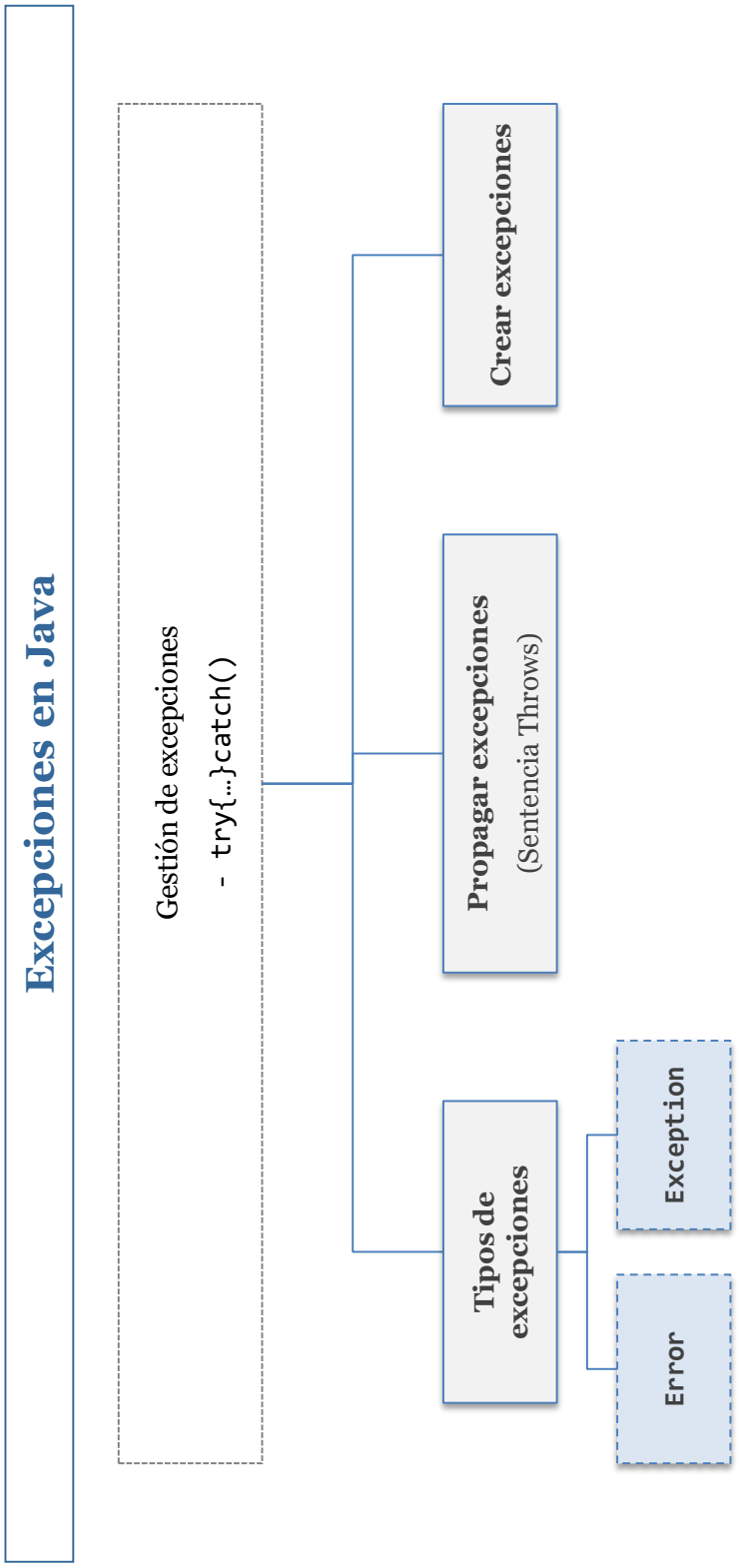
Excepciones

TEMA 4

Esquema



Ideas clave

4.1. ¿Cómo estudiar este tema?

Para estudiar este tema debes leer las **páginas 171-198** del siguiente libro, disponible en la Biblioteca Virtual de UNIR:

García, L. F. (2010). *Todo lo básico que debería saber sobre programación orientada a objetos en Java*. Barranquilla: Uninorte.

Para comprobar si has comprendido los conceptos realiza el test de autoevaluación del tema.

En este tema vamos a ver qué son las excepciones y cómo las podemos utilizar dentro de los programas Java. Veremos:

- Qué son las excepciones.
- Captura y gestión de excepciones.
- Lanzamiento y creación de excepciones.

4.2. Excepciones

Excepción

Es un **suceso en tiempo de ejecución** que puede causar que una **rutina fracase**.

Funcionamiento de excepciones

- Generación de una excepción dentro de un método.
- Construcción del objeto `Exception`.
- Lanzamiento de la excepción.
- Captura de la excepción:
  - Dentro del propio método.
  - Fuera del método a través de la pila de llamadas.
- Ejecución del manejador de la excepción.

Ventajas del uso de excepciones

- Separación código - manejo de errores
- Propagación de los errores sobre la pila de llamadas
- Agrupación de errores y diferenciación

4.3. Captura y gestión de excepciones

Las excepciones que se producen en un bloque `try` pueden ser capturadas por una manejador de excepciones especificado por un bloque `catch`.

La sintaxis es la siguiente:

```
try{
//sentencia(s) que pueden generar excepciones
}
catch (tipoExcepción1 objEspecifico){
//sentencia(s) para el tratamiento de la excepción recogida
}
...
catch (tipoExcepción2 objGeneral){
//sentencia(s) para el tratamiento de la excepción recogida
}
finally{
//sentencia(s)
}
```

El **bloque `try` o bloque de intento** define el conjunto de sentencias que podrían lanzar una excepción.

Cada **bloque `catch`** contiene un **manejador de excepciones** para el tipo de excepción indicada en su argumento, el cual debe ser el nombre de una clase heredada de la clase `Throwable`.

El **bloque `finally`** define el **conjunto de sentencias** que se ejecutan siempre, se haya producido o no una excepción.

Todo **bloque `try`** deber ir acompañado de al menos un **bloque `catch` o `finally`**.

Cuando se lanza una **excepción** se busca el manejador cuyo argumento coincida con el tipo de la excepción lanzada. Si se encuentra un manejador para una excepción no se continúa la propagación, así que siempre deben ponerse primero los manejadores de excepciones específicas y al final los de excepciones generales.

4.4. Lanzamiento de excepciones

El método `throw` se utiliza para elevar o lanzar una excepción. Su sintaxis es:

```
throw objetoThrowable;
```

La cláusula `throws` en un método indica las excepciones que el método puede lanzar. Su sintaxis es:

```
throws listaExcepciones;
```

Las excepciones que un método lanza explícitamente o que son lanzadas por **métodos invocados** por ese método, se deben declarar en la **lista de excepciones**.

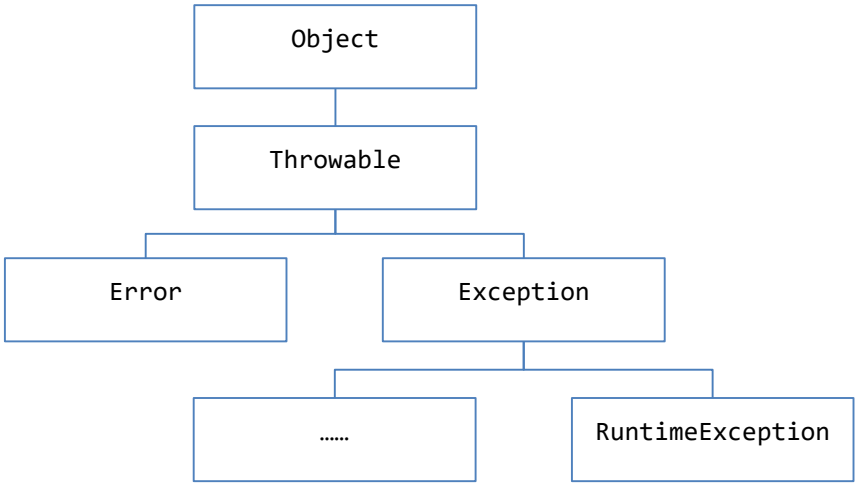
4.5. La clase `Throwable`

La clase `Throwable` es la superclase de todas las clases de excepción.

- Constructores:
  - `Throwable()`
  - `Throwable(String mensaje)`
  - `Throwable(String mensaje, Throwable causa)`
  - `Throwable(Throwable causa)`

- Algunos métodos:
  - `String getMessage()`
  - `void printStackTrace()`
  - `Throwable initCause(Throwable causa)`
  - `Throwable getCause()`

Tipos de excepciones



- **`Error`**: errores internos y por agotamiento de recursos, generado por la máquina virtual. Normalmente no se capturan.
- **`Exception`**: problema no grave, excepción generada por el programa.
- **`RuntimeException`**: se comete por un error de programación (moldeado incorrecto, `array` fuera de límites, acceso a referencia nula, etc.).
- **Excepciones no comprobadas**: las que derivan de `Error` o de `RuntimeException`.

4.6. Creación de excepciones

- Objetos excepción
  - Crear una clase que tenga como clase base a `Exception` (directa o indirectamente).
  - Implementar los constructores (al menos uno sin argumentos y otro que reciba un `string`).
  - Añadir los métodos que se estimen necesarios.

- Elevación de la excepción
  - Sintaxis: `throw AlgunObjetoThrowable;`
  - En el momento en que se lanza una excepción, se abandona la ejecución de las sentencias posteriores y se busca el bloque `catch` más cercano.

Ejemplo:

```
class MiExcepcion extends Exception{
	public MiExcepcion(){super();}
	public MiExcepcion(String s){super(s);}
}
//SIN THROWS unreported exception MiExcepcion; must be caught or
declared to be throw
class ClaseC{
	public void metodoC() throws MiExcepcion{
		System.out.println("C-> lanza excepcion");
		throw new MiExcepcion("Casi que no");
		//System.out.println("Aqui no se llega");
	} }
class ClaseB{
	public void metodoB() throws MiExcepcion{
		ClaseC c=new ClaseC();
		try{
			c.metodoC();
		}catch(MiExcepcion e){
			System.out.println("B-> captura excepcion");
			System.out.println("B-> propaga excepcion");
			throw e;
		} } }
class ClaseA{
	public void metodoA() throws MiExcepcion{
		ClaseB b=new ClaseB();
		b.metodoB();
		System.out.println("A-> propaga excepcion"); //no se
ejecuta
	} }
public class PruebaExcepcion{
	public static void main(String [] args){
		ClaseA a=new ClaseA();
		try{
			a.metodoA();
		}catch(MiExcepcion e){
			System.out.println("main-> captura excepcion");
			System.out.println(e.getMessage());
		} } }
```

La salida es:

```
C-> lanza excepcion
B-> captura excepcion
B-> propaga excepcion
main-> captura excepcion
Casi que no
```

Lo + recomendado

No dejes de leer…

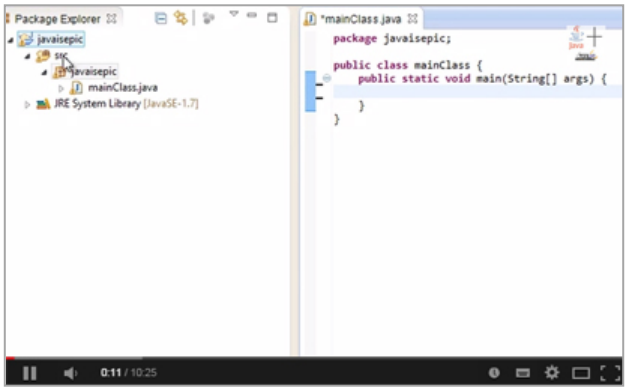
Gestionar y crear excepciones

En la página web de Oracle puedes encontrar más información sobre cómo gestionar y crear excepciones.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

No dejes de ver…

Cómo gestionar excepciones en Java



Vídeo en el que se explica el uso de los bloques `try` y `catch` y de la sentencia `throws`.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=UCHRLzCgVfc

Excepciones en Java

Definición y explicación de cómo tratar las excepciones en Java.

Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:
http://www.youtube.com/watch?v=cPYXto03Xp0

+ Información

A fondo

Excepciones de Oracle

Ejercicios sobre excepciones de Oracle, te va a servir para saber si has comprendido correctamente los conceptos.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://docs.oracle.com/javase/tutorial/essential/exceptions/QandE/questions.html

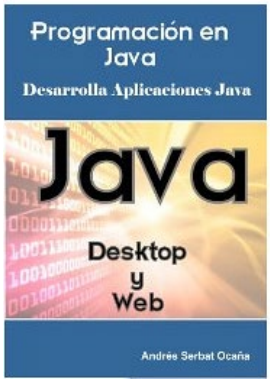
Excepciones desde un punto de vista menos técnico

Documento que hace un análisis de cómo se pueden entender las excepciones desde un punto de vista menos técnico.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.jite.org/documents/Vol11/JITEv11IIPp327-352Rashkovits1157.pdf

Programación en Java: Desarrolla Aplicaciones Java

Serbat, A. (2014). *Programación en Java: Desarrolla Aplicaciones Java* [Ed. Kindle]. Recuperado de Amazon.com



Libro en el que se hace un análisis de cómo desarrollar aplicaciones en Java para escritorio y web.

Test

1. Una excepción es:

A. Un suceso en tiempo de ejecución.

B. Un suceso en tiempo de compilación.

C. Un error en tiempo de compilación.

D. Un error de sintaxis.

2. Dentro del funcionamiento de las excepciones:

A. Siempre son lanzadas por el sistema.

B. Pueden ser lanzadas por el usuario.

C. No pueden ser lanzadas por el usuario.

D. El usuario las puede lanzar pero no crear nuevas excepciones.

3. Cuando una sentencia lanza una excepción:

A. El bloque `try….catch` es opcional.

B. El bloque `try…catch` es obligatorio.

C. El bloque `try…catch` se usa para propagar la excepción.

D. El bloque `try…catch` se utiliza solo en la última sentencia que lanza la excepción.

4. Los tipos de excepciones que existen son:

A. `Error` y `Exception.`

B. `Error`, `exception` y problema.

C. Todos heredan de `Error.`

D. Todos heredan de `Exception.`

5. Para poder lanzar una excepción:

A. El objeto debe derivar de la clase `Throwable`.

B. El objeto debe derivar de cualquier otra clase.

C. El objeto puede ser de cualquier tipo.

D. El objeto debe ser instanciado con `Thowable.`

6. Las excepciones siempre:

A. Deben ser gestionadas donde se generan.

B. Pueden propagarse al objeto que invocó el método que lo ha generado.

C. Si no se propagan mueren.

D. Siempre se deben propagar.

7. Un bloque `try`:

A. Obligatoriamente debe tener varios `catch`.

B. Puede tener varios `finally`.

C. Puede tener uno o varios `catch`.

D. Un `try` puede dejarse sin cerrar.

8. La sentencia `finally`:

A. Es similar a `catch`, hace lo mismo.

B. Puede aparecer varias veces en un `try`.

C. Puede usarse en lugar de `try`.

D. Tiene el conjunto de sentencia que se ejecutan siempre que hay una excepción.

9. Viendo el siguiente código ¿existe algún error?

```
try{

   }catch(Exception e){

   }catch(IOException e){

   }
```

A. Tiene errores de compilación.

B. Un try no puede tener varios `catch`.

C. El orden de los `catch` no es correcto.

D. Debería aparecer un bloque `finally`.

10. Cuando creamos un bloque `try….catch`:

A. En los `catch` se deben poner las capturas de excepciones desde la más específica a la más general.

B. Se deben indicar desde la más general a la más específica.

C. El orden de los `catch` da igual.

D. Siempre se debe incluir en primer lugar el `finally`.