JUnit Testing Exercises

Exercise 1: Setting Up JUnit

Scenario:

You need to set up JUnit in your Java project to start writing unit tests.

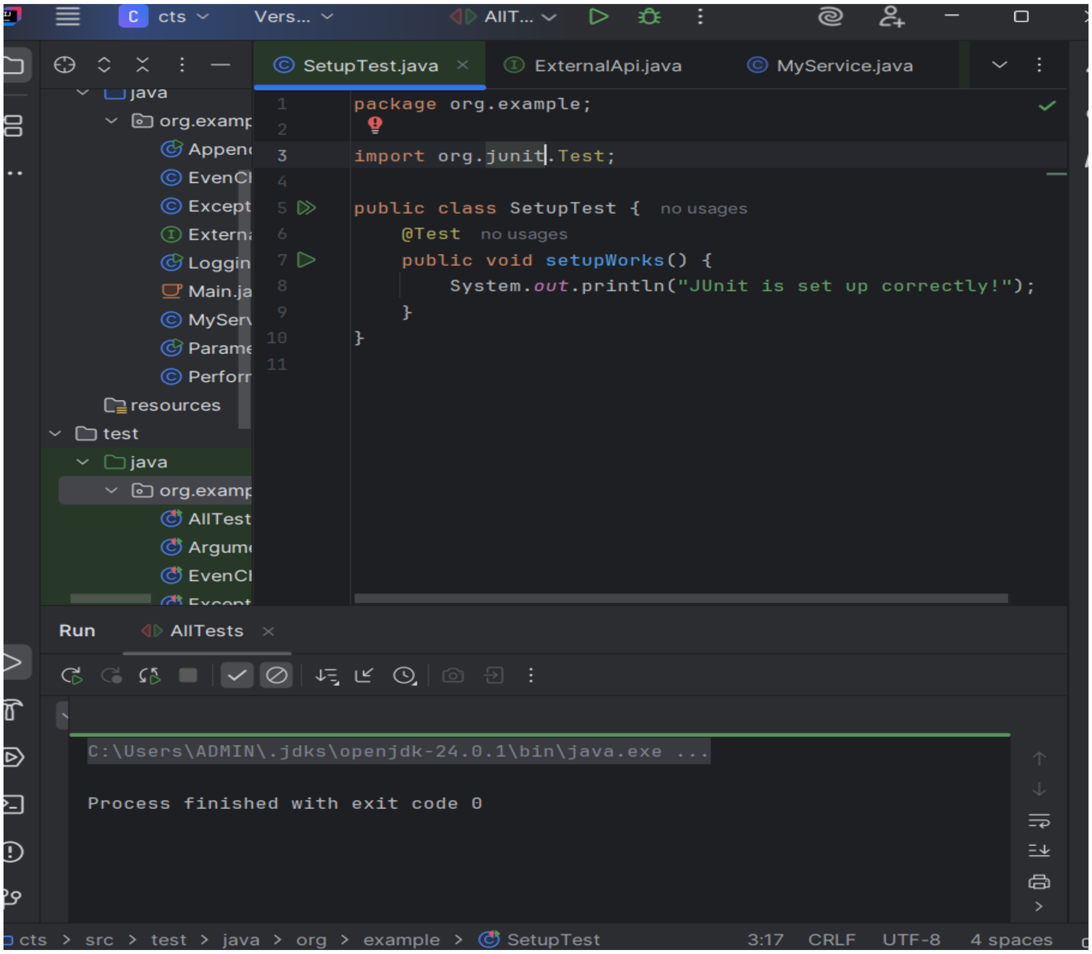
Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

3. Create a new test class in your project.



```
package org.example;

import org.junit.Test;

public class SetupTest {  no usages
    @Test  no usages
    public void setupWorks() {
        System.out.println("JUnit is set up correctly!");
    }
}
```

```
C:\Users\ADMIN\.jdks\openjdk-24.0.1\bin\java.exe ...

Process finished with exit code 0
```

Exercise 2: Writing Basic JUnit Tests

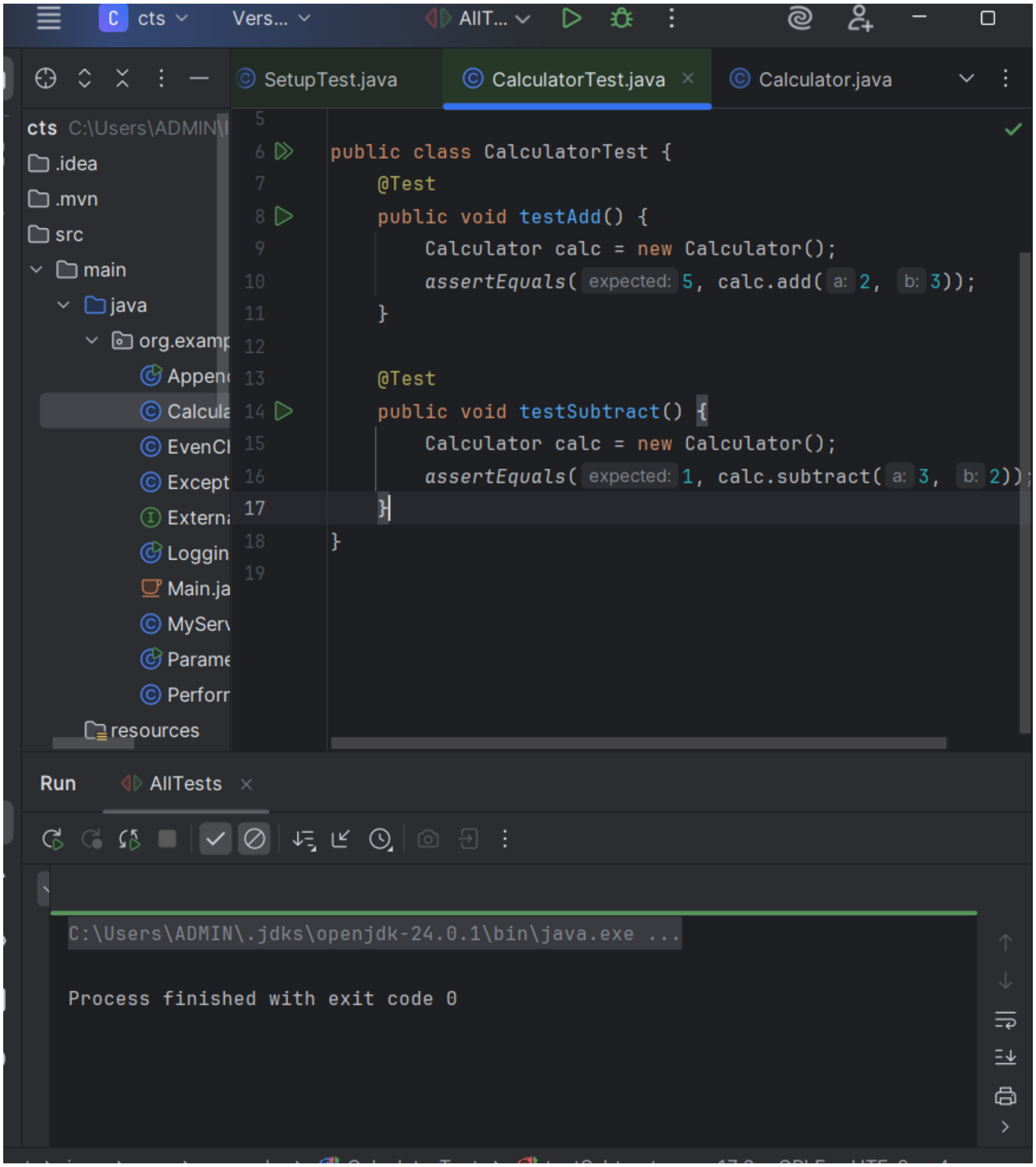
Scenario:

You need to write basic JUnit tests for a simple Java class.

Steps:

1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods.



```
public class CalculatorTest {
    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        assertEquals( expected: 5, calc.add( a: 2,  b: 3));
    }

    @Test
    public void testSubtract() {
        Calculator calc = new Calculator();
        assertEquals( expected: 1, calc.subtract( a: 3,  b: 2));
    }
}
```

```
C:\Users\ADMIN\.jdks\openjdk-24.0.1\bin\java.exe ...

Process finished with exit code 0
```

Exercise 3: Assertions in JUnit

Scenario:

You need to use different assertions in JUnit to validate your test results.

Steps:

1. Write tests using various JUnit assertions.

Solution Code:

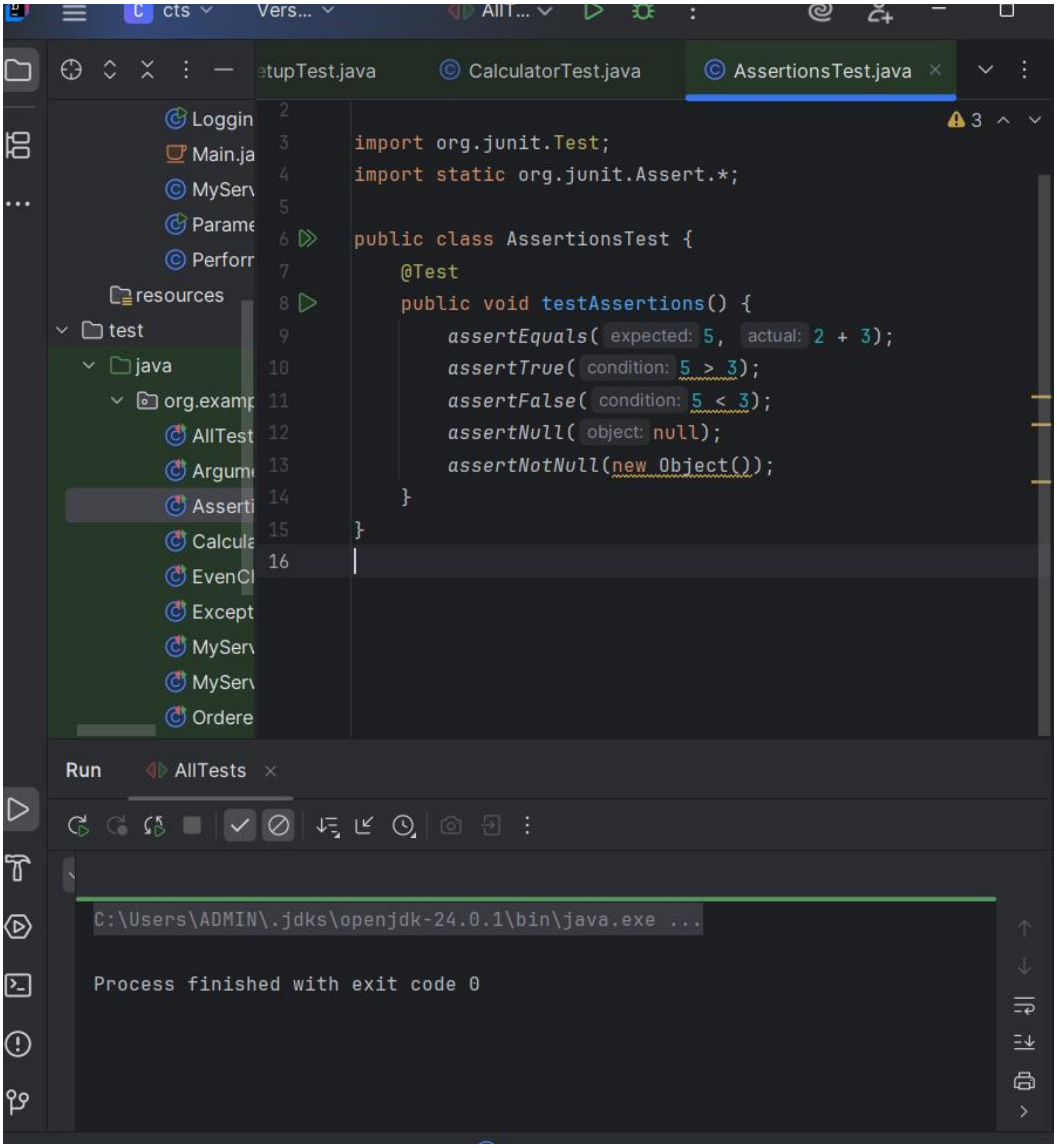
```
public class AssertionsTest {
   @Test     public void
testAssertions() {
      // Assert equals
assertEquals(5, 2 + 3);

      // Assert true
assertTrue(5 > 3);

      // Assert false
assertFalse(5 < 3);

      // Assert null
assertNull(null);

      // Assert not null
      assertNotNull(new Object());
   }
}
```



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

public class AssertionsTest {
    @Test
    public void testAssertions() {
        assertEquals( expected: 5,  actual: 2 + 3);
        assertTrue( condition: 5 > 3);
        assertFalse( condition: 5 < 3);
        assertNull( object: null);
        assertNotNull(new Object());
    }
}
```

```
C:\Users\ADMIN\.jdks\openjdk-24.0.1\bin\java.exe ...

Process finished with exit code 0
```

Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

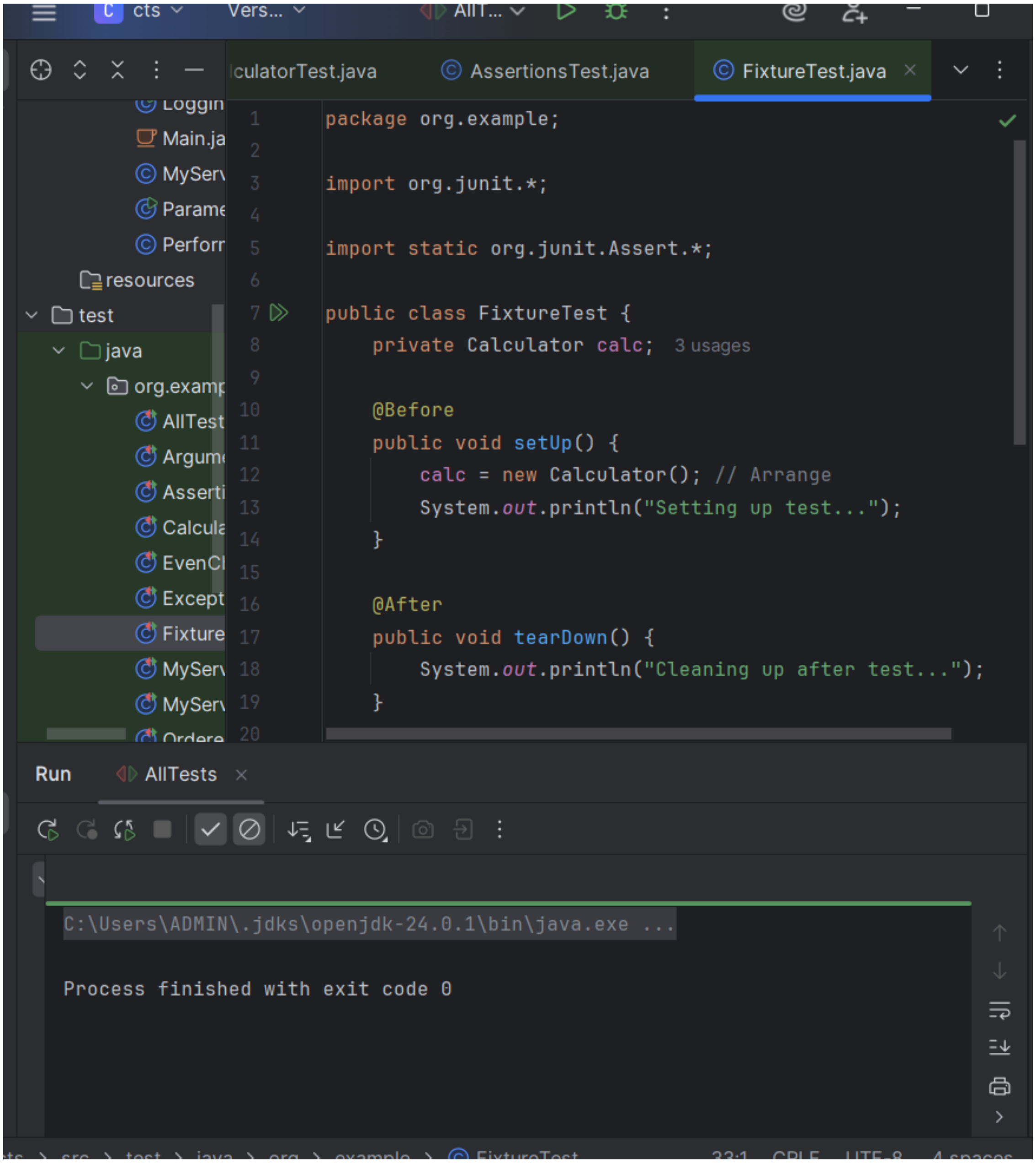
Scenario:

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.



```java
package org.example;

import org.junit.*;

import static org.junit.Assert.*;

public class FixtureTest {
    private Calculator calc;

    @Before
    public void setUp() {
        calc = new Calculator(); // Arrange
        System.out.println("Setting up test...");
    }

    @After
    public void tearDown() {
        System.out.println("Cleaning up after test...");
    }

```

```
C:\Users\ADMIN\.jdks\openjdk-24.0.1\bin\java.exe ...

Process finished with exit code 0
```