# 🔹 What is an Exception?

An **exception** is an event that disrupts the normal flow of the program. It is an **object** that is thrown at runtime. It occurs when something unexpected happens, like accessing an invalid index, dividing by zero, or trying to open a file that does not exist.

**🔸 Categories:**

| **Type** | **Description** |
| --- | --- |
| **Checked Exception** | Checked at compile-time. Must be handled. e.g. IOException, SQLException |
| **Unchecked Exception** | Runtime exceptions. e.g. NullPointerException, ArithmeticException |
| **Error** | Serious issues, not meant to be caught. e.g. OutOfMemoryError, StackOverflowError |

java.lang.Object   
 └── java.lang.Throwable   
 ├── java.lang.Exception   
 │ ├── Checked Exceptions (e.g., IOException)   
 │ └── RuntimeException (Unchecked)  
 └── java.lang.Error

**🔹 Key Concepts**

| **Concept** | **Description** |
| --- | --- |
| try | Block of code to monitor for exceptions |
| catch | Block that handles the exception |
| finally | Executes whether or not an exception occurs |
| throw | Used to explicitly throw an exception |
| throws | Declares exceptions a method might throw |

# 🔹 Custom Exception

class MyException extends Exception {  
 public MyException(String message) {  
 super(message);  
 }  
}

public class Test {  
 public static void main(String[] args) throws MyException {  
 throw new MyException("Custom exception thrown!");  
 }  
}

# 🔹 Multiple Catch Blocks

try {  
int[] arr = new int[5];  
 System.out.println(arr[10]);  
} catch (ArithmeticException e) {  
 System.out.println("Arithmetic Exception");  
} catch (ArrayIndexOutOfBoundsException e) {  
 System.out.println("Array Index Out Of Bounds");  
} catch (Exception e) {  
 System.out.println("General Exception");  
}

# 🔹 try-with-resources (Java 7+)

For auto-closing resources like files, streams, etc.

try (BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {  
 System.out.println(br.readLine());  
 } catch (IOException e) {  
 e.printStackTrace();  
}

# 🔹 Best Practices

* Catch **specific** exceptions first.
* Avoid empty catch blocks.
* Use **finally** to close resources (or prefer try-with-resources).
* Do **not catch Throwable** unless absolutely necessary.
* Document exceptions with @throws in Javadoc.