

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;
    }
}
```

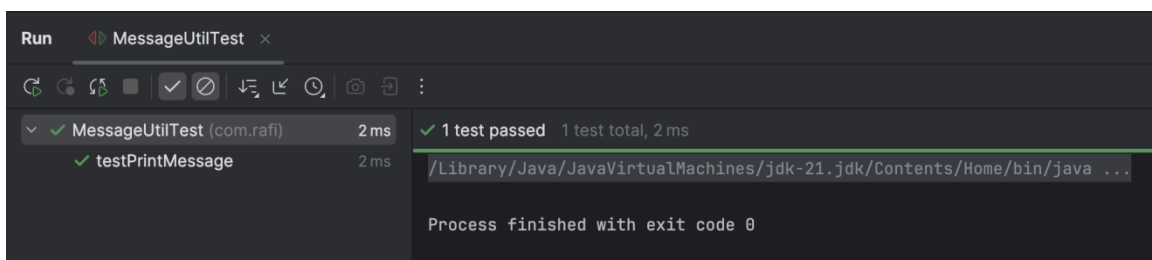
MessageUtilTest.java :

```
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);

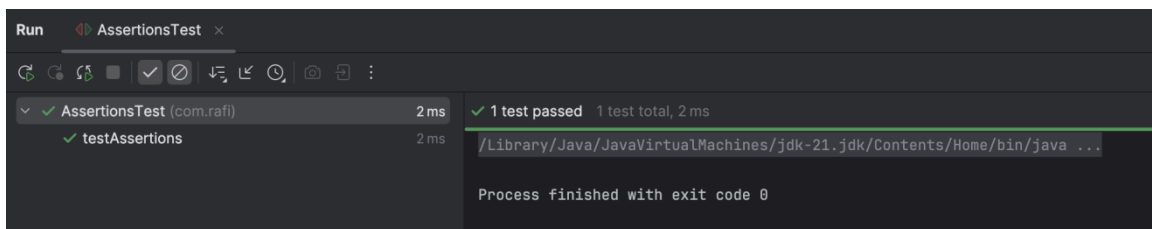
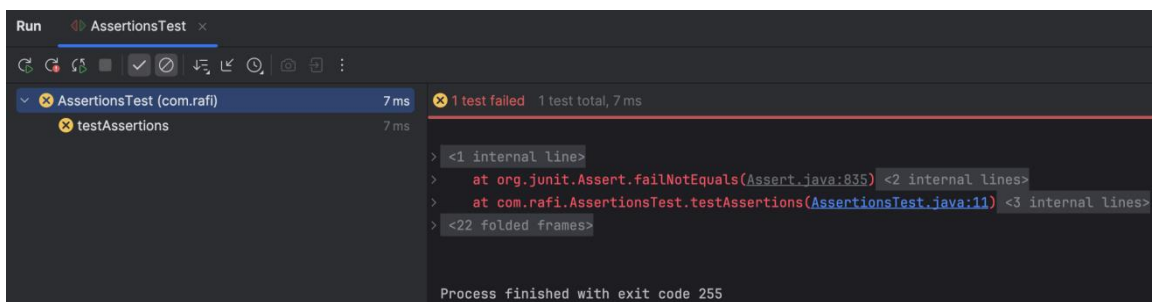
        //assertEquals(5, 2 + 2); this will fail

        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;
```

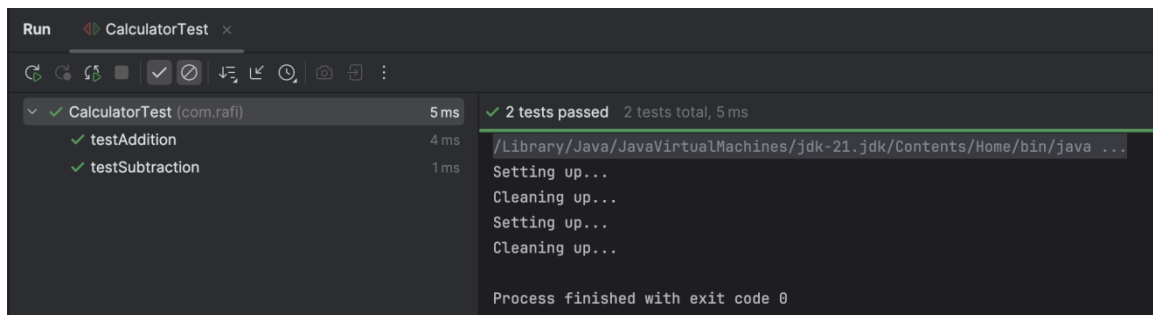
```
    // Act
```

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

```
    // Assert
```

```
    assertEquals(6, result);
```

OUTPUT :



The screenshot shows the 'Run' window of an IDE. The title bar says 'Run' and 'CalculatorTest'. Below the title bar is a toolbar with icons for running, debugging, and other actions. The main area is divided into two panes. The left pane shows a tree view of the test results: 'CalculatorTest (com.rafi)' with a green checkmark and '5 ms', 'testAddition' with a green checkmark and '4 ms', and 'testSubtraction' with a green checkmark and '1 ms'. The right pane shows the output of the tests: '2 tests passed 2 tests total, 5 ms'. Below this, there is a list of system messages: '/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...', 'Setting up...', 'Cleaning up...', 'Setting up...', 'Cleaning up...', and 'Process finished with exit code 0'.

```
Run CalculatorTest x
```

✓ CalculatorTest (com.rafi) 5 ms

✓ testAddition 4 ms

✓ testSubtraction 1 ms

✓ 2 tests passed 2 tests total, 5 ms

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...

Setting up...

Cleaning up...

Setting up...

Cleaning up...

Process finished with exit code 0

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 :

```
Run MyServiceTest.testExternalApi() x
✓ MyServiceTest (com.rafi) 579 ms ✓ 1 test passed 1 test total 579 ms
  ✓ testExternalApi() 579 ms
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...
WARNING: A Java agent has been loaded dynamically (/Users/rafi/.m2/repository/net/bytebuddy/byte-buddy-agent/1.14.12/byte-buddy-agent-1.14.12.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
Process finished with exit code 0
```

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);
```

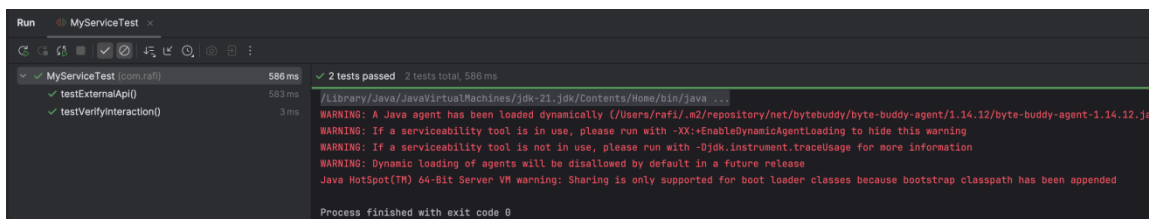
```
when(mockApi.getData()).thenReturn("Mock Data");
MyService service = new MyService(mockApi);
String result = service.fetchData();
assertEquals("Mock Data", result);
}
```

```
@Test
public void testVerifyInteraction() {
```

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

```
}
```

OUTPUT :



```
Run MyServiceTest x
✓ 2 tests passed 2 tests total, 586 ms
MyServiceTest (com.rafi) 586 ms
  ✓ testExternalApi() 583 ms
  ✓ testVerifyInteraction() 3 ms
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java ...
WARNING: A Java agent has been loaded dynamically (/Users/rafi/.m2/repository/net/bytebuddy/byte-buddy-agent/1.14.12/byte-buddy-agent-1.14.12.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.Instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
Process finished with exit code 0
```

SL4J Logging exercises

Exercise 1 : Logging Error Messages and Warning Levels

CODE :

LoggingExample.java :

```
package com.rafi;

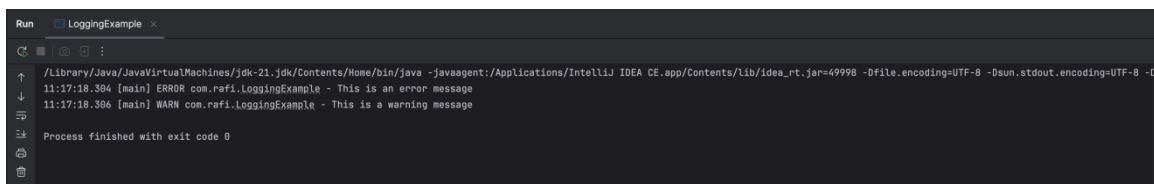
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

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 :



```
Run LoggingExample x
/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/Lib/idea_rt.jar-49998 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -D
11:17:18.394 [main] ERROR com.rafi.LoggingExample - This is an error message
11:17:18.396 [main] WARN com.rafi.LoggingExample - This is a warning message
Process finished with exit code 0
```