**Lab Tasks**

**Lab Task 1: Basic Exception Handling**

Objective: Create a Java program that reads an integer from the user and handles the xxxxException if the user enters a non-integer value.  
Instructions: Write a program that prompts the user to enter an integer. Use a try-catch block to catch an xxxxException and display a message indicating that the input is invalid.

**Lab Task 2: ArithmeticException**

Objective: Implement nested try-catch blocks to handle exceptions occurring at different levels.

Steps:

Write a program that has a try-catch block inside another try-catch block.

In the inner try block, handle xxxException by converting a string to a number.

In the outer try block, handle xxxException by performing a division operation.

Print different messages depending on which exception occurs.

**Lab Task 3: Handling Multiple Exceptions**

Objective: Write a Java program that handles multiple exceptions (e.g., xxxException and xxxxException).  
Instructions: Create a program that includes a division operation and an array. Use try-catch blocks to catch bothxxxException and xxxException. Display different messages for each exception.

**Lab Task 4: Custom Exception**

Objective: Create and use a custom exception class to handle insufficient balance in a bank account.

Steps:

Define a custom exception class: Create a class called InsufficientBalanceException that extends Exception.

Write a method to check balance: Write a method that checks if a bank account balance is sufficient to make a withdrawal. Let's say the balance should not go below $100 after a withdrawal.

Throw InsufficientBalanceException: If the withdrawal would cause the balance to drop below $100, throw InsufficientBalanceException.

Use try-catch block: In the main program, call the withdrawal method inside a try-catch block. Handle the InsufficientBalanceException and print an appropriate message to the user.

**Lab Task 5: Using finally Block**

Objective: Demonstrate the use of the finally block in Java.  
Instructions: Write a program that attempts to open a file and read its contents. Use a try-catch block to handle FileNotFoundException. Use a finally block to close the file, ensuring that the file is closed regardless of whether an exception occurred.

**Lab Task 6: Propagating Exceptions**

Objective: Illustrate how exceptions can be propagated in Java.  
Instructions: Create a method that throws an IOException. Call this method from the main method and handle the exception in the main method using a try-catch block. Explain how the exception is propagated and handled.

**More to be added**