**Week-2**

**JUnit Testing**

**Exercise 1: Setting Up Junit**

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

**Step1** : Create a new Java project in your IDE(Eclipse)

**Step2** : Add JUnit dependency to your project.

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.junitsetup</groupId>

<artifactId>practice</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Step 3:**

**GreetingService.java**

package practice;

public class GreetingService {

public String greet(String name) {

if (name == null || name.isEmpty()) {

return "Hello, Guest!";

}

return "Hello, " + name + "!";

}

}

**GreetingServiceTest.java**

package practice;

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

import org.junit.Test;

public class GreetingServiceTest {

@Test

public void testGreetWithName() {

GreetingService service = new GreetingService();

String result = service.greet("Jyothika");

*assertEquals*("Hello, Jyothika!", result);

}

@Test

public void testGreetWithEmptyString() {

GreetingService service = new GreetingService();

String result = service.greet("");

*assertEquals*("Hello, Guest!", result);

}

@Test

public void testGreetWithNull() {

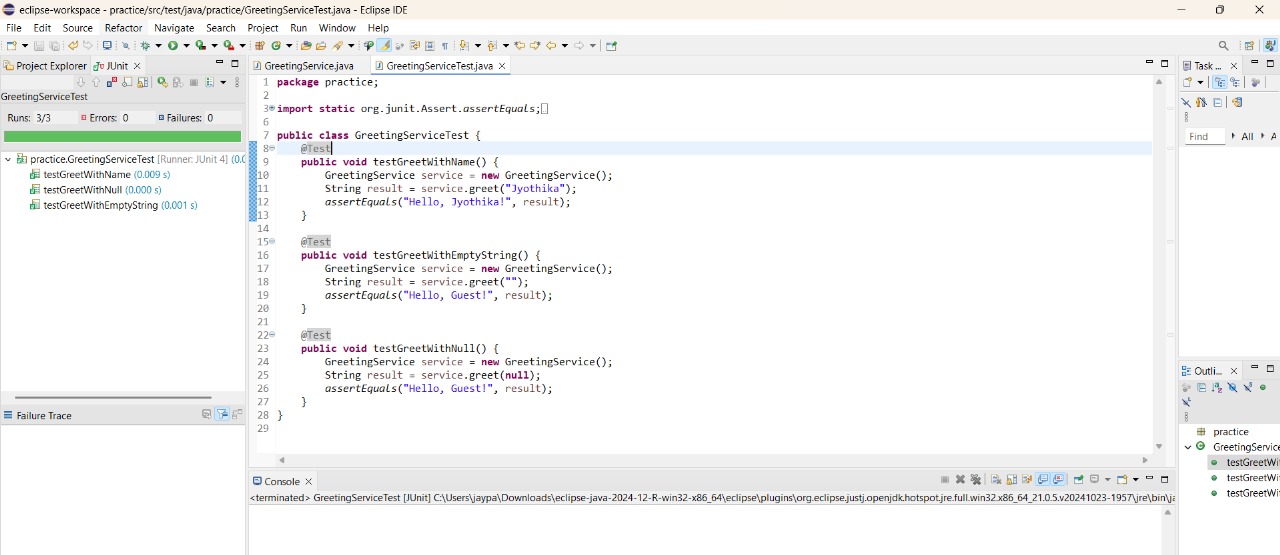
GreetingService service = new GreetingService();

String result = service.greet(null);

*assertEquals*("Hello, Guest!", result);

}

}

****

**Exercise 3: Assertions in Junit**

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

**AssertionsTest.Java**

**package** practice;

**import** org.junit.Test;

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

**public** **class** AssertionsTest {

@Test

**public** **void** testVariousAssertions() {

// Assert equals

*assertEquals*("Sum should be 10", 10, 7 + 3);

// Assert true

*assertTrue*("This should be true", "OpenAI".startsWith("O"));

// Assert false

*assertFalse*("Should be false", 100 < 50);

// Assert null

String name = **null**;

*assertNull*("Name should be null", name);

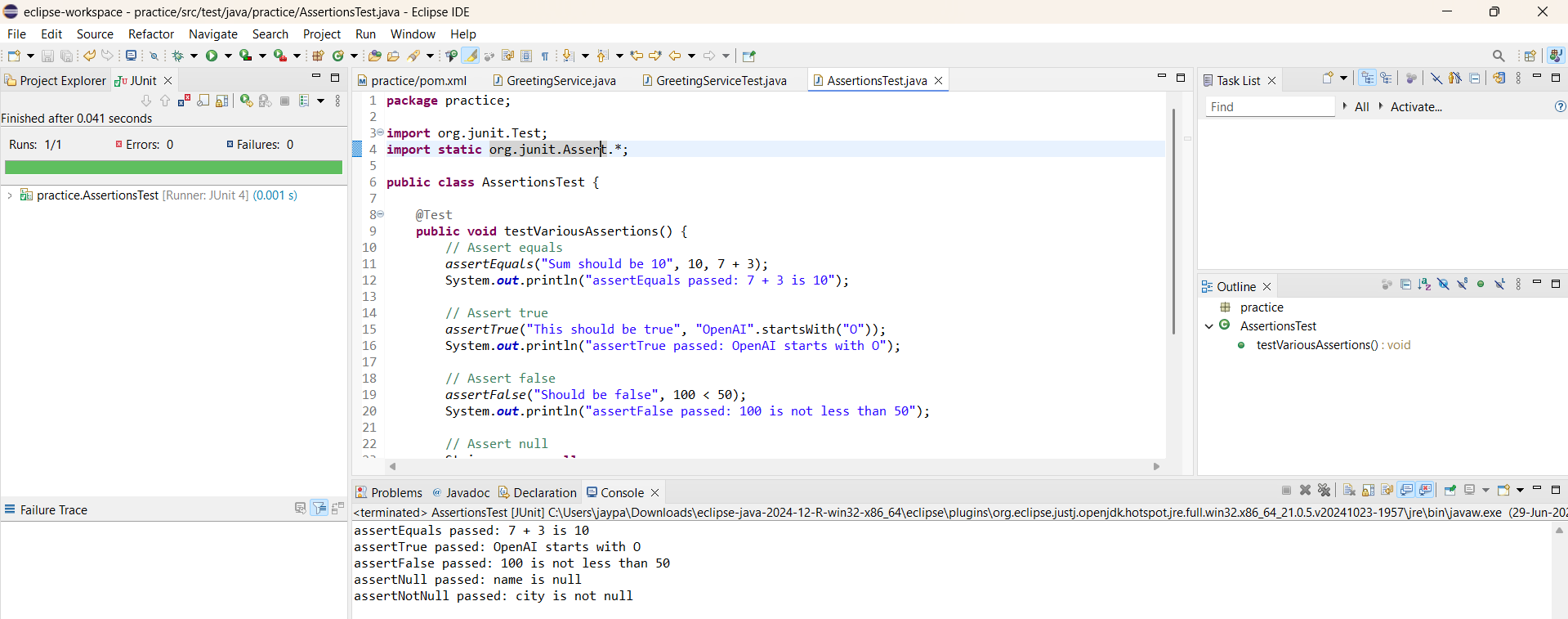
// Assert not null

String city = "Hyderabad";

*assertNotNull*("City should not be null", city);

}

}



**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.**

**ServiceTest.java**

package practice;

public class ServiceTest {

public String greet(String name) {

return (name == null || name.isEmpty()) ? "Hello, Guest" : "Hello, " + name;

}

}

**ExampleServiceTest.java**

package practice;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class ExampleServiceTest {

private ServiceTest service;

@Before

public void setUp() {

service = new ServiceTest();

}

@After

public void tearDown() {

service = null;

}

@Test

public void testGreetWithName() {

String input = "Jyo";

String result = service.greet(input);

*assertEquals*("Hello, Jyo", result);

System.out.println("testGreetWithEmpty passed with result: " + result);

}

@Test

public void testGreetWithEmpty() {

String input = "";

String result = service.greet(input);

*assertEquals*("Hello, Guest", result);

System.out.println("testGreetWithEmpty passed with result: " + result);

}

}

