**Week 2 – JUnit_Mandatory_HandsOn**
**Skill: JUnit Testing**
**Candidate: Projita Kar**
**Superset ID: 6407705**
**Type: Mandatory Hands-On**

# Exercise 1: Setting Up JUnit

**Scenario:**
**You need to set up JUnit in your Java project to start writing unit tests.**

**Steps:**

**1. Create a new Java project in your IntelliJ IDE.**

Project Name: JUnitSetup

**2. Add JUnit dependency to your project. If you are using Maven, add the following to your**
**pom.xml:**

```
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.13.2</version>
 <scope>test</scope>
</dependency>
```

**Code:-**

JUnitSetup\pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>JUnitSetup</artifactId>
   <version>1.0-SNAPSHOT</version>

   <properties>
       <maven.compiler.source>21</maven.compiler.source>
       <maven.compiler.target>21</maven.compiler.target>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

```
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

3. **Create a new test class in your project.**

- Created Java class Calculator.java:- JUnitSetup\src\main\java\com\example\Calculator.java

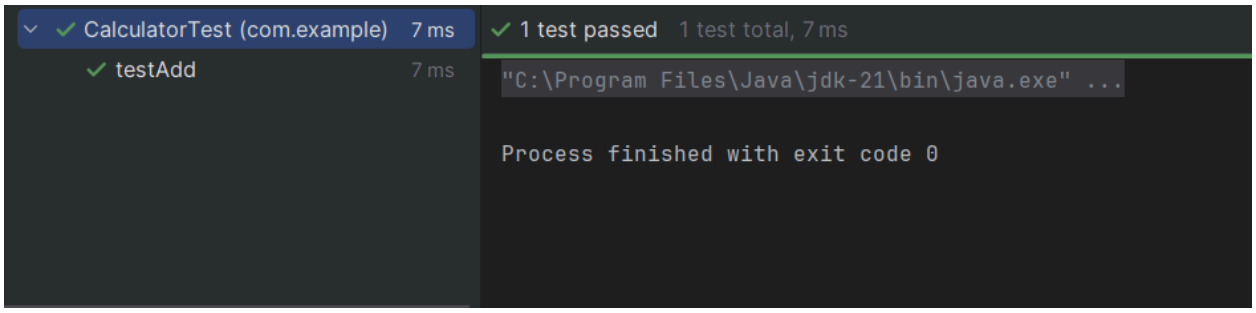
```
package com.example;



public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

}
```

- Created test class CalculatorTest.java: JUnitSetup\src\test\java\com\example\CalculatorTest.java

```
package com.example;

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        int result = calc.add(3, 4);
```

```
        assertEquals(7, result);
    }
}
```

**Output:- Run CalculatorTest.java**




# Exercise 3: Assertions in JUnit

**Scenario:**
**You need to use different assertions in JUnit to validate your test results.**

**Steps:**
**1. Write tests using various JUnit assertions.**

**Solution Code:**

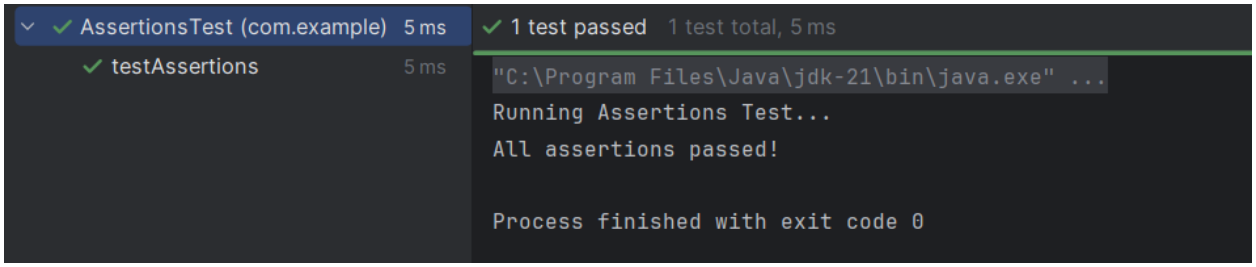
```
public class AssertionsTest {
 @Test
 public void testAssertions() {
 // Assert equals
 assertEquals(5, 2 + 3);
 // Assert true
 assertTrue(5 > 3);
 // Assert false
 assertFalse(5 < 3);
 // Assert null
 assertNull(null);
 // Assert not null
 assertNotNull(new Object());
 }
}
```

**Code:-**

src/test/java/com/example/AssertionsTest.java

```java
package com.example;

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

public class AssertionsTest {

   @Test
   public void testAssertions() {

       System.out.println("Running Assertions Test...");

       assertEquals(5, 2 + 3);


       assertTrue(5 > 3);

       assertFalse(5 < 3);

       assertNull(null);

       assertNotNull(new Object());

       System.out.println("All assertions passed!");
   }
}
```

**Output**:-




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

**Scenario:**

**You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup**
**and teardown methods.**
**Steps:**
**1. Write tests using the AAA pattern.**
**2. Use @Before and @After annotations for setup and teardown methods.**

**Note**: Using the same Calculator class from Exercise 1.

**Code:-**

src/test/java/com/example/CalculatorAdvancedTest.java

```
package com.example;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class CalculatorAdvancedTest {

   private Calculator calculator;

   // 🔹 Setup method: runs before each test
   @Before
   public void setUp() {
       calculator = new Calculator();  // Arrange
       System.out.println("Setup complete");
   }

   // 🔹 Teardown method: runs after each test
   @After
   public void tearDown() {
       calculator = null;
       System.out.println("Teardown complete");
   }

   // 🔹 Test 1: Using Arrange-Act-Assert
   @Test
   public void testAddPositiveNumbers() {
       // Act
       int result = calculator.add(10, 20);
       // Assert
       assertEquals(30, result);
   }

   // 🔹 Test 2
```

```java
    @Test
    public void testAddNegativeNumbers() {
        int result = calculator.add(-5, -3);
        assertEquals(-8, result);
    }
}
```

**Output:-**

```
CalculatorAdvancedTest (com.e 9 ms        2 tests passed  2 tests total, 9 ms
    testAddPositiveNumbers    8 ms        "C:\Program Files\Java\jdk-21\bin\java.exe" ...
    testAddNegativeNumbers    1 ms        Setup complete
                                          Teardown complete
                                          Setup complete
                                          Teardown complete

                                          Process finished with exit code 0
```