**Assignments on Junit:**

**1.**

**package** junitpractice;

**import** java.lang.reflect.Array;

**import** java.util.Arrays;

**//create a class of MiniMaxFinder**

**public** **class** MiniMaxFinder {

**public** **int**[] arr(**int** [] numbers) {

Arrays.*sort*(numbers);

**int** [] arr1= {numbers[0],numbers[numbers.length-1]};

**return** arr1;

}

}

**//create a Junit test case of MiniMaxFind**

package junitpractice;

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

import org.junit.jupiter.api.Test;

class MiniMaxFinderTest {

@Test

void testMinMaxFind() {

MiniMaxFinder mnf = new MiniMaxFinder();

int expedted[] = new int[] {3,56};

assertArrayEquals(expedted, mnf.arr(new int[] {56,34,7,3,54,3,34,34,53}));

}

@Test

void testMinMaxFind1() {

MiniMaxFinder mnf1 = new MiniMaxFinder();

int expedted1[] = new int[] {0,99};

assertArrayEquals(expedted1, mnf1.arr(new int[] {30,1,10,25,56,99,87,45,0}));

}

@Test

void testMinMaxFind2() {

MiniMaxFinder mnf2 = new MiniMaxFinder();

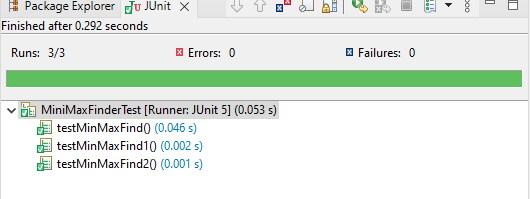
int expedted2[] = new int[] {101,999};

assertArrayEquals(expedted2, mnf2.arr(new int[] {999,101,205,665,777,854,465}));

}

}

**Output:**

****

**3.**

**//create a BankAccount Class**

**package** junitpractice;

**import** javax.naming.InsufficientResourcesException;

**public** **class** BankAccount {

**int** a;

**int** BankAccountBalance = 20000;

**public** String Withdraw(**int** a) **throws** InsufficientFundException {

**if**(a< BankAccountBalance) {

**return** ("wait for a momment");

}

**else**

{

**throw** **new** InsufficientFundException("Insufficient Funds");

}

}

}

**//create a BankAcoountTest Class**

package junitpractice;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertThrows;

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

import javax.naming.InsufficientResourcesException;

import org.junit.jupiter.api.Test;

class BankAccountTest {

@Test

void testwithdraw() {

BankAccount a = new BankAccount();

assertThrows(InsufficientFundException.class, ()-> a.Withdraw(20000),"An Exception may be occurred" );

}

@Test

void testwithdraw1() throws InsufficientFundException {

BankAccount a1 = new BankAccount();

String expected = "wait for a momment";

assertEquals(expected, a1.Withdraw(19999));

}

}

**4.**

**package** junitpractice;

**//create a javaUnit project**

**public** **class** MyJuintProject {

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

**return** a+b;

}

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

**return** a-b;

}

}

**//create a java unit test class**

**package** junitpractice;

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

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

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

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

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

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

**class** MyClassCase {

MyJuintProject junit;

@BeforeAll

**static** **void** beforeAllInit() {

System.***out***.println("this nedds to run before all");

}

@AfterAll

**static** **void** afterAll() {

System.***out***.println("We are at the end of the Programming");

}

@BeforeEach

**void** init() {

junit = **new** MyJuintProject();

}

@AfterEach

**void** afterEach() {

System.***out***.println("The code run successfully");

}

@Test

**void** addtest() {

**int** result = junit.add(10, 20);

*assertEquals*(30, result);

}

@Test

**void** subtracttest() {

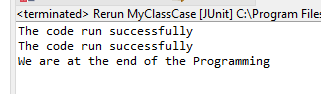
**int** result = junit.subtraction(10, 9);

*assertEquals*(1, result);

}

}

**Output:**

****