**J-Unit**

**1)pom.xml**

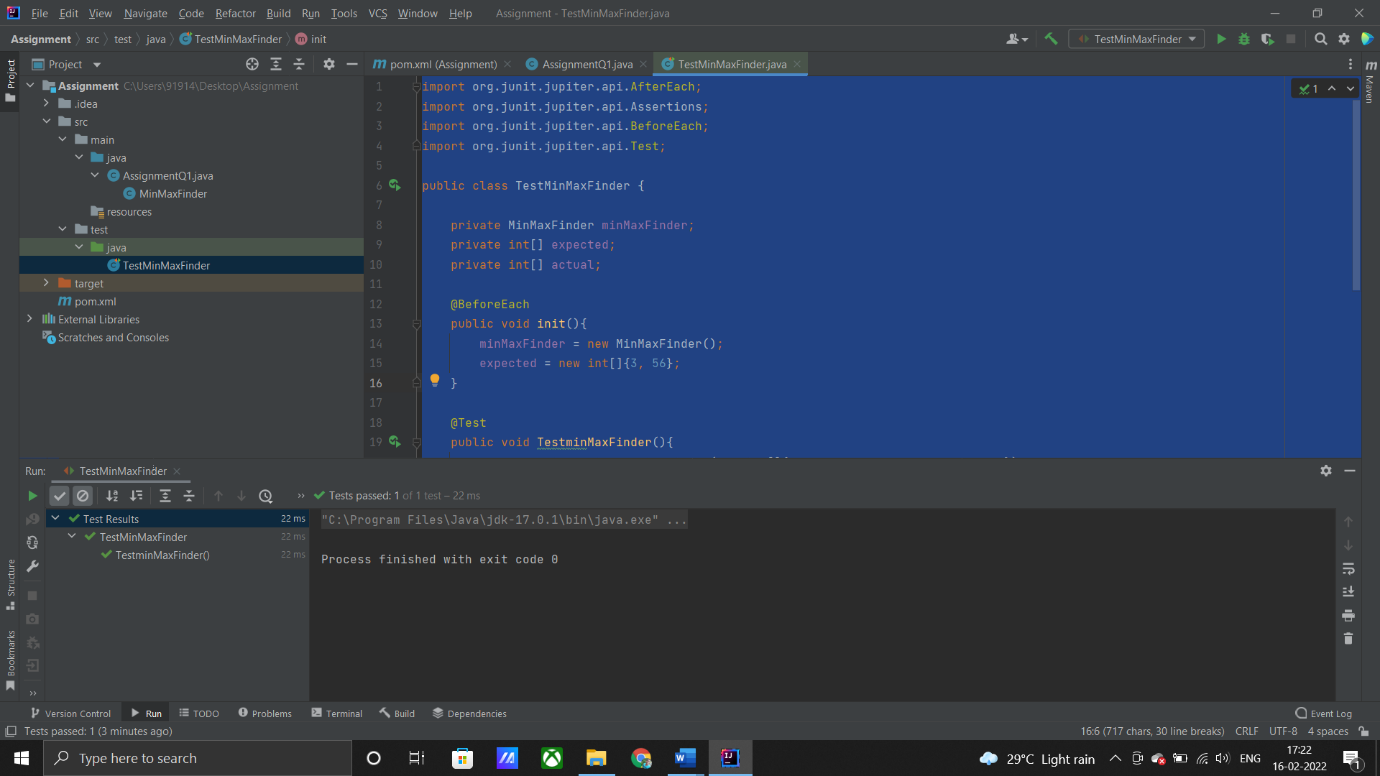
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>Assignment</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
  
 <dependencies>  
  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
  
</project>

**AssignmentQ1.java**

import java.util.Arrays;  
  
class MinMaxFinder{  
  
 public int[] findMinMax(int[] arr){  
 int[] ans = new int[2];  
  
 Arrays.*sort*(arr);  
  
 ans[0] = arr[0];  
 ans[1] = arr[arr.length-1];  
 return ans;  
 }  
}

**TestMinMaxFinder.java**

import org.junit.jupiter.api.AfterEach;  
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
public class TestMinMaxFinder {  
  
 private MinMaxFinder minMaxFinder;  
 private int[] expected;  
 private int[] actual;  
  
 @BeforeEach  
 public void init(){  
 minMaxFinder = new MinMaxFinder();  
 expected = new int[]{3, 56};  
 }  
  
 @Test  
 public void TestminMaxFinder(){  
 actual = minMaxFinder.findMinMax(new int[]{56, 34, 7,3, 54, 3, 34, 34, 53});  
 Assertions.*assertArrayEquals*(expected,actual);  
 }  
  
 @AfterEach  
 public void Clean(){  
 minMaxFinder = null;  
 expected = actual = null;  
 }  
}

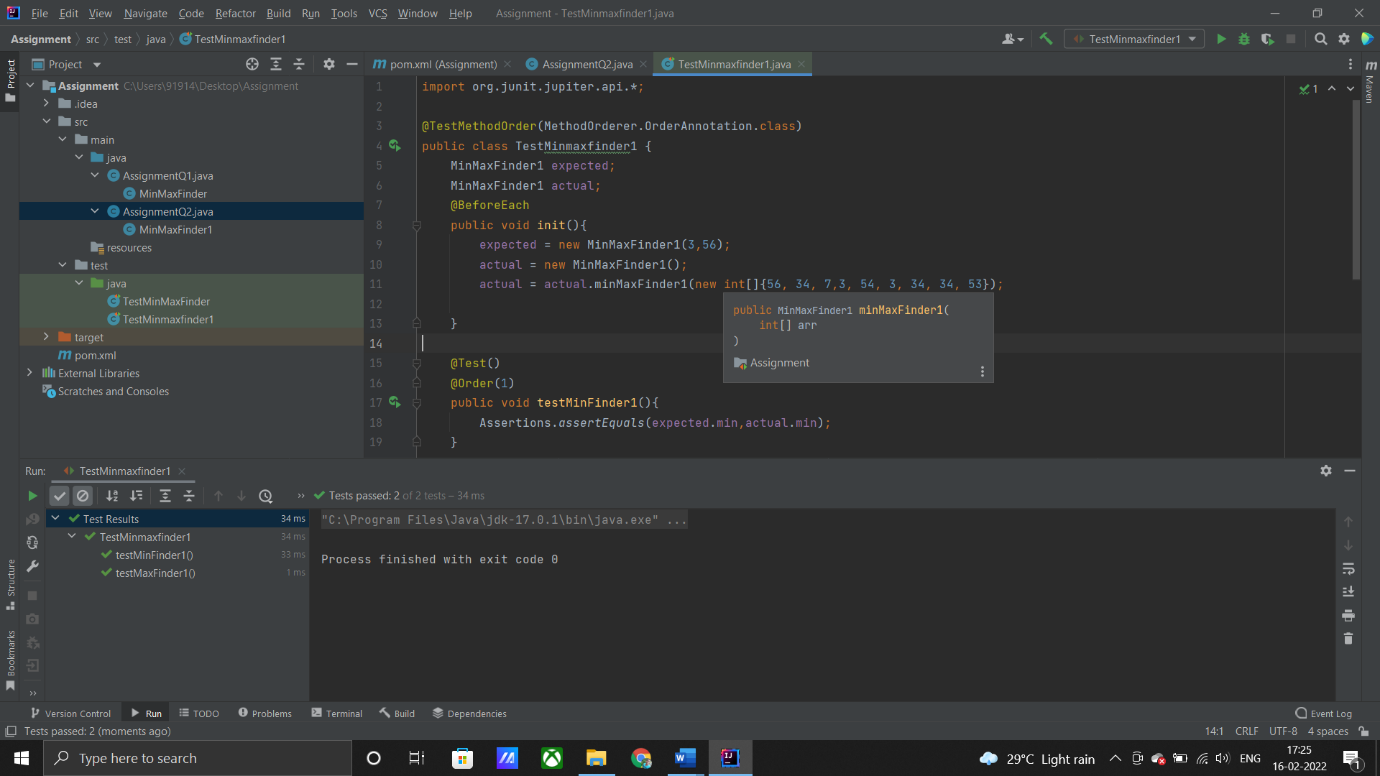
****

**2)AssignmentQ2**

import java.util.Arrays;  
class MinMaxFinder1{  
 int max;  
 int min;  
 MinMaxFinder1(){}  
  
 MinMaxFinder1(int min, int max){  
 this.min = min;  
 this.max = max;  
  
 }  
 public MinMaxFinder1 minMaxFinder1(int[] arr){  
 int min;  
 int max;  
 Arrays.*sort*(arr);  
 min = arr[0];  
 max = arr[arr.length-1];  
  
 return new MinMaxFinder1(min,max);  
 }  
}

TestMinmaxfinder1.java

import org.junit.jupiter.api.\*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
public class TestMinmaxfinder1 {  
 MinMaxFinder1 expected;  
 MinMaxFinder1 actual;  
 @BeforeEach  
 public void init(){  
 expected = new MinMaxFinder1(3,56);  
 actual = new MinMaxFinder1();  
 actual = actual.minMaxFinder1(new int[]{56, 34, 7,3, 54, 3, 34, 34, 53});  
  
 }  
  
 @Test()  
 @Order(1)  
 public void testMinFinder1(){  
 Assertions.*assertEquals*(expected.min,actual.min);  
 }  
  
 @Test  
 @Order(2)  
 public void testMaxFinder1(){  
 Assertions.*assertEquals*(expected.max,actual.max);  
 }  
  
 @AfterEach  
 public void Clean(){  
 actual = expected = null;  
 }  
}



3)

AssignmentQ3.java

class BankAccount{  
 private final long amount;  
  
 BankAccount(long amount){  
 this.amount = amount;  
 }  
 public long withDraw(long withdrawalAmount) throws InsufficientFundsExpcetion{  
 if(amount-withdrawalAmount<=20000){  
 throw new InsufficientFundsExpcetion();  
 }  
 return amount-withdrawalAmount;  
 }  
  
 private class InsufficientFundsExpcetion extends Exception {  
 }  
}

BankAccountTest.java

import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
public class TestBankAccount {  
 private BankAccount bankAccount;  
 private long withDrawAmount;  
 @BeforeEach  
 public void init(){  
 //Initialize the Bank Amount;  
 bankAccount = new BankAccount(50000);  
 }  
  
 @Test  
 public void testBankAccount(){  
 withDrawAmount = 40000;  
 Assertions.assertThrows(InsufficientFundsExpcetion.class,()->{  
 System.*out*.println(bankAccount.withDraw(withDrawAmount));  
 },"YOU HAVE INSUFFICIENT BALANCE!!!");  
  
 }  
  
 @Test  
 public void testBankAccount1(){  
 withDrawAmount = 20000;  
 Assertions.*assertDoesNotThrow*(()->{  
 System.*out*.println("BALANCE AFTER WITHDRAWAL : "+bankAccount.withDraw(withDrawAmount));  
 });  
 }  
  
 private class InsufficientFundsExpcetion {  
 }  
}

4)

AssignmentQ4.java

import org.junit.jupiter.api.\*;  
  
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)  
class Assignment4 {  
  
 @BeforeAll  
 static void meth1(){  
 System.*out*.println("BeforeAll :- This will execute only once and before all the test methods: \n");  
 }  
  
 @BeforeEach  
 public void meth2(){  
 System.*out*.println("BeforeEach :- This will always execute before all the test methods: ");  
 }  
  
 @Test  
 @Order(1)  
 public void meth5(){  
 System.*out*.println("HELLO FROM TEST METHOD-1");  
 }  
 @Test  
 @Order(2)  
 public void meth6(){  
 System.*out*.println("HELLO FROM TEST METHOD-2");  
 }  
 @Test  
 @Order(3)  
 public void meth7(){  
 System.*out*.println("HELLO FROM TEST METHOD-3");  
 }  
 @AfterEach  
 public void meth3(){  
 System.*out*.println("AfterEach :- This will always execute after all the test methods: \n");  
 }  
  
 @AfterAll  
 static void meth4(){  
 System.*out*.println("AfterAll :- This will execute only once and after all the test methods: \n");  
 }  
}

output

