**ASSIGNMENT 4**

**1. COURSE REGISTERATION**

**Student Class:**

**package** registrationSystem;

**class** Student

{

**protected** String studName;

**protected** String studID;

**public** Student(String name, String id)

{

**this**.studName=name;

**this**.studID=id;

}

**public** String getName()

{

**return** **this**.studName;

}

**public** String getID()

{

**return** **this**.studID;}}

**Course Class:**

**package** registrationSystem;

**class** Course

{

**public** String courseName;

**public** **int** numberOfStudent=0;

**public** Student[] regStudents=**new** Student[10];

**public** Course(String title)

{

**this**.courseName=title;

}

**public** Student[] getStudents()

{

**return** **this**.regStudents;

}

**public** **boolean** isFull()

{

**if**(**this**.numberOfStudent==10)

{

**return** **true**;

}

**return** **false**;

}

**public** String getTitle()

{

**return** **this**.courseName;

}

**public** **int** getNumberOfStudents()

{

**return** **this**.numberOfStudent;

}

**public** **boolean** regStudent(Student student)

{ **int** j;String name1,name2,id1,id2;

id1=student.studID;

name1=student.studName;

**if**(!isFull())

{

**for**(j=0;j<**this**.numberOfStudent;j++)

{

id2=regStudents[j].studID;

name2=regStudents[j].studName;

**if**(name1==name2 && id1==id2)

{

System.***out***.println("Student is already registered");

**return** **false**;

}

}

**this**.regStudents[j]=student;

**this**.numberOfStudent=**this**.numberOfStudent+1;

System.***out***.println("Number of students now"+**this**.numberOfStudent);

}

**else**

{

System.***out***.println("Course is full");

**return** **false**;

}

**return** **true**;

}}

**TDD:**

**package** registrationSystem;

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

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

**import** org.junit.Before;

**import** registrationSystem.Course;

**import** registrationSystem.Student;

**import** org.junit.Test;

**public** **class** registrationTest

{

Student stud, stud1,stud2,stud3,stud4,stud5,stud6,stud7,stud8,stud9,stud10,stud11;

Course course;

@Before

**public** **void** beforeTest()

{

stud= **new** Student("Name1","01");

stud2= **new** Student("Name2","02");

**this**.course= **new** Course("Course1");

**this**.course.regStudent(stud);

**this**.course.regStudent(stud2);

System.***out***.println(course.getNumberOfStudents());

//course.regStudent(stud2);

}

@Test

**public** **void** regstudent()

{ **boolean** flag=**false**;

//happy case : number of student is less than 10 and student is registered

Student stud3= **new** Student("Name3","03");

course.regStudent(stud3);

Student students[]=course.getStudents();

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

{

**if**(students[i]==stud3)

flag=**true**;

}

**assert**(flag);

}

@Test

**public** **void** regagain()

{

stud4=**new** Student("Name1","01");

*assertFalse*(course.regStudent(stud4));

}

@Test

**public** **void** regNewCourse()

{ **boolean** flag=**false**;

Course course2=**new** Course("secondCourse");

stud5= **new** Student("Name1","01");

course2.regStudent(stud5);

Student studentList[]=course2.getStudents();

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

{

**if**(studentList[i]==stud5)

flag=**true**;

}

**assert**(flag);

}

@Test

**public** **void** regStudentMoreThanLimt()

{

// trying to register when already 10 are there

Course c=**new** Course("registerMoreThan10");

stud1=**new** Student("Name1","01");

stud2=**new** Student("Name2","02");

stud3= **new** Student("Name3", "03");

stud4= **new** Student("Name4", "04");

stud5= **new** Student("Name5", "05");

stud6= **new** Student("Name6", "06");

stud7= **new** Student("Name7", "07");

stud8= **new** Student("Name8", "08");

stud9= **new** Student("Name9", "09");

stud10= **new** Student("Name10", "10");

c.regStudent(stud1);

c.regStudent(stud2);

c.regStudent(stud3);

c.regStudent(stud4);

c.regStudent(stud5);

c.regStudent(stud6);

c.regStudent(stud7);

c.regStudent(stud8);

c.regStudent(stud9);

c.regStudent(stud10);

**int** n1=c.getNumberOfStudents();

stud11=**new** Student("Name11","11");

**boolean** flag3=c.regStudent(stud11);

**int** n2=c.getNumberOfStudents();

**if**(n1==n2)

{

*assertFalse*(flag3);

}

**else**

{

*fail*("Registration exceeded limit test failed");

}

}

}

**2. PURCHASE A BOOK**

**Book Class:**

**package** Assignment3;

**public** **class** Book

{

**float** price;

String title;

**int** quantity;

Book(String title,**float** price,**int** quan)

{

**this**.price=price;

**this**.title=title;

**this**.quantity=quan;

}

**public** **float** getPrice()

{

**return** **this**.price;

}

**public** String getTitle()

{

**return** **this**.title;

}

**public** **int** getQuanBook()

{

**return** **this**.quantity;

}

**public** **boolean** getBook(String title)

{

**if**(**this**.title==title)

{

**if**(**this**.quantity>0)

{

System.***out***.println(**this**.title+"Book is present");

**return** **true**;

}

}

**return** **false**;

}

**public** **void** setQuantity(**int** quan)

{

**this**.quantity=quan;

}}

**Customer Class:**

**package** Assignment3;

**import** Assignment3.Book;

**public** **class** Customer

{

**float** price=0;

**int** buyQuan=0;

**public** **float** purchaseBook(Book book,**int** quantity)

{

**if**(book.getBook(book.title))

{

**if**(book.getQuanBook()>=quantity)

{

buyQuan=quantity;

}

**else** **if**(book.getQuanBook()<quantity)

{

buyQuan=book.getQuanBook();

}

}

book.setQuantity(book.getQuanBook()-buyQuan);

**this**.price=book.getPrice()\*buyQuan+**this**.price;

**return** **this**.price;

}

**public** **static** **void** main(String args[])

{

Book b1=**new** Book("abc",10.00f,5);

Book b2=**new** Book("cde",20.00f,10);

Customer c=**new** Customer();

c.purchaseBook(b1,1);

Customer c2=**new** Customer ();

c2.purchaseBook(b1, 2);

c2.purchaseBook(b2, 5);

System.***out***.println("Price for first customer"+c.price);

System.***out***.println("Price for second customer"+c2.price);}}

**3. GAS LEVEL**

**package** Assignment3;

**public** **class** GasTask {

**private** **double** *amt* = 0;

**public** **void** addGas(**double** x)

{

*amt* =*amt*+ x;

System.***out***.println (*amt*);

}

**public** **static** **void** useGas(**double** y)

{

*amt*= amt- y;

**if** (*amt* < 0)

{

*amt* = 0;

}

System.***out***.println (*amt*);

}

**public** **static** **double** getGasLevel()

{

System.***out***.println (*amt*);

**return** *amt*;

}

**public** **static** **void** main(String[] args)

{

*addGas*(125.5563);

*useGas*(0.56);

*getGasLevel*();

}

}

**TDD**

package Assignment3;

import static org.junit.Assert.\*;

import org.junit.Test;

public class GasTaskTest

{

double amount;

@Test

public void test1()

{

GasTask gt = new GasTask();

gt.addGas(1.00);

gt.useGas(0.5);

amount=gt.getGasLevel();

assertEquals(amount,0.5,0);

}

@Test

public void test2()

{

GasTask gt = new GasTask();

gt.addGas(1);

gt.useGas(0);

amount= gt.getGasLevel();

assertEquals(amount, 1,0);

}

@Test

public void test3()

{

GasTask gt = new GasTask();

gt.addGas(0);

gt.useGas(1);

amount= gt.getGasLevel();

assertEquals(amount, 0,0);

}

@Test

public void test4()

{

GasTask gt = new GasTask();

gt.addGas(0.5);

gt.useGas(1.0);

amount= gt.getGasLevel();

assertEquals(amount, 0.5,0);

}}

**4. CAR CLASS**

**package** Assignment3;

**public** **class** Car {

**private** String make ="Ford 2000", model="Ford";

**private** **int** year = 1996;

**public** Car(String make, String model, **int** year)

{

**this**.make = make;

**this**.model = model;

**this**.year = year;

}

**public** **void** setMake(String make)

{

**this**.make = make;

}

**public** String getMake()

{

**return** **this**.make;

}

**public** **void** setModel(String model)

{

**this**.model = model;

}

**public** String getModel()

{

**return** **this**.model;

}

**public** **void** setYear(**int** year)

{

**this**.year = year;

}

**public** **int** getYear()

{

**return** **this**.year;

}

**public** String toString()

{

**return** "This car " + getMake() + "model " +getModel()+ getYear();

}

}

**5. DRIVER CLASS**

**package** Assignment3;

**public** **class** Driver {

**public** **boolean** drive(Car c, GasTask gt)

{

**if**((c.getMake()!=**null**) &&(c.getModel()!=**null**) && (c.getYear()!=0))

{

**if**(gt.getGasLevel()>0)

**return** **true**;

**else**

{

System.***out***.println("no fuel");

**return** **false**;

}

}

**else**

**return** **false**;

}

**public** **static** **void** main(String[] args)

{

Driver d= **new** Driver();

Car c= **new** Car("Honda","Honda Civic",2017);

GasTask gt=**new** GasTask();

gt.addGas(1.00);

**if**(d.drive(c,gt))

{

System.***out***.println("Successful Drive");

}

**else**

{

System.***out***.println("No car or No fuel");

}

}

}

**Extra Credit:**

1. Given an array containing n distinct numbers taken from 0, 1, 2 …, n, find the one that is missing from the array. For example, given nums = {0, 1, 3}, return 2.

**Answer:**

**package** Assignment3;

**public** **class** missingNumber{

**public** **int** missingNumbe(**int**[] nums)

{

**int** sum=0;

**for**(**int** i=1; i<nums.length; i++){

sum+=nums[i];

}

**int** n=nums.length;

**return** n\*(n+1)/2-sum;

}

**public** **static** **void** main(String[] args) {

**int**[] a = {0,1,2,4};

missingNumber m = **new** missingNumber();

System.***out***.println("The missing number in the array is " +m.missingNumbe(a));

}

}

1. Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements. For example, given nums = {0, 1, 0, 3, 12}, after calling your function, nums should be {1, 3, 12, 0, 0}, You must do this in-place without making a copy of the array.

**Answer:**

**package** Assignment3;

**public** **class** moveZeroes

{

**public** **static** **void** movezero(**int** arr[], **int** n)

{

**int** count = 0;

**for** (**int** i = 0; i < n; i++)

**if** (arr[i] != 0)

arr[count++] = arr[i];

**while** (count < n)

arr[count++] = 0;

}

**public** **static** **void** main (String[] args)

{

**int** arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};

**int** n = arr.length;

*movezero*(arr, n);

System.***out***.println("Aftre moving zeros: ");

**for** (**int** i=0; i<n; i++)

System.***out***.print(arr[i]+" ");

}

}

1. Given a non-empty array of integers, return the third maximum distinct number in this array. If it does not exist, return the maximum number. For example, given{3,2,1}, return 1; given{1,2}, return 2; given{2,2,3,1}, return 1.

**Answer:**

**package** Assignment3;

**public** **class** thirdMax

{

**public** **int** thirdmax(**int**[] nums)

{

**int** max, mid, small, count;

max = mid = small = Integer.***MIN\_VALUE***;

count = 0;

**for**( **int** x: nums) {

**if**( x == max || x == mid ) {

**continue**;

}

**if** (x > max) {

small = mid;

mid = max;

max = x;

count++;

} **else** **if**( x > mid) {

small = mid;

mid = x;

count++;

} **else** **if** ( x >= small) {

small = x;

count++;

}

}

**if**( count >= 3) {

**return** small;

} **else** {

**return** max;

}

}

**public** **static** **void** main(String[] args)

{

**int**[] a = {12,0,5,10};

thirdMax m = **new** thirdMax();

System.***out***.println("The third highest number in the array is "+ m.thirdmax(a));

}

}