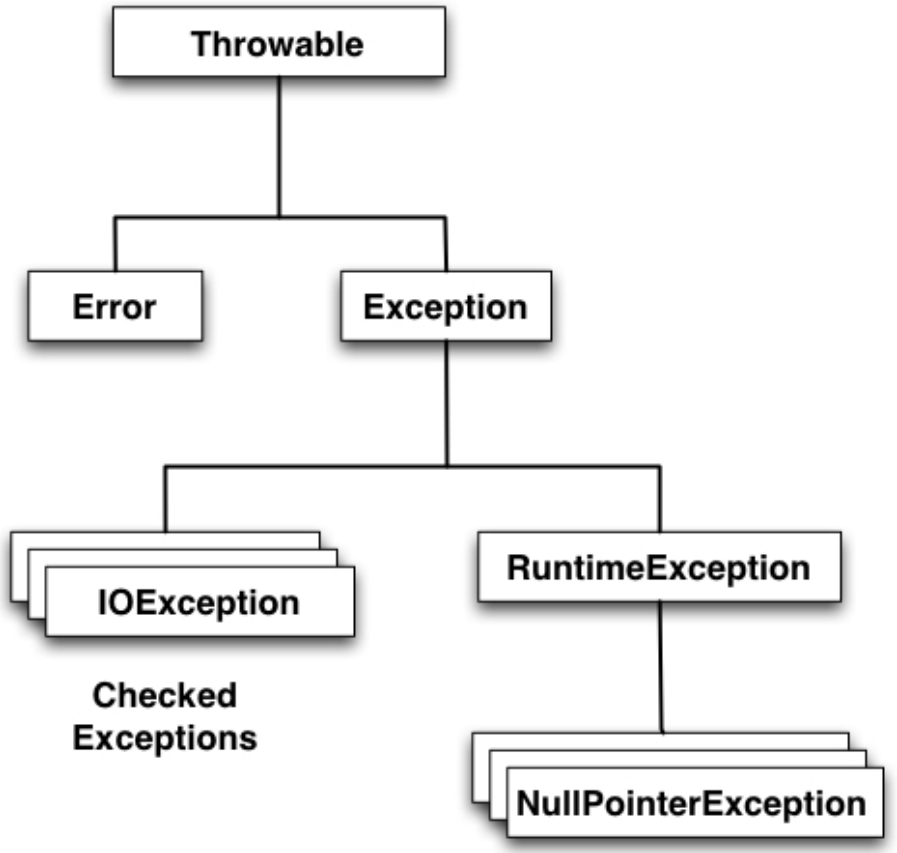
La clase **java.lang.Throwable** describe cualquier clase que pueda ser lanzada como excepción.

Existen dos tipos de clases **Throwable**:

- **java.lang.Error** representa errores de compilación y errores del sistema.
- **java.lang.Exception** representa las excepciones generadas por la aplicación.

Existe un tipo especial de clases **Exception**:

- **java.lang.RuntimeException** representa excepciones generadas por la aplicación cuya gestión no es obligatoria.



Excepciones verificadas(checked) vs. no verificadas (unchecked)

Unidad 7 - Excepciones

Excepciones verificadas (checked) vs. no verificadas (unchecked)

Verificadas (checked):

- Su tratamiento es obligatorio y el compilador así lo chequea. Todas aquellas clases hijas de **java.lang.Exception** que no lo sean de **java.lang.RuntimeException**

No verificadas (unchecked):

- Su tratamiento no es obligatorio y el compilador no lo chequea. Todas aquellas clases hijas de **java.lang.Error** o de **java.lang.RuntimeException**

Manejo de excepciones

Unidad 7 - Excepciones

Manejo de excepciones

Se manejan mediante bloques try & catch:

```
try {
    // Código susceptible de lanzar una excepción.
} catch(tipo-excepción ex) {
    // Código de tratamiento de la excepción.
} catch(tipo-excepción ex) {
    // Código de tratamiento de la excepción.
} finally {
    // Código que se ejecuta siempre.
}
```

Unidad 7 - Excepciones

Manejo de excepciones

El **código susceptible de lanzar o producir una excepción** debe estar incluido en un bloque try.

```
try
{
    // Código susceptible de lanzar una excepción.
}
```

El **código a ejecutar para tratar una excepción** concreta debe estar incluido en un bloque catch.

```
catch(Exception ex) {
    // Código de tratamiento de la excepción
}
```

Unidad 7 - Excepciones

Manejo de excepciones

El **código a ejecutar independientemente de que se produzca o no una excepción** debe estar incluido en un bloque finally.

```
finally
{
  // Código que se ejecuta siempre.
}
```

Atención con las **variables locales** definidas en cualquiera de los bloques porque no son accesibles desde fuera.

Unidad 7 - Excepciones

Manejo de excepciones

Los bloques try & catch pueden estar anidados en cualquiera de las estructuras

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class TestExcepciones {

    public static void main(String[] args) {
        File f = new File("./test.txt");
        FileInputStream fis = null;
        try {
            fis = new FileInputStream(f);
        } catch (FileNotFoundException e) {
            System.out.println("Archivo no encontrado");
        } finally {
            try {
                fis.close();
            } catch (IOException e) {
                System.out.println("Error al cerrar el archivo");
            }
        }
    }

}
```

Lanzamiento de excepciones

Unidad 7 - Excepciones

Lanzamiento de excepciones

Un método indica que puede lanzar o (relanzar) una excepción mediante la palabra **throws**.

Un método crea y lanza una excepción mediante la palabra **throw**.

```
public class FileInputStream extends Inputstream{
    public FileInputStream(File aFile) throws IOException{
        if(condicion){
                throw new IOException("No existe el fichero.");
        }
    }
}
```

Unidad 7 - Excepciones

Lanzamiento de excepciones

Si una excepción es relanzada mediante throws consecutivamente y llega a la JVM, esta se detendrá mostrando la excepción.

```
Exception occurred during event dispatching:
java.lang.NumberFormatException: test
at java.lang.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1194)
at java.lang.Double.parseDouble(Double.java:198)
at edu.upco.einf.practica12.EuroConversor.actionPerformed(EuroConversor.java:65)
at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1461)
at javax.swing.AbstractButton.ForwardActionEvents.actionPerformed(AbstractButton.java:1515)
at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:392)
at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:264)
at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:254)
at java.awt.Component.processMouseEvent(Component.java:3799)
at java.awt.Component.processEvent(Component.java:3628)
at java.awt.Container.processEvent(Container.java:1202)
at java.awt.Component.dispatchEventImpl(Component.java:2678)
at java.awt.Container.dispatchEventImpl(Container.java:1251)
at java.awt.Component.dispatchEvent(Component.java:2581)
```

Unidad 7 - Excepciones

Lanzamiento de excepciones

Ejemplo de definición:

```
public class FactorNegativoException extends Exception {
    private static final long serialVersionUID = 697497904559054369 8L;

    public FactorNegativoException(double param) {
        super("Has introducido un valor erroneo: " + param);
    }
}
```

*Nota: **java.lang.Exception** tiene varios constructores, pero el más usado es el que recibe un String con el motivo de la excepción. Este String será accesible mediante el método **getMessage()**.*

Unidad 7 - Excepciones

Lanzamiento de excepciones

```
public void escalar(double factor) throws FactorNegativoException {
    if (factor < 0)
        throw new FactorNegativoException(factor);
    else
        radio *= factor;
}
```

Unidad 7 - Excepciones

Lanzamiento de excepciones

Ejemplo

```java
// En este método se lanza la excepción en caso de producirse
public Circulo metodoCalculo(double r, double f) throws FactorNegativoException {
    Circulo c = new Circulo(r);
    c.escalar(f);
    return c;
}

// En este método se trata la excepción en caso de producirse
public Circulo metodoCalculo2(double r, double f) {
    Circulo c = new Circulo(r);
    try {
        c.escalar(f);
    } catch (FactorNegativoException e) {
        e.printStackTrace();
    }
    return c;
}
```

Algunos métodos útiles

Unidad 7 - Excepciones

Algunos métodos útiles

public void printStackTrace();

Imprime por la salida estándar la pila (stack) de llamadas incluyendo los números de línea y ficheros donde se ha producido la excepción.

- **public String getMessage();**

  Devuelve una cadena de caracteres con la descripción de la excepción.

- **public String toString();**

  Devuelve la representación en cadena de caracteres de la excepción.

Algunas excepciones y errores

Unidad 7 - Excepciones

Algunas excepciones y errores

Todas las excepciones y errores Java tienen un apartado específico dentro de la API.

- **java.lang.NoClassDefFoundError** (no verificada)

  Lanzada cuando se intenta instanciar una clase y no se encuentra.

- **java.lang.IllegalArgumentException** (no verificada)
  - Lanzada cuando se llama a un método con un parámetro erróneo.

- **java.lang.IndexOutOfBounds** (no verificada)
  - Lanzada cuando se intenta acceder a una posición inexistente en una colección (array, vector, etc.....).

Unidad 7 - Excepciones

Algunas excepciones y errores

java.lang.InterruptedException (verificada)

Lanzada cuando se despierta un thread que estaba sleep.

java.lang.NoSuchFieldError (no verificada)

Lanzada cuando se intenta acceder a un atributo inexistente.

java.lang.NoSuchMethodError (no verificada)

Lanzada cuando se intenta acceder a un método inexistente.

Unidad 7 - Excepciones

Algunas excepciones y errores

java.lang.NumberFormatException (no verificada)

Lanzada cuando se intenta convertir una cadena de caracteres a un número y tiene caracteres no numéricos.

java.lang.OutOfMemoryError (no verificada)

Lanzada cuando la JVM no puede instanciar un objeto por falta de memoria física.

java.lang.ClassCastException (no verificada)

Lanzada cuando se realiza un casting de un objeto mediante una clase de la que no es instancia.

Unidad 7 - Excepciones

Algunas excepciones y errores

java.lang.ArithmeticException (no verificada)

Lanzada cuando se produce una situación anómala en una operación matemática (por ejemplo, divisiones por 0).

java.io.IOException (verificada)

Lanzado cuando ocurre un problema de I/O genérico.

java.io.EOFException (verificada)

Lanzada cuando se detecta el final de un stream de manera inesperada.

Unidad 7 - Excepciones

Algunas excepciones y errores

java.io.FileNotFoundException (verificada)

Lanzada cuando se intenta acceder a un fichero en modo lectura y no se encuentra.

java.lang.NullPointerException (no verificada)

Lanzada cuando se accede a un objeto cuyo valor es null.