# Report on Binary Class Implementation and Testing

## Introduction

The Binary class has been designed to perform bitwise operations on binary numbers in a console-based application. This report provides an overview of the class's design and implementation, improvements made to the App.java file for better user interaction, and a detailed review of the testing strategies applied.

## Implementation Details

### Binary Class

The Binary class comes with the following key features:

Constructor Validation: Ensures that input strings are valid binary numbers. If an invalid input is detected, it defaults to "0".

Bitwise Operations:

OR: Performs a bitwise OR operation by aligning the lengths of input binary numbers and applying the OR operation bit by bit.

AND: Similar to OR, but applies the bitwise AND operation instead.

Multiply: Implements binary multiplication using repeated addition and bit-shifting techniques.

Helper Methods:

add: Adds two binary numbers while accounting for carry bits.
shiftLeft: Shifts a binary number to the left, effectively multiplying it by 2.

Utility Methods:

equals: Compares two Binary objects to check for equality.
toString: Returns the string representation of the binary number.

## App.java Enhancements

The App.java file acts as the user interface for interacting with the Binary class. It has been enhanced to provide a seamless user experience with the following features:

Input Handling: Allows users to input binary numbers and select the desired operations.

Interactive Menu: Enables users to choose between OR, AND, and Multiply operations, and displays results in real time.

Enhanced Validation: Ensures all user inputs are valid binary numbers and provides appropriate feedback for invalid entries.

## Testing

### Binary Class Testing

JUnit was used to test the Binary class, ensuring its reliability and correctness. Key testing areas included:

#### Test Coverage:

OR, AND, Multiply: Each method was tested with both standard and edge cases, including scenarios where binary numbers had differing lengths.
Normalization Checks: Verified that binary outputs were normalized by removing leading zeros.

### App.java Testing

The testing for App.java focused on ensuring smooth user interaction. Key areas of testing included:

Input Tests: Validated that the application only accepts proper binary inputs.

Menu Logic Tests: Confirmed that the interactive menu handles both valid and invalid selections accurately.

End-to-End Tests: Simulated complete user workflows, from input to operation selection, to validate the robustness of the system.

## Conclusion

The Binary class, along with its integration into the App.java file, demonstrates a robust and efficient design for performing bitwise operations on binary numbers. The thorough testing framework ensures that the system is reliable and performs as expected under a variety of conditions. These features make the application user-friendly, accessible, and highly effective for working with binary operations.