**JUnit\_Basic Testing Exercises**

**1. SETTING UP JUNIT**

**CODE**

**App.java**

package com.example;  
  
public class App {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

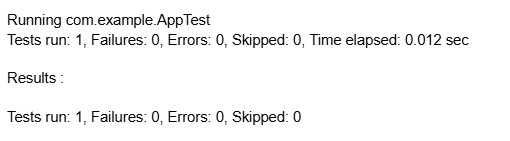
**AppTest.java**

package com.example;  
  
import static org.junit.Assert.\*;  
import org.junit.Test;  
  
public class AppTest {  
 @Test  
 public void testAddition() {  
 App app = new App();  
 assertEquals(8, app.add(5, 3));  
 }  
}

**pom.xml**

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

**OUTPUT:**



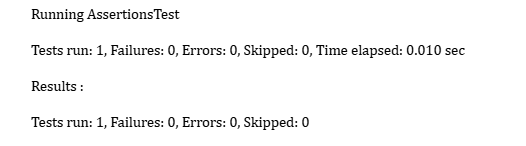
**3.ASSERTIONS IN JUNIT**

**CODE:**

**AssertionsTest.java**

import static org.junit.Assert.\*;  
import org.junit.Test;  
  
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());   
 }   
}

**OUTPUT:**



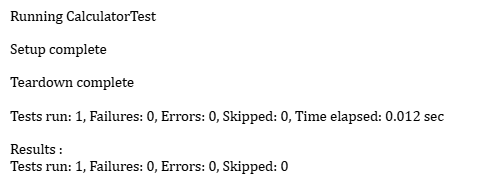
**4.Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**CODE:**

**CalculatorTest.java**

import static org.junit.Assert.\*;  
import org.junit.Before;  
import org.junit.After;  
import org.junit.Test;  
  
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("Teardown complete");  
 }  
  
 @Test  
 public void testAddition() {  
 // Arrange is done in setUp()  
 // Act  
 int result = calculator.add(4, 6);  
 // Assert  
 assertEquals(10, result);  
 }  
}  
  
class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

**OUTPUT:**



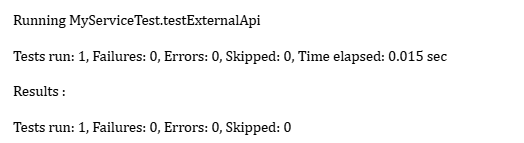
**Mockito exercises**

**1. Mocking and Stubbing**

**CODE**

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);   
 }   
}  
  
interface ExternalApi {  
 String getData();  
}  
  
class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**OUTPUT:**



**2.** **Verifying Interactions**

**CODE:**

import static org.mockito.Mockito.\*;   
import org.junit.jupiter.api.Test;   
import org.mockito.Mockito;   
  
public class MyServiceTest {   
 @Test   
 public void testVerifyInteraction() {   
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);   
 MyService service = new MyService(mockApi);   
 service.fetchData();   
 verify(mockApi).getData();   
 }   
}  
  
interface ExternalApi {  
 String getData();  
}  
  
class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}  
**OUTPUT:**

