Excepciones y principio “Single Responsibility”

Grupo 3

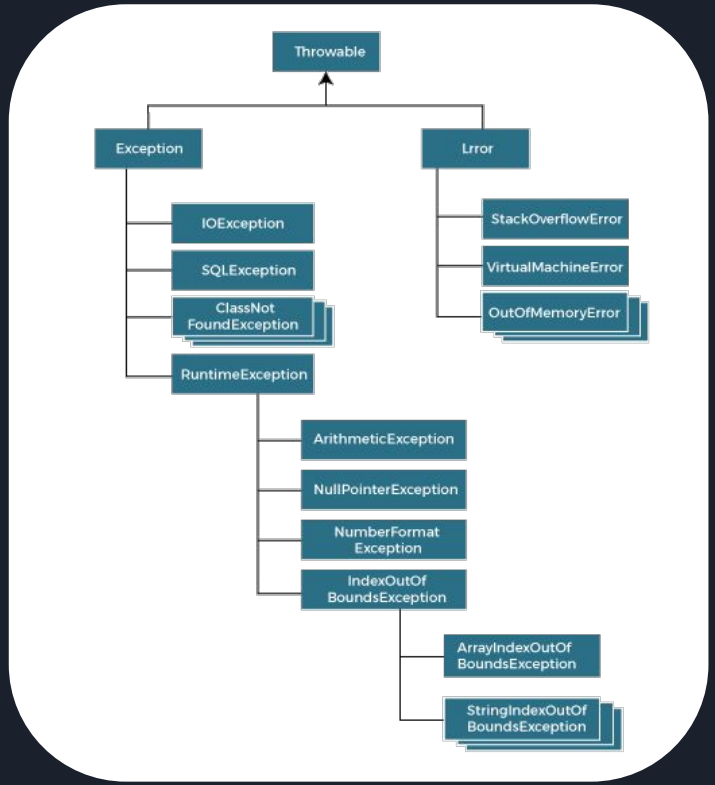
Amanda Jesus Montero Zuniga
Mynell Jemuel Myers Hall
Paolo José Ruiz Zuniga
Cristina Urbina Cespedes

¿Qué son las excepciones?

Los errores en el tiempo de ejecución son conocidos como “excepciones”.

Ocurren principalmente cuando ocurre un error en la ejecución de las instrucciones del programa.

Tipos de excepciones



Manejo de excepciones

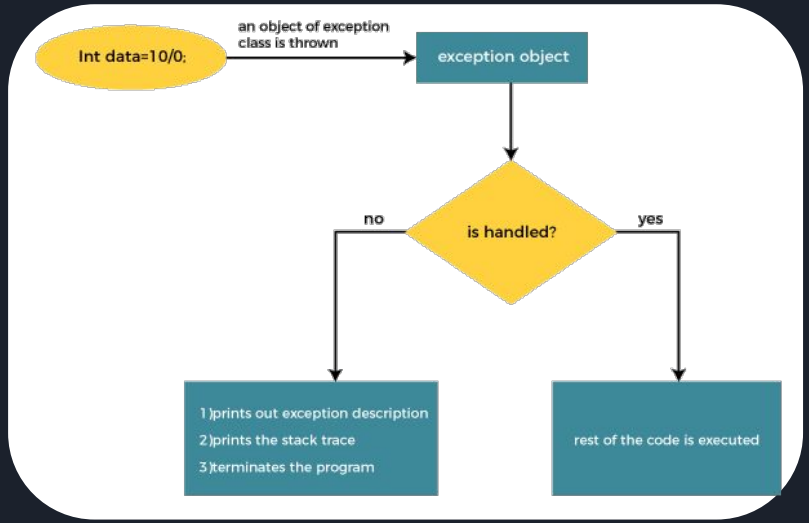
Las excepciones pueden ser manejadas por las siguientes instrucciones en Java:

- ➔ Try
- ➔ Catch
- ➔ Finally
- ➔ Throw

Excepciones try...catch

El par de bloques “try” y “catch” se utilizan en conjunto para el manejo de excepciones en Java.

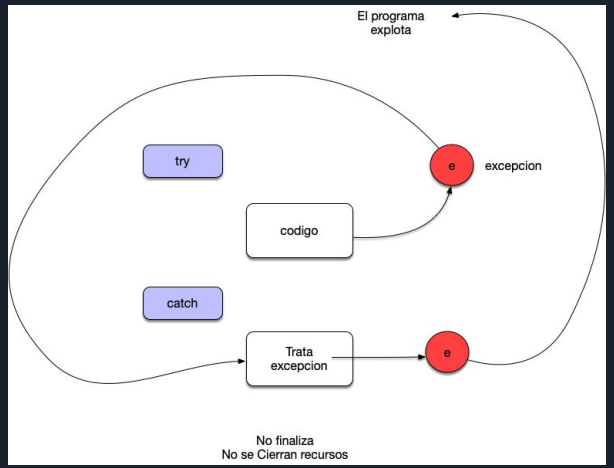
- Bloque “try”: Contiene las instrucciones donde es posible que ocurra una excepción.
- Bloque “catch”: Maneja las excepciones, cuando una excepción ocurre en el bloque “try” es su bloque “catch” asociado el que maneja dicha excepción.



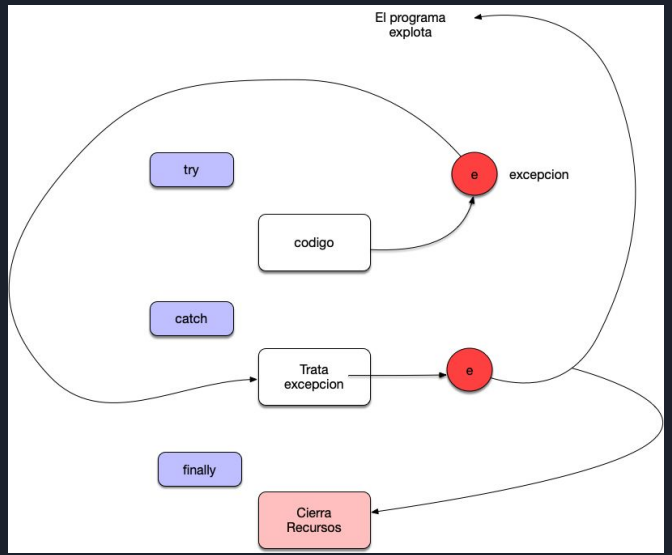
Excepción finally

El propósito de una excepción Finally es ejecutar algo después de que se “catchea” un error o no haya error, y ejecutar un código para intentar minimizar el impacto del error.

Sin Finally



Con Finally



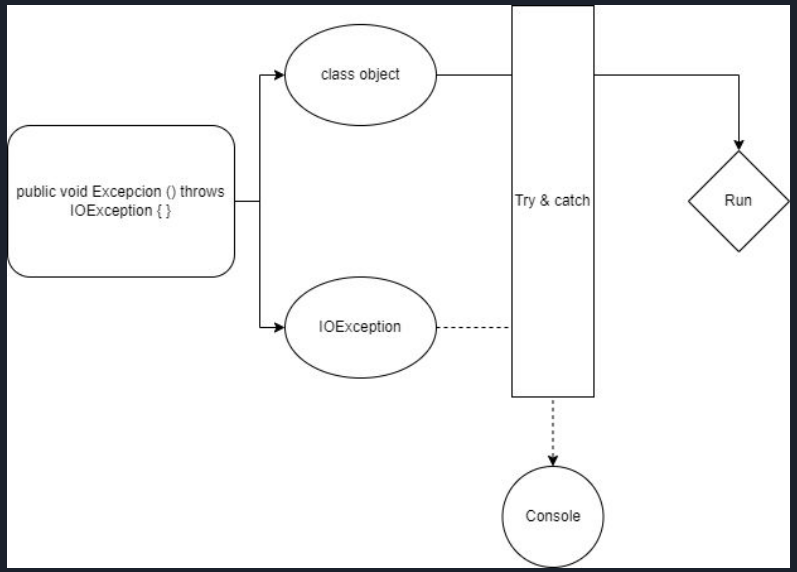
Excepción throw

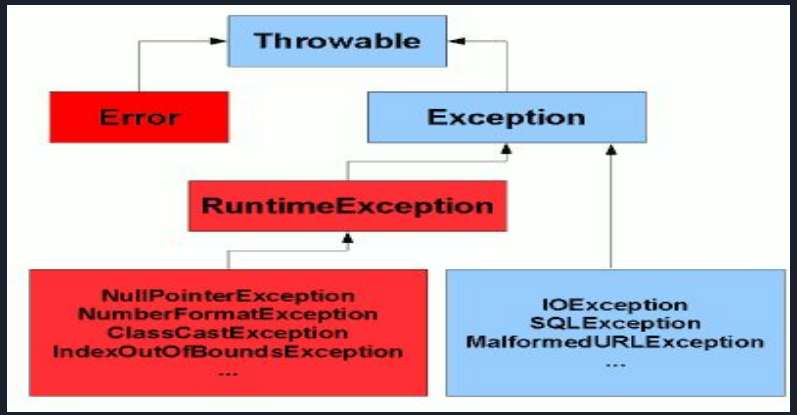
Con el uso de throw y el bloque try-catch en el método donde posiblemente se genere el error, "avisamos" a Java, capturamos el error y prevenimos que se congele el programa.

La función de Throws es permitir al método que lo incluye, hacer dos cosas:

1- Si todo va bien, devolvernos un objeto.

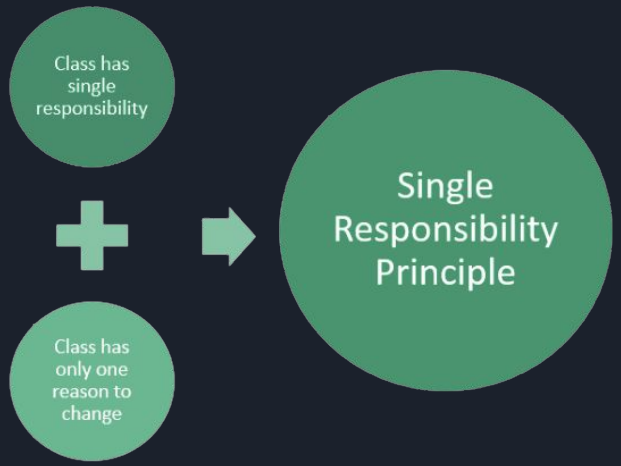
2- Si surge algún imprevisto, en vez de realizar su trabajo, va a lanzar una clase IOException, para capturar el error y poder continuar con el resto del programa.





Principio Single Responsibility

Establece que una clase debe tener una sola razón para cambiar. Esto significa que una clase debe tener solo una responsabilidad o una razón para existir.



Principio Single Responsibility

Código más **mantenible y comprensible**, asegurando que cada clase tenga un propósito claro y específico.

Para cumplir con el Principio de Responsabilidad Única, las responsabilidades de una clase deben identificarse y separarse en diferentes clases. Cada una de ellas debe entonces ser responsable de **sólo un aspecto específico** de la funcionalidad general.

Ventajas

1. Código más mantenible.
2. Reutilización de código.
3. Reducción de la complejidad.
4. Mejora en la legibilidad.
5. Reducción de la probabilidad de errores.

Desventajas

1. Aumento en el número de clases.
2. Mayor complejidad en las interacciones entre las clases.
3. Mayor tiempo de desarrollo.
4. Más difícil de implementar.
5. Aumento en el esfuerzo de pruebas(testing).

Bibliografía

Caules, C. A. (2023, 13 enero). Java Finally y el cierre de recursos. Arquitectura Java. https://www.arquitecturajava.com/java-finally-y-el-cierre-de-recursos/

Shekhawat, S. S. (s. f.). SOLID - Single Responsibility Principle With C#. https://www.c-sharpcorner.com/article/solid-single-responsibility-principle-with-c-sharp/

Singh, C. (2022, 13 septiembre). Try Catch in Java – Exception handling. BeginnersBook. https://beginnersbook.com/2013/04/try-catch-in-java/

Richard. (2017, 2 junio). Excepciones (Exception) en Java, con ejemplos. Jarroba. https://jarroba.com/excepciones-exception-en-java-con-ejemplos/

Ungureanu, V. (2021, 10 diciembre). Single Responsibility Principle - Vlad Ungureanu. Medium. https://medium.com/@learnstuff.io/single-responsibility-principle-ad3ae3e264bb

Exception Handling in Java | Java Exceptions - javatpoint. (s. f.). www.javatpoint.com. https://www.javatpoint.com/exception-handling-in-java

Java try-catch - javatpoint. (s. f.). www.javatpoint.com. https://www.javatpoint.com/try-catch-block

Throw y throws en Java ¿Qué son y cómo se usan? (s. f.). https://open-bootcamp.com/cursos/java/throw-y-throws