What is Exception Handling in Java?

Exception handling in Java is a mechanism used to handle runtime errors, allowing the program to continue executing instead of terminating abruptly. An **exception** is an event that disrupts the normal flow of the program's instructions, often due to errors like dividing by zero, accessing an array out of bounds, or reading from a non-existent file.

Java provides a powerful and flexible way to handle such runtime errors using **try-catch** blocks, along with other constructs such as `throw`, `throws`, and `finally`.

Key Concepts

1. **Exception**: An object representing an error or unusual condition that occurs during the execution of a program. It can be **checked** (e.g., file handling exceptions) or **unchecked** (e.g., `NullPointerException`, `ArithmeticException`).
2. **try**: A block of code that might throw an exception. If an exception occurs inside the `try` block, Java will look for a corresponding `catch` block to handle it.
3. **catch**: A block of code that handles the exception thrown from the `try` block. It specifies the type of exception to catch (e.g., `ArithmeticException`, `IOException`) and contains code to deal with that exception.
4. **finally**: A block of code that is always executed, regardless of whether an exception was thrown or caught. It's usually used for cleanup actions, like closing files or releasing resources.
5. **throw**: Used to explicitly throw an exception in a method. For example, you can throw a new `Exception` or any of its subclasses.
6. **throws**: Used in a method signature to declare that the method might throw certain exceptions, allowing them to be handled by the caller of the method.

Exception Hierarchy

All exception classes in Java are derived from the `java.lang.Throwable` class, which has two main subclasses:

- **Exception**: Represents exceptions that can be caught and handled. It has two main types:
    - **Checked Exceptions**: These must be either handled in a `try-catch` block or declared using `throws`. Examples: `IOException`, `SQLException`.

    - **Unchecked Exceptions**: Also called runtime exceptions, these do not need to be explicitly handled. Examples: `NullPointerException`, `ArrayIndexOutOfBoundsException`.
- **Error**: Represents serious errors that a typical application should not try to handle, like `OutOfMemoryError` or `StackOverflowError`.