/\* Finish before Feb 12, 2017 23:59 PST. Try to write JUnit Test for every problem. When writing method, try to use TDD to write. Please notice, there’s no standard answer for question 1-5. If you have limited time, you can skip extra credits. \*/

1. Design a simple registration system that allows Student to register in a course using 2 classes: class Student & class Course. Implement the scenarios in class Test’s main method.

Each student has a name and an id variables. Each object of class Student is initialized using values of name and id passed to constructor. Class Student has accessor methods for its instance variables

Each Course has a name, and a variable numberOfStudent representing the number of registered students. A course can have a maximum number of 10 students registered in it. Class Course store the registered students in students which is an array of type Student. When a student register in a course, he is added to the array. Each object of class Course is initialized using the title. Class Course has the following methods: method getStudents(): return the array of registered students; method boolean isFull(): return true if the course is full, accessor method for the title and numberOfStudent field, method registerStudent (Student student): if the course is not full, register a student in course.

**class** Student{

**int** id;

String name;

Student(**int** id,String name){

**this**.id=id;

**this**.name=name;

}

**int** getId(){

**return** id;

}

String getName(){

**return** name;

}

}

**class** Course{

String title;

**int** numberOfStudents;

Student[] students=**new** Student[10];

Student[] getStudents(){

**return** students;

}

**boolean** isFull(){

**for**(**int** i=0;i<10;i++){

**if**(students[i]==**null**){

**return** **false**;

}

}

**return** **true**;

}

String getTitle(){

**return** title;

}

**int** getNumberOfStudents(){

**for**(**int** i=0;i<10;i++){

**if**(students[i]!=**null**){

numberOfStudents++;

}

}

**return** numberOfStudents;

}

**void** setTitle(String title){

**this**.title=title;

}

**void** setStudents(Student[] students){

**this**.students=students;

}

**boolean** registerStudent (Student student){

**if**(!isFull()){

students[ getNumberOfStudents()]=student;

**return** **true**;

}

Else **false**;

}

}

**class** Test{

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

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

c.setTitle("Java");

c.registerStudent(**new** Student(101,"Mounika"));

c.registerStudent(**new** Student(102,"Anushka"));

c.registerStudent(**new** Student(103,"Prabhas"));

System.out.println("Is it Full? :"+c.isFull());

System.out.println(c.getStudents()[2].getName());

}

}

Junit Test:

import org.junit.Assert;

import org.junit.Test;

public class TestClass{

@Test

public void TestStudent(){

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

c.setTitle("Java");

c.registerStudent(**new** Student(101,"Mounika"));

c.registerStudent(**new** Student(102,"Anushka"));

c.registerStudent(**new** Student(103,"Prabhas"));

System.out.println("Is it Full? :"+c.isFull());

Assert.**assert**(c.isFull()==**false**);

}

}

2.Design and implement class Book and class Customer. You need to create necessary attributes for these two classes and methods if needed. Every customer can purchase books, and every book has price and title. Besides, you need to write the main method to test some scenarios like this: a customer is buying several books, and it prints out the total price for this purchase on the screen.

**public** **class** Book {

**int** id;

String title;

**float** price;

**public** Book(**int** id, String title, **float** price) {

**super**();

**this**.id = id;

**this**.title = title;

**this**.price = price;

}

**public** **class** Customer {

**int** id;

String name;

String mailId;

String mobileNo;

**public** Customer() {

}

**public** Customer(**int** id, String name, String mailId, String mobileNo) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.mailId = mailId;

**this**.mobileNo = mobileNo;

}

Book[] cart;

**void** purchaseBooks(Book[] books){

cart=books;

}

**public** **float** totalPrice(){

**float** total=0;

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

**if**(cart[i]!=**null**){

total+=cart[i].price;

}

}

**return** total;

}

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

System.out.println(totalPrice);

}

}

Junit Test:

package assessment4;

import org.junit.Assert;

import org.junit.Test;

public class TestClass {

@Test

public void TestBooks(){

Book[] books=new Book[10];

books[0]=new Book(101,"My style of living",300.89f);

books[1]=new Book(102,"My style of living",300.89f);

Customer c=new Customer(301,"Rakesh","r@mail.com","77534876784");

c.purchaseBooks(books);

float t=c.totalPrice();

Assert.assertTrue(t==601.78f);

}

}

3.Write a class named GasTank containing:

An instance variable named amount of type double, initialized to 0.

A method named addGas that accepts a parameter of type double . The value of the amount instance variable is increased by the value of the parameter.

A method named useGas that accepts a parameter of type double . The value of the amount instance variable is decreased by the value of the parameter.

A method named getGasLevel that accepts no parameters. getGasLevel returns the value of the amount instance variable.

**public** **class** GasTank {

**double** amount;

**void** addGas(**double** amount){

**this**.amount+=amount;

}

**void** useGas(**double** amount){

**this**.amount-=amount;

}

**double** getGasLevel(){

**return** amount;

}

}

**class** MainClass{

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

GasTank g=**new** GasTank();

g.addGas(40.5);

g.useGas(32.8);

System.***out***.println(g.getGasLevel());

}

}

Junit Test:

import org.junit.Assert;

import org.junit.Test;

public class TestClass {

@Test

public void TestGas(){

double amount=0;

GasTank g=new GasTank();

g.addGas(40.0);

g.useGas(32.8);

g.getGasLevel();

Assert.assertTrue(g.getGasLevel()==7.2f);

}

}

4.Design and implement a class called Car. You need to create necessary attributes for this class, and method if needed. Define the Car constructor to initialize these values (in that order). Include getter and setter methods for all instance data.

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

**int** modelNum;

String modelName;

String color;

GasTank gasTank;

**public** Car(**int** modelNum, String modelName, String color) {

**super**();

**this**.modelNum = modelNum;

**this**.modelName = modelName;

**this**.color = color;

}

**public** **int** getModelNum() {

**return** modelNum;

}

**public** **void** setModelNum(**int** modelNum) {

**this**.modelNum = modelNum;

}

**public** String getModelName() {

**return** modelName;

}

**public** **void** setModelName(String modelName) {

**this**.modelName = modelName;

}

**public** String getColor() {

**return** color;

}

**public** **void** setColor(String color) {

**this**.color = color;

}

**boolean** drive(){

**if**(gasTank.getGasLevel()>0){

System.***out***.println("Car has Started");

**return** **true**;

}**else**{

System.***out***.println("Car has very low gas level please fill");

**return** **false**;

}

}

}

Junit Test:

import org.junit.Assert;

import org.junit.Test;

public class TestClass {

@Test

**public** **void** TestCar() {

Car c=**new** Car(1102,"XUV","Brown");

Assert.assertTrue(c.color=="Brown");

}

5.Combine with problem 3 and 4, define a class named Driver that contains methods like drive and addGas so that the driver can drive the car.

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

**void** drive(Car c){

c.drive();

}

**void** addGas(Car c,**double** amount){

c.gasTank.addGas(amount);

}

}

Junit Test:

import org.junit.Assert;

import org.junit.Test;

public class TestClass {

@Test

**public** **void** TestDriver() {

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

c.gasTank=**new** GasTank();

d.addGas(c, 40.5);

Assert.assertTrue(Driver.drive(c)==**true**);

c.gasTank.useGas(32.5);

Assert.assertTrue(c.gasTank.getGasLevel()==8.0);

}

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.

public static missingNumber(int[] nums) {

**public** **static** **void** missingNumber(**int**[] arr){

**int** num=arr[0];

**for**(**int** i=0;i<arr.length;i++,num++){

**if**(arr[i]!=num){

System.out.println(num);

}

}

}

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.

public static void moveZeroes(int[] nums) {

**int** count=0,temp;

**for**(**int** i=0;i<arr.length-2;i++){

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

count++;

**if**(arr[i+1]!=0){

arr[i]=arr[i+1];

arr[i+1]=0;

count--;

}

**else**{

temp=arr[i];

arr[i]=arr[i+2];

arr[i+2]=temp;

}

}

}

**for**(**int** j=0;j<arr.length;j++){

System.out.println(arr[j]);

}

}

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.

public int thirdMax(int[] nums) {

**int** i, j,n=1,max;

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

**for**(j=0; j< n ; j++)

{

**if**(arr[i] == arr[j])

**break**;

}

**if** (j==n )

arr[n++] = arr[i];

}

max=arr[0];

**if**(n<3 && (arr[1]>max)){

max=arr[1];

}

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

**int** a;

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

{

**for** (j = i + 1; j < n; ++j)

{

**if** (arr[i] > arr[j])

{

a = arr[i];

arr[i] = arr[j];

arr[j] = a;

}

}

}

max=arr[n-3];

}

**return** max;

}