Write a program to demonstrate standard tests ?

Operation.java [com.service] :-

**package** com.service;

**public** **class** Operation {

**public** **int** add(**int** a,**int** b) {

**int** add = a + b;

**return** add;

}

}

operationTest.java [com.test]:-

package com.test;

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

import org.junit.jupiter.api.Test;

import com.service.Operation;

class OperationTest {

@Test

void testAdd() {

// fail("Not yet implemented");

Operation op = new Operation();

int result = op.add(100, 200);

assertEquals(300, result);

}

}

Write a program to demonstrate assertions. ?

Operation.java [com.service] :-

**package** com.service;

**public** **class** Operation {

**public** **int** add(**int** a,**int** b) {

**int** add = a + b;

**return** add;

}

}

operationTest.java [com.test]:-

package com.test;

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

import org.junit.jupiter.api.Test;

import com.service.Operation;

class OperationTest {

@Test

void testAdd() {

// fail("Not yet implemented");

Operation op = new Operation();

int result = op.add(100, 200);

assertEquals(300, result);

}

}

Write a program to demonstrate conditional test executions ?

Employee.java [com.service] :-

**package** com.service;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** **float** salary;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

**public** Employee(**int** id, String name, **float** salary) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

}

**public** Employee() {

**super**();

// **TODO** Auto-generated constructor stub

}

}

EmployeeService.java [com.service] :-

package com.service;

import java.util.ArrayList;

import java.util.List;

public class EmployeeService {

public String checkUser(String name, String pass) {

if(name.equals("Harsh") && pass.equals("123")) {

return "success";

}else {

return "failure";

}

}

}

EmployeeServiceTest.java [com.test] :-

package com.test;

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

import java.util.List;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Test;

import com.service.Employee;

import com.service.EmployeeService;

class EmployeeServiceTest {

@Test

@DisplayName("Check user Authentication")

void testCheckUser() {

//fail("Not yet implemented");

EmployeeService es = new EmployeeService();

String result = es.checkUser("Harsh", "123");

assertEquals("success", result);

String result2 = es.checkUser("Shubham", "1234");

assertEquals("failure", result2);

}

}

Write a program to demonstrate nested and repeated tests ?

Operation.java [com.service] :-

**package com.test;**

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

**import org.junit.jupiter.api.Test;**

**import com.service.Operation;**

**class OperationTest {**

**@Test**

**void testAdd() {**

**// fail("Not yet implemented");**

**Operation op = new Operation();**

**int result = op.add(100, 200);**

**assertEquals(300, result);**

**}**

**@Test**

**void testMinus() {**

**//fail("Not yet implemented");**

**Operation op = new Operation();**

**int result = op.minus(200, 100);**

**assertEquals(100, result);**

**}**

**@Test**

**void testMul() {**

**//fail("Not yet implemented");**

**Operation op = new Operation();**

**int result = op.mul(100, 200);**

**assertEquals(20000, result);**

**}**

**@Test**

**void testDiv() {**

**//fail("Not yet implemented");**

**Operation op = new Operation();**

**int result = op.div(200, 100);**

**assertEquals(2, result);**

**}**

**}**

operationTest.java [com.test]:-

**package** com.service;

**public** **class** Operation {

**public** **int** add(**int** a,**int** b) {

**int** add = a + b;

**return** add;

}

**public** **int** minus(**int** a,**int** b) {

**int** add = a - b;

**return** add;

}

**public** **int** mul(**int** a,**int** b) {

**int** add = a \* b;

**return** add;

}

**public** **int** div(**int** a,**int** b) {

**int** add = a / b;

**return** add;

}

}

Write a program to demonstrate dynamic tests?

Employee.java [com.service] :-

**package** com.service;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** **float** salary;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

**public** Employee(**int** id, String name, **float** salary) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

}

**public** Employee() {

**super**();

// **TODO** Auto-generated constructor stub

}

}

EmployeeService.java [com.service] :-

package com.service;

import java.util.ArrayList;

import java.util.List;

public class EmployeeService {

public String checkUser(String name, String pass) {

if(name.equals("Harsh") && pass.equals("123")) {

return "success";

}else {

return "failure";

}

}

}

EmployeeServiceTest.java [com.test] :-

package com.test;

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

import java.util.List;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Test;

import com.service.Employee;

import com.service.EmployeeService;

class EmployeeServiceTest {

@Test

@DisplayName("Check user Authentication")

void testCheckUser() {

//fail("Not yet implemented");

EmployeeService es = new EmployeeService();

String result = es.checkUser("Harsh", "123");

assertEquals("success", result);

String result2 = es.checkUser("Shubham", "1234");

assertEquals("failure", result2);

}

}

Write a program to demonstrate a dependency injection ?

Pom.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>

DpendencyInjection :-

**package** com.ecommerce.tests;

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

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

**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.jupiter.api.condition.DisabledIf;

**import** org.junit.jupiter.api.condition.EnabledOnOs;

**import** org.junit.jupiter.api.condition.OS;

**import** org.junit.platform.runner.JUnitPlatform;

**import** org.junit.runner.RunWith;

**@DisplayName("JUnit 5 Dependency Injection Example")**

**@RunWith(JUnitPlatform.class)**

**public** **class** DependencyInjection {

DependencyInjection(TestInfo testInfo) {

assertEquals("JUnit 5 Dependency Injection Example", testInfo.getDisplayName());

}

**@BeforeEach**

void init(TestInfo testInfo) {

**String** displayName = testInfo.getDisplayName();

assertTrue(displayName.equals("TEST 1") || displayName.equals("test2()"));

}

**@Test**

**@DisplayName("TEST 1")**

**@Tag("my-tag")**

void test1(TestInfo testInfo) {

assertEquals("TEST 1", testInfo.getDisplayName());

assertTrue(testInfo.getTags().contains("my-tag"));

}

**@Test**

void test2() {

}

}

You are given a project to demonstrate RESTful with Spring Boot?

demoTest.java [com.main] :-

package com.main;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

@SpringBootApplication(scanBasePackages = "com")

@EntityScan(basePackages = "com.bean")

public class demoTest {

public static void main(String[] args) {

SpringApplication.run(demoTest.class, args);

System.out.println("Server Up");

}

}

sampleController.java [com.controller] :-

package com.controller;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

public class sampleController {

@RequestMapping(value="say",method = RequestMethod.GET)

public String sayHello() {

return "Welcome to Spring boot with Rest API";

}

}