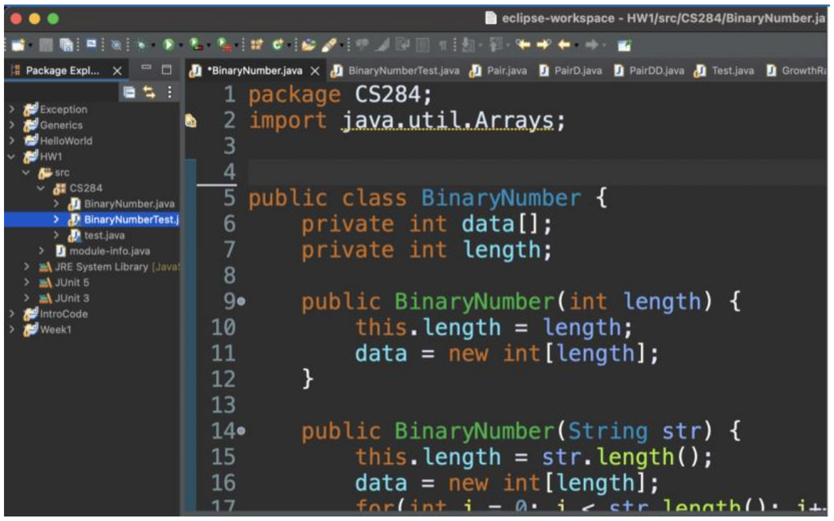
CS 284: Recitation Week 5 - JUnit

This week you'll learn how to use JUnit to test your code. This recitation helps you get started with JUnit.

1. Open your HW1 (BinaryNumber) project in Eclipse



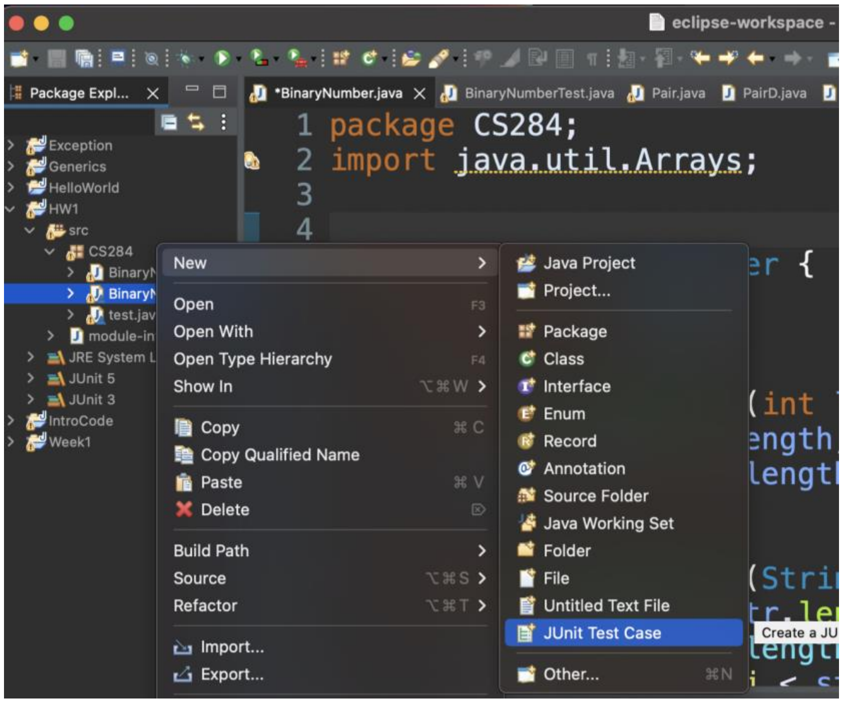
```
package CS284;
import java.util.Arrays;

public class BinaryNumber {
    private int data[];
    private int length;

    public BinaryNumber(int length) {
        this.length = length;
        data = new int[length];
    }

    public BinaryNumber(String str) {
        this.length = str.length();
        data = new int[length];
        for(int i = 0; i < str.length(); i+
```

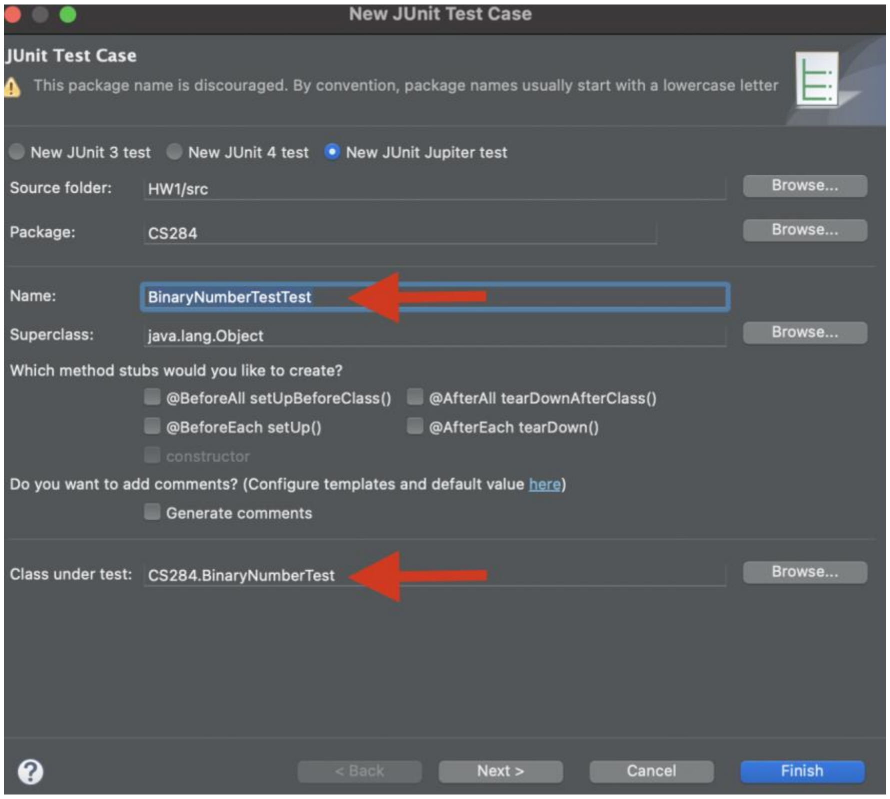
2**. Set Up JUnit Testing**: In Eclipse, you create a JUnit test case by selecting in the main window menubar File -> New -> JUnit Test Case.



```
package CS284;
import java.util.Arrays;
```

Once you clicked on the item, a big dialog should pop out.

In the pop-up, you can choose the package and class name of your test case. It will suggest the default name "BinaryNumberTest" and mark cs284.BinaryNumberas the “class under test”.


New JUnit Test Case

JUnit Test Case

This package name is discouraged. By convention, package names usually start with a lowercase letter

( ) New JUnit 3 test ( ) New JUnit 4 test [✓ New JUnit Jupiter test]

Source folder: HW1/src

Package: CS284

Name: BinaryNumberTestTest

Superclass: java.lang.Object

Which method stubs would you like to create?

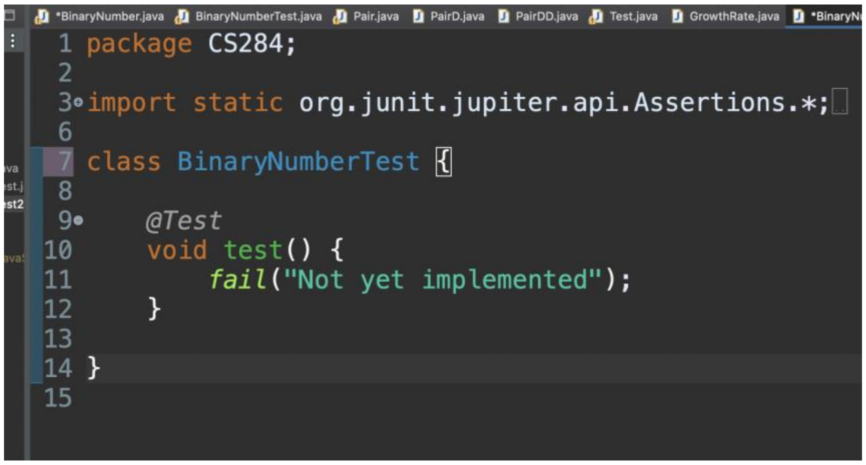
- [ ] @BeforeAll setUpBeforeClass()
- [ ] @AfterAll tearDownAfterClass()
- [ ] @BeforeEach setUp()
- [ ] @AfterEach tearDown()
- [ ] constructor

Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

Class under test: CS284.BinaryNumberTest

3. This is the default BinaryNumberTest



```java
package CS284;

import static org.junit.jupiter.api.Assertions.*;

class BinaryNumberTest {

    @Test
    void test() {
        fail("Not yet implemented");
    }

}

```

4. To mark a method as a test method, annotate it with the `@Test` annotation. This method executes the code under test.

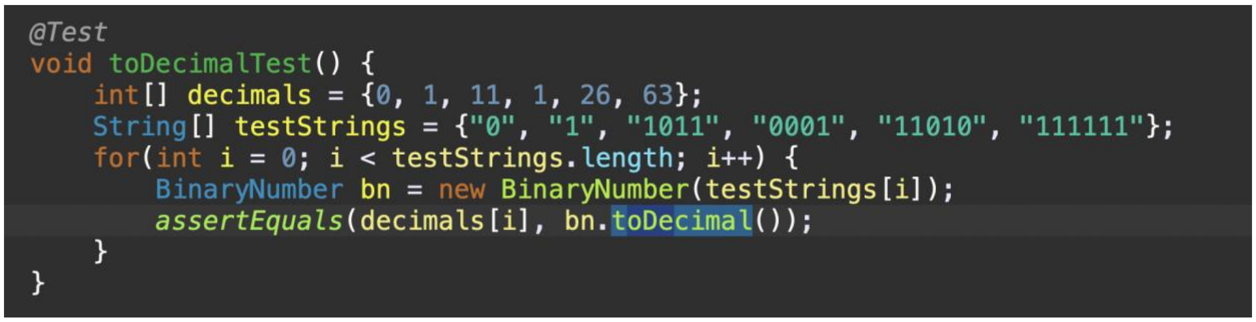
You can use assert methods, provided by JUnit to check an expected result versus the actual result. JUnit's Assertions class offers a number of static methods which are useful for writing the test cases.

JUnit 5 allows to use `static imports` for its assertStatements which allows fields and methods defined in a class as public static to be used without specifying the class in which the field is defined.

Some useful Assert Methods:

- `assertEquals(expected, actual`**)** – check whether **expected.equal(actual)**.
- `assertEquals(expected, got, message)` - in addition to above, it allows to define messages which are shown if the test fails
- `assertTrue(boolean condition)` – check the true condition.
- `assertFalse(boolean condition)`   – check the false condition.
- `assertArrayEquals` – check whether two arrays are equal to each other or not

5. Now create your first test case, write JUnit tests for your BinaryNumber implementation
Here an example to help you get started: `ToDecimalTesr()`



```
@Test
void toDecimalTest() {
    int[] decimals = {0, 1, 11, 1, 26, 63};
    String[] testStrings = {"0", "1", "1011", "0001", "11010", "111111"};
    for(int i = 0; i < testStrings.length; i++) {
        BinaryNumber bn = new BinaryNumber(testStrings[i]);
        assertEquals(decimals[i], bn.toDecimal());
    }
}
```

6. Run a Junit: Within the Package Explorer, locate your JUnit test. Right-click and select **Run As > JUnit Test**. This will execute your test and report results within the JUnit Eclipse view. The result of the tests are displayed in the JUnit view. A green bar shows a single test passed.