W2 LIVE CODDING 1

Write the java program that creates 3 objects of students and prints the name of student with highest total. Implement the code as follows:

Define class Student with following members:

• String name, double[] marks (array length is 3) as instance variables which rep- resents the name and marks of student

• Constructor to initialize instance variables

• Accessor method getName() that returns the name of student

• Method findTotal() that returns the total marks of the student

Define class Test that has the following members:

• Method getMax() which takes Student [] as parameter and returns the name of the student with highest total.

• main() method, that creates 3 instances of Student, stores them in an array and invokes the method getMax() by passing that Student []

```
import java.util.*;
class Student {
   private String name;
   private double marks [];
   public Student(String name, double[] marks) {
      this.name = name;
      this.marks = marks;
   }
   public String getName() {
      return (this.name);
   }
   public double findTotal() {
      double total = 0.0;
      for (double i : this.marks) {
         total = total + i;
```

```
        }
        return (total);
    }
}
public class Test {
    public static String getMax(Student[] student) {
        double max_marks = 0.0;
        String max_student = "";
        for (Student i : student) {
            double total_marks = i.findTotal();
            if (total_marks > max_marks) {
                max_student = i.getName();
                max_marks = total_marks;
            }
        }
        return max_student;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Student[] arr = new Student[3];
        for(int i = 0; i < 3; i++){
            String name = sc.next();
            double[] m = {sc.nextDouble(), sc.nextDouble(), sc.nextDouble()};
            arr[i] = new Student(name, m);
        }
        System.out.println(getMax(arr));
    }
}
```

W2 LIVE CODING 2

Employee el does a set of projects. Employee e2 also does all the projects did by e1 except the first project, in place of which e2 does another project. Write a program that defines two classes Employee and Test. Define copy constructor to create e2 from e1 in such a way that changing the values of instance variables of either e2 or e1 should not affect the other one. The code takes name of e2 and new project done by e2 as input. Complete the program as specified below.

• Class Employee that has the following members.

Private instance variables String name and String[] projects to store name and projects respectively

Define required constructor(s)

- Accessor methods getName() and getProject() to get name of employee and project at specific index.

- Mutator methods setName() and setProject() to set name of employee and project at specific index. • Class Test that has the method main which does the following.

Two objects of Employee e1 and e2 are created. e2 is created using e1

- name of e2 and second item bought by c2 are updated by taking the input

name of e1, e2 and first project done by e1 and e2 are printed

```
import java.util.*;
class Employee{
   String name;
   String[] projects;
   public Employee(String n, String[] proj){
      name = n;
      projects = proj;
   }
   public Employee(Employee e){
```

```
        this.name = e.name;
        int l = e.projects.length;
        this.projects = new String[l];
        for(int i = 0; i < l; i++){
            this.projects[i] = e.projects[i];
        }
    }
    public void setName(String n) {
        name = n;
    }
    public void setProject(int index, String proj) {
        projects[index] = proj;
    }
    public String getName() {
        return name;
    }
    public String getProject(int indx) {
        return projects[indx];
    }
}
public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] proj =  {"PJ1", "PJ2", "PJ3"};
        Employee e1 = new Employee("Surya", proj);
        Employee e2 = new Employee(e1);
        e2.setName(sc.next());
        e2.setProject(0, sc.next());
        System.out.println(e1.getName() + ": " + e1.getProject(0));
```

```
        System.out.println(e2.getName() + ": " + e2.getProject(0));
    }
}
```

W3 LIVE CODING 3

Write the Java code as instructed.

Class Faculty has the following members.

- String name, double salary as private instance variables

- Constructor to initialize the instance variables

- Method public double bonus (float percent) that returns the bonus com- puted as (percent/100.0)*salary

You should define method getDetails() to display name and salary of an

Faculty

- You should overload method getDetails() as getDetails (float percent) to display the name, salary and bonus of an Faculty

• Class Hod extends class Faculty and has the following members.

- String personalAssistant as private instance variable - Constructor to initialize the instance variables of Hod that includes the instance variables of Faculty

- You should override method public double bonus (float percent) that re-

turns the bonus of a Hod as 50 percent less bonus than a regular faculty You should override method getDetails() to display the name, salary and personalAssistant of Hod.

- You should override method get Details (float percent) to display the name, salary, personalAssistant and bonus of a Hod

Class InheritanceTest has the main method and the following functionality.

- It creates objects of Faculty and Hod types, and also accepts the required input values.

- Invokes methods getDetails() and getDetails (float percent) on Faculty and Hod objects.

```
import java.util.Scanner;

class Faculty{
```

```
    private String name;
    private double salary;
    public Faculty(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    public double bonus(float percent){
        return (percent/100.0)*salary;
    }
     public String getDetails() {
         return name + ", " + salary;
     }
     public String getDetails(float percent ) {
         return getDetails()+ ", bonus = "+bonus(percent);
     }
}
class Hod extends Faculty{
     private String personalAssistant;
     public Hod(String name, double salary, String pa) {
         super(name, salary);
         this.personalAssistant = pa;
     }
     public double bonus(float percent){
         return 0.5*super.bonus(percent);
     }
     public String getDetails() {
         return super.getDetails()+", "+ personalAssistant;
     }
     public String getDetails(float percent ) {
```

```
        return getDetails()+", "+bonus(percent);
    }
}
public class InheritanceTest{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Faculty obj1 = new Faculty(sc.next(), sc.nextDouble());
        Faculty obj2 = new Hod(sc.next(), sc.nextDouble(), sc.next());
        System.out.println(obj1.getDetails());
        System.out.println(obj1.getDetails(10));
        System.out.println(obj2.getDetails());
        System.out.println(obj2.getDetails(10));
    }
}
```

W4 LIVE CODING 1

Write a Java program as instructed.

Abstract class UPIPayment has two abstract methods: abstract void payment () and abstract void rewards()

Class PhonePay extends class UPIPayment and has the following members.

- private int amount as private instance variable - Constructor to initialize the instance variable

- You should override method public void payment() such that the overridden method displays the message "Phone pay: Payment success:" and invokes method rewards ().

- You should override method public void rewards () as follows.

* If amount < 500, then display the message "Sorry no rewards".

* Else, if amount >= 500, then display the message "10 off on next mobile rc".

• Class Paytm extends class UP IPayment and has the following members.

- private int amount as private instance variable - Constructor to initialize the instance variable

- You should override method public void payment() such that the overridden method displays the message "Paytm : Payment success:" and invokes method rewards().

You should override method public void rewards() as follows.

* If amount < 500, then display the message "Sorry no rewards". * Else, if amount >= 500, then display message "10 off on next DTH rc".

Class UPIUser has the following method.

- Method public void transfer AndGet Rewards (UPIPayment obj) that invokes method payment () using obj.

Class AbstractTest has the main method, and has the following functionality.

- Creates an object of UPIUser

- Takes two amounts a1 for PhonePay and a2 for Paytm

- Invokes method transfer AndGetRewards () first by passing an object of PhonePay with amount a1 as parameter, and then by passing an object of Paytm with amount a2 as parameter.

```
import java.util.*;
abstract class UPIPayment{
   abstract void payment();
   abstract void rewards();
}
class PhonePay extends UPIPayment{
   private int amount;
   public PhonePay(int amount) {
      this.amount = amount;
   }
   public void payment() {
      System.out.println("Phone pay:Payment success:");
      rewards();
   }
   public void rewards() {
      if(amount < 500)
         System.out.println("Sorry no rewards");
      else if(amount >= 500)
```

```
            System.out.println("10 off on next mobile rc");
    }
}
class Paytm extends UPIPayment{
    private int amount;
    public Paytm(int amount) {
        this.amount = amount;
    }
    public void payment() {
        System.out.println("Paytm:Payment success:");
        rewards();
    }
    public void rewards() {
        if(amount < 500)
            System.out.println("Sorry no rewards");
        else if(amount >= 500)
            System.out.println("10 off on next DTH rc");
    }
}
class UPIUser{
    public void transferAndGetRewards(UPIPayment obj) {
        obj.payment();
    }
}
public class AbstractTest {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a1 = sc.nextInt();
        int a2 = sc.nextInt();
```

```
        UPIUser u = new UPIUser();
        u.transferAndGetRewards(new PhonePay(a1));
        u.transferAndGetRewards(new Paytm(a2));
    }
}
```

W4 LIVE CODING 2

Write Java code as instructed.

• Define an interface Appraisable that has the following members:

- Default method default void appraisal (Teacher t) that increments the salary of an Employee by (stuPassPer/100) +5000.

- Abstract method public abstract void checkAndUpdateSalary()

• Define an interface Special Appraisable that extends Appraisable and has the

following members: - Default method default void spAppraisal (Teacher t) that increments the salary of an Employee by (stuPassPer/100)*10000.

• Class Teacher that implements the interface Special Appraisable and has the following members:

- String name, double salary and private double stuPassPer as private in-

stance variables

- Constructor to initialize the instance variables

Mutator method to update salary

-Accessor method to access salary

- Accessor method to access stuPassPer

- Override method toString() that returns name, salary and stuPassPer of

the Teacher as a single concatenated string (each separated by a single space) - Overriding method public void checkAndUpdateSalary() that has the fol-

lowing functionality.

* If stuPassPer>=60 and stuPassPer<75 then invoke the appraisal () method from Appraisable interface

* Else, if stuPassPer>=75 and stuPassPer<=100 then invoke thespAppraisal() method from Special Appraisable interface

Class Interface Test that has the following members:

You should define method public static void printUpdatedTeachList (Teacher [] tList) that has the following functionality

* Iterate over array tList and invoke method checkAndUpdateSalary() on each Teacher object.

* Then, iterate over tList and display each Teacher object.

- main method has the following functionality * Creates and initializes an array tArr of three Teacher objects

* Invokes method printUpdatedTeachList (Teacher [] tList) to print the updated details of each Teacher after the appraisal is applied

```
import java.util.*;
interface Appraisable{
   default void appraisal(Teacher t){
      t.setSalary(t.getSalary()+(t.getstuPassPer()/100)*5000);
   }
   public abstract void checkAndUpdateSalary();
}
interface SpecialAppraisable extends Appraisable{
   default void spAppraisal(Teacher t){
      t.setSalary(t.getSalary()+(t.getstuPassPer()/100)*10000);
   }
}
class Teacher implements SpecialAppraisable{
   private String name;
   private double salary;
   private double stuPassPer;
   public Teacher(String name, double salary, double stuPassPer) {
      this.name = name;
      this.salary = salary;
```

```
        this.stuPassPer = stuPassPer;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public double getstuPassPer() {
        return stuPassPer;
    }
    public String toString() {
        return name + ", " + salary + ", " + stuPassPer;
    }
    public void checkAndUpdateSalary() {
        if(stuPassPer >= 60 && stuPassPer < 75)
            appraisal(this);
        else if(stuPassPer >= 75 && stuPassPer <= 100)
            spAppraisal(this);
     }
}
public class InterfaceTest {
    public static void printUpdatedTeachList(Teacher[] tList) {
        for (int i = 0; i < tList.length; i++)
            tList[i].checkAndUpdateSalary();
        for (int i = 0; i < tList.length; i++)
            System.out.println(tList[i]);
    }
public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);

Teacher tArr[] = new Teacher[3];

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

tArr[i] = new Teacher(sc.next(), sc.nextDouble(), sc.nextDouble());

InterfaceTest.printUpdatedTeachList(tArr);

}

}
```

W4 LIVE CODING 3

Write a Java program that accepts list of student objects and updates the regular fees, based on the number of backlogs that each student has.

• An interface Generatable that has the following member:

- Abstract method abstract void feeGenerate (Students)

Class Student that has the following members:

- String name, double fee and int backlogs as private instance variable.

- Constructor to initialize the name and backlogs.

- Mutator method to update the fee

-Accessor method to access the backlogs

- Override method toString() that returns name, fee and backlogs of the Student as a single concatenated string (each separated by comma(,))

Class ExamBranch that has the following members: - A private inner class Regular that implements the interface Generatable and overrides method public void feeGenerate (Student s) that, in turn, initializes the regular student fee as 1500.

- A private inner class Supple that implements the interface Generatable and overrides method public void feeGenerate (Students) that, in turn, has the following functionality.

*If the student backlogs 1, then update the student fee to 2000. ==

*Else, if the student backlogs 2, then update the student fee to 2500. ==

* Else, if the student backlogs >= 3, then update the student fee to 3500.

You should define method getRegularFee () that returns an object of Regular

class.

- You should define method getSuppleFee () that returns an object of Supple

class.

Class PrivateClassTest has the following members:

- You should define method public static Student [] getStudentsFee (Student sList[]) that does the following:

* Iterates over array sList such that in each iteration, invoke method feeGenerate (Students) on each Student object from Regular class, if student backlogs 0. Else, invoke method feeGenerate (Student s) == on each student object from Supple class.

main method has the following functionality * Creates and initializes an array sArr of three Student objects

* Invokes method getStudentsFee (sArr) to get the updated details of each Student after the fee is updated

* Prints the updated list

```
import java.util.Scanner;

interface Generatable{
   abstract void feeGenerate(Student s);
}
class Student {
   private String name;
   private double fee;
   private int backlogs;
   public Student(String name, int backlogs) {
      this.name = name;
      this.backlogs = backlogs;
   }
   public void setFee(double fee) {
      this.fee = fee;
   }
```

```
    public int getBacklogs() {
        return backlogs;
    }
    public String toString() {
        return name + ", " + fee + ", " + backlogs;
    }
}
class ExamBranch{
    private class Regular implements Generatable{
        public void feeGenerate(Student s) {
            s.setFee(1500.00);
        }
    }
    private class Supple implements Generatable{
        public void feeGenerate(Student s) {
            if(s.getBacklogs() == 1)
                s.setFee(1500 + 500);
            else if(s.getBacklogs() == 2)
                s.setFee(1500 + 1000);
            else if(s.getBacklogs() >=3 )
                s.setFee(1500 + 2000);
    }
}
    public Generatable getRegularFee() {
        return new Regular();
    }
    public Generatable getSuppleFee() {
        return new Supple();
    }
```

```
}
public class PrivateClassTest {
   public static Student[] getStudentsFee(Student sList[]){
      ExamBranch obj;
      for (int i = 0; i < sList.length; i++) {
         if(sList[i].getBacklogs()==0) {
            obj=new ExamBranch();
            obj.getRegularFee().feeGenerate(sList[i]);
         }
         else {
            obj=new ExamBranch();
            obj.getSuppleFee().feeGenerate(sList[i]);
         }
       }
       return sList;
    }
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      Student[] sArr = new Student[3];
      for (int i = 0; i < sArr.length; i++) {
         sArr[i] = new Student(sc.next(), sc.nextInt());
      }
      sArr = PrivateClassTest.getStudentsFee(sArr);
      for (int i = 0; i < sArr.length; i++) {
         System.out.println(sArr[i]);
      }
   }
}
```

W5 LIVE CODING 1

Write a program that, given two integers, two doubles as x and y coordinates of two points p1(x1,y1) and p2(x2,y2) on a two-dimensional plane as input, finds the mid- point p3(x3, y3) of the line segment formed by p1 and p2 using the formula: Generic class Point has the following members.

x1 + x2 2 x3 = and y3 = 2

• Instance variables x and y

• A constructor to initialize x and y

• A method mid (Point p) to return the mid-point of the line segment joining the current point to p

• A method that overrides the method toString() in the Object class to format the output

Class Test has the main method.

• The main method accepts the two input points. The first line of input will be two integers of point p1. The second line of input will be two doubles of point p2. It then invokes the method mid of one of the objects.

```
import java.util.*;
class Point<T extends Number>{
   T x, y;
   public Point(T x, T y) {
      this.x = x;
      this.y = y;
   }
   public Point<Double> mid(Point<?> p){
      Point<Double> pt = new Point<Double>(0.0, 0.0);
      pt.x = (this.x.doubleValue() + p.x.doubleValue()) / 2;
      pt.y = (this.y.doubleValue() + p.y.doubleValue()) / 2;
      return pt;
   }
```

```
    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
public class Test{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Point<Integer> p1 = new Point<Integer>(sc.nextInt(), sc.nextInt());
        Point<Double> p2 = new Point<Double>(sc.nextDouble(), sc.nextDouble());
        Point<Double> p3 = p1.mid(p2);
        System.out.println(p3);
    }
}
```

W6 LIVE CODING 1

Write a Java program that takes as input 4 Shop objects and the list of Shop objects with attributes shop name, and number of items sold nsold. The program should add each customer name as key and number of items as value to the map object. After adding all objects to the map, display the shop name which has sold maximum number of items as shown in the test cases. Complete the program as specified below:

• Class Shop that has the following members:

String name, int nsold as private instance variable

- Constructor to initialize the name and nsold

- Accessor methods to all instance variables

• Class MapTest has the following members:

You should define method public static void printShopName (ArrayList<Shop> sList) that does the following:

* Iterates over array sList such that in each iteration, add each customer name as key and number of items as value to the map object.

* Print the shop name which has sold maximum number of items.

- main method has the following functionality

* Creates a list of 4 Shop objects.

* Invokes method print ShopName (list) to print the shop name which has sold maximum number of items.

```
import java.util.*;
class Shop{
   private String name;
   private int nsold;
   public Shop(String s, int ns){
         this.name = s;
         this.nsold = ns;
   }
   public String getName(){
         return name;
   }
   public int getItemSold(){
         return nsold;
   }
}
public class MapTest {
         public static void printShopName(ArrayList<Shop> sList) {
            Map<String, Integer> m = new LinkedHashMap<String, Integer>();
            String shop = "";
            int sold = 0;
for(Shop s: sList)
    m.put(s.getName(), m.getOrDefault(s.getName(),0)+s.getItemSold());

for (HashMap.Entry<String, Integer> entry : m.entrySet()){
```

```
    if(entry.getValue()> sold) {
            shop = entry.getKey();
       sold = entry.getValue();
    }
}
       System.out.println(shop+" : "+sold);
    }
    public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            ArrayList<Shop> list = new ArrayList<Shop>();
            for (int i = 0; i < 4; i++) {
                    list.add(new Shop(sc.next(), sc.nextInt()));
            }
            printShopName(list);
    }
}
```