Report for Source Code and Testing of Binary Operations - 100825241

Mohammad Daiyan

1. Introduction

This report provides an overview of the design and implementation of the **OR**, **AND**, and **Multiply** operations for the Binary class, as well as the associated testing code. The purpose of these additions is to enhance the functionality of the **Binary** class, enabling bitwise and arithmetic operations on binary variables.

The report also discusses the structure, logic, and methodology behind the testing code to ensure the correctness and reliability of the implemented methods.

---

2. Source Code Overview

2.1 Binary Class

The Binary class serves as a utility for representing and manipulating binary numbers. The new methods, **OR**, **AND**, and **Multiply**, were implemented as static methods within this class. Below is an explanation of each method:

a. OR Method

This method performs a bitwise logical OR operation between two binary numbers. It aligns the numbers by padding with zeros to the same length and iterates through each bit to compute the OR result.

Key Steps:

1. Pad the shorter binary number with leading zeros to match the length of the longer number.
2. Iterate through each bit of the two numbers.
3. Use logical OR to compute the result for each bit.
4. Construct and return a new Binary object with the result.

b. AND Method

This method computes a bitwise logical AND between two binary numbers. Similar to the OR method, the numbers are aligned before computing the result.

Key Steps:

1. Pad the shorter binary number with leading zeros to match lengths.
2. Iterate through each bit of the two numbers.
3. Use logical AND to compute the result for each bit.
4. Construct and return a new Binary object with the result.

c. Multiply Method

This method multiplies two binary numbers using repeated addition. The add method, already defined in the Binary class, is reused for this purpose.

Key Steps:

1. Initialize the result as "0".
2. For each 1 in the second binary number (from right to left), shift the first binary number by the position of the bit.
3. Add the shifted binary number to the result.
4. Construct and return a new Binary object with the result.

2.2 App.java

The App class serves as the main entry point for demonstrating the functionality of the Binary class. The program:

1. Prompts the user to input two binary numbers.
2. Displays the results of addition, OR, AND, and multiplication operations.
3. Uses the updated methods from the Binary class interactively.

3. Testing Code Overview

The BinaryTest class uses JUnit to perform unit testing on the Binary class. Each new method is tested with multiple scenarios to ensure correctness.

3.1 Test Cases for OR Method

- **Normal Input:** Tests OR operation with two standard binary numbers.
- **OR with Zero:** Tests OR operation when one of the numbers is zero.

3.2 Test Cases for AND Method

- **Normal Input:** Tests AND operation with two standard binary numbers.
- **AND with Zero:** Tests AND operation when one of the numbers is zero.

3.3 Test Cases for Multiply Method

- **Normal Multiplication:** Verifies multiplication of two binary numbers.
- **Multiplication by Zero:** Ensures the result is zero when one of the numbers is zero.
- **Multiplication by One:** Ensures the result equals the first binary number when multiplied by one.

4. Example Test Cases

Below are examples of the test cases implemented in the BinaryTest class:

OR Test Example

Input:

- Binary1: "1010"
- Binary2: "0110"

Expected Output:

- Result: "1110"

AND Test Example

Input:

- Binary1: "1010"
- Binary2: "0110"

Expected Output:

- Result: "0010"

Multiply Test Example

Input:

- Binary1: "101"
- Binary2: "11"

Expected Output:

- Result: "1111"

5. Conclusion

The **Binary** class has been successfully enhanced with OR, AND, and Multiply operations. The associated testing code ensures the reliability and correctness of these methods under various scenarios, including edge cases. The updated App class demonstrates the practical usage of these operations interactively, making the program user-friendly and functional.

These additions provide robust support for bitwise and arithmetic operations, further improving the utility of the Binary class for handling binary variables.