Software Project Management Deliverable 1

Maven and Unit testing

Name	**Student ID**
Liam Rea	100743012
Sarah Wedge	100785532
Alden O'Cain	100558599
Osamah Al-Bayati	100782415
Michael Barrett	100779360

Introduction:

This project aims to design and implement a program in Java that can handle binary numbers, specifically a Binary class that can store binary numbers as strings, truncate leading zeros, and perform basic arithmetic operations such as addition. The program also includes a test file that uses JUnit to test the functionality of the Binary class and its methods. The goal of this project is to demonstrate the ability to work with binary numbers in Java and to provide a better understanding of how binary arithmetic works.

The importance of software project management and testing cannot be overstated in software development. Effective project management ensures that the project is completed on time, within budget, and meets the desired specifications. It involves planning, organising, and monitoring the various aspects of the project, such as the scope, schedule, budget, and resources [2]. Project management also involves effective communication and collaboration among team members to ensure that the project is moving in the right direction. By having a well-defined project management plan in place, it helps to identify and resolve any issues that may arise during the development process [1].

Testing, on the other hand, is crucial for ensuring that the software is reliable, efficient, and meets the users' needs. It is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. It helps to identify and fix bugs, improve the quality of the code, and ensure that the software behaves as intended [5]. It is an essential step in the software development process as it helps to ensure that the software is fit for its intended purpose. Testing also helps to identify potential security vulnerabilities and performance bottlenecks in the software [5].

Unit testing, in particular, is a way to ensure that individual units of code are working as expected. It is a software testing method where individual units or components of a software application are tested in isolation [8]. By writing unit tests for the Binary class, we can ensure that the basic functionality of the class is working as intended and that any changes made to the class do not break existing functionality.

Together, project management and testing ensure that the software is developed to the highest standards and is fit for its intended purpose. Project management provides a structure and a plan for the development process, while testing ensures that the software meets the requirements and is of high quality. By having a well-defined project management plan in place and by performing thorough testing, we can ensure that the software is reliable, efficient, and meets the needs of the users.

Designing Source Code:

The source code consists of a program named Binary.java and a file called BinaryTest.java which lists test cases for Binary.java. The Binary.java file consists of a Binary class with a constructor that stores a binary number as a String and truncates any leading zeros. It has a getValue method to return the String sequence of binary numbers which represent the current Binary object. The add function was also given. It is a static function that takes two Binary objects, adds them by bit, and returns a new Binary object representing the result.

The BinaryTest.java file contains test cases that check the functionality of the Binary class and its methods, including the add function. It uses JUnit, a popular Java testing framework, to create and run the test cases. Overall, the program is designed to demonstrate the ability to work with binary numbers in Java, including performing basic arithmetic operations on them.

To design the source code, we first had to look at how every other function was created within the Binary.java file. According to the file, each object would take a binary number stored as a string and truncate the leading zeros. This would cause some problems later during testing and design would have to compensate.

Several enhancements were made to the Binary.java source code. The main goal was to develop three methods to perform the bitwise and, or, and multiplication of any pair of binary numbers. The and() function first checks the length of both binary arguments and adjusts any differences by concatenating zeros to the left side of the shorter number. This addition to the code was fundamental as it ensured the numbers could be easily compared at each bit index. The “and” operation was performed on each pair of bits and their result was added to a String variable. Once all bits were evaluated, the resulting String was converted to a Binary object and returned.

The next function added was the or() function. It is responsible for performing the bitwise “or” of two binary objects. Similar to the and() function, the or() function begins by comparing the lengths of both binary inputs and resolving their length differences, if encountered. The “or” operation was performed on each pair of bits and their result was added to a String variable which was converted to a Binary object and returned.

The final addition was the multiplication function. The first step was to select what method of multiplication would be applied. We decided to use “repeated addition” as our technique which would allow us to utilise the previously created add() function. In terms of binary multiplication, repeated addition works by taking the multiplier and locating the spot of the first digit “1” starting from the left most bit. Then the multiplicand is shifted to the left so its rightmost bit lines up with the leftmost “1” bit of the multiplier. The multiplicand, including its extra zeros that were appended upon shifting, are added to the resulting sum, which begins at 0. The leftmost “1” bit of the multiplier is changed to 0 and the procedure is repeated again until the multiplicand consists of only zeros [12].

To implement this functionality in the code, three methods were utilised. The multiplication method mult() was created to perform the multiplication. Another method called shiftLeft() was prepared to perform left shifting on any binary object based on a provided number of shifts. The previously made add() method was also utilised within mult() in order to perform a bitwise summation of the multiplicand each time it lined up with the multiplier's leading 1.

The last piece of source code is the App.java file. The App.java file is the main class of the program, it has a main method which is used to run the program and demonstrate the functionality of the Binary class. The AppTest.java file contains test cases that check the functionality of the App class.

Maven:

Maven is a build automation tool that is primarily used for Java projects. It helps to manage the project's build, dependencies, and documentation. It provides a standard structure for Java projects and automates many of the common tasks that are involved in building a Java project, such as compiling, packaging, and running tests.

Maven uses a Project Object Model (POM) to manage the project's build. The POM is an XML file that contains the project's information, such as the project's name, version, and dependencies. The POM also specifies the build plugins that should be used for the project, such as the compiler plugin, the jar plugin, and the surefire plugin (for running tests). Maven uses the information in the POM to execute the appropriate plugins and perform the desired build tasks [9].

In this project, Maven is used to manage the project's build and dependencies. The POM file is used to specify the project's dependencies, such as the JUnit library, which is used for testing [7]. Maven then automatically downloads the necessary dependencies and adds them to the project's classpath. Maven is also used to compile, package and run the test cases for the Binary class using the plugins defined in the POM file [9].

Maven also provides a standard project structure, which helps to organise the project's source code and resources. The standard project structure includes directories for source code, resources, and tests, which helps to keep the project organised and makes it easier to navigate [10]. It also helps to standardise the build process across different Java projects, which makes it easier to manage dependencies and perform common build tasks.

Therefore, Maven is a powerful tool for managing the build and dependencies of a Java project. It provides a standard structure for Java projects, automates many of the common tasks involved in building a Java project, and makes it easier to manage dependencies. Maven is used in this project to manage the project's build, dependencies and run test cases for the Binary class using JUnit.

Using Maven in the development process brought a number of benefits to the project. One of the main benefits is the ability to manage dependencies. Maven makes it easy to manage dependencies by specifying them in the POM file and automatically downloading them. This eliminates the need to manually download and configure dependencies, which can save time and reduce the risk of errors [11].

Another benefit of using Maven is the ability to standardise the build process. Maven provides a standard project structure and a set of plugins that can be used to perform common build tasks. This makes it easy to automate the build process and ensures that the build process is consistent across different environments [11]. This also makes it easier to manage the project and ensure that it is built correctly.

Maven also provides a built-in test runner, which makes it easy to run tests for the project. The Surefire plugin, that is defined in the POM file, can be used to run tests using JUnit and generate test reports. This makes it easy to ensure that the project is working correctly and that it meets the requirements. Maven also provides a central place for documentation, which makes it easy to keep track of the project's status. Information in the POM file can be used to generate project documentation, such as a project summary or a list of dependencies [10].

The following is a snippet of the POM file for the project, which shows how the dependencies for JUnit and the Surefire plugin are defined:

```
<dependencies>
   <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
   </dependency>
</dependencies>
<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.22.1</version>
      </plugin>
   </plugins>
</build>
```

This snippet shows that the project depends on JUnit version 4.12 and the Surefire plugin version 2.22.1. Maven will automatically download these dependencies when the project is built, ensuring that they are available to the project at runtime.

Another way Maven is integrated with the project is through the use of build profiles. Build profiles allow for different build settings to be defined for different environments. In summary, Maven is integrated into the project through the use of the POM file, which defines the project's dependencies and build settings. Additionally, build profiles are used to define different build settings for different environments, allowing for flexibility in the development process.

Unit Testing:

Unit testing is the process of creating individual tests to run various functions of our code through [8]. This is used to test the inputs and outputs of the function in a black box testing method in order to ensure that each function produces the correct result. In our case, this is accomplished using Junit as imported by Maven [7].

When designing a function, the goal is to give the correct output for every input, however it can be hard to consider every possible test and specific input. The goal of the tester is to find these combinations of inputs that break the function. Should they find a bugged input, they report it to the developer who then needs to adjust the code in order to make it work with that state. It can be even harder to find the bug if the issue is caused by a much lower level of code, rather than what is being built on top of it.

During our unit testing for our new functions, we ran into this exact issue in the form of unequal digits. In the Binary class, the binary number is stored as a string with any leading zeros truncated as they do not contribute to the value of the number. During our designing of tests for functions OR and AND, we ran into the issue of unequal numbers of digits, i.e. one number would be 1010 and the second would be 110. Our design for the function looked at each position and compared the two, outputting 1 if both numbers were 1 for AND and if only one number was 1 for OR.

Naturally, when it came to the highest value digit, there would be no number to compare it to and the program would error. Several tests were designed around this, comparing numbers when the digits were equal, when value one had less digits and when value 2 had less digits. To fix this issue, the lengths of numbers were compared and the difference between numbers filled with leading 0’s. These would allow the functions to continue on running and return a value, unaffected by the difference in digit count. However some tests still returned false which were resolved by fixing logical errors in the code.

The multiplication function also faced problems related to truncation. These problems were discovered by performing unit tests with various binary number input sequences. This led us to discover that since the multiplication uses shifting, its logic was heavily reliant on the position of the bits. If the multiplier was truncated, resulting in it being shorter than the multiplicand, then the “for” loop in the mult() function, which checks the position of the first leading 1 within the multiplicand, would perform incorrectly.

The leading 1 would be the first bit encountered, so the for loop would call the shiftLeft() method with its counter value (i) equal to 0. This means the multiplicand would not be shifted at all since it would have no zeros appended as the argument “n” was equal to 0. This changes the result of the multiplication, making it incorrect. Once the truncation issue was assessed, the tests no longer failed for the multiplication method .

The design of our test cases converted our returned numbers back to the Binary class, which would Truncate any leading 0’s that were a part of the return value. When designing the assert statement, we assumed that the output should be something like 0010, however this doesn’t make sense. If the leading zeros serve no purpose, then the expected output should be adjusted to suit it. If we were to keep the zeros, it would require editing the lower level functions of the Binary class that we did not design and result in potentially other test cases failing that relied on these functions. We decided that the expected output should also truncate the zeros.

This resulted in all test cases passing and provided some good insight into how testing, designing tests and what the expected output should be.

Conclusion:

This project explores the design and implementation of a Java program that can store binary numbers as strings, truncate leading zeros, and perform basic arithmetic operations such as addition. Additionally, the project also includes a test file that makes use of JUnit to evaluate how well the binary class and its methods function.

Project management and testing are used to assist, detect, and resolve any difficulties that may arise during the development process. Effective project management guarantees that the project is completed on time, within budget, and to the user's standards. The project's numerous components, including the scope, schedule, budget, and resources, are planned, organised, and monitored.

Testing is a component of a dependable and effective piece of software that meets the needs of its users. Testing is the process of assessing a system and/or its components in order to identify and solve problems, uncover potential security vulnerabilities, and improve product quality. In order to make sure that the program meets the needs of users and shareholders, testing is a crucial step in the software development cycle. Software can be tested in isolation using unit testing to make sure that each individual code block and the software as a whole function as planned. Unit tests for the binary class guarantee that the class's core functionality is operational and that any changes do not interfere with the functionality that has already been implemented.

In order to ensure that the software is created to the highest standards and is suitable for its intended use, this project investigates project management and testing.

Project management provides structure and direction for the development process, and testing ensures that the software is of a high standard and complies with user requirements. We can make sure that the software is dependable, efficient, and meets user needs by putting in place a well-defined project management plan and doing thorough testing.

References

[1] E. Chappell, “16 unmissable benefits of Project Management Software,” *ClickUp*, 30-Sep-2021. [Online]. Available: https://clickup.com/blog/benefits-of-project-management-software/. [Accessed: 27-Jan-2023].

[2] D. Huizinga and A. Kolawa, Automated defect prevention: Best practices in software management. Hoboken, NJ: Wiley-Interscience, 2007.

[3] J. U. L. T. junit-lambda-team@googlegroups.com, “Using JUnit 5 Platform,” Maven Surefire plugin – using junit 5 platform, 14-May-2018. [Online]. Available: https://maven.apache.org/surefire/maven-surefire-plugin/examples/junit-platform.html. [Accessed: 27-Jan-2023].

[4] J. van Z. V. Siveton, “Maven Getting Started Guide,” Maven, 01-Nov-2006. [Online]. Available: https://maven.apache.org/guides/getting-started/index.html#What_is_Maven. [Accessed: 27-Jan-2023].

[5] P. Parthiban, “7 reasons why software testing is important,” *Indium Software*, 14-Apr-2021. [Online]. Available: https://www.indiumsoftware.com/blog/why-software-testing/. [Accessed: 27-Jan-2023].

[6] S. B. Stefan Bechtold, “JUnit 5 User Guide,” JUnit 5 user guide. [Online]. Available: https://junit.org/junit5/docs/current/user-guide/. [Accessed: 27-Jan-2023].

[7] S. Gulati and R. Sharma, Java unit testing with junit 5: Test Driven Development with junit 5. Berkeley, CA: Apress, 2017.

[8] C. BasuMallick, “What is unit testing? types, tools, and best practices,” *Spiceworks*, 20-Sep-2022. [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-unit-testing/. [Accessed: 27-Jan-2023].

[9] “What is Maven?,” *Maven*, 23-Jan-2023. [Online]. Available: https://maven.apache.org/what-is-maven.html. [Accessed: 27-Jan-2023].

[10] J. van Zyl, “Introduction to the standard directory layout,” Maven, 09-Mar-2014. [Online]. Available: https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html. [Accessed: 27-Jan-2023].

[11] P. Pedamkar, “What is Maven?: Basic concept and advantages of Maven,” *EDUCBA*, 14-Sep-2022. [Online]. Available: https://www.educba.com/what-is-maven/. [Accessed: 27-Jan-2023].

[12] “Binary Multiplication by Repeated Addition,” *Binary multiplication*, 10-Aug-2001. [Online]. Available: https://courses.cs.vt.edu/~cs1104/BuildingBlocks/multiply.020.html. [Accessed: 27-Jan-2023].