**Exception**An exception is an unexpected event that occurs during program execution, disrupting the normal flow of control. It can be caused by various reasons, such as invalid input, resource exhaustion, or network errors.

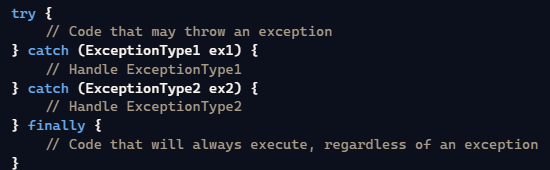
**Handling**  
It is a mechanism to handle runtime errors such as classNotFound, IO, SQL etc

**Error**  
Our code is not responsible to raise error (non-recoverable)  
Cause due to lack of system resources (Memory problem, hardware problem)

* Example: OutOfMemoryError, StackOverflowError

**Checked**   
These are checked at compile-time. You must handle them using try-catch blocks or declare them using the throws keyword.

* Example: IO Exception, SQL Exception

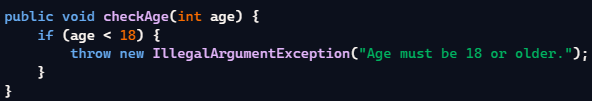


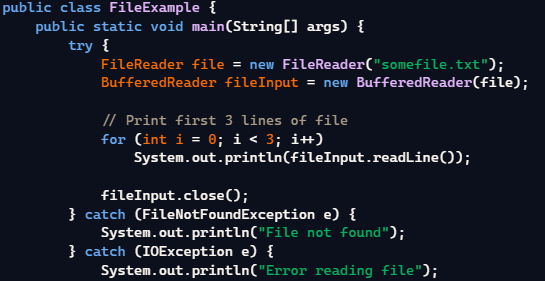
Throws : Used to duck the exception

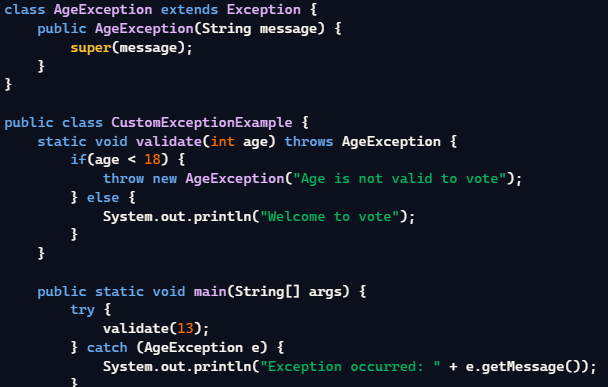


**Unchecked**  
These are not checked at compile-time but occur at runtime. They are subclasses of Runtime Exception.

* Example: NullPointerException, ArrayIndexOutOfBoundsException

**Throw**   
It is used to Re-Throw an exception (Throwing the handled exception object)  
Lines below the throw keyword will not be executed  


**Example 1: Handling a File Not Found Exception  
**

**Example 2: Throwing and Catching Custom Exceptions  
**

Interview Questions

**Level 1: Beginner**

**1. What are exceptions in Java, and how are they different from errors?**

Exceptions are events that disrupt the normal flow of a program. They are usually conditions that a program can handle, while errors are serious problems that applications shouldn't try to catch.

* **Example:** IOException vs OutOfMemoryError.

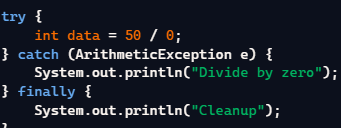
**2. Explain the difference between checked and unchecked exceptions.**

Checked exceptions are checked at compiletime. Examples include IOException.

Unchecked exceptions are checked at runtime and are subclasses of RuntimeException, like NullPointerException.

* **Example:** IOException vs ArithmeticException.

**3. How does the try-catch-finally block work in Java?**

The try block contains code that may throw an exception.   
The catch block handles the exception.   
The finally block executes regardless of an exception.  


**Level 2: Intermediate**

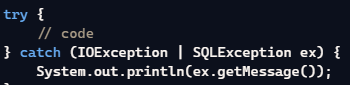
**1. Can you explain the difference between the throw and throws keywords?**

The throw keyword is used to explicitly throw an exception from a method or any block of code.   
The throws keyword is used in a method signature to declare exceptions that might be thrown by the method.  


**2. What is a custom exception, and how would you create one in Java?**

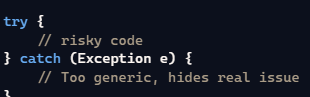
A custom exception is a user-defined exception to handle specific application scenarios. You create one by extending the Exception class  

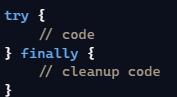

**3. How do you handle multiple exceptions in a single catch block?**

Use the multi-catch feature introduced in Java 7 by separating exception types with a vertical bar |.  


**Level 3: Advanced**

**1. Why should you avoid catching the Exception or Throwable classes directly?**

Catching Exception or Throwable directly can hide programming errors, making debugging difficult and potentially leading to unforeseen issues.  
  
  
**2. How does the**finally**block work, and when would you use it?**

The finally block executes whether an exception is thrown or not. Use it for cleanup activities like closing file streams or database connections.  


**3. Explain exception chaining in Java.**

Exception chaining allows one exception to be caused by another. You can set the cause of   
an exception using the constructor or initCause method.  
