Name: Lutika Kolhe

# JUnit Assignment

1. Write a class called MinMaxFinder. Define a method in it called find Min Max() which accepts an int array and retums new array of size 2, wherein the 0th index will have the min value of the array and 1st index will have maxvalue of the array. Perform Junit testing of the method findMin Max with as many test cases you can think of (min 3 test cases)

E.g.

Min MaxFinder.findMinMaxt new int[] (56, 34, 7,3, 54, 3, 34, 34, 53) ); should return a new array with min and max values (3, 56) at 0th and 1st index respectively.

Ans- Here I created 2 class first is MinMaxFinder.java and another is JUnitTesting.java

Class1 : MinMaxFinder.java

**package** JUnit;

**package** org.junit.app;

**public** **class** MinMaxFinder {

**public** **int**[] minmaxFinder(**int**[] a) {

**int**[] arr=**new** **int**[2];

**int** min = a[0],max=0;

**for**(**int** i=0; i<a.length; i++ ) {

**if**(a[i]<min) {

min = a[i];

}

}

arr[0]=min;

**for**(**int** i=0; i<a.length; i++ ) {

**if**(a[i]>max) {

max = a[i];

}

}

arr[1]=max;

**int** r[] = {min,max};

System.***out***.println("Minimum:" + r[0] +" "+ "Maximum:" + r[1]);

**return** r;

}

}

Class2 : JUnitTesting.java

**package** JUnit;

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

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

**class** JUnitTesting {

**int**[] expected1= {4,21};

**int**[] expected2= {5,98};

**int**[] expected3= {1,6};

@Test

**void** test() {

MinMaxFinder mm=**new** MinMaxFinder();

**int**[] a= {20,10,4,15,21};

**int**[] min=mm.minmaxFinder(a);

*assertArrayEquals*(expected1,min); //WithArray

}

@Test

**void** test1() {

MinMaxFinder mm=**new** MinMaxFinder();

**int**[] a= {45,7,48,5,98};

**int**[] min=mm.minmaxFinder(a);

*assertArrayEquals*(expected2,min);

}

@Test

**void** test2() {

MinMaxFinder mm=**new** MinMaxFinder();

**int**[] a= {2,5,4,1,6};

**int**[] min=mm.minmaxFinder(a);

*assertArrayEquals*(expected3,min);

}

}

**OutPut:**

Minimum:4 Maximum:21

Minimum:5 Maximum:98

Minimum:1 Maximum:6

1. Modify the above method to return a single object representing min and max value of the pass array. Define new sets of Junit Test cases of this modified method.

Ans- Here I created 2 class first is Modify.java and another is TestCase.java

Class 1 : Modify.java

**package** JUnit;

**import** java.util.Arrays;

**public** **class** Modify {

**public** Object[] minmaxfinder(**int**[] a) {

**int**[] arr=**new** **int**[2];

**int** min = a[0],max=0;

**for**(**int** i=0; i<a.length; i++ ) {

**if**(a[i]<min) {

min = a[i];

}

}

arr[0]=min;

**for**(**int** i=0; i<a.length; i++ ) {

**if**(a[i]>max) {

max = a[i];

}

}

arr[1]=max;

Object r[] = {min,max};

System.***out***.println("Minimum, Maximum: "+Arrays.*toString*(r));

**return** r;

}

}

Class 2 : TestCase.java

**package** JUnit;

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

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

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

**class** TestCase {

Modify objminmax;

Object[] expected1= {4,21};

Object[] expected2= {5,98};

Object[] expected3= {1,6};

@Test

**void** test() {

Modify mm=**new** Modify();

**int**[] a= {20+1,10-6,8,15,19};

Object[] min=mm.minmaxfinder(a);

*assertArrayEquals*(expected1,min);

}

@Test

**void** test1() {

Modify mm=**new** Modify();

**int**[] a= {45,7,48+2,5,98};

Object[] min=mm.minmaxfinder(a);

*assertArrayEquals*(expected2,min);

}

@Test

**void** test2() {

Modify mm=**new** Modify();

**int**[] a= {2,5,4,1,6};

Object[] min=mm.minmaxfinder(a);

*assertArrayEquals*(expected3,min);

}

}

**OUTPUT:**

Minimum, Maximum: [4, 21]

Minimum, Maximum: [5, 98]

Minimum, Maximum: [1, 6]

1. Write a Bank Account class with method withdraw which accepts amount to be withdrawn from the account (amount to be deducted from the balance of the account). In case there are insufficient funds a InsufficientFunds Expcetion should be raised. After defining the method perform Junit testing to check whether the InsufficientFunds Exception is raised when you try to withdraw amount that is over and above the account balance.

bankAccount withdraw(20,000); should raise the Insufficient FundsException if the balance in the account is less than 20,000.

Ans- Here I created 3 class which is Banking.java, InsuicientFundsException.java and BankTest.java

Class 1 : Banking.java

**package** JUnit;

**public** **class** Banking {

**int** balance;

**public** **int** getBalance() {

**return** balance;

}

**public** **void** setBalance(**int** balance) {

**this**.balance = balance;

}

**public** **void** withdraw(**int** withdrawamt) **throws** InsufficientFundsException {

**if**(withdrawamt >=balance) {

**throw** **new** InsufficientFundsException();

}

**else** {

System.***out***.println("New balance amount is:"+(balance - withdrawamt));

}

}

}

Class 2 : InsufficientFundsException.java

**package** JUnit;

**public** **class** InsufficientFundsException **extends** Exception {

**public** **void** InsufficientFundsException() {

}

}

Class 3 : BankTest.java

**package** JUnit;

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

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

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

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

**class** BankTest {

Banking b;

@BeforeEach

**void** initialise() {

b=**new** Banking();

}

@Test

@DisplayName("checking InsufficientFundsException")

**public** **void** testWithdraw() {

b.setBalance(2000);

*assertThrows*(InsufficientFundsException.**class**,()->b.withdraw(25000),"your balance is less than withdraw amount are equals to withdraw amount");

}

@Test

@DisplayName("checking the balance ")

**public** **void** testWithdrawWithoutException() {

b.setBalance(2500);

**int** expected=2000;

**int** actual=2000;

*assertEquals*(expected,actual,"invalid balance");

}

}

**OUTPUT:**

New balance amount is: 1500

1. Write a Junit Testing to show the use of Lifecycle hooks annotation such as @BeforeAll, @BeforeEach @AfterEach and @AfterAll

Ans- Class1 : LifeCycle.java

**package** JUnit;

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

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

**import** org.junit.After;

**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;

**public** **class** LifeCycle

{

**public** **void** LifecycleTest() {

System.***out***.println("Constructor");

}

@BeforeAll

**static** **void** beforeTheEntireTestFixture() {

System.***out***.println("Before the entire test fixture");

}

@AfterAll

**static** **void** afterTheEntireTestFixture() {

System.***out***.println("After the entire test fixture");

}

@BeforeEach

**void** beforeEachTest() {

System.***out***.println("Before each test");

}

@AfterEach

**void** afterEachTest() {

System.***out***.println("After each test");

}

@Test

**void** firstTest() {

System.***out***.println("First test");

}

@Test

**void** secondTest() {

System.***out***.println("Second test");

}

}

**OUTPUT:**

Before the entire test fixture

Before each test

First test

After each test

Before each test

Second test

After each test

After the entire test fixture