



FACULTY OF ENGINEERING AND APPLIED SCIENCE

SOFE 3980U Software Quality Winter 2023

LAB ONE

Software Project Management and Comprehension Tool

Ontario Tech University

SOFE 3980U Software Quality

CRN: 73385

Instructor(s): Mohamed El-Darieby

2022-01-22

Tasfia Alam – 100584647

Ahmaad Ansari – 100785574

Jacky Lee – 100787870

Mohammad Mohammadkhani – 100798165

Ashley Tyrone – 100786450

Table of Contents

I. Introduction	3
A. Apache Maven	3
B. Unit Testing	3
C. Video Submission	3
II. Discussion	4
A. Additional Binary Class Functions	4
B. Test Cases	5
C. Sample Output	7
III. Conclusion	8

I. Introduction

A. Apache Maven

Apache Maven is a powerful build automation tool that is widely used in the Java development community. It is an open-source tool designed to simplify the process of managing dependencies, building, and testing Java-based projects. In this lab report, we will explore the features and capabilities of Maven, including its installation and configuration, as well as its use in managing dependencies and building projects. We will also demonstrate how Maven can be integrated with popular Java development tools such as Eclipse and IntelliJ IDEA. Overall, this lab report aims to introduce Maven and its role in modern Java development comprehensively.

B. Unit Testing

Unit testing is a crucial aspect of software development, as it allows developers to verify that individual units of code are working as expected. In this lab report, we will explore the use of unit testing in Java development. We will cover the basics of creating and running unit tests, as well as best practices for writing effective and maintainable unit tests. Overall, this lab report aims to provide a comprehensive introduction to unit testing in Java, highlighting its importance in ensuring the quality and reliability of code.

C. Video Submission

Click [here](#) for the video submission. If the hyperlink is not working copy and paste the following link into your browser:

`https://drive.google.com/file/d/13sd7ofn_Dev4jATmsoz4llkaK0V7Gpyf/view?usp=share_link`

The video can also be found in our [GitHub](#) repository. If the hyperlink is not working copy and paste the following link into your browser:

`https://github.com/ahmaad-ansari/SOFE3980U-Lab1-D2/tree/main`

II. Discussion

A. Additional Binary Class Functions

Source Code: Bitwise OR Function

```
public static Binary or(Binary num1, Binary num2)
{
    // stores an integer version of the string binary value using getValue()
    int intNum1 = Integer.parseInt(num1.getValue(), 2);
    int intNum2 = Integer.parseInt(num2.getValue(), 2);

    // calculates the bitwise OR of the binary integers created
    int intNum3 = intNum1 | intNum2;

    // creates a result of type Binary to be returned
    Binary result = new Binary(Integer.toBinaryString(intNum3));
    return result;
}
```

This code defines a static method called **or** that takes in two objects of type `Binary` and returns a new `Binary` object. Inside the method, the code first converts the binary value of the two input `Binary` objects, **num1** and **num2**, to integers using the **parseInt** method and the base 2 radix. Then it performs a bitwise OR operation on these two integers, which results in a new integer. Next, the code converts this new integer back to a binary string using the **toBinaryString** method. Lastly, it creates a new `Binary` object using this binary string and returns it as the result of the method.

Source Code: Bitwise AND Function

```
public static Binary and(Binary num1, Binary num2)
{
    // stores an integer version of the string binary value using getValue()
    int intNum1 = Integer.parseInt(num1.getValue(), 2);
    int intNum2 = Integer.parseInt(num2.getValue(), 2);

    // calculates the bitwise AND of the binary integers created
    int intNum3 = intNum1 & intNum2;

    // creates a result of type Binary to be returned
    Binary result = new Binary(Integer.toBinaryString(intNum3));
    return result;
}
```

This code defines a static method called **and** that takes in two objects of type `Binary` and returns a new `Binary` object. Inside the method, the code first converts the

binary value of the two input Binary objects, **num1** and **num2**, to integers using the **parseInt** method and the base 2 radix. Then it performs a bitwise AND operation on these two integers, which results in a new integer. Next, the code converts this new integer back to a binary string using the **toBinaryString** method. Lastly, it creates a new Binary object using this binary string and returns it as the result of the method.

Source Code: Multiplication Function

```
public static Binary multiply(Binary num1, Binary num2)
{
    // stores an integer version of the string binary value using getValue()
    int intNum1 = Integer.parseInt(num1.getValue(), 2);
    int intNum2 = Integer.parseInt(num2.getValue(), 2);

    // calculates the product of the binary integers created
    int intNum3 = intNum1 * intNum2;

    // creates a result of type Binary to be returned
    Binary result = new Binary(Integer.toBinaryString(intNum3));
    return result;
}
```

This code defines a static method called **multiply** that takes in two objects of type Binary and returns a new Binary object. Inside the method, the code first converts the binary value of the two input Binary objects, **num1** and **num2**, to integers using the **parseInt** method and the base 2 radix. Then it multiplies these two integers, which results in a new integer. Next, the code converts this new integer back to a binary string using the **toBinaryString** method. Lastly, it creates a new Binary object using this binary string and returns it as the result of the method.

B. Test Cases

Test Case Name	Input Parameter	Expected Output	Actual Output	Valid Output	Report
or	(1000, 1111)	1111	1111	Pass	
or2	(1010, 11)	1011	1011	Pass	
or3	(11, 1010)	1011	1011	Pass	
or4	(0, 1010)	1010	1010	Pass	

or5	(0, 0)	0	0	Pass	
and	(1000, 1111)	1000	1000	Pass	
and2	(1010, 11)	10	10	Pass	
and3	(11, 1010)	10	10	Pass	
and4	(0, 1010)	0	0	Pass	
and5	(0, 0)	0	0	Pass	
multiply	(1000, 1111)	1111000	1111000	Pass	
multiply2	(1010, 11)	11110	11110	Pass	
multiply3	(11, 1010)	11110	11110	Pass	
multiply4	(0, 1010)	0	0	Pass	
multiply5	(0, 0)	0	0	Pass	

Three functions were added to the Binary class. These functions are the or, and, and multiply functions. Five test functions were created for each of these functions which can be depicted in the table above. Below is the source code for one of the fifteen generated test functions provided as a sample of how the other test functions were structured.

Source Code: Sample Test Case

```

/**
 * Test The and functions with two binary numbers of the same length
 */
@Test
public void and()
{
    Binary binary1=new Binary("1000");
    Binary binary2=new Binary("1111");
    Binary binary3=Binary.and(binary1,binary2);
    assertTrue( binary3.getValue().equals("1000"));
}

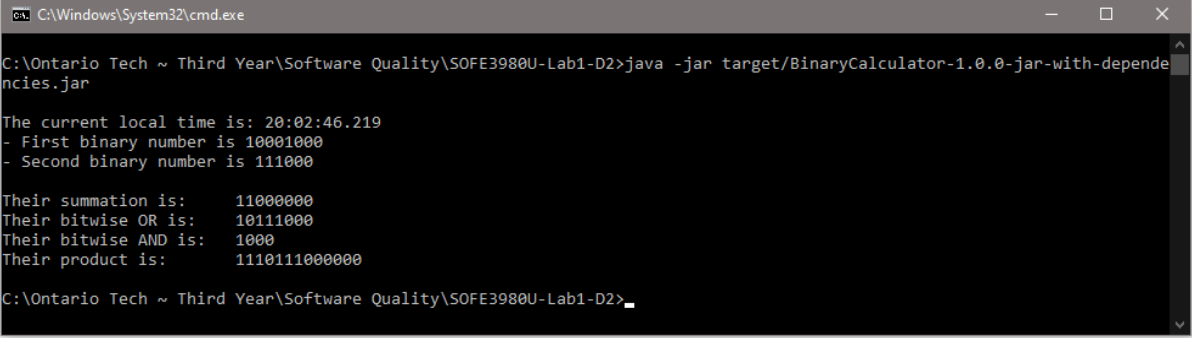
```

This sample test case is checking the **and** function of a Binary class. The test creates two Binary objects, **binary1** and **binary2**, with the values "1000" and "1111"

respectively. Then, it calls the **and** function on these two objects and assigns the result to a new Binary object, **binary3**. Finally, the test asserts that the value of **binary3** is equal to "1000" using the **assertTrue** method.

C. Sample Output

Sample Output: App.java



```
C:\Windows\System32\cmd.exe

C:\Ontario Tech ~ Third Year\Software Quality\SOF3980U-Lab1-D2>java -jar target/BinaryCalculator-1.0.0-jar-with-dependencies.jar

The current local time is: 20:02:46.219
- First binary number is 10001000
- Second binary number is 111000

Their summation is:      11000000
Their bitwise OR is:     10111000
Their bitwise AND is:    1000
Their product is:        1110111000000

C:\Ontario Tech ~ Third Year\Software Quality\SOF3980U-Lab1-D2>
```

The screenshot above provides the output for the **App.java** class. The output consists of the current local time, as well as two binary numbers and the results of four different operations applied to the binary numbers:

- Binary Summation,
- Bitwise OR,
- Bitwise AND, and
- Binary Multiplication.

III. Conclusion

In conclusion, this lab was aimed at familiarizing participants with Maven as a software project management tool. Through the process of installing Maven on a Windows operating system, participants were able to gain a deeper understanding of how to create, configure, and build Maven projects. The lab also provided hands-on experience in automatically generating documentation for a project, configuring the project to automatically add dependencies to the jar file, and writing and running tests on the project. Overall, this lab aimed to give participants a comprehensive understanding of Maven's capabilities and how to use them effectively in their own software development projects.