**HANDSON EXERCISES - WEEK 2**

**Skill : Test driven development and Logging framework**

**JUnit\_Basic Testing Exercises**

**Exercise 1 : Setting Up JUnit**

**CODE :**

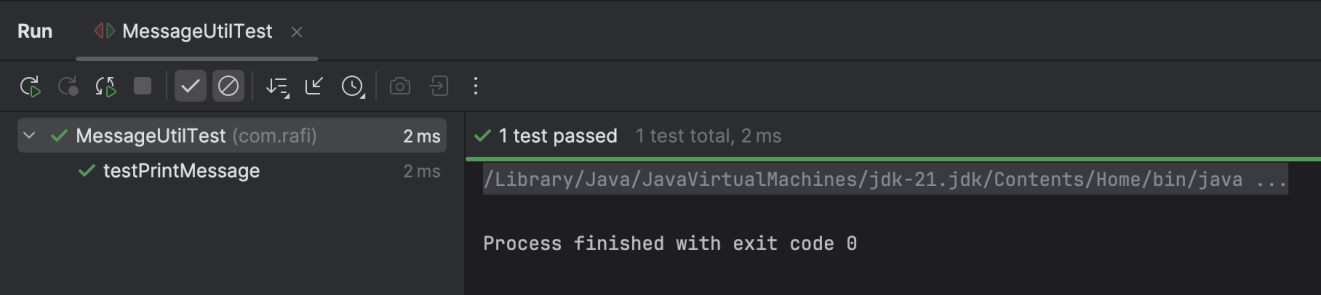
**MessageUtil.java :**

package com.rafi;  
  
public class MessageUtil {  
 private String message;  
  
 public MessageUtil(String message) {  
 this.message = message;  
 }  
  
 public String printMessage() {  
 return message;  
 }  
}

**MessageUtilTest.java :**

package com.rafi;  
  
import org.junit.Test;  
import static org.junit.Assert.assertEquals;  
  
public class MessageUtilTest {  
  
 @Test  
 public void testPrintMessage() {  
 MessageUtil messageUtil = new MessageUtil("Hello World");  
 String message = messageUtil.printMessage();  
 assertEquals("Hello World", message);  
 }  
}

**OUTPUT :**



**Exercise 3 : Assertions in JUnit**

**CODE :**

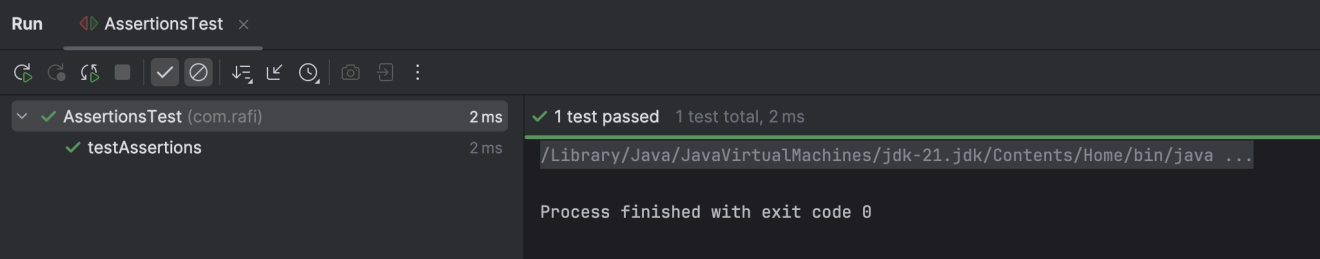
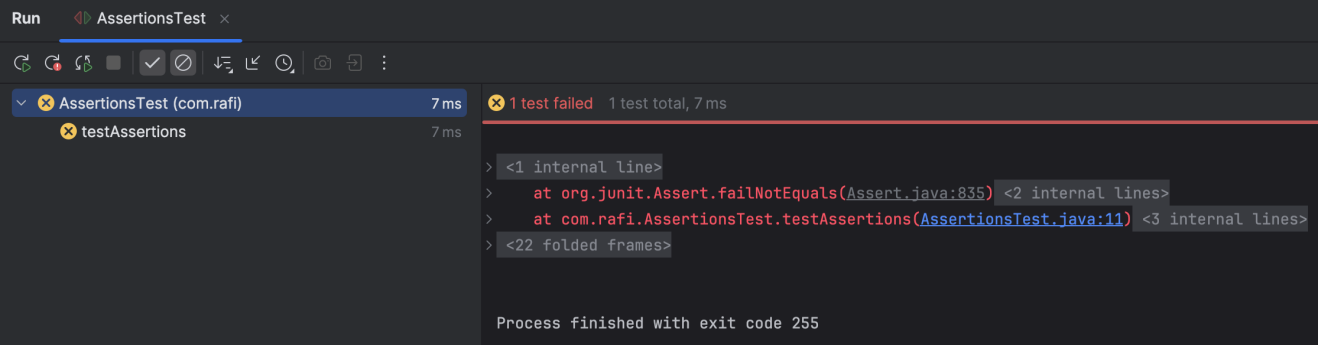
**AssertionTest.java :**

package com.rafi;  
  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class AssertionsTest {  
  
 @Test  
 public void testAssertions() {  
 assertEquals(5, 2 + 3);

*//assertEquals*(5, 2 + 2); this will fail

assertTrue(5 > 3);  
  
 assertFalse(5 < 3);  
  
 assertNull(null);  
  
 assertNotNull(new Object());  
 }  
}

**OUTPUT :**

**Success case**   
  
  
**Sample Failure case**

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

**CODE :**

**Calculator.java :**

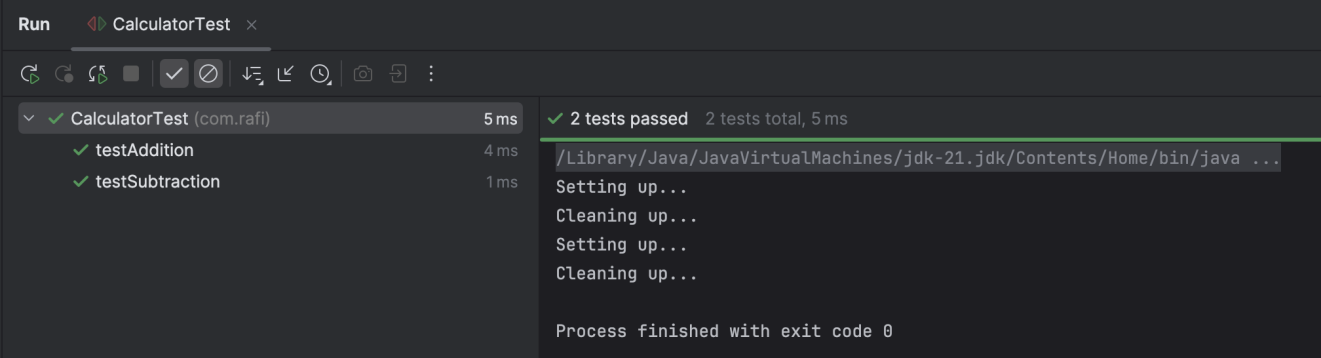
package com.rafi;  
  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
}

**CalculatorTest.java :**

package com.rafi;  
  
import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class CalculatorTest {  
  
 private Calculator calculator;  
  
 @Before  
 public void setUp() {  
 System.*out*.println("Setting up...");  
 calculator = new Calculator();  
 }  
  
 @After  
 public void tearDown() {  
 System.*out*.println("Cleaning up...");  
 calculator = null;  
 }  
  
 @Test  
 public void testAddition() {  
 // Arrange  
 int a = 5;  
 int b = 3;

int result = calculator.add(a, b);  
  
 *assertEquals*(8, result);  
 }  
  
 @Test  
 public void testSubtraction() {  
 // Arrange  
 int a = 10;  
 int b = 4;  
  
 // Act  
 int result = calculator.subtract(a, b);  
  
 // Assert  
 *assertEquals*(6, result);  
 }  
}

**OUTPUT :**



**Mockito exercises**

**Exercise 1 : Mocking and Stubbing**

**CODE :**

**ExternalApi.java :**

package com.rafi;  
  
public interface ExternalApi {  
 String getData();  
}

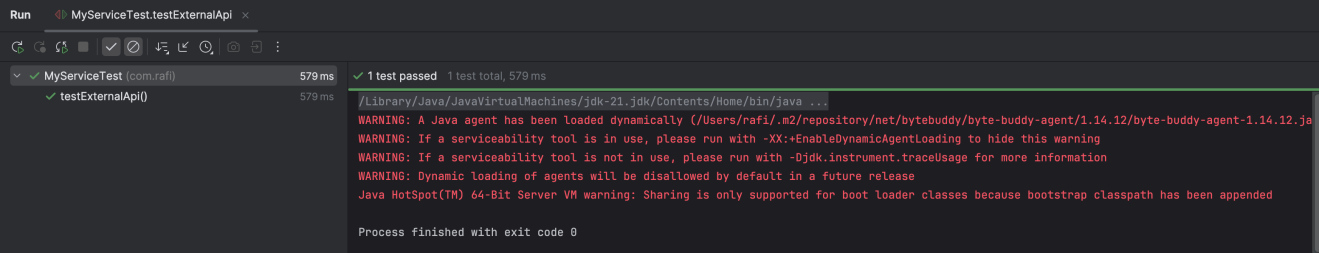
**MyService.java :**

package com.rafi;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**MyServiceTest.java :**

package com.rafi;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.assertEquals;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
  
 ExternalApi mockApi = mock(ExternalApi.class);  
 when(mockApi.getData()).thenReturn("Mock Data");  
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
 assertEquals("Mock Data", result);  
  
 }  
}

**OUTPUT :**



**Exercise 2 : Verifying Interactions**

**CODE :**

**ExternalApi.java :**

package com.rafi;  
  
public interface ExternalApi {  
 String getData();  
}

**MyService.java :**

package com.rafi;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**MyServiceTest.java :**

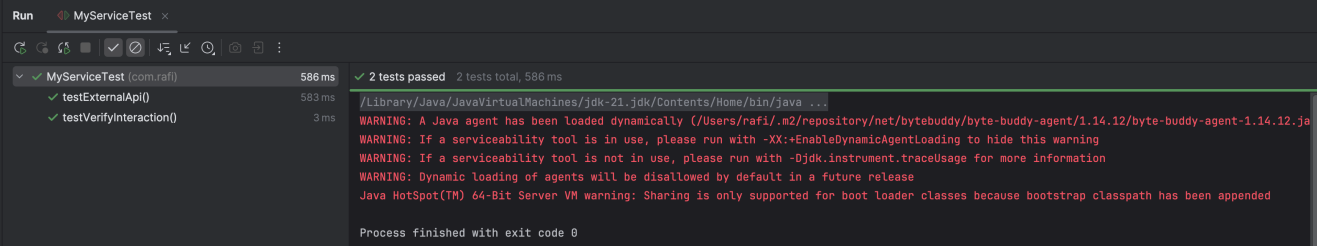
package com.rafi;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
  
 ExternalApi mockApi = *mock*(ExternalApi.class);

*when*(mockApi.getData()).thenReturn("Mock Data");  
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
 *assertEquals*("Mock Data", result);  
 }  
  
 @Test  
 public void testVerifyInteraction() {

ExternalApi mockApi = *mock*(ExternalApi.class);  
 MyService service = new MyService(mockApi);  
 service.fetchData();  
 *verify*(mockApi).getData();

}  
}

**OUTPUT :**



**SL4J Logging exercises**

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

**CODE :**

**LoggingExample.java :**

package com.rafi;  
  
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 :**

