1.JUnit Testing Exercises

Exercise 1: Setting Up JUnit

Step 1: Created a project with name ProgrammingTesting in intellij

Step 2: Added dependency to pom.xml:

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

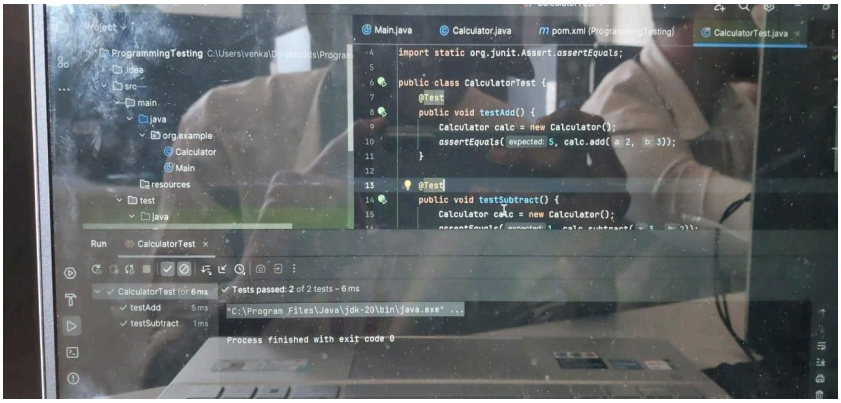
Step 3: Sample Test Class : CalculatorTest.java

```
package org.example;

import org.junit.Test;
import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.Before;


public class CalculatorTest {

   private Calculator calculator;


   @Test
   public void testAdd() {
       calculator = new Calculator();
       // Arrange
       int a = 10;
       int b = 5;

       // Act
       int result = calculator.add(a, b);

       // Assert
       assertEquals(15, result);
   }

}
```

File: Calculator.java

```
package org.example;

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

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

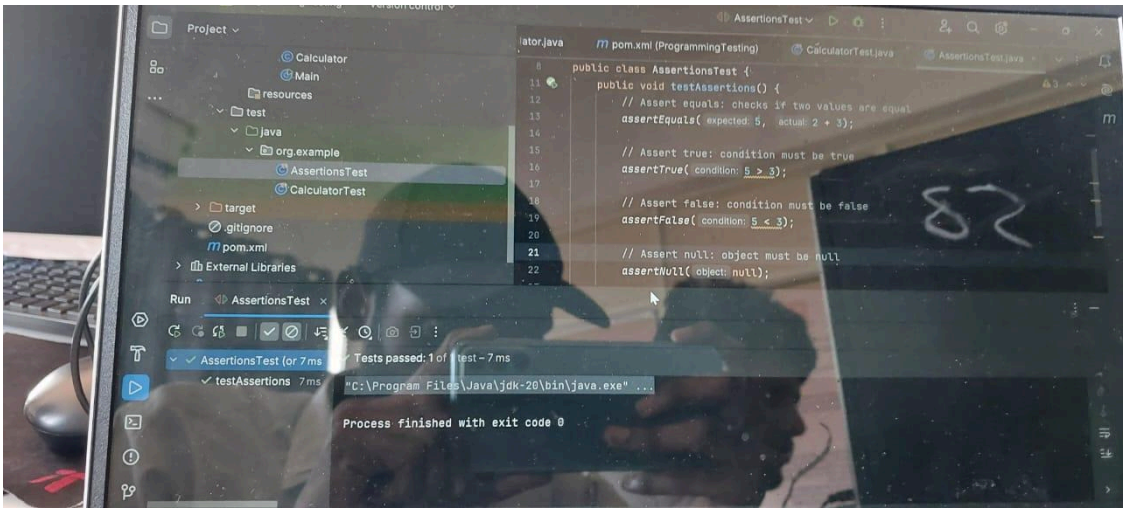
Output:



Exercise 3: Assertions in JUnit

```java
package org.example;

import org.junit.Test;

import static org.junit.Assert.*;


public class AssertionsTest {

    @Test
    public void testAssertions() {
        // Assert equals: checks if two values are equal
        assertEquals(5, 2 + 3);

        // Assert true: condition must be true
        assertTrue(5 > 3);

        // Assert false: condition must be false
        assertFalse(5 < 3);

        // Assert null: object must be null
        assertNull(null);

        // Assert not null: object must not be null
        assertNotNull(new Object());
    }
}
```

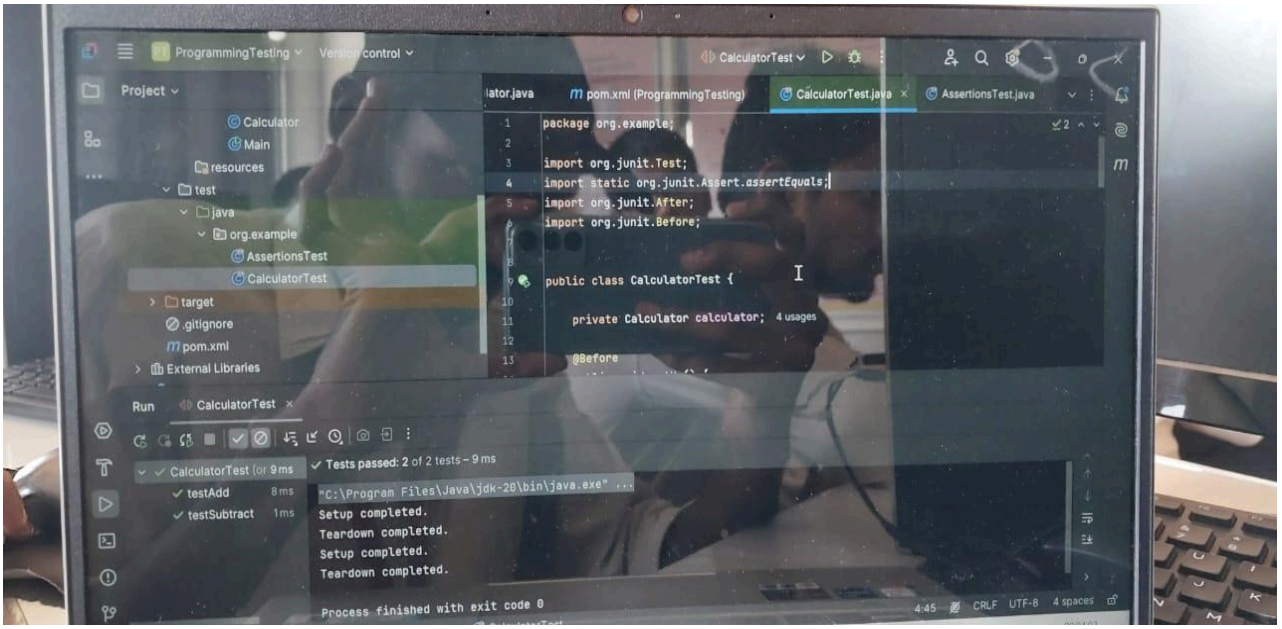
Output:



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

```java
package org.example;

import org.junit.Test;
import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.Before;


public class CalculatorTest {

    private Calculator calculator;

    @Before
    public void setUp() {
        // Setup: Runs before each test
        calculator = new Calculator();
        System.out.println("Setup completed.");
    }

    @After
    public void tearDown() {
        // Teardown: Runs after each test
        calculator = null;
        System.out.println("Teardown completed.");
    }

    @Test
    public void testAdd() {
        // Arrange
        int a = 10;
        int b = 5;

        // Act
        int result = calculator.add(a, b);

        // Assert
        assertEquals(15, result);
    }

    @Test
    public void testSubtract() {
        // Arrange
        int a = 10;
        int b = 3;

        // Act
        int result = calculator.subtract(a, b);

        // Assert
        assertEquals(7, result);
    }
}
```

Output:


ProgrammingTesting
Version control
CalculatorTest
Project
Calculator
Main
resources
test
java
org.example
AssertionsTest
CalculatorTest
target
.gitignore
pom.xml
External Libraries
pom.xml (ProgrammingTesting)
CalculatorTest.java
AssertionsTest.java
package org.example;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.Before;
public class CalculatorTest {
private Calculator calculator; 4 usages
@Before
Run
CalculatorTest
Tests passed: 2 of 2 tests – 9 ms
CalculatorTest (or 9 ms
testAdd 8 ms
testSubtract 1 ms
"C:\Program Files\Java\jdk-20\bin\java.exe" ...
Setup completed.
Teardown completed.
Setup completed.
Teardown completed.
Process finished with exit code 0
4:45
CRLF
UTF-8
4 spaces

Mockito Hands-On Exercises :

Exercise 1: Mocking and Stubbing

Step 1: Adding dependency

```
<dependency>
   <groupId>org.mockito</groupId>
   <artifactId>mockito-junit-jupiter</artifactId>
   <version>5.14.2</version>
   <scope>test</scope>
</dependency>
```

Step 2:

FileName: ExternalApi.java

```
package org.example;

public interface ExternalApi {
   String getData();
}
```

FileName: MyService.java

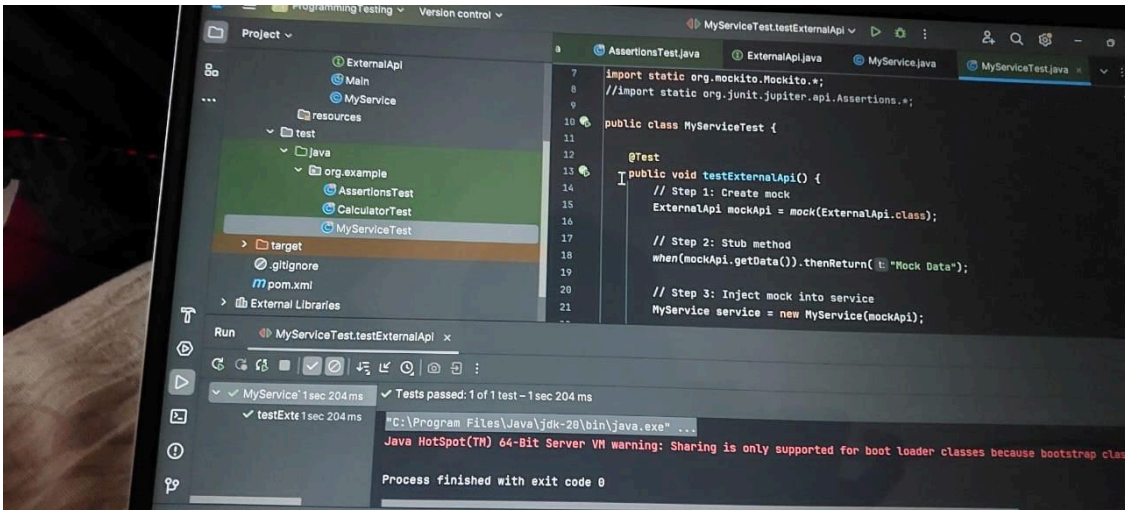
```
package org.example;


public class MyService {
   private final ExternalApi externalApi;

   public MyService(ExternalApi externalApi) {
       this.externalApi = externalApi;
   }

   public String fetchData() {
       return externalApi.getData();
   }
}
```

FileName: MyServiceTest.java

```
package org.example;


import org.junit.Test;

import static org.junit.Assert.*;
import static org.mockito.Mockito.*;
//import static org.junit.jupiter.api.Assertions.*;

public class MyServiceTest {

   @Test
   public void testExternalApi() {
       // Step 1: Create mock
       ExternalApi mockApi = mock(ExternalApi.class);

       // Step 2: Stub method
       when(mockApi.getData()).thenReturn("Mock Data");

       // Step 3: Inject mock into service
```

```
        MyService service = new MyService(mockApi);

        // Step 4: Verify result
        String result = service.fetchData();
        assertEquals("Mock Data", result);
    }
}
```

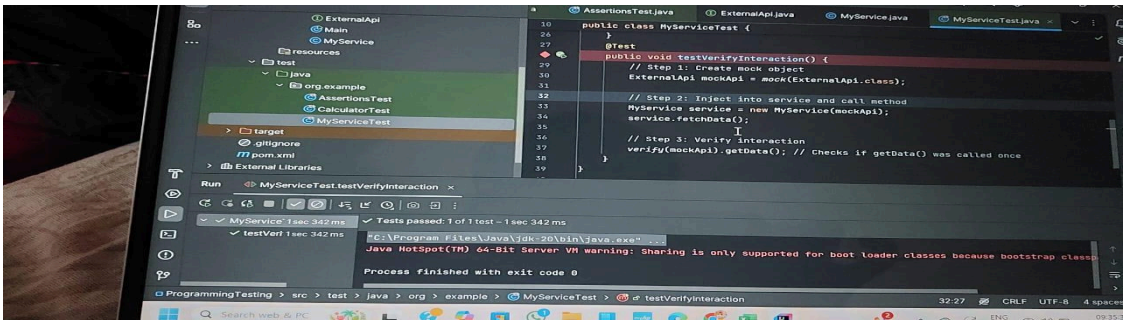
Output:



Exercise 2: Verifying Interactions

```
package org.example;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.mockito.Mockito.*;
//import static org.junit.jupiter.api.Assertions.*;

public class MyServiceTest {
    @Test
    public void testVerifyInteraction() {
        // Step 1: Create mock object
        ExternalApi mockApi = mock(ExternalApi.class);

        // Step 2: Inject into service and call method
        MyService service = new MyService(mockApi);
        service.fetchData();

        // Step 3: Verify interaction
        verify(mockApi).getData(); // Checks if getData() was called once
    }
}
```

Output:



Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

Step 1: Add SLF4J and Logback dependencies to your `pom.xml` file:

```
 <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-api</artifactId>
       <version>1.7.30</version>
   </dependency>
   <dependency>
       <groupId>ch.qos.logback</groupId>
       <artifactId>logback-classic</artifactId>
       <version>1.2.3</version>
   </dependency>
</dependencies>
```

Step 2: Create a Java class that uses SLF4J for logging:
FileName: LoggingExample.java

```
package org.example;


import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LoggingExample {
   private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);

   public static void main(String[] args) {
       logger.error("This is an error message");
       logger.warn("This is a warning message");
   }
}
```

Output: