

**Faculty of Engineering and Applied Science**

**SOFE 3980U**

**Software Quality & project management**

**Lab 1**

**Pritkumar Vasudevbhai Raiyani**

**100868794**

**Date:** January 2025

**Lab Section CRN 75766**

**Introduction**

The report discusses on the Binary calculator application that performs operations like ***Addition, Multiplication, bitwise OR, and bitwise AND***. It also has an interactive interface where user can enter two binary numbers and select the operations they want to see the results for, they also exists the interface when they want by selecting the appropriate option. The Binary.java has a binary class that has all the logic for these operations, which makes it easier to use and modify. The BinaryTest.java has the test cases for all the methods to ensure that the methods work correctly. The report also explain the source and testing code for the design.

**Source Code Analysis**

* **App.java**

The App.java is the main entry point forthe Binary Calculator program. It provides an interactive interface to the user where the user can input two binary numbers and perform the four binary operations ***Addition, Multiplication, bitwise OR, and bitwise AND.*** There is also a menu based system that allows the user to choose the operations they want to perform and also an option to exit the program.

**Features**

1. User Input
   * The program asks the user to enter two numbers.
   * The input is stored as strings and passed to the Binary class constructor to validate and initialize Binary objects.
2. Menu-Based system

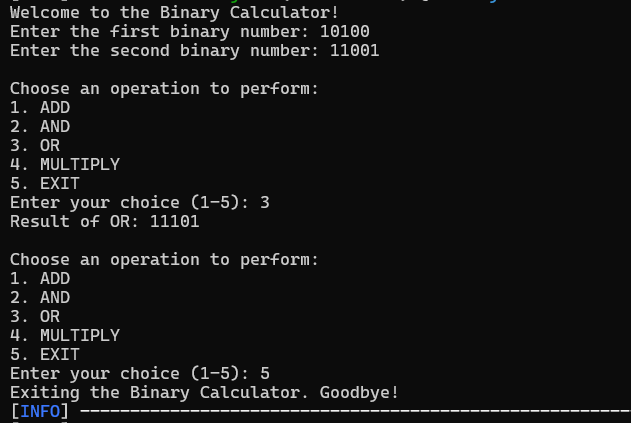
* A menu is displayed to the user for the operations that can be performed and a exit option.
* It also has a corresponding number assigned to it from 1 to 5.

1. Dynamic option selection

* Based on the option selected by the user it performs the corresponding operation and the result is displayed.
* If an invalid option is selected it asks the user to enter the correct option.

1. Program Termination

* The programs ends when the user sects the end option that is 5.



**Binary.java**

The Binary class was extended to include the following methods:

1. **or(Binary num1, Binary num2)**
   * This method computes the bitwise OR operation between two binary numbers.
   * **Design**: The two binary numbers are aligned by their length. For each bit, if either bit is 1, the result bit is 1; otherwise, it is 0.
   * **Key Features**:
     + Handles binary numbers of different lengths by padding with 0s.
     + Returns a new Binary object with the resulting value.

Example:

Binary binary1 = new Binary("1010");

Binary binary2 = new Binary("1100");

Binary result = Binary.or(binary1, binary2); // Result: "1110"

1. **and(Binary num1, Binary num2)**
   * This method computes the bitwise AND operation between two binary numbers.
   * **Design**: Similar to the OR operation, but both bits must be 1 for the result bit to be 1.
   * **Key Features**:
     + Handles binary numbers of different lengths by padding with 0s.

Example:

Binary binary1 = new Binary("1010");

Binary binary2 = new Binary("1100");

Binary result = Binary.and(binary1, binary2); // Result: "1000"

1. **multiply(Binary num1, Binary num2)**
   * This method multiplies two binary numbers using repeated addition (implemented via the existing add method).
   * **Design**:
     + For each 1 bit in the second binary number, the first binary number is shifted and added to the result.
     + Handles edge cases like multiplying by 0.

Example:

Binary binary1 = new Binary("101");

Binary binary2 = new Binary("11");

Binary result = Binary.multiply(binary1, binary2); // Result: "1111"

**App.java**

The main program was updated to make the application more interactive and user-friendly. It:

* Accepts two binary numbers from the user.
* Performs all four operations: add, or, and, and multiply.
* Displays the results clearly.

Example Interaction:

Welcome to the Binary Calculator!

Enter the first binary number: 1010

Enter the second binary number: 1101

First binary number: 1010

Second binary number: 1101

Sum: 10111

OR: 1111

AND: 1000

Multiply: 111110

**Testing Code Analysis**

**BinaryTest.java**

The test class for the Binary class was extended to include three tests for each of the new methods (or, and, multiply).

1. **Testing or() Method**
   * **Purpose**: Ensure correctness of the OR operation for binary numbers of varying lengths.
   * **Test Cases**:
     + Same-length binary numbers.
     + Binary numbers of different lengths.
     + Binary number and 0.

Example:

@Test

public void orTest() {

Binary binary1 = new Binary("1010");

Binary binary2 = new Binary("1100");

Binary result = Binary.or(binary1, binary2);

assertTrue(result.getValue().equals("1110"));

}

1. **Testing and() Method**
   * **Purpose**: Ensure correctness of the AND operation.
   * **Test Cases**:
     + Same-length binary numbers.
     + Binary numbers of different lengths.
     + Binary number and 0.

Example:

@Test

public void andTest() {

Binary binary1 = new Binary("1010");

Binary binary2 = new Binary("1100");

Binary result = Binary.and(binary1, binary2);

assertTrue(result.getValue().equals("1000"));

}

1. **Testing multiply() Method**
   * **Purpose**: Validate correctness of binary multiplication.
   * **Test Cases**:
     + Two non-zero binary numbers.
     + Binary number and 0.
     + Binary numbers of different lengths.

Example:

@Test

public void multiplyTest() {

Binary binary1 = new Binary("101");

Binary binary2 = new Binary("11");

Binary result = Binary.multiply(binary1, binary2);

assertTrue(result.getValue().equals("1111"));

}

**AppTest.java**

The AppTest file includes a basic sanity check to ensure that the application runs without errors. While minimal, this can be extended to include tests for specific application behavior if needed.

**Summary**

The project successfully extended the Binary class with three new operations (or, and, multiply) and updated the application to be more interactive. Comprehensive unit tests were added to verify the correctness of the new methods, ensuring reliability and robustness.

Future recommendations include:

1. Adding input validation in App.java for better user error handling.
2. Expanding the AppTest class to include functional tests for the interactive parts of the application.

Overall, the design and testing of the new functionalities were implemented effectively, meeting the project requirements.