Creating a Project

```
C:\Users\Hisha\Documents\labs>mvn archetype:generate ^
More? -DgroupId=com.ontariotechu.sofe3980U ^
More? -DartifactId=BinaryCalculator ^
More? -Dversion=1.0.0 ^
More? -DarchetypeArtifactId=maven-archetype-quickstart ^
More? -DarchetypeVersion=1.4 ^
More? -DinteractiveMode=false
```

```
[INFO] Parameter: groupId, Value: com.ontariotechu.sofe3980U
[INFO] Parameter: artifactId, Value: BinaryCalculator
[INFO] Parameter: version, Value: 1.0.0
[INFO] Parameter: package, Value: com.ontariotechu.sofe3980U
[INFO] Parameter: packageInPathFormat, Value: com/ontariotechu/sofe3980U
[INFO] Parameter: package, Value: com.ontariotechu.sofe3980U
[INFO] Parameter: groupId, Value: com.ontariotechu.sofe3980U
[INFO] Parameter: artifactId, Value: BinaryCalculator
[INFO] Parameter: version, Value: 1.0.0
[INFO] Project created from Archetype in dir: C:\Users\Hisha\Documents\labs\
BinaryCalculator
[INFO] ------------------------------------------------------------------------
---
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

App.java

C: > Users > Hisha > Documents > labs > BinaryCalculator > src > main > java > com > ontariotechu > sofe3980U > App.java

```java
package com.ontariotechu.sofe3980U;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}

```

Running the Project

```
<artifactId>maven-jar-plugin</artifactId>
<version>3.0.2</version>
  <configuration>
    <archive>
      <manifest>
        <addClasspath>true</addClasspath>
        <mainClass>com.ontariotechu.sofe3980U.App</mainC
      </manifest>
    </archive>
  </configuration>
</plugin>
```

Adding Source Files to the Project

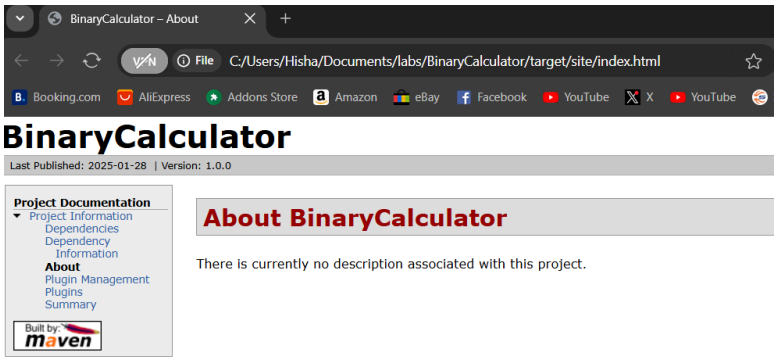
src > main > java > com > ontariotechu > sofe3980U

Sort View

Name	Date modified	Type	Size
App	2025-01-28 9:45 PM	Java Source File	1 KB
Binary	2025-01-28 9:45 PM	Java Source File	3 KB

```
C:\Users\Hisha\Documents\labs\BinaryCalculator>java -jar target/Binary
Calculator-1.0.0.jar
First binary number is 10001000
Second binary number is 111000
Their summation is 11000000
```

Generate Documentation for the Project




```
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>3.4.1</version>
      </plugin>
    </plugins>
  </reporting>
</project>
```

Class Binary

java.lang.Object
com.ontariotechu.sofe3980U.Binary

```
public class Binary
extends Object
```

Unsigned integer Binary variable

Constructor Summary

Constructors

Constructor	Description
`Binary(String number)`	A constructor that generates a binary object.

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method

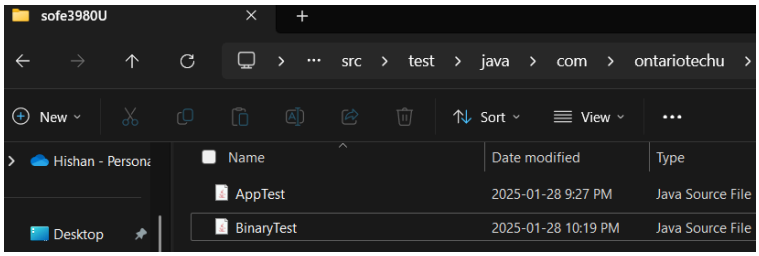
Add Dependencies

```
    </dependency>
    <dependency>
      <groupId>joda-time</groupId>
      <artifactId>joda-time</artifactId>
      <version>2.9.2</version>
    </dependency>
    </dependencies>
```

```
    </plugin>
    <!-- assembly lifecycle, generate jar with dependencies-->
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <archive>
      <manifest>
        <addClasspath>true</addClasspath>
        <mainClass>com.ontariotechu.sofe3980U.App</mainClass>
      </manifest>
      </archive>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
  </plugin>
    <!-- site lifecycle, see https://maven.apache.org/ref/current/mave
```

```
C:\Users\Hisha\Documents\labs\BinaryCalculator>java -jar target/Binary
Calculator-1.0.0-jar-with-dependencies.jar
The current local time is: 22:12:19.965
First binary number is 10001000
Second binary number is 111000
Their summation is 11000000
```

Add Test Cases




```
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>3.4.1</version>
      </plugin>
      <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-report-plugin</artifactId>
  <version>3.0.0-M7</version>
</plugin>
    </plugins>
  </reporting>
</project>
```

BinaryCalculator

Last Published: 2025-01-28 | Version: 1.0.0 BinaryCalcula

Project Documentation
- Project Information
- Project Reports
  - Javadoc
  - Test Javadoc
  - **Surefire Report**

Built by: maven

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
12	0	0	0	100%	0.032

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Design

OR Function

```java
/**
 * Perform bitwise OR operation on two binary numbers.
 *
 * @param num1 The first binary object
 * @param num2 The second binary object
 * @return A binary variable with the result of <i>num1 OR num2</i>.
 */
public static Binary OR(Binary num1, Binary num2) {
    StringBuilder result = new StringBuilder();
    int maxLength = Math.max(num1.number.length(), num2.number.length());

    // Pad the shorter binary number with leading zeros
    String paddedNum1 = padWithZeros(num1.number, maxLength);
    String paddedNum2 = padWithZeros(num2.number, maxLength);

    for (int i = 0; i < maxLength; i++) {
        char bit1 = paddedNum1.charAt(i);
        char bit2 = paddedNum2.charAt(i);
        result.append((bit1 == '1' || bit2 == '1') ? '1' : '0');
    }

    return new Binary(result.toString());
}
/**
```

AND Function

```java
/**
 * Perform bitwise AND operation on two binary numbers.
 *
 * @param num1 The first binary object
 * @param num2 The second binary object
 * @return A binary variable with the result of <i>num1 AND num2</i>.
 */
public static Binary AND(Binary num1, Binary num2) {
    StringBuilder result = new StringBuilder();
    int maxLength = Math.max(num1.number.length(), num2.number.length());

    // Pad the shorter binary number with leading zeros
    String paddedNum1 = padWithZeros(num1.number, maxLength);
    String paddedNum2 = padWithZeros(num2.number, maxLength);

    for (int i = 0; i < maxLength; i++) {
        char bit1 = paddedNum1.charAt(i);
        char bit2 = paddedNum2.charAt(i);
        result.append((bit1 == '1' && bit2 == '1') ? '1' : '0');
    }

    return new Binary(result.toString());
}
```

MULTIPLY Function

```java
/**
 * Multiply two binary numbers.
 *
 * @param num1 The first binary object
 * @param num2 The second binary object
 * @return A binary variable with the result of <i>num1 * num2</i>.
 */
public static Binary multiply(Binary num1, Binary num2) {
    Binary result = new Binary("0");
    Binary tempNum1 = new Binary(num1.number);

    for (int i = num2.number.length() - 1; i >= 0; i--) {
        if (num2.number.charAt(i) == '1') {
            result = add(result, tempNum1);
        }
        tempNum1 = new Binary(tempNum1.number + "0"); // Shift left (equivalent to multiplying by 2)
    }

    return result;
}
```

Updated App.Java File

```java
    // Create a Scanner object for user input
    Scanner scanner = new Scanner(System.in);

    // Prompt the user to enter the first binary number
    System.out.println("Enter the first binary number:");
    String input1 = scanner.nextLine();
    Binary binary1 = new Binary(input1);

    // Prompt the user to enter the second binary number
    System.out.println("Enter the second binary number:");
    String input2 = scanner.nextLine();
    Binary binary2 = new Binary(input2);

    // Perform operations
    Binary sum = Binary.add(binary1, binary2);
    Binary orResult = Binary.OR(binary1, binary2);
    Binary andResult = Binary.AND(binary1, binary2);
    Binary multiplyResult = Binary.multiply(binary1, binary2);

    // Display results
    System.out.println("\nResults:");
    System.out.println("First binary number: " + binary1.getValue());
    System.out.println("Second binary number: " + binary2.getValue());
    System.out.println("Sum: " + sum.getValue());
    System.out.println("OR: " + orResult.getValue());
    System.out.println("AND: " + andResult.getValue());
    System.out.println("Multiply: " + multiplyResult.getValue());

    // Close the scanner
    scanner.close();
}
```

Test Cases

```java
// New tests for OR function

@Test
public void OR_SameLength() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("1100");
    Binary result = Binary.OR(binary1, binary2);
    assertTrue(result.getValue().equals("1110"));
}

@Test
public void OR_DifferentLength() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("11");
    Binary result = Binary.OR(binary1, binary2);
    assertTrue(result.getValue().equals("1011"));
}

@Test
public void OR_WithZero() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("0000");
    Binary result = Binary.OR(binary1, binary2);
    assertTrue(result.getValue().equals("1010"));
}
```

```java
// New tests for AND function

@Test
public void AND_SameLength() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("1100");
    Binary result = Binary.AND(binary1, binary2);
    assertTrue(result.getValue().equals("1000"));
}

@Test
public void AND_DifferentLength() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("11");
    Binary result = Binary.AND(binary1, binary2);
    assertTrue(result.getValue().equals("10"));
}

@Test
public void AND_WithZero() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("0000");
    Binary result = Binary.AND(binary1, binary2);
    assertTrue(result.getValue().equals("0"));
}
```

```java
// New tests for Multiply function

@Test
public void Multiply_SameLength() {
    Binary binary1 = new Binary("1010"); // 10
    Binary binary2 = new Binary("1100"); // 12
    Binary result = Binary.multiply(binary1, binary2);
    assertTrue(result.getValue().equals("1111000")); // 120
}

@Test
public void Multiply_DifferentLength() {
    Binary binary1 = new Binary("101"); // 5
    Binary binary2 = new Binary("10"); // 2
    Binary result = Binary.multiply(binary1, binary2);
    assertTrue(result.getValue().equals("1010")); // 10
}

@Test
public void Multiply_WithZero() {
    Binary binary1 = new Binary("1010");
    Binary binary2 = new Binary("0");
    Binary result = Binary.multiply(binary1, binary2);
    assertTrue(result.getValue().equals("0"));
}
```