Java Exception Handling Lab Exercises

Programming Lab

Lab Programming Questions on Exception Handling in Java

Question 1: Basic Exception Handling

Write a Java program that reads two integers from the user and performs division. Handle `ArithmeticException` when dividing by zero and `InputMismatchException` when invalid input is provided. Display appropriate error messages for each scenario.

Question 2: Custom Exception Class

Create a custom exception class `InvalidAgeException` that is thrown when an age value is less than 0 or greater than 150. Write a program that takes age as input and throws this custom exception with appropriate messages for invalid age values.

Question 3: Multiple Catch Blocks

Write a program that creates an array of 5 integers and asks the user for an index to display. Handle `ArrayIndexOutOfBoundsException` for invalid indices and `InputMismatchException` for non-integer input using multiple catch blocks.

Question 4: Finally Block Demonstration

Create a program that opens a file for reading using `FileInputStream`. Handle `FileNotFoundExceptior` and ensure that the file stream is properly closed in the `finally` block. Demonstrate that the `finally` block executes regardless of whether an exception occurs or not.

Question 5: Exception Propagation

Write a program with three methods: `methodA()`, `methodB()`, and `methodC()`. `methodC()` should throw an `ArithmeticException`. Let the exception propagate through `methodB()` to `methodA()`, where it should be caught and handled. Demonstrate exception propagation in the call stack.

Question 6: Try-with-Resources

Implement a program that uses try-with-resources to automatically manage a `FileInputStream` and `FileOutputStream`. The program should copy content from one file to another. Handle `IOException` and demonstrate that resources are automatically closed.

Question 7: Throw and Throws

Create a method `validatePassword()` that takes a password string and throws a custom `WeakPasswordException` if the password length is less than 8 characters. Use the `throw` keyword to throw the exception and declare it in the method signature using `throws`. Write a main method to test this functionality.

Question 8: Nested Try-Catch

Write a program with nested try-catch blocks. The outer block should handle `NumberFormatException` and the inner block should handle `ArithmeticException`. Demonstrate how exceptions are handled in nested scenarios and show the flow of execution.

Question 9: Exception Chaining

Create a program that demonstrates exception chaining. Catch a `FileNotFoundException`, then throw a custom `DataProcessingException` with the original exception as the cause. Show how to retrieve the chained exception using `getCause()` method.

Question 10: Comprehensive Banking Application

Develop a simple banking application with a `BankAccount` class that has methods for deposit and withdrawal. Implement the following exception handling:

- `InsufficientFundsException` - thrown when withdrawal amount exceeds balance
- `InvalidAmountException` - thrown for negative deposit/withdrawal amounts
- `AccountLockedException` - thrown when operating on a locked account

Create a menu-driven program to test all these scenarios.

Note: For each question, write complete Java programs with proper class structure, exception handling mechanisms, and appropriate test cases to demonstrate the functionality.