

Lecture 12

Exceptions

- When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.
- When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

Exceptions

- The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.
- There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception.

Exceptions

- Checked Exception: The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.
- Unchecked Exception: The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

try and catch

- The **try** statement allows you to define a block of code to be tested for errors while it is being executed.
- The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.
- The **try** and **catch** keywords come in pairs

try block

- Java **try** block is used to enclose the code that might throw an exception. It must be used within the method.
- If an exception occurs at the particular statement in the try block, the rest of the block code will not execute. So, it is recommended not to keep the code in try block that will not throw an exception.
- Java try block must be followed by either catch or finally block.

catch block

- Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception (i.e., Exception) or the generated exception type. However, the good approach is to declare the generated type of exception.
- The catch block must be used after the try block only. You can use multiple catch block with a single try block.

Internal Working of Java try-catch block

- The JVM firstly checks whether the exception is handled or not. If exception is not handled, JVM provides a default exception handler that performs the following tasks:
 - Prints out exception description.
 - Prints the stack trace (Hierarchy of methods where the exception occurred).
 - Causes the program to terminate.
- But if the application programmer handles the exception, the normal flow of the application is maintained, i.e., rest of the code is executed.

The throw keyword

- The **throw** statement allows you to create a custom error.
- The throw statement is used together with an exception type. There are many exception types available in Java:
- ArithmeticException, FileNotFoundException, Array IndexOutOfBoundsException, SecurityException, etc

The throw keyword

- The Java throw keyword is used to throw an exception explicitly.
- We specify the **exception** object which is to be thrown. The Exception has some message with it that provides the error description. These exceptions may be related to user inputs, server, etc.
- We can throw either checked or unchecked exceptions in Java by throw keyword. It is mainly used to throw a custom exception. We will discuss custom exceptions later in this section.
- We can also define our own set of conditions and throw an exception explicitly using throw keyword. For example, we can throw ArithmeticException if we divide a number by another number. Here, we just need to set the condition and throw exception using throw keyword.

The throws keyword

- The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception. So, it is better for the programmer to provide the exception handling code so that the normal flow of the program can be maintained.
- Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmers' fault that he is not checking the code before it being used.