Nested and Repeated Tests

Step 1: Configuring pom.xml to add JUnit5 dependencies

```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>UsingJUnit</groupId>
  <artifactId>UsingJUnit</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
     <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.4.2</version>
     </dependency>

      <dependency>
      <groupId>org.junit.platform</groupId>
      <artifactId>junit-platform-launcher</artifactId>
      <version>1.2.0</version>
   </dependency>

  </dependencies>
</project>
```

Step 2: Create a Java class Calculator for testing purposes.

package com.ecommerce.tests;

```java
public class Calculator
{
    public int add(int a, int b) {
        return a + b;
    }
}
```

Step 3: Creating a JUnit class NestedCases

```java
package com.ecommerce.tests;

import org.junit.jupiter.api.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.junit.platform.runner.JUnitPlatform;
import org.junit.runner.RunWith;

@DisplayName("JUnit 5 Nested Example")
@RunWith(JUnitPlatform.class)
public class NestedCases {

    @BeforeAll
    static void beforeAll() {
        System.out.println("Before all test methods");
    }

    @BeforeEach
    void beforeEach() {
        System.out.println("Before each test method");
    }

    @AfterEach
    void afterEach() {
        System.out.println("After each test method");
```

```java
    }

    @AfterAll
    static void afterAll() {
        System.out.println("After all test methods");
    }

    @Nested
    @DisplayName("Tests for the method A")
    class A {

        @BeforeEach
        void beforeEach() {
            System.out.println("Before each test method of the A class");
        }

        @AfterEach
        void afterEach() {
            System.out.println("After each test method of the A class");
        }

        @Test
        @DisplayName("Example test for method A")
        void sampleTestForMethodA() {
            System.out.println("Example test for method A");
        }

        @Nested
        @DisplayName("When X is true")
        class WhenX {

            @BeforeEach
            void beforeEach() {
                System.out.println("Before each test method of the WhenX class");
            }

            @AfterEach
```

```java
        void afterEach() {
            System.out.println("After each test method of the WhenX
class");
        }

        @Test
        @DisplayName("Example test for method A when X is true")
        void sampleTestForMethodAWhenX() {
            System.out.println("Example test for method A when X is
true");
        }
      }
    }
}
```

Step 4: Creating a JUnit class RepeatedTests

```java
package com.ecommerce.tests;

import org.junit.jupiter.api.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.junit.platform.runner.JUnitPlatform;
import org.junit.runner.RunWith;
import static org.junit.jupiter.api.Assertions.assertEquals;

@DisplayName("JUnit 5 Repeated Tests Example")
@RunWith(JUnitPlatform.class)
public class RepeatedTests {
    @BeforeAll
    public static void init(){
        System.out.println("Before All init() method called");
    }

    @BeforeEach
```

```java
        public void initEach(){
            System.out.println("Before Each initEach() method called");
        }

        @Test
        @DisplayName("Add operation test")
        @RepeatedTest(5)
        void addNumber(TestInfo testInfo) {
            Calculator calculator = new Calculator();
            Assertions.assertEquals(2, calculator.add(1, 1), "1 + 1 should equal 2");
            System.out.println("===addNumber testcase executed===");
        }

        @AfterEach
        public void cleanUpEach(){
            System.out.println("After Each cleanUpEach() method called");
        }

        @AfterAll
        public static void cleanUp(){
            System.out.println("After All cleanUp() method called");
        }
}
```