**TEST DRIVEN DEVELOPMENT AND LOGGING FRAMEWORK**

**JUNIT BASIC TESTING**

**EXERCISE – 1 : SETTING UP JUNIT:**

**Steps to include the JUNIT library in Eclipse:**

1. After creating a new project we need to right click on the project.
2. Now select Build paths -> Configure build path.
3. After go to the Libraries section and add a new library named “Junit5”
4. After that click on “Finish” and then “Apply and Close”.

**PalindromeChecker.java:**

package Junit;

public class PalindromeChecker {

public boolean isCleanPalindrome(String input) {

if (input == null) return false;

String converted = input.replaceAll("[^a-zA-Z]", "").toLowerCase();

String reversed = new StringBuilder(converted).reverse().toString();

return converted.equals(reversed);

}

}

**PalindromeTest.java:**

package Junit;

import org.junit.Test;

import static org.junit.Assert.\*;

public class PalindromeTest {

@Test

public void testIsPalindrome() {

PalindromeChecker checker = new PalindromeChecker();

*assertTrue*(checker.isCleanPalindrome("apa"));

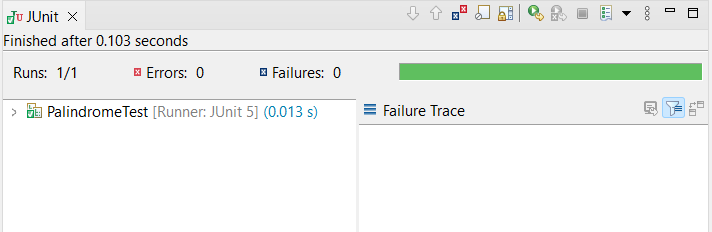
*assertFalse*(checker.isCleanPalindrome("hello"));

}

}

**OUTPUT:**

1. Right click on the ‘PalindromeTest.java’ file.
2. Now select Run As -> JUnit test



**EXERCISE – 3: ASSERTIONS IN JUNIT:**

**Assertions.java:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class Assertions {

@Test

public void testAssertions() {

*assertEquals*(5, 2 + 3);

*assertTrue*(5 > 3);

*assertFalse*(5 < 3);

*assertNull*(null);

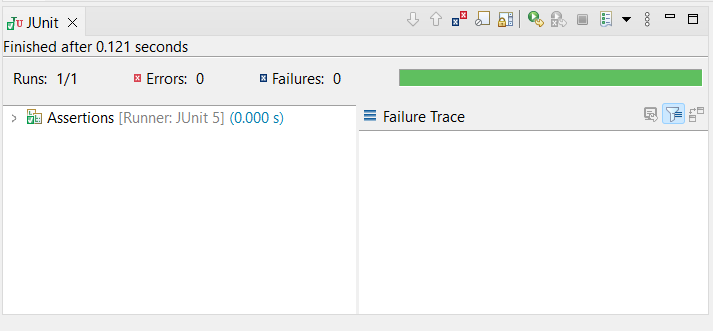
*assertNotNull*(new Object());

}

}

**OUTPUT:**

1. Right click on the ‘Assertions.java’ file.
2. Now select Run As -> JUnit test



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

**UsernameValidator.java:**

package Junit;

public class UsernameValidator {

public boolean isValid(String username) {

if (username == null) return false;

if (username.length() < 5) return false;

if (!Character.*isLetter*(username.charAt(0))) return false;

return true;

}

}

**UsernameValidatorTest.java:**

package Junit;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class UsernameValidatorTest {

private UsernameValidator validator;

@Before

public void setUp() {

validator = new UsernameValidator();

System.*out*.println("Setup complete.");

}

@After

public void tearDown() {

validator = null;

System.*out*.println("Teardown complete.");

}

@Test

public void testValidUsername() {

boolean result = validator.isValid("John123");

*assertTrue*(result); //Assert

}

@Test

public void testTooShortUsername() {

boolean result = validator.isValid("ab1");

*assertFalse*(result);

}

@Test

public void testStartsWithDigit() {

boolean result = validator.isValid("1User");

*assertFalse*(result);

}

@Test

public void testNullInput() {

boolean result = validator.isValid(null);

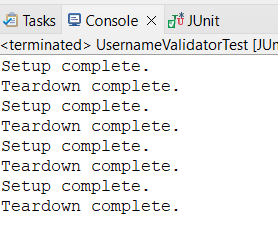
*assertFalse*(result);

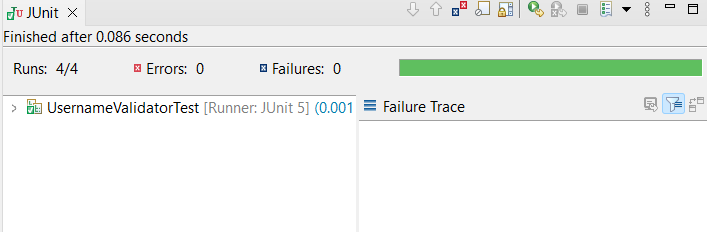
}

}

**OUTPUT:**

1. Right click on the ‘UsernameValidatorTest.java’ file.
2. Now select Run As -> JUnit test





**MOCKITO EXERCISES**

**EXERCISE – 1: MOCKING AND STUBBING**

1.Create a new Maven project in Eclipse.

2.Now add JUnit 5 and Mockito dependencies to pom.xml

**pom.xml**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

3.Create java classes

**ExternalApi.java**

package com.example.mock\_demo;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example.mock\_demo;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example.mock\_demo;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testFetchDataWithMock() {

ExternalApi mockApi = *mock*(ExternalApi.class);

*when*(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

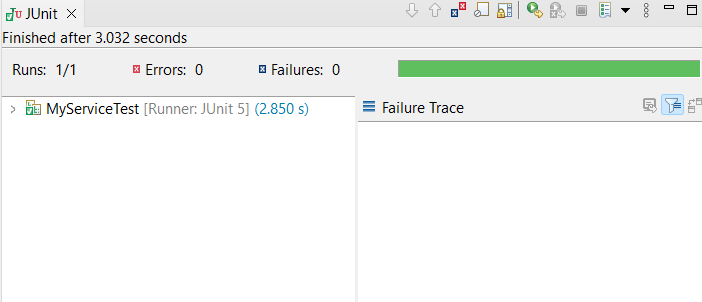
*assertEquals*("Mock Data", result);

}

}

**OUTPUT:**

1. Right click on the ‘MyServiceTest.java’ file.
2. Now select Run As -> JUnit test



**EXERCISE – 2: VERIFYING INTERACTIONS**

1.Create a new Maven project in Eclipse.

2.Now add JUnit 5 and Mockito dependencies to pom.xml

**pom.xml**

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

3.Create java classes

**ExternalApi.java**

package com.example.mock\_demo;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example.mock\_demo;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example.mock\_demo;

import static org.mockito.Mockito.\*;

import org.mockito.Mockito;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.*mock*(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

*verify*(mockApi).getData();

}

}

**OUTPUT:**

1. Right click on the ‘MyServiceTest.java’ file.
2. Now select Run As -> JUnit test

