LEXICON

Exercise 1: Basic Exception Handling

Task: Write a program that asks the user to input two integers and then divides the first number by the second. Implement exception handling to manage the scenario where the user inputs zero as the second number.

Hint: Use try-catch to handle ArithmeticException.

Exercise 2: Multiple Exception Types

Task: Create a method that reads an integer from the user and checks whether it is within a certain range (e.g., 1 to 100). Handle exceptions for invalid inputs (e.g., non-integer input) and out-of-range values.

Hint: Use try-catch to handle InputMismatchException and a custom exception for out-of-range values.

Exercise 3: Custom Exception

Task: Define a custom exception InsufficientBalanceException that is thrown when a withdrawal amount exceeds the account balance. Implement a simple banking system that allows deposits and withdrawals, and handles the custom exception appropriately.

Hint: Create a BankAccount class and handle the custom exception using throw and throws.

Exercise 4: Nested try-catch Blocks

Task: Write a program that attempts to read a file and parse its contents as integers. Implement nested try-catch blocks to handle potential exceptions like NoSuchFileException, IOException and NumberFormatException.

Hint: Use nested try-catch blocks where the outer block handles file-related exceptions, and the inner block handles parsing-related exceptions.

Exercise 5: finally Block

Task: Modify the program from Exercise 1 to include a finally block that always executes, printing a message to the console, regardless of whether an exception was thrown or not.

Hint: The finally block should follow the catch block.

Exercise 6: Throwing Exceptions

Task: Write a method that takes a string as input and checks if it is a valid email address. If not, throw an IllegalArgumentException with an appropriate message.

Hint: Use the throw keyword to manually throw an exception when the input is invalid.

Exercise 7: Custom Exception for List Operations

Task: Create a program that manages a list of names. Implement two methods: one for finding a name in the list and another for adding a name to the list.

- If the name cannot be found, throw a custom exception NameNotFoundException.
- If a duplicate name is added to the list, throw another custom exception DuplicateNameException.

LEXICON

Exercise 8: Writing to a File Using Try-with-Resources

Task: Create a Java application that writes a string of text to a file. Use the try-with-resources statement to ensure that resources are automatically closed after the operation is complete.

Hint: The try-with-resources statement automatically closes resources such as file streams that implement the AutoCloseable interface.