Excepciones: detección y tratamiento

Introducción

- Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias. Cuando no es capturada por el programa, es capturada por el gestor de excepciones por defecto que retorna un mensaje y detiene el programa.
- Las excepciones en Java son objetos de clases derivadas de la clase base **Exception** que a su vez es una clase derivada de la clase base **Throwable**

Capturar excepciones

- Para capturar una excepción se utiliza el bloque **try-catch**.
- Se encierra en un bloque **try** el código que puede generar una excepción, este bloque va seguido por uno o más bloques **catch**.
- Cada bloque **catch** especifica el tipo de excepción que puede atrapar y contiene un manejador de excepciones.
- Después del último bloque **catch** puede aparecer un bloque **finally** (opcional) que siempre se ejecuta haya ocurrido o no la excepción; se utiliza el bloque **finally** para cerrar ficheros o liberar recursos del sistema después de que ocurra una excepción:

```
try {
	//Código que puede generar excepciones

} catch (excepcion1 e1) {

	//manejo de la excepcion1
} catch (excepcion2 e2) {

	//manejo de la excepcion2
}
//etc …..
finally {

	//Se ejecuta después de try o catch
}
```

- Para capturar cualquier excepción que se pueda producir utilizamos la clase **Exception**. Si se usa habrá que ponerla al final de la lista de manejadores para evitar que los manejadores que vienen después queden ignorados.

```
try {
        //Código que puede generar excepciones

} catch (NumberFormatException e) {
        //manejo de la excepcion
} catch (ArithmeticException ex) {
        //manejo de la excepcion
} catch (ArrayIndexOutOfBoundsException ex) {
        //manejo de la excepcion
} catch (Exception ex) {
        //tratamiento si se produce cualquier otra excepción
finally {
        //Se ejecuta haya o no excepción
}
```

- Para obtener más información sobre la excepción se puede llamar a los métodos de la clase base **Throwable**, algunos son:
  - **String getMessage()** → Devuelve la cadena de error del objeto
  - **String toString()** → Devuelve una breve descripción del objeto
  - **void printStackTrace()** → Visualiza el objeto y la traza de pila de llamadas lanzada.

Veamos un ejemplo:

```
public class ejemploExcepciones {
	public static void main (String[] args) {
		int [] arraynum = new int [4];
		try {
			arraynum[10] = 20;
		} catch (Exception ex) {
				System.err.println ("toString	=> " + ex.toString());
				System.err.println ("getMessage	=> " +
ex.getMessage());
			ex.printStackTrace();
		}
		finally {
					System.out.println ("SE EJECUTA SIEMPRE");
			}
			}
		}
```

El ejemplo anterior tendrá la siguiente salida:

```
toString    => java.lang.ArrayIndexOutOfBoundsException: 10
getMessage    => 10
java.lang.ArrayIndexOutOfBoundsException: 10
        at ejemploExcepciones.main(ejemploExcepciones.java:5)
SE EJECUTA SIEMPRE
```

- Una sentencia **try** puede estar dentro de un bloque de otra sentencia **try.** Si la sentencia interna no dispone de un manejador **catch**, se busca el manejador en las sentencias **try** más externas.

Especificar excepciones

- Para especificar excepciones utilizamos la palabra clave **throws** seguida de la lista de todos los tipos de excepciones potenciales; si un método decide no gestionar una excepción (mediante **try-catch**), debe especificar que puede lanzar esa excepción. El siguiente ejemplo indica que el método main() puede lanzar las excepciones IOException y ClassNotFoundException:

```
public static void main (String [] args) throws IOException, ClassNotFoundException { ...
```

- Aquellos métodos que pueden lanzar excepciones, deben saber cuáles son esas excepciones en su declaración. Una forma de averiguarlo es compilando el programa. Al compilarlo aparecerán los errores que pueden ser lanzados.