Java Exception Handling Overview

1. Unchecked Exceptions

Unchecked exceptions are subclasses of java.lang.RuntimeException. They are not checked at compile time and usually indicate programming bugs.

Common Unchecked Exceptions:

- NullPointerException
- ArrayIndexOutOfBoundsException
- ArithmeticException
- ClassCastException
- IllegalArgumentException
  - NumberFormatException
  - IllegalStateException
- ConcurrentModificationException

Package Hierarchy:

java.lang

RuntimeException

ArithmeticException

ArrayIndexOutOfBoundsException

ClassCastException

IllegalArgumentException

NumberFormatException

IllegalStateException

NullPointerException

ConcurrentModificationException

2. Checked Exceptions

Checked exceptions are subclasses of java.lang.Exception (excluding RuntimeException). They must be handled with try-catch or declared with throws.

Common Checked Exceptions:

- IOException
  - FileNotFoundException
  - EOFException
- SQLException
- ClassNotFoundException
- ParseException
- InterruptedException
- InvocationTargetException
- NoSuchMethodException

Package Hierarchy:

java.lang

Exception

IOException

FileNotFoundException

SQLException

ClassNotFoundException

ParseException

InterruptedException

InvocationTargetException

NoSuchMethodException

3. try-catch vs throws

---

try-catch:

- Handles the exception locally.

- Good for immediate logging, retry, or fallback logic.

throws:

- Declares that a method might throw an exception.

- Lets the caller handle it.

- Useful in layered architecture or libraries.

Use try-catch when:

- You want to handle the issue where it occurs.

Use throws when:

- You want to pass the responsibility to the caller.