
✅ Checked vs Unchecked Exceptions in Java

🔷 What is an Exception?

An **exception** is an event that disrupts the normal flow of the program. In Java, exceptions are objects that are thrown at runtime when something goes wrong.

All exceptions are part of the **Throwable** class hierarchy.

Throwable

├── Exception ← Recoverable (for code errors)

│ ├── Checked Exceptions

│ └── Unchecked Exceptions (RuntimeException)

└── Error ← Unrecoverable (for JVM errors)

✅ 1. Checked Exceptions

🔹 Definition:

Checked exceptions are **checked at compile time**. The compiler forces you to **handle or declare** them using try-catch or throws.

🔹 Examples:

- IOException
- SQLException
- FileNotFoundException
- ClassNotFoundException
- ParseException

🔹 Must Handle or Declare:

You must either:

- use a try-catch block
- or declare using throws

✅ Example:

```
import java.io.*;


public class Example {
   public static void main(String[] args) throws IOException {
      FileReader file = new FileReader("abc.txt"); // Checked Exception
      file.read();
      file.close();
   }
}
```

🔸 If you don't handle or declare the exception, the **compiler will give an error**.

✅ **2. Unchecked Exceptions (Runtime Exceptions)**

🔹 **Definition:**

Unchecked exceptions are **not checked at compile time**, only at **runtime**. The compiler doesn't force you to handle them.

🔹 **Examples:**

- ArithmeticException
- NullPointerException
- ArrayIndexOutOfBoundsException
- IllegalArgumentException
- NumberFormatException

🔹 **Optional to Handle:**

You **can** handle them using try-catch, but it's **not mandatory**.

✅ **Example:**

```
public class Example {
   public static void main(String[] args) {
      int a = 10 / 0;  // Runtime Exception: ArithmeticException
   }
```

}

🔸 The above code compiles successfully but throws a **java.lang.ArithmeticException** at runtime.

📓 Key Differences Between Checked and Unchecked Exceptions

Feature	Checked Exception	Unchecked Exception
Checked at Compile-time?	Yes	No
Must Handle or Declare?	Yes (try-catch or throws)	No
Belongs to	Exception class (excluding RuntimeException)	RuntimeException and its subclasses
Examples	IOException, SQLException	NullPointerException, ArithmeticException
Compiler Enforces?	Yes	No

✅ Good Practice

- **Checked Exceptions** → Use when the program can **recover** from a situation (e.g., missing file, database not connected).
- **Unchecked Exceptions** → Use when the error is due to **programming mistakes** (e.g., null pointer, bad logic).

🧠 Tip for Exam

- IOException, SQLException → Checked
- NullPointerException, ArithmeticException → Unchecked
