Píldora Formativa: Manejo de Excepciones en Java

Introducción

El manejo de excepciones es una herramienta fundamental en Java para garantizar que nuestros programas puedan responder de manera adecuada a situaciones inesperadas. Una excepción es un evento que interrumpe el flujo normal de la ejecución de un programa y ocurre debido a condiciones que el programa no puede manejar directamente, como un intento de dividir por cero,
acceder a un índice fuera de los límites de un array o tratar de abrir un archivo inexistente.

Esta guía cubrirá las definiciones esenciales, ejemplos prácticos y buenas prácticas para trabajar con excepciones en Java.

1. ¿Qué son las Excepciones?

Una excepción es un objeto que representa un error o un comportamiento anómalo que ocurre durante la ejecución de un programa. Cuando una excepción ocurre, Java crea un objeto que contiene información sobre el error y detiene el flujo normal del programa, a menos que el programador haya preparado código para manejarla.

Las excepciones se generan de dos maneras principales:
- Por el propio entorno de ejecución de Java (Runtime Environment).
- Manualmente, mediante la palabra clave `throw`.

Tipos de Excepciones

1. Checked Exceptions (Excepciones verificadas):

- Son detectadas en tiempo de compilación. El programador debe manejar estas excepciones obligatoriamente usando bloques `try-catch` o declarándolas en el encabezado del método con la palabra clave `throws`.
- Ejemplo: `IOException`, `SQLException`.

2. Unchecked Exceptions (Excepciones no verificadas):

- Son detectadas en tiempo de ejecución. No es obligatorio manejarlas, pero es buena práctica hacerlo si se pueden anticipar.
- Ejemplo: `NullPointerException`, `ArithmeticException`.

3. Errores (`Error`):

- Representan problemas graves relacionados con el entorno de ejecución, como falta de memoria o desbordamiento de pila. Por lo general, no se manejan directamente.
- Ejemplo: `StackOverflowError`, `OutOfMemoryError`.

2. Manejo de Excepciones en Java

El manejo de excepciones en Java se realiza mediante los bloques `try`, `catch`, y `finally`. Estas palabras clave permiten capturar y gestionar situaciones excepcionales de forma controlada.

Bloque `try-catch`

El bloque `try` contiene el código que puede generar una excepción, mientras que el bloque `catch` maneja el tipo específico de excepción que se produce.

Bloque `finally`

El bloque `finally` se ejecuta siempre, independientemente de si ocurre una excepción o no. Es útil para liberar recursos, cerrar archivos o limpiar código.

3. Lanzar Excepciones

El programador puede generar (lanzar) una excepción manualmente utilizando la palabra

clave `throw`. Además, los métodos pueden declarar que lanzan excepciones utilizando `throws`.

Ejemplo con `throws`:

```
public static int dividir(int numerador, int denominador) throws ArithmeticException {
   if (denominador == 0) {
      throw new ArithmeticException("No se puede dividir por cero");
   }
   return numerador / denominador;
}
```

4. Crear Excepciones Personalizadas

Es posible definir nuestras propias excepciones extendiendo la clase `Exception` o `RuntimeException`.

```
public class MiExcepcion extends Exception {
   public MiExcepcion(String mensaje) {
      super(mensaje);
   }
}
```

5. Principales Excepciones en Java

Java proporciona una amplia gama de excepciones predefinidas en su biblioteca estándar. Aquí tienes algunas de las más comunes:

Excepciones Comunes

1. **ArithmeticException**: Ocurre cuando se realiza una operación matemática inválida, como dividir por cero.
    - Ejemplo: `int resultado = 10 / 0;`
2. **NullPointerException**: Se lanza cuando se intenta acceder a un objeto que es `null`.
    - Ejemplo: `String texto = null; texto.length();`
3. **ArrayIndexOutOfBoundsException**: Se lanza cuando se intenta acceder a un índice fuera de los límites de un array.
    - Ejemplo:
      ```
      int[] numeros = {1, 2, 3};
      System.out.println(numeros[5]);
      ```
4. **ClassCastException**: Se lanza cuando se intenta convertir un objeto de una clase a otra clase incompatible.
    - Ejemplo: `Object obj = new Integer(10); String str = (String) obj;`
5. **InputMismatchException**: Ocurre cuando el tipo de dato proporcionado no coincide con el esperado.
    - Ejemplo:
      ```
      Scanner scanner = new Scanner(System.in);
      int numero = scanner.nextInt(); // Introducir texto en lugar de un número.
      ```
6. **IOException**: Representa un error relacionado con la entrada y salida de datos, como problemas al leer un archivo.
    - Ejemplo:
      ```
      FileReader reader = new FileReader("archivo_inexistente.txt");
      ```
7. **FileNotFoundException**: Una subclase de `IOException`, ocurre cuando no se encuentra un archivo especificado.
    - Ejemplo: `new FileInputStream("archivo.txt");`
8. **NumberFormatException**: Se lanza cuando se intenta convertir una cadena en un número, pero la cadena no tiene un formato válido.
    - Ejemplo: `int numero = Integer.parseInt("texto");`
9. **IllegalArgumentException**: Indica que un argumento pasado a un método no es apropiado.
    - Ejemplo: `Thread.sleep(-100); // Tiempo negativo`

10. **IllegalStateException**: Se lanza cuando el estado de un objeto no permite la operación solicitada.

- Ejemplo: `Scanner scanner = new Scanner(System.in); scanner.close(); scanner.next();`

7. Manejo de la Pila de Excepciones

¿Qué es el Stack Trace?

La pila de excepciones, o **stack trace**, es una representación jerárquica del flujo de métodos que se ejecutaron antes de que ocurriera una excepción. Esta información es crucial para depurar programas, ya que muestra:

1. **El tipo de excepción lanzada.**
2. **El mensaje asociado a la excepción.**
3. **El rastro de métodos llamados antes del error.**

Cuando ocurre una excepción, Java automáticamente genera el stack trace. Este puede visualizarse mediante los métodos:

- `Throwable.printStackTrace()`: Imprime la pila de excepciones en la consola.
- `Throwable.getStackTrace()`: Devuelve un array con el rastro de la pila para análisis programático.

Ejemplo de Uso del Stack Trace

```
public class StackTraceEjemplo {
    public static void main(String[] args) {
        try {
            metodo1();
        } catch (Exception e) {
            System.out.println("Se capturó una excepción: " + e.getMessage());
            System.out.println("Rastro de la pila:");
            e.printStackTrace();
        }
    }

    public static void metodo1() {
        metodo2();
    }

    public static void metodo2() {
        throw new RuntimeException("Error simulado");
    }
}
```

Salida:

```
Se capturó una excepción: Error simulado
Rastro de la pila:
java.lang.RuntimeException: Error simulado
    at StackTraceEjemplo.metodo2(StackTraceEjemplo.java:15)
    at StackTraceEjemplo.metodo1(StackTraceEjemplo.java:11)
    at StackTraceEjemplo.main(StackTraceEjemplo.java:6)
```