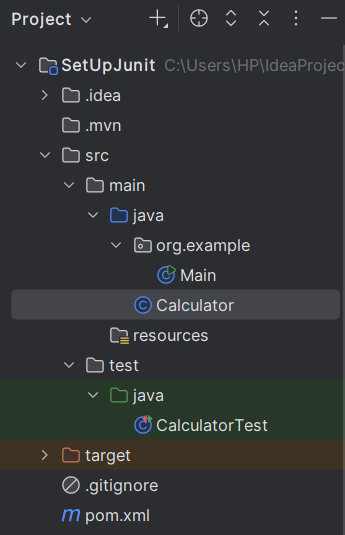
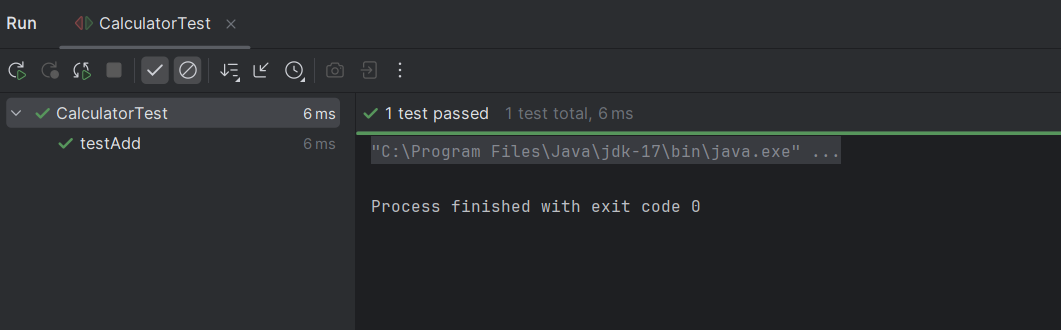
**JUnit Testing Exercises**

**Exercise 1-: Setting up JUnit**

****

|  |
| --- |
| **Calculator.java**  **public class Calculator {  public int add(int a, int b) {  return a + b;  } }**  **CalculatorTest.java**  **import org.junit.Test; import static org.junit.Assert.\*;  public class CalculatorTest {   @Test  public void testAdd() {  Calculator calc = new Calculator();  int result = calc.add(2, 3);  *assertEquals*(5, result);  } }** |

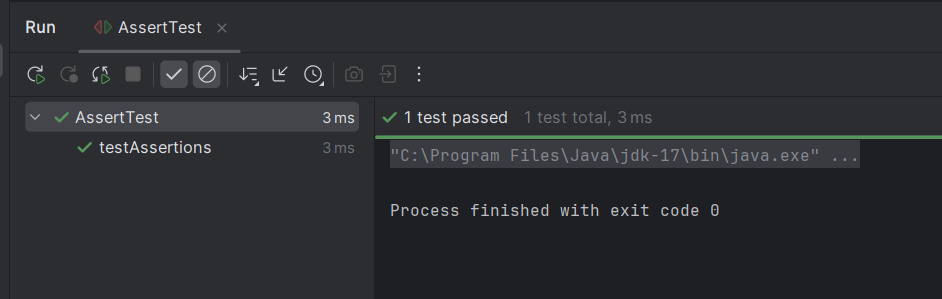
**OUPUT-:**

****

**Exercise 3-: Assertions in Junit**

|  |
| --- |
| import org.junit.Test; import static org.junit.Assert.\*;  public class AssertTest {   @Test  public void testAssertions() {  *assertEquals*(5, 2 + 3);   *assertTrue*(5 > 3);   *assertFalse*(5 < 3);    *assertNull*(null);    *assertNotNull*(new Object());  } } |

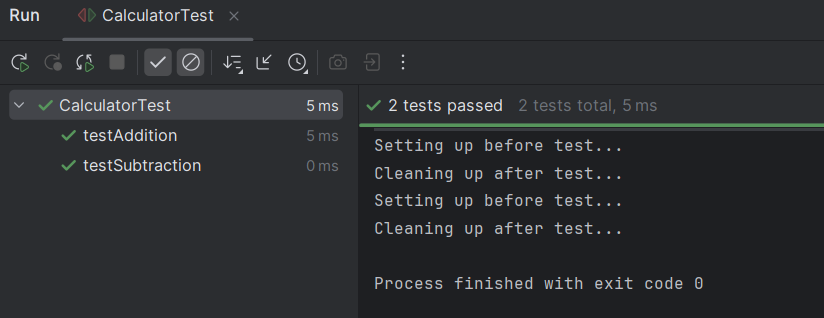
**OUTPUT -:**

****

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

|  |
| --- |
| **Calculator.java**  **public class Calculator {  public int add(int a, int b) {  return a + b;  }  public int sub(int a, int b) {  return a - b;  } }**  **CalculatorTest.java**  **import org.junit.Before; import org.junit.After; import org.junit.Test;  import static org.junit.Assert.\*;  public class CalculatorTest {   private Calculator calc;   @Before  public void setUp() {  calc = new Calculator();  System.*out*.println("Setting up before test...");  }   @After  public void tearDown() {  System.*out*.println("Cleaning up after test...");  }   @Test  public void testAddition() {  int a = 10;  int b = 5;   int result = calc.add(a, b);   *assertEquals*(15, result);  }   @Test  public void testSubtraction() {  int a = 10;  int b = 3;   int result = calc.sub(a, b);   *assertEquals*(7, result);  } }** |

**OUTPUT-:**

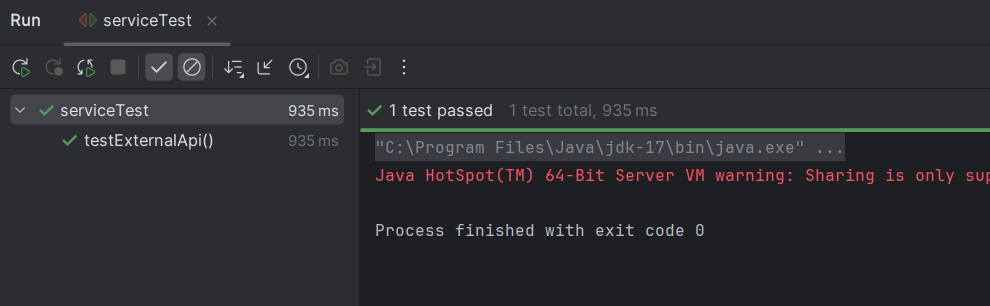
****

**Mockito Hands-On Exercises**

**Exercise 1: Mocking and Stubbing**

|  |
| --- |
| **public interface ExternalAPI {  String getData(); }**  **service.java**  **public class service {  private ExternalAPI api;   public service(ExternalAPI api) {  this.api = api;  }   public String fetchData() {  return api.getData();  } }**  **serviceTest.java**  **import static org.mockito.Mockito.\*; import static org.junit.jupiter.api.Assertions.\*;  import org.junit.jupiter.api.Test; import org.mockito.Mockito;  public class serviceTest {   @Test  public void testExternalApi() {    ExternalAPI mockApi = Mockito.*mock*(ExternalAPI.class);   *when*(mockApi.getData()).thenReturn("Mock Data");   service ser = new service(mockApi);  String result = ser.fetchData();   *assertEquals*("Mock Data", result);  } }** |

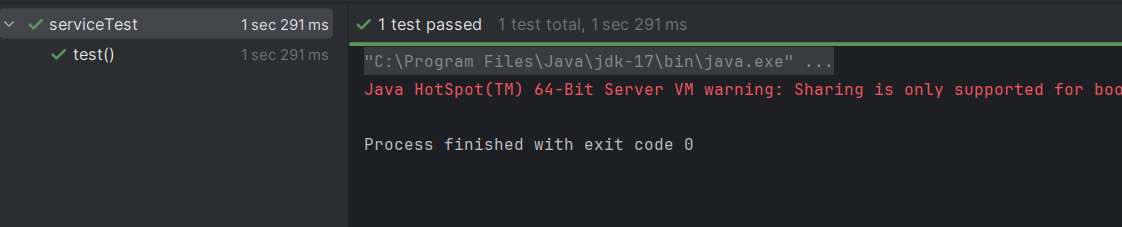
**OUTPUT-:**

****

**Exercise 2: Verifying Interaction**

|  |
| --- |
| **public interface ExternalAPI {  void authenticate();  String getData();  void disconnect(); }**  **service.java**  **public class service {   private ExternalAPI api;   public service(ExternalAPI api) {  this.api = api;  }   public String fetchDataWithLifecycle() {  api.authenticate();  String data = api.getData();  api.disconnect();  return data;  } }**  **serviceTest.java**  **import org.junit.jupiter.api.Test; import org.mockito.InOrder;  import static org.mockito.Mockito.\*; import static org.junit.jupiter.api.Assertions.\*;  public class serviceTest {   @Test  public void test() {   ExternalAPI mockApi = *mock*(ExternalAPI.class);   *when*(mockApi.getData()).thenReturn("Mock Data");   service service = new service(mockApi);  String result = service.fetchDataWithLifecycle();   *assertEquals*("Mock Data", result);   InOrder inOrder = *inOrder*(mockApi);  inOrder.verify(mockApi).authenticate();  inOrder.verify(mockApi).getData();  inOrder.verify(mockApi).disconnect();  } }** |

**OUTPUT-:**

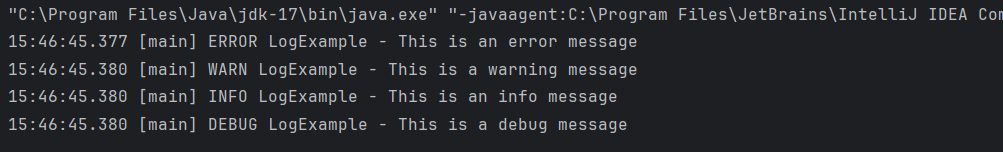
****

**Logging using SLF4J**

**Exercise 1: Logging Error Messages and Warning Levels**

|  |
| --- |
| **LogExample.java**  **import org.slf4j.Logger; import org.slf4j.LoggerFactory;  public class LogExample {  private static final Logger *logger* = LoggerFactory.*getLogger*(LogExample.class);   public static void main(String[] args) {  *logger*.error("This is an error message");  *logger*.warn("This is a warning message");  *logger*.info("This is an info message");  *logger*.debug("This is a debug message");  } }** |

**OUTPUT-:**

****