 



Conceptos fundamentales de Java 6-2: Manejo de errores Actividades prácticas

# Objetivos de la lección:

* Descripción de distintos tipos de errores que pueden ocurrir y cómo se manejan en Java
* Descripción de excepciones utilizadas en Java
* Determinación de las excepciones que se arrojan para una clase base
* Escribir un código para manejar una excepción arrojada mediante el método de una clase base

# Vocabulario:

Identifique el término correspondiente a cada una de las definiciones a continuación.

|  |  |
| --- | --- |
| Try | Palabra clave en Java que indica que el bloque siguiente de código maneja una excepción especificada. |
| Checked Exception | Una excepción que se maneja de manera opcional |
| Unchecked exception | Una excepción que se DEBE manejar. |
| Syntax error | Indica que hay un problema con la interpretación de su programa. |
| Throw | Evita que el intérprete ejecute el resto del código hasta que éste encuentre una palabra catch. |
| Encoding error | Error que indica un problema con el formato de codificación. |
| Runtime exception | Error que se produce mientras se ejecuta el programa, también denominado excepción. |
| Logical error | Error que se produce como resultado de una lógica de programación incorrecta. |
| Catch | Bloque de código que maneja excepciones al tratar casos donde se arroja una excepción. |
| Exception | Errores que se producen durante el tiempo de ejecución y que pueden ser corregidos o manejados por el código. |

# Inténtelo/resuélvalo:

1. Describa la diferencia entre error de sintaxis, error lógico y excepción.

* R// **Error de sintaxis:** Este tipo de error ocurre cuando se escribe código que no sigue las reglas gramaticales del lenguaje de programación. El compilador nos indicará que hay un error y no podrá ejecutar el programa hasta que se haya corregido el error. Por ejemplo, si olvidamos cerrar una llave de un bloque de código en Java, el compilador nos indicará que hay un error de sintaxis.
* **Error lógico:** Este tipo de error ocurre cuando el programa se ejecuta correctamente, pero no produce los resultados esperados. Por ejemplo, si un programa debería sumar dos números y en lugar de eso los resta, estamos frente a un error lógico.
* **Excepción:** Una excepción es un evento que ocurre durante la ejecución de un programa y que interrumpe su flujo normal. Las excepciones se utilizan para manejar situaciones inesperadas, las excepciones se manejan mediante código adicional en el programa que puede detectar la excepción y tomar medidas para manejarla adecuadamente.

1. ¿Cuál es el problema del siguiente código? Debe imprimir " Hello World! " cuatro veces a la pantalla.

String str = “Hello World”;

for(int i = 0; i < 4; i++);

{

System.out.println(str); str+= ”!”;

}

R// El código no tiene problemas, pero al imprimir el “Hello World!” solo imprime uno porque tiene un ; de más en la línea de:

for(int i = 0; i < 4; i++);

por lo cual imprime solo un “Hello World!”, si le quitamos él ; ahora sí imprime los 4 “Hello World!”.

1. Describa una excepción que haya experimentado antes en su programa. Explique de qué manera se podría manejar con un bloque de código try/catch.

**package** punto3y4;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**public** **class** Punto3 {

**public** **static** **void** main(String[] args) {

**try** {

File file = **new** File("ruta/del/archivo.txt");

Scanner scanner = **new** Scanner(file);

**while** (scanner.hasNextLine()) {

String line = scanner.nextLine();

System.***out***.println(line);

}

scanner.close();

} **catch** (FileNotFoundException e) {

System.***out***.println("El archivo no se ha encontrado: " + e.getMessage());

}

}

}

El bloque try/catch es una forma de manejar excepciones que pueden ocurrir durante la ejecución del programa. El código que podría causar una excepción se coloca dentro del bloque try, y el código que maneja la excepción se coloca dentro del bloque catch. Esto permite que el programa continúe ejecutándose y maneje las excepciones de manera adecuada en lugar de fallar y salir del programa.

1. Escriba un segmento de código que tenga:
   1. Un error de sintaxis.
   2. Un error lógico.
   3. Una excepción.

R// A) **package** punto4;

**public** **class** Punto4\_A {

//Ejemplo de syntax error

**public** **static** **void** main(String[] args) {

**int** x = 5;

**int** y = 10

**int** z = x + y;

System.out.println("La suma es: " + z);

}

}

B) **package** punto4;

**public** **class** Punto4\_B {

//Ejemplo de error lógico

**public** **static** **void** main(String[] args) {

**int** x = 5;

**int** y = 10;

**int** z = x \* y;

System.***out***.println("La suma es: " + z);

}

}

C) **package** punto4;

**public** **class** Punto4\_C {

//Ejemplo de una excepción

**public** **static** **void** main(String[] args) {

**int** x = 5;

**int** y = 0;

**int** z = 0;

**try** {

z = x / y;

} **catch** (ArithmeticException e) {

System.***out***.println("Se ha producido una excepción: " + e.getMessage());

}

System.***out***.println("El resultado es: " + z);

}

}

1. ¿Cuál es la diferencia entre una excepción marcada y una excepción sin marcar?

R// la diferencia principal entre una excepción marcada y una excepción no marcada en Java es que la excepción marcada debe ser declarada o manejada, mientras que la excepción no marcada no necesita ser declarada o manejada.