JUNIT BASIC TESTING

Superset ID : 6384831

Name : Mohana Priya N

E-mail : mohanapriya.2205056@srec.ac.in

Mandatory Questions:

1) Exercise 1: Setting Up JUnit

Solution:

// Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>JUnitDemo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

// HelloWorld.java

```
public class HelloWorld {
    public String sayHello() {
        return "Hello, JUnit!";
    }}
```

// HelloWorldTest.java

```
  import org.junit.Test;
import static org.junit.Assert.*;
public class HelloWorldTest {
    @Test
    public void testSayHello() {
        HelloWorld hello = new HelloWorld();
        assertEquals("Hello, JUnit!", hello.sayHello());
    }
}
```

2) Exercise 3: Assertions in JUnit

Solution:

```
import org.junit.Test;
import static org.junit.Assert.*;
public class AssertionsTest {
    @Test
    public void testAssertions() {
        assertEquals(5, 2 + 3);
        assertTrue(5 > 3);
        assertFalse(2 > 10);
        Object obj1 = null;
        assertNull(obj1);
        Object obj2 = new Object();
        assertNotNull(obj2);
    }
}
```

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

Solution:

//Calculator.java

```
public class Calculator {
    public int multiply(int a, int b) {
        return a * b;
    }
    public int divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Divide by zero");
        }
        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 calculator;
    @Before
    public void setUp() {
        calculator = new Calculator();
        System.out.println("Setup complete.");
    }
    @After
    public void tearDown() {
        System.out.println("Test finished.\n");
    }
    @Test
    public void testMultiply() {
        int a = 5, b = 4;
        int result = calculator.multiply(a, b);
        assertEquals(20, result);
    }
    @Test
    public void testDivide() {
        assertEquals(2, calculator.divide(10, 5));
    }
    @Test(expected = IllegalArgumentException.class)
    public void testDivideByZero() {
        calculator.divide(10, 0);
    }
}
```

Other Questions:

4) Exercise 2: Writing Basic JUnit Tests

Solution:

//Calculator.java

```
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

```
import org.junit.Test;
import static org.junit.Assert.*;
```

```
public class CalculatorTest {
    @Test
    public void testAddition() {
        Calculator calc = new Calculator();
        assertEquals(15, calc.add(10, 5));
    }
    @Test
    public void testSubtraction() {
        Calculator calc = new Calculator();
        assertEquals(5, calc.subtract(10, 5));
    }
}
```

JUNIT ADVANCED TESTING

Other Questions:

1) Exercise 1: Parameterized Tests

Solution:

// EvenChecker.java

```
public class EvenChecker {
    public boolean isEven(int number) {
        return number % 2 == 0;
    }
}
```

//EvenCheckerTest.java

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;
import static org.junit.jupiter.api.Assertions.*;
public class EvenCheckerTest {
    EvenChecker checker = new EvenChecker();
    @ParameterizedTest
    @ValueSource(ints = {2, 4, 6, 8, 10})
    public void testIsEven(int number) {
        assertTrue(checker.isEven(number));
    }
    @ParameterizedTest
    @ValueSource(ints = {1, 3, 5, 7, 9})
    public void testIsNotEven(int number) {
        assertFalse(checker.isEven(number));
    }
}
```

2) Exercise 2: Test Suites and Categories

Solution:

//Test class 1:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class MathTests {
    @Test
    public void testAdd() {
        assertEquals(4, 2 + 2);
    }
}
```

//Test class 2:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class StringTests {
    @Test
    public void testLength() {
        assertEquals(5, "Hello".length());
    }
}
```

//AllTests.java

```
import org.junit.platform.suite.api.SelectClasses;
import org.junit.platform.suite.api.Suite;
@Suite
@SelectClasses({
    MathTests.class,
    StringTests.class
})
public class AllTests {
}
```

3) Exercise 3: Test Execution Order

Solution:

// **OrderedTests.java**

```
import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class OrderedTests {
    @Test
    @Order(1)
    public void testStart() {
        System.out.println("Test 1: Start");
        assertTrue(true);
    }
    @Test
    @Order(2)
    public void testMiddle() {
        System.out.println("Test 2: Middle");
        assertEquals(10, 5 + 5);
    }
    @Test
    @Order(3)
    public void testEnd() {
        System.out.println("Test 3: End");
        assertNotNull("JUnit");
    }
}
```

4) Exercise 4: Exception Testing

Solution:

// ExceptionThrower.java

```
public class ExceptionThrower {
    public void throwException() {
        throw new IllegalArgumentException("Invalid input!");
    }
}
```

// ExceptionThrowerTest.java

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class ExceptionThrowerTest {
    @Test
    public void testExceptionThrown() {
        ExceptionThrower obj = new ExceptionThrower();
        assertThrows(IllegalArgumentException.class, obj::throwException);
    }
}
```

5) Exercise 5: Timeout and Performance Testing

Solution:

```
//PerformanceTester.java
public class PerformanceTester {
   public void performTask() throws InterruptedException {
      Thread.sleep(100);
   }
}
// PerformanceTesterTest.java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import java.time.Duration;
public class PerformanceTesterTest {
   @Test
   public void testPerformanceWithinTime() {
      PerformanceTester tester = new PerformanceTester();
      assertTimeout(Duration.ofMillis(200), () -> {
         tester.performTask();
      });
   }
}
```