

**MP0485**  
**Programación**  
**UF8. Excepciones**

## **8.2. Excepciones personalizadas**

# Índice

---

☰	Objetivos	3
☰	Introducción	4
☰	Crear excepción no comprobada	5
☰	Crear excepción comprobada	11
☰	OJO: throw frente a throws	14
☰	Resumen	15

# Objetivos

---

Con esta unidad perseguimos los siguientes objetivos:

1

Implementar clases capaces de desencadenar excepciones. Por ejemplo, una cuenta bancaria podría desencadenar una excepción cuando intentamos retirar más dinero de una cuenta del que tenemos.

---

¡Ánimo y adelante!

## Introducción

---

Cuando creamos una clase personalizada también deberíamos crear las clases de excepción asociadas que sean necesarias para gestionar las situaciones excepcionales o de error.

---

Una clase personalizada de excepción debe heredar de *Exception* o de *RuntimeException*. Lo más habitual es heredar de *Exception*.

- Si creamos una clase que hereda de *Exception* será una excepción comprobada, es decir, es obligatorio gestionarla usando *try ... catch* o propagarla como hemos visto en la unidad anterior.
- Si creamos una clase que hereda de *RuntimeException* será una excepción de tipo "no comprobada", es decir, no es obligatorio gestionarla.
- Por convención, es altamente conveniente que las clases de excepción personalizadas terminen con la palabra *Exception*.

# Crear excepción no comprobada

En este apartado aprenderás por medio de un ejercicio guiado cómo se crea una excepción personalizada de tipo "no comprobada".

Una clase *Coche* compuesta por las propiedades *marca*, *modelo* y *velocidad* tiene los métodos *acelerar()* y *frenar()*. Queremos considerar una situación excepcional el hecho de que la velocidad exceda de los 120 km/hora. Para ello crearemos una excepción llamada *ExcesoVelocidadException* de tipo "no comprobada".

Partiremos de esta sencilla clase *Coche* que todavía no considera el exceso de velocidad como una situación excepcional:

```
public class Coche {  
    private String marca;  
    private String modelo;  
    private int velocidad;  
  
    public Coche(String marca, String modelo) {  
        this.marca = marca;  
        this.modelo = modelo;  
        this.velocidad = 0;  
    }  
  
    @Override  
    public String toString() {  
        return "Coche [marca=" + marca + ", modelo=" + modelo + ",  
    }  
  
    public void acelerar(int cuanto) {  
        this.velocidad = this.velocidad + cuanto;  
    }  
  
    public void frenar(int cuanto) {  
        this.velocidad = this.velocidad - cuanto;  
    }  
}
```

Ahora puedes crear la clase *Principal*, con método *main* y comprobar que evidentemente puedes acelerar el coche todo lo que quieras.

```
public class Principal {  
    public static void main(String[] args) {  
        Coche miCoche = new Coche("Ford", "Fiesta");  
  
        miCoche.acelerar(100);  
        miCoche.acelerar(50);  
        System.out.println(miCoche);  
    }  
}
```

Recuerda que *System.out.println(miCoche)*; equivale a poner *System.out.println(miCoche.toString())*; ya que *toString()* es el método que se ejecuta por defecto.

---

Pasos a seguir para considerar el exceso de velocidad una situación excepcional:

1

Crear una clase que extienda de *Exception* o *RuntimeException*.

Como queremos que nuestra excepción sea de tipo "no comprobada", extenderemos de *RuntimeException*. Nuestra nueva clase de excepción podría quedar así:

```
public class ExcesoVelocidadException extends RuntimeException {  
    private static final long serialVersionUID = 1L;  
    private int nuevaVelocidad;  
  
    public ExcesoVelocidadException(int nuevaVelocidad) {  
        super("Exceso de velocidad");  
        this.nuevaVelocidad = nuevaVelocidad;  
    }  
  
    @Override  
    public String toString() {  
        return "ExcesoVelocidadException [nuevaVelocidad=" + nuevaV  
    }  
}
```

Vamos a analizar un poco el código:

- En primer lugar observa que **nuestra nueva clase *ExcesoVelocidadException* termina con la palabra *Exception***. Esto es una buena práctica, y aunque sea opcional puedes considerarlo como obligatorio, ya que es la forma de que se distinga claramente que se trata de una clase que representa una excepción.
- La clase *ExcesoVelocidadException* extiende de *RuntimeException*, lo que la convierte en una **excepción de tipo "no comprobada"**.
- En el constructor, con la sentencia *super("Exceso de velocidad")* estamos pasando un valor al constructor de *RuntimeException*. El texto pasado como argumento será lo que finalmente devuelva el método *getMessage()* del objeto de excepción.
- Nuestra clase de excepción, como cualquier otro tipo de clase, puede tener todas las propiedades y todos los métodos que sea necesario para cumplir correctamente con su función. En el ejemplo, hemos añadido la propiedad *nuevaVelocidad*, que contendrá la velocidad del coche después de la aceleración que provocó la excepción. El valor para la propiedad *nuevaVelocidad* se lo estamos pasando al constructor por medio de un parámetro.
- La sentencia *private static final long serialVersionUID = 1L;* es necesaria para evitar un *warning* o alerta que nos aparece en Eclipse. Esto se debe a que *RuntimeException* es una clase serializable y necesita un número de versión. Este concepto lo veremos en otra unidad, por ahora confórmate con saber que hay que añadir esa línea y ya está.
- Por último hemos sobrescrito el método *toString()*.

## 2

### Desencadenar la excepción dentro de la clase *Coche*

La clase *Coche* tendrá que desencadenar la excepción cuando la velocidad sea superior a 120 km/hora y lo hará creando un nuevo objeto de la clase *ExcesoVelocidadException* con un tipo de declaración específica para este fin que tiene el siguiente formato:

```
throw new ClaseDeExcepción(parámetros);
```

La excepción se desencadenará dentro del método *acelerar*. Modifica el código del método *acelerar* de la siguiente manera:

```
public void acelerar(int cuanto) {
    this.velocidad = this.velocidad + cuanto;
    if (this.velocidad > 120) {
        throw new ExcesoVelocidadException(this.velocidad);
    }
}
```

3

### Provocar la excepción

Puesto que es una excepción de tipo "no comprobada" no estamos obligados a gestionarla, Eclipse no nos está dando ninguna pista sobre la posible excepción.

```
public class Principal {
    public static void main(String[] args) {
        Coche miCoche = new Coche("Ford", "Fiesta");
        System.out.println(miCoche.toString());
        miCoche.acelerar(100);
        miCoche.acelerar(50);
        System.out.println(miCoche);
    }
}
```

Si ejecutas el programa, como no hemos gestionado la excepción, se abortará justo en la segunda aceleración y el programa terminará lanzando la traza de la excepción, tal y como puedes a continuación.

 Coche [marca=Ford, modelo=Fiesta, velocidad=0]  
Exception in thread "main" ExcesoVelocidadException [nuevaVelocidad=150]  
at Coche.acelerar(Coche.java:20)  
at Principal.main(Principal.java:6)

Llamamos traza de la excepción a un mensaje de texto que muestra el nombre de la excepción, los argumentos si los tiene, el método que provocó la excepción y la pila de llamadas a los distintos métodos de donde proviene. La parte de texto en rojo del cuadro de información anterior es la traza del error.

También, por supuesto, podemos optar por gestionar la excepción usando *try ... catch*.

```
public class Principal {  
    public static void main(String[] args) {  
        Coche miCoche = new Coche("Ford", "Fiesta");  
        System.out.println(miCoche.toString());  
        try {  
            miCoche.acelerar(100);  
            miCoche.acelerar(50);  
        } catch (ExcesoVelocidadException e) {  
            System.out.println("Cuidado, has excedido la velocidad");  
            System.out.println(e.toString());  
        }  
        System.out.println(miCoche);  
    }  
}
```

En ocasiones resulta interesante mostrar la traza del error dentro del bloque *catch* para obtener mayor información sobre lo ocurrido. Esto se hace con la sentencia *e.printStackTrace()*;

```
public class Principal {  
    public static void main(String[] args) {  
        Coche miCoche = new Coche("Ford", "Fiesta");  
        System.out.println(miCoche.toString());  
        try {  
            miCoche.acelerar(100);  
            miCoche.acelerar(50);  
        } catch (ExcesoVelocidadException e) {  
            e.printStackTrace();  
        }  
        System.out.println(miCoche);  
    }  
}
```

# Crear excepción comprobada

En esta ocasión realizaremos el mismo ejemplo de antes, pero esta vez la excepción *ExcesoVelocidadException* será de tipo "comprobada".

Volveremos a realizar los mismos pasos del anterior apartado, pero esta vez para crear una excepción de tipo "comprobada". Te aconsejo que crees un proyecto nuevo para quedarte con los dos ejemplos. ¡Vamos allá!

1

Crear una clase que extienda de *Exception*

```
public class ExcesoVelocidadException extends Exception {  
    private static final long serialVersionUID = 1L;  
    private int nuevaVelocidad;  
  
    public ExcesoVelocidadException(int nuevaVelocidad) {  
        super("Exceso de velocidad");  
        this.nuevaVelocidad = nuevaVelocidad;  
    }  
  
    @Override  
    public String toString() {  
        return "ExcesoVelocidadException [nuevaVelocidad=" + nuevaV  
    }  
}
```

Solo hemos modificado el nombre de la clase base *RuntimeException* por *Exception*, el resto del código se ha quedado igual.

2

### Desencadenar la excepción dentro de la clase *Coche* y propagarla hacia arriba.

En este caso, el método *acelerar* no solo tiene que **incluir la declaración *throw new*** para **desencadenar la excepción**, sino que además debe **propagarla hacia arriba con la declaración *throws ExcesoVelocidadException*** en la cabecera de la función para que sea recogida por el método que la provocó. En nuestro ejemplo, la excepción será provocada en el método *main* de la clase *Principal*.

```
public void acelerar(int cuanto) throws ExcesoVelocidadException {  
    this.velocidad = this.velocidad + cuanto;  
    if (this.velocidad > 120) {  
        throw new ExcesoVelocidadException(this.velocidad);  
  
    }  
}
```

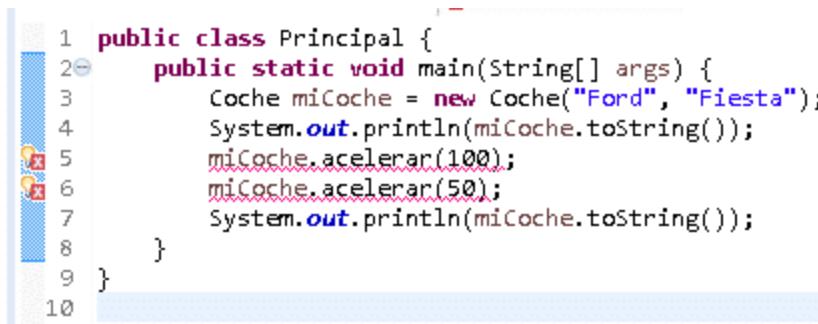
3

### Provocar la excepción

Comenzaremos por crear una clase *Coche* y hacer uso del método *acelerar* sin preocuparnos de nada más.

```
public class Principal {  
    public static void main(String[] args) {  
        Coche miCoche = new Coche("Ford", "Fiesta");  
        System.out.println(miCoche.toString());  
        miCoche.acelerar(100);  
        miCoche.acelerar(50);  
        System.out.println(miCoche);  
    }  
}
```

Eclipse ya nos está mostrando errores de compilación porque ahora tenemos una excepción de tipo "controlado" y estamos obligados a gestionarla.



```
1 public class Principal {
2     public static void main(String[] args) {
3         Coche miCoche = new Coche("Ford", "Fiesta");
4         System.out.println(miCoche.toString());
5         miCoche.acelerar(100);
6         miCoche.acelerar(50);
7         System.out.println(miCoche.toString());
8     }
9 }
10
```

Debemos encerrar el código en un bloque *try ... catch* o propagar la excepción. En esta ocasión vamos a utilizar un bloque *try ... catch*.

```
public class Principal {
    public static void main(String[] args) {
        Coche miCoche = new Coche("Ford", "Fiesta");
        System.out.println(miCoche.toString());
        try {
            miCoche.acelerar(100);
            miCoche.acelerar(50);
        } catch (ExcesoVelocidadException e) {
            System.out.println(e.getMessage());
            System.out.println(e.toString());
        }
        System.out.println(miCoche);
    }
}
```

## OJO: throw frente a throws

No confundas las declaraciones tipo **throws** con las declaraciones tipo **throw new**.

Las declaraciones **throws** se utilizan para relanzar o propagar hacia arriba una excepción.

```
public static void ejecutarTarea2() throws FileNotFoundException {
    FileReader fichero = new FileReader("c:/datos.txt");
    System.out.println("El fichero ha sido abierto");
}
```

Relanzamos la excepción al método que invocó a *ejecutarTarea2()*.

Las declaraciones tipo **throw new** se utilizan para desencadenar una excepción.

```
public void acelerar(int cuanto) {
    this.velocidad = this.velocidad + cuanto;
    if (this.velocidad > 120) {
        throw new ExcesoVelocidadException(this.velocidad);
    }
}
```

## Resumen

---

---

Has terminado la lección, vamos a ver los puntos más importantes que hemos tratado.

- Cuando desarrollamos una librería de clases para gestionar algún problema común de ingeniería de software, debemos implementar también las clases necesarias para responder a las situaciones de excepción que puedan producirse.
- Para crear una clase de excepción personalizada no comprobada tenemos que crear una clase que derive de *RuntimeException*.
- Para crear una clase de excepción personalizada comprobada tenemos que crear una clase que derive de *Exception*.



**PROEDUCA**