Excepciones en Java

Las excepciones en Java son eventos anómalos que pueden ocurrir durante la ejecución de un programa y que alteran el flujo normal de ejecución. Estos eventos representan situaciones inesperadas o errores que deben ser manejados de manera adecuada para garantizar que el programa continúe ejecutándose para así evitar interrupciones.

Tipos de excepciones en Java

Las excepciones se dividen en tres categorías principales, cada una heredada de la clase base Throwable. Estas categorías son:

- **Checked exceptions**: Son excepciones que se deben declarar en la firma del método o capturar explícitamente en un bloque try-catch. Si una excepción comprobada no se maneja correctamente, el código no se compilará. Estas excepciones se heredan de la clase Exception.
- **Unchecked exceptions**: Son excepciones que no están obligadas a ser manejadas explícitamente. Estas excepciones ocurren durante la ejecución del programa y no se requiere que sean declaradas en la firma del método o capturadas mediante un bloque try-catch. Se heredan de la clase RuntimeException.
- **Errors**: Son problemas graves que generalmente están fuera del control del programador y no deben manejarse explícitamente. Estos errores indican problemas serios que deberían detener la ejecución del programa. Se heredan de la clase Error.

Manejo de excepciones en Java

El manejo de excepciones en Java es fundamental. A continuación, veremos cómo manejar excepciones utilizando bloques try, catch y finally. El bloque try se utiliza para envolver el código que puede generar una excepción. Si se produce una excepción dentro del bloque try, se captura en uno o más bloques catch asociados.

Por otro lado, el bloque finally se utiliza para ejecutar código que debe ejecutarse, independientemente de si se ha producido una excepción o no. Se usa para liberar recursos que deben cerrarse, como conexiones a bases de datos o archivos.

```
try {
    // Código que puede generar una excepción
    // ...
} catch (ArithmeticException e) {
    // Captura de excepción de división por cero
    System.out.println("Ocurrió una excepción de división por cero: " + e.getMessage());
} catch (Exception e) {
    // Captura de cualquier otra excepción
    System.out.println("Ocurrió una excepción: " + e.getMessage());
}
```

Multicatch

También existe en Multicatch que apareció en Java 7 lo que nos permite manejar varias excepciones en un mismo catch, no es necesario que las excepciones tengan herencia en común, el orden no altera el resultado y la excepción puede retorna cada uno de los tipos.

```
try {
    // Código que puede lanzar múltiples excepciones
} catch (IOException | SQLException e) {
    // Manejo común para ambas excepciones
    System.out.println("Ocurrió un error de I/O o de base de datos: " + e.getMessage());
}
```

Try with resources

En Java, el try with resources es una estructura introducida en **Java 7** que facilita el manejo automático de recursos, como archivos, sockets, conexiones de base de datos entre otros.

Sirve para cerrar automáticamente los recursos abiertos aun cuando ocurre una excepción dentro del bloque try. Esto permite cerrar correctamente los recursos aun cuando una excepción es lanzada.

Referencias

Jorge Lopez Blasco, (2023, 23 de noviembre). *Introduccion a POO en Java: Excepciones.* https://openwebinars.net/blog/introduccion-a-poo-en-java-excepciones/

Oracle. (2014). *The Java™ Tutorials: Essential Classes – Exceptions*. Oracle Corporation. https://docs.oracle.com/javase/tutorial/essential/exceptions/

Baeldung. (2020). *Java Exception Handling - A Complete Guide*. Baeldung. https://www.baeldung.com/java-exceptions