SuperSet ID : 6314238 **BAPANAPALLI PREM SAI SIDDHIK**

HANDSON EXERCISES - WEEK 2

Skill : Test driven development and Logging framework

JUnit_Basic Testing Exercises

Exercise 1 : Setting Up JUnit

CODE :

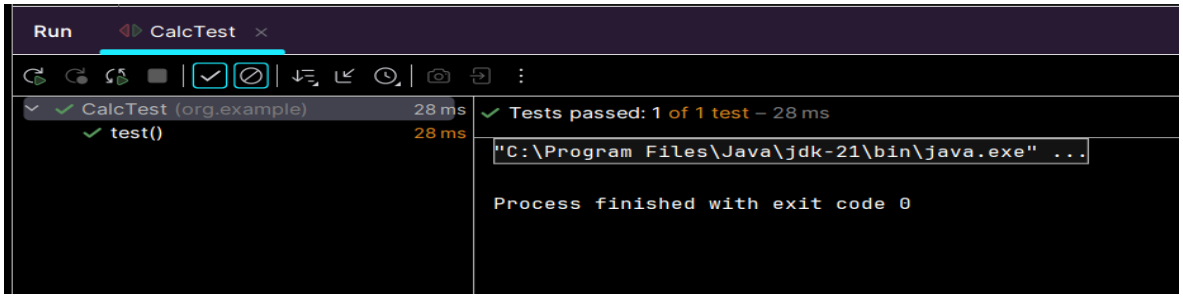
Calc.java :

```
package org.example;
public class Calc {
   public int divide(int num1,int num2)
   {
      return num1/num2;
   }
}
```

MessageUtilTest.java :

```
package org.example;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
class CalcTest {
   @Test
   public void test()
   {
      Calc c=new Calc();
      assertEquals(2,c.divide(10,5));
   }
}
```

OUTPUT :



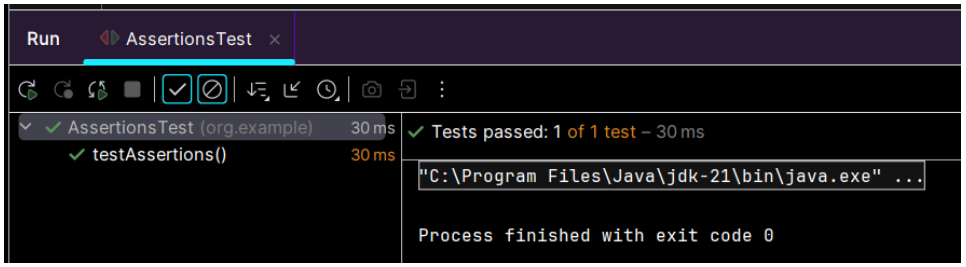
Exercise 3 : Assertions in JUnit

CODE:

AssertionsTest.java :

```
package org.example;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class AssertionsTest {
    @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

CODE :

Calculator.java :

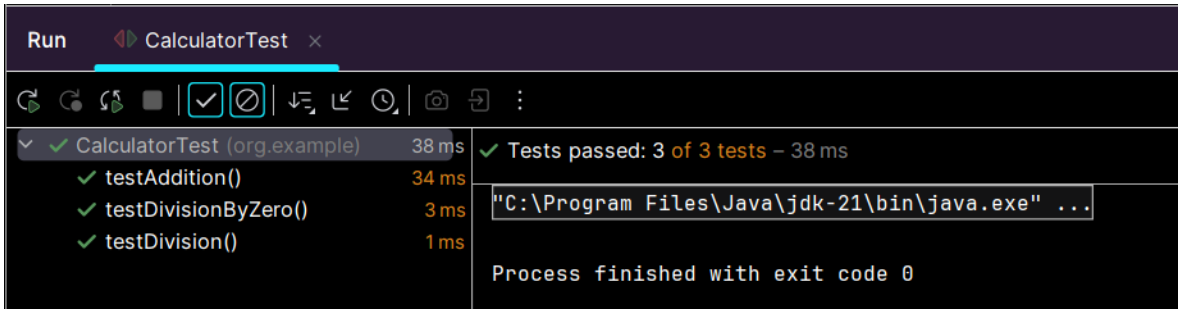
```
package org.example;
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
    public int divide(int a, int b) {
        if (b == 0) throw new ArithmeticException("Divide by zero");
        return a / b;
    }
}
```

CalculatorTest.java :

```
package org.example;

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() {
        calculator = new Calculator();
    }

    @After
    public void tearDown() {
        calculator = null;
    }

    @Test
```

```
    public void testAddition() {
        int result = calculator.add(10, 5);
        assertEquals(15, result);
    }

    @Test
    public void testDivision() {
        int result = calculator.divide(20, 4);
        assertEquals(5, result);
    }

    @Test(expected = ArithmeticException.class)
    public void testDivisionByZero() {
        calculator.divide(10, 0);
    }
}
```

OUTPUT :



Mockito exercises

Exercise 1 : Mocking and Stubbing

CODE :

ExternalApi.java :

```
package org.example;

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

MyService.java :

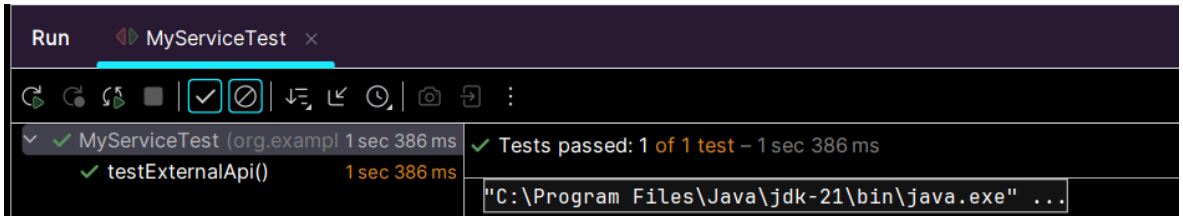
```
package org.example;
public class MyService {
    private ExternalApi api;

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

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

MyServiceTest.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 MyServiceTest {
```

```
  @Test
  public void testExternalApi() {
      ExternalApi mockApi = Mockito.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 :

<u>ExternalApi.java :</u>

```
package org.example;

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

<u>MyService.java :</u>

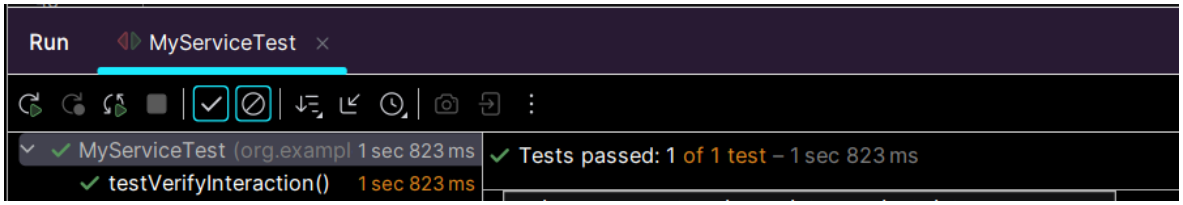
```
package org.example;

public class MyService {
    private final ExternalApi api;

    public MyService(ExternalApi api) {
        this.api = api;
    }
```

```
    public String fetchData() {

        return api.getData();
    }

    public void fetchAndProcess() {

        api.getData();
    }
}
```

MyServiceTest.java :

```
package org.example;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.verify;

@ExtendWith(MockitoExtension.class)
public class MyServiceTest {

    @Mock
    ExternalApi mockApi;

    @InjectMocks
    MyService service;

    @Test
    void testVerifyInteraction() {
        service.fetchAndProcess();
        verify(mockApi).getData();
    }
}
```

OUTPUT :



SL4J Logging exercises

Exercise 1 : Logging Error Messages and Warning Levels

CODE :

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 :