Day 8:

Task 1: Write a set of JUnit tests for a given class with simple mathematical operations (add, subtract, multiply, divide) using the basic @Test annotation.

Class with Mathematical Operations:

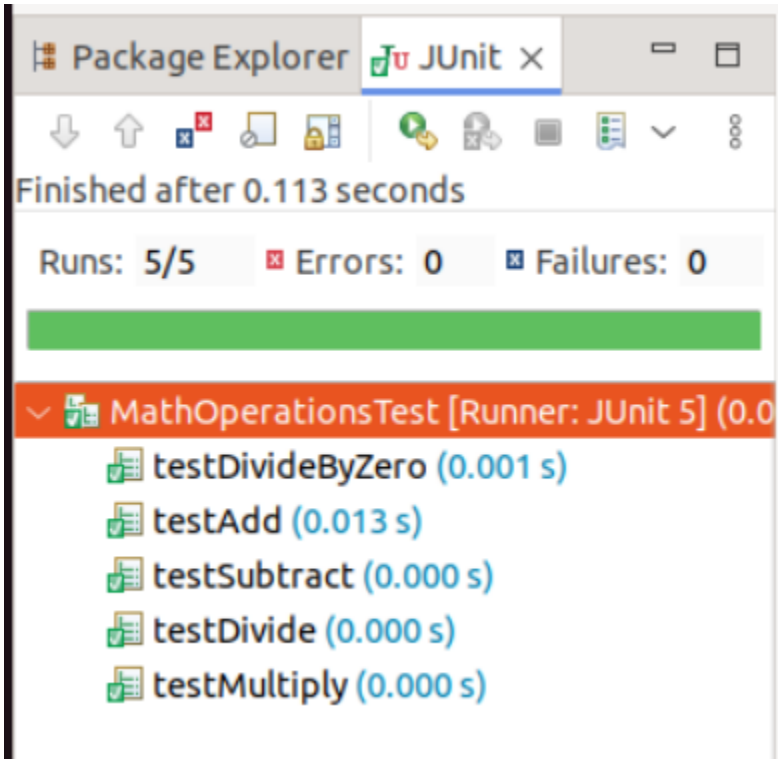
```
public class MathOperations {

    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public int divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Cannot divide by zero");
        }
        return a / b;
    }
}
```

JUnit Test Class:

```
import static org.junit.Assert.*;
import org.junit.Test;

public class MathOperationsTest {

    MathOperations mathOps = new MathOperations();

    @Test
    public void testAdd() {
        assertEquals(5, mathOps.add(2, 3));
        assertEquals(-1, mathOps.add(-2, 1));
    }

    @Test
    public void testSubtract() {
        assertEquals(1, mathOps.subtract(3, 2));
        assertEquals(-3, mathOps.subtract(-2, 1));
    }

    @Test
    public void testMultiply() {
        assertEquals(6, mathOps.multiply(2, 3));
        assertEquals(-2, mathOps.multiply(-1, 2));
    }

    @Test
    public void testDivide() {
        assertEquals(2, mathOps.divide(6, 3));
        assertEquals(-2, mathOps.divide(-4, 2));
    }

    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
```

```
        mathOps.divide(1, 0);
    }
}
```

Output



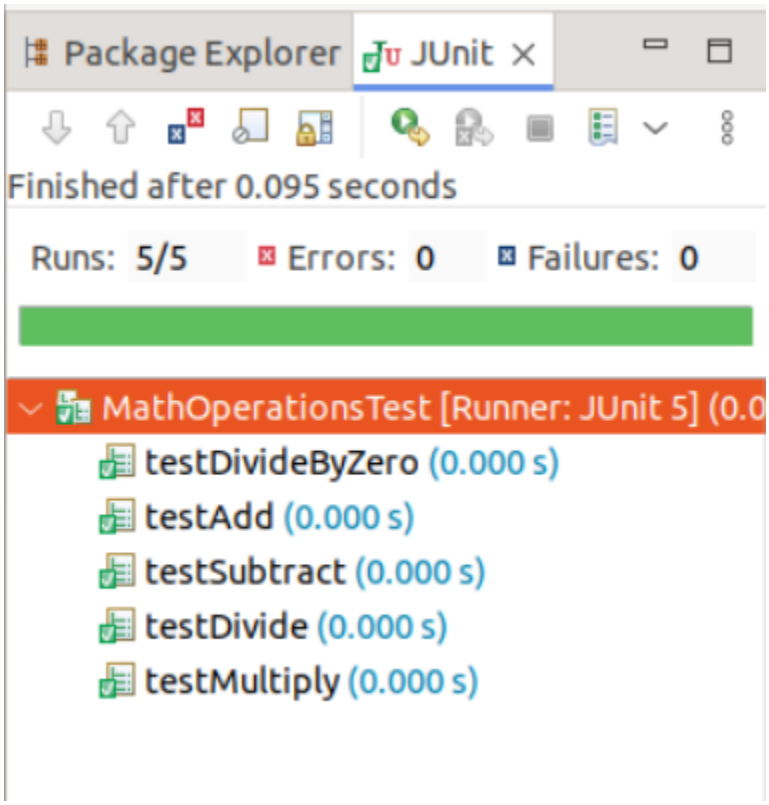
Task 2: Extend the above JUnit tests to use @Before, @After, @BeforeClass, and @AfterClass annotations to manage test setup and teardown.

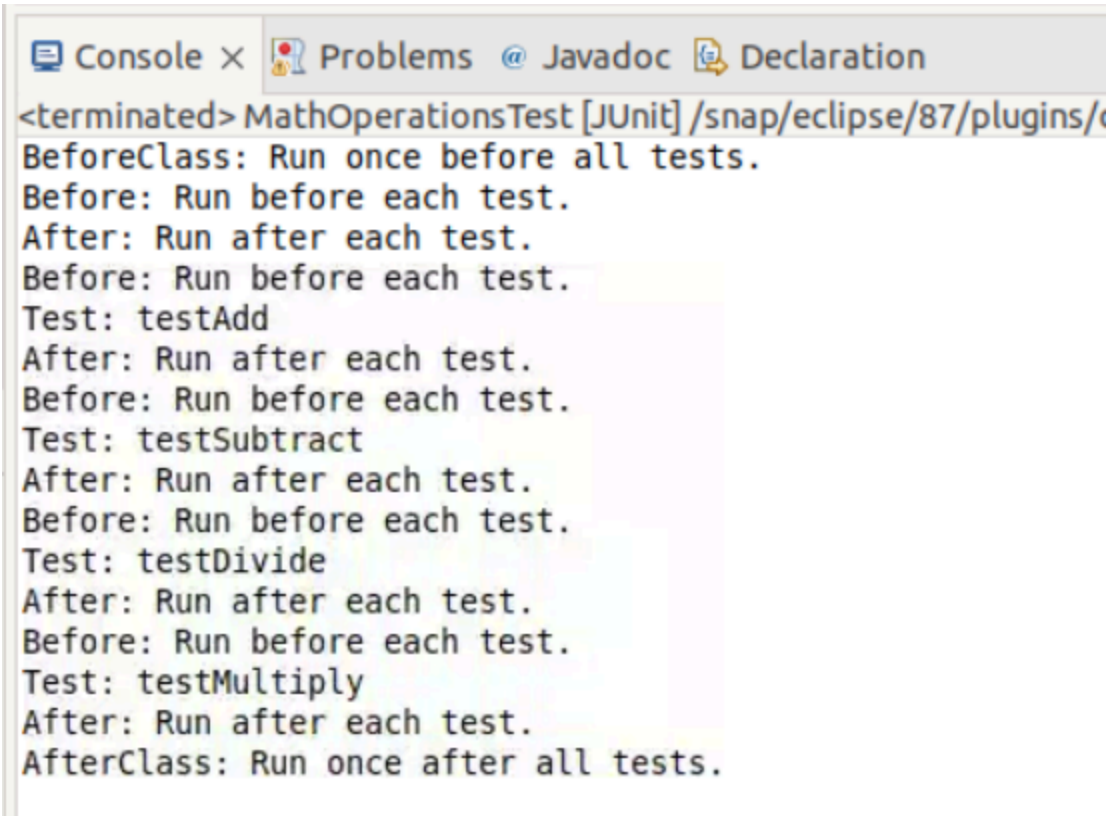
```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

public class MathOperationsTest {

    private MathOperations mathOps;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        // Code executed once before any test methods run
        System.out.println("BeforeClass: Run once before all tests.");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        // Code executed once after all test methods have run
        System.out.println("AfterClass: Run once after all tests.");
    }

    @Before
    public void setUp() throws Exception {
        // Code executed before each test method
        mathOps = new MathOperations();
        System.out.println("Before: Run before each test.");
    }
```

```
@After
public void tearDown() throws Exception {
    // Code executed after each test method
    System.out.println("After: Run after each test.");
}

@Test
public void testAdd() {
    assertEquals(5, mathOps.add(2, 3));
    assertEquals(-1, mathOps.add(-2, 1));
    System.out.println("Test: testAdd");
}

@Test
public void testSubtract() {
    assertEquals(1, mathOps.subtract(3, 2));
    assertEquals(-3, mathOps.subtract(-2, 1));
    System.out.println("Test: testSubtract");
}

@Test
public void testMultiply() {
    assertEquals(6, mathOps.multiply(2, 3));
    assertEquals(-2, mathOps.multiply(-1, 2));
    System.out.println("Test: testMultiply");
}

@Test
public void testDivide() {
    assertEquals(2, mathOps.divide(6, 3));
    assertEquals(-2, mathOps.divide(-4, 2));
    System.out.println("Test: testDivide");
}
```

```
    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        mathOps.divide(1, 0);
        System.out.println("Test: testDivideByZero");
    }
}
```

Output




Console
Problems
Javadoc
Declaration
<terminated> MathOperationsTest [JUnit] /snap/eclipse/87/plugins/
BeforeClass: Run once before all tests.
Before: Run before each test.
After: Run after each test.
Before: Run before each test.
Test: testAdd
After: Run after each test.
Before: Run before each test.
Test: testSubtract
After: Run after each test.
Before: Run before each test.
Test: testDivide
After: Run after each test.
Before: Run before each test.
Test: testMultiply
After: Run after each test.
AfterClass: Run once after all tests.

Task 3: Create test cases with assertEquals, assertTrue, and assertFalse to validate the correctness of a custom String utility class.

StringUtils Class:

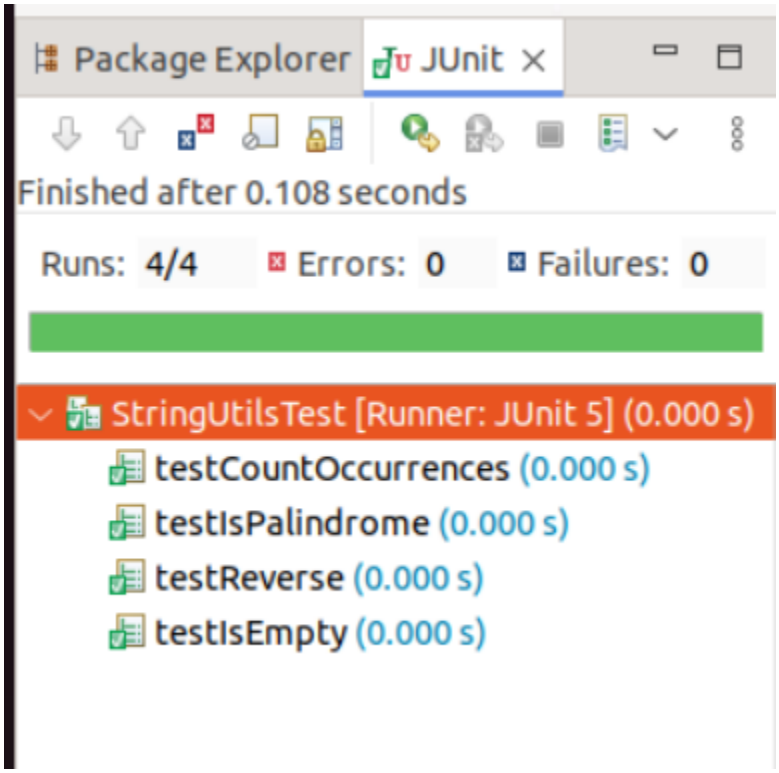
```
public class StringUtils {

    public boolean isEmpty(String str) {
        return str == null || str.isEmpty();
    }

    public String reverse(String str) {
        if (str == null) {
            return null;
        }
        return new StringBuilder(str).reverse().toString();
    }

    public boolean isPalindrome(String str) {
        if (str == null) {
            return false;
        }
        String reversed = reverse(str);
        return str.equals(reversed);
    }

    public int countOccurrences(String str, char ch) {
        if (str == null) {
            return 0;
        }
        int count = 0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i) == ch) {
                count++;
```

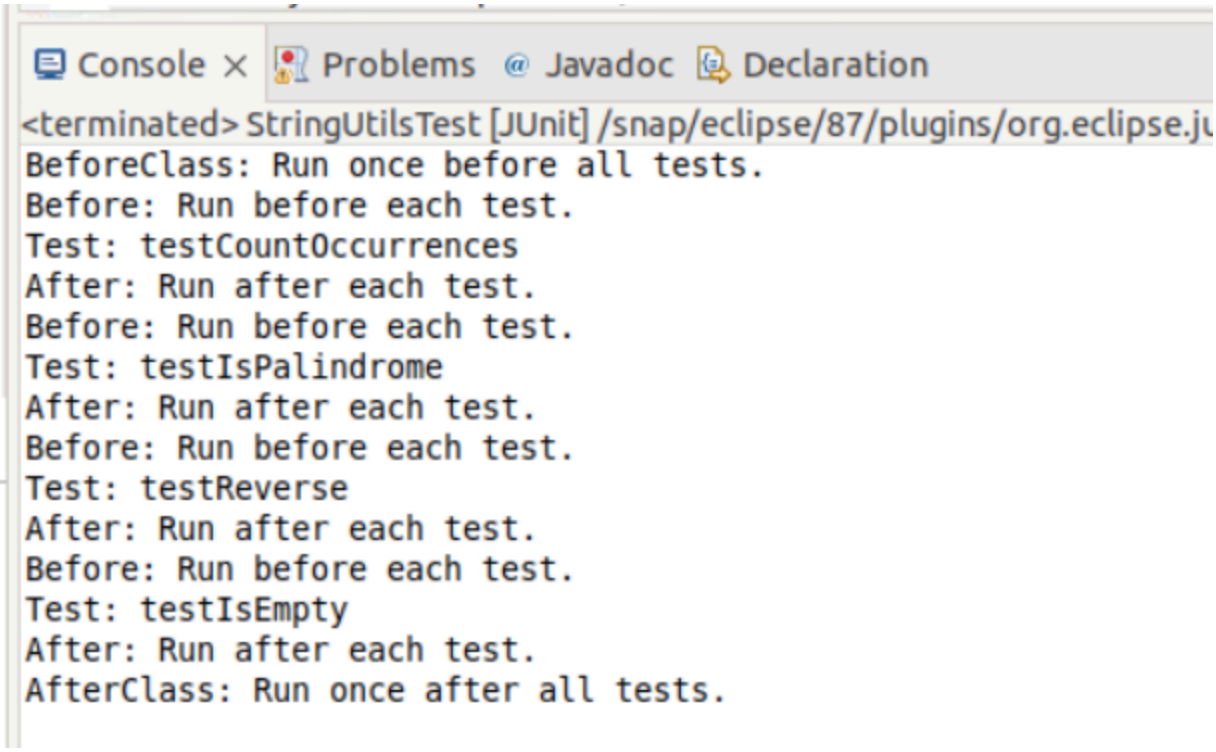
```
            }
        }
        return count;
    }
}
```

StringUtilsTest Class:

```

```

Output




Console
Problems
Javadoc
Declaration
<terminated> StringUtilsTest [JUnit] /snap/eclipse/87/plugins/org.eclipse.ju
BeforeClass: Run once before all tests.
Before: Run before each test.
Test: testCountOccurrences
After: Run after each test.
Before: Run before each test.
Test: testIsPalindrome
After: Run after each test.
Before: Run before each test.
Test: testReverse
After: Run after each test.
Before: Run before each test.
Test: testIsEmpty
After: Run after each test.
AfterClass: Run once after all tests.