Exception Handling

Exceptions are events that disrupt the normal flow of a program and represent error conditions or unexpected situations.

In exception hierarchy, Throwable class is the root, with subclasses Error and Exception.

Exception class is the super class of all exceptions.

Exception class is the super class of all checked exceptions.

RuntimeException class is the super class of all unchecked exceptions.

Checked exceptions must be caught or declared using throws, while unchecked exceptions do not require explicit handling.

Try block is used to enclose code that might throw an exception and catch block is usedto handle the exception that occur within the try block.

We can handle different types of exceptions with multiple catch blocks.

Catch the most specific exceptions first before more general ones - order of catch blocks.

Finally block is used to execute the code that should always run, regardless of exceptions.

Try block execution flow:

1) The code inside the try block is executed sequentially.

2) If an exception occurs during the execution of the try block, the remaining code inside the block is skipped.

3) The runtime immediately jumps to the corresponding catch block that matches the type of the thrown exception.

4) If there is no matching catch block, the program searches for a suitable exception handler in the calling methods.

5) If a matching catch block is found, the code inside that catch block is executed to handle the exception.

6) After executing the catch block, the program continues to execute any code in the finally block (if present).

7) If no exception occurs, the catch block(s) are skipped, and the program continues executing any code in the finally block (if present).

8) The program continues with the code following the try-catch-finally construct.

We can use the throw keyword to explicitly throw exceptions.

throw keyword is used to raise an exception explicitly within the code, while throws is used in a method header to indicate that the method might throw specific checked exceptions, which must be handled by the calling code.

We create custom exception classes by extending Exception or RuntimeException.

We can catch multiple exceptions in a single catch block using a vertical bar (|).

Use finally block for resource cleanup ex. closing file

Try-with-resources automatically releases external resources without requiring finally block.

Catch specific exception in catch block, avoid using Exception class in catch block to catch exceptions.

“throw” keyword is used to throw exception explicitly based on specific conditions.

Use throws clause to declare checked exceptions that a method may throw.

Errors represent critical problems and are not meant to be caught or handled.

Do not catch exceptions without proper handling - don't use empty catch blocks.

Subclass RuntimeException, to create custom/use-defined unchecked exceptions.

Subclass Exception, to create custom/user-defined checked exceptions.

Exception represents exceptional conditions that Java program can anticipate and handle ex. ArithmeticException, ArrayIndexOutOfBoundsException etc.

Error represents exceptional conditions that are typically caused by external factors or severe system-level issues that are beyond the control of the application. Errors are generally not recoverable, and attempting to handle them might lead to more severe problems or unexpected behavior. Ex. OutOfMemoryError, StackOverflowError

The finally block is entered even if a return statement is encountered.

Finally block is not entered on the execution of System.exit(0).