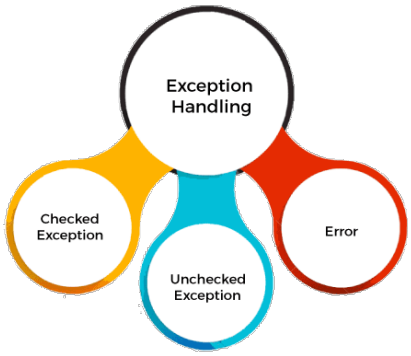
EXCEPTION HANDLING

Introduction to Exception

In Java, Exception is an unwanted or unexpected event, which occurs during the execution of a program, i.e. at run time, that disrupts the normal flow of the program's instructions. Exceptions can be caught and handled by the program.

When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

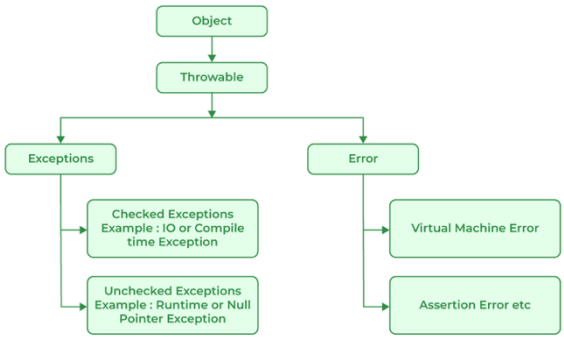


Difference Between Error and Exception

- Error: An Error indicates a serious problem that a reasonable application should not try to catch.
- Exception: Exception indicates conditions that a reasonable application might try to catch.

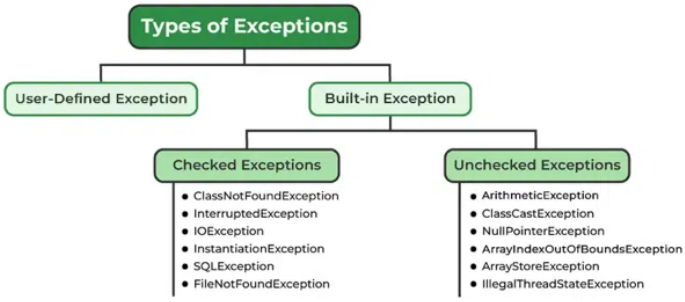
Exception Hierarchy

All exception and error types are subclasses of the class **Throwable**, which is the base class of the hierarchy. One branch is headed by **Exception**. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, **Error** is used by the Java run-time system to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.



Types of Exceptions

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Exceptions can be categorized in two ways:

1. Built-in Exceptions
Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

- Checked Exceptions: Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.

- Unchecked Exceptions: The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

2. User-Defined Exceptions:
Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, users can also create exceptions, which are called 'user-defined Exceptions'.

Keywords Used In Exception Handling

1.try The try block contains code that might throw an exception. When an exception occurs in the try block, the control is transferred to the appropriate catch block. A try block must be followed by at least one catch or finally block.

If no exception occurs, the catch blocks are skipped, and control moves to the code after the try-catch structure (or the finally block if present). Nested try blocks are allowed.

Execution enters the try block. When the exception is thrown, the rest of the try block is skipped, and control goes to the catch block.

2. catch The catch block handles the exception thrown by the try block. It specifies the type of exception it can handle. We can define multiple catch blocks to handle different types of exceptions.

Each catch block is evaluated in the order it appears. The first matching exception type is executed. We can catch exceptions using a superclass (e.g., Exception), but it must appear *after* more specific exception types.

catch blocks can use the exception object which was created when the exception was evaluated in try block, to get details about the error.

3. finally The finally block always executes, regardless of whether an exception is thrown or caught.

Typically used for resource cleanup (e.g., closing files, releasing locks, disconnecting databases). If the JVM exits (System.exit()), the finally block might not execute.

4. throw
The throw keyword is used to explicitly throw an exception from a method or block of code. throw keyword is used to explicitly throw an exception either pre-defined or user-defined.

5. throws
The throws keyword is used in a method signature to declare exceptions that the method might throw. It allows checked exceptions to be propagated to the caller for handling. Multiple exceptions can be declared, separated by commas.

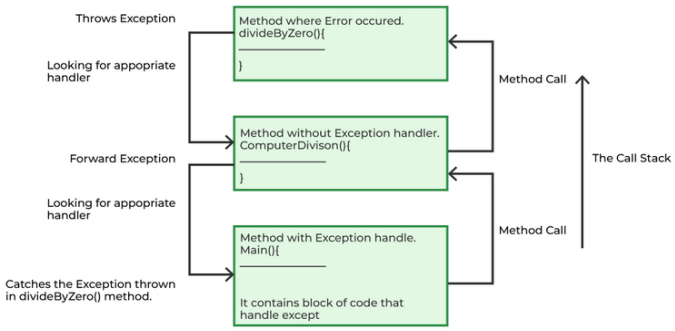
Methods to print the Exception Information

1.printStackTrace() This method prints exception information in the format of the Name of the exception: description of the exception, stack trace.

2. toString() The toString() method prints exception information in the format of the Name of the exception: description of the exception.

3. getMessage() The getMessage() method prints only the description of the exception.

How Does JVM Handle an Exception?



The Call Stack and searching the call stack for exception handler.

Default Exception Handling: Whenever inside a method, if an exception has occurred, the method creates an Object known as an Exception Object and hands it off to the run-time system(JVM). The exception object contains the name and description of the exception and the current state of the program where the exception has occurred. Creating the Exception Object and handling it in the run-time system is called throwing an Exception. There might be a list of the methods that had been called to get to the method where an exception occurred. This ordered list of methods is called Call Stack. Now the following procedure will happen.

- The run-time system searches the call stack to find the method that contains a block of code that can handle the occurred exception. The block of the code is called an Exception handler.
- The run-time system starts searching from the method in which the exception occurred and proceeds through the call stack in the reverse order in which methods were called.
- If it finds an appropriate handler, then it passes the occurred exception to it. An appropriate handler means the type of exception object thrown matches the type of exception object it can handle.
- If the run-time system searches all the methods on the call stack and couldn't have found the appropriate handler, then the run-time system handover the Exception Object to the default exception handler, which is part of the run-time system. This handler prints the exception information in the following format and terminates the program abnormally.

How Programmer Handle an Exception?

Customized Exception Handling: Java exception handling is managed via five keywords: try, catch, throw, throws, and finally. Briefly, here is how they work. Program statements that you think can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown. Your code can catch this exception (using catch block) and handle it in some rational manner. System-generated exceptions are automatically thrown by the Java run-time system. To manually throw an exception, use the keyword throw. Any exception that is thrown out of a method must be specified as such by a throws clause. Any code that absolutely must be executed after a try block completes is put in a finally block.