Java Mérida Academy | Junio 2025
Alumno: Raúl Antonio Villanueva Hernández

Excepciones

Una excepción es un evento inesperado que ocurre durante la ejecución de un programa, es decir, en tiempo de ejecución, y que interrumpe el flujo normal de las instrucciones del programa. Las excepciones pueden ser divididas en dos tipos: las marcadas (checked) y las no marcadas (unchecked).

Marcadas: se comprueban en tiempo de compilación y obligan al programador a manejarlas.

No Marcadas: se comprueban en tiempo de ejecución y no requieren un manejo por parte del programador en tiempo de compilación.

Marcadas	**No Marcadas**
Se comprueban en tiempo de compilación y se emplean try-catch para manejarlas.	Se comprueban en tiempo de ejecución.
Deriva de Exception.	Deriva de RuntimeException.
Causado por factores externos como conexiones a bases de datos.	Causado por errores de programación.
Ejemplos: • ClassNotFoundException • InterruptedException • IOException • InstantiationException • SQLException • FileNotFoundException	Ejemplos: • ArithmeticException • ClassCastException • NullPointerException • ArrayIndexOutOfBoundsException • ArrayStoreException • IllegalThreadStateException

Estructura try-with-resources

El propósito de la estructura try-with-resources es simplificar el cierre automático de recursos, lo que solía hacerse dentro del bloque finally. Un recurso es un objeto que debe cerrarse una vez que el programa ha terminado con él. Esta estructura garantiza que cada recurso se cierre al final de la instrucción. Cualquier objeto que implemente la interfaz java.lang.AutoCloseable, lo cual incluye todos los que implementan java.io.Closeable, se puede usar como recurso en un bloque try-with-resources.

Multicatch

A partir de Java 7 se permite que un solo bloque catch puede manejar más de un tipo de excepción. Esta característica permite reducir la duplicación de código. Se usa “|” para separar los tipos de excepciones en una sentencia catch. Si se usa multicatch, el parámetro de dicha estructura es final, por ejemplo:

```
    catch (IOException|SQLException ex) {
        logger.log(ex);
        throw ex;
    }
```

En el código anterior “ex” es final, esto quiere decir que no se podría reasignar ya que se generaría un error de compilación.