BSCCS2005: Sep 2023 OPE1 Questions with Test Cases and Solutions

Overview

The exam will be conducted in two sessions. In each session, a student will be presented with 6 questions out of which 4 are graded and 2 are optional (challenging). Out of the 4 graded questions, he/she should answer 3 questions correctly, passing all the private test cases to get a full score. In case they attempt all 4, only the best 3 scores will be considered. We configure 2 copies of each type on the portal, and using a randomization script, every student sees one copy of each type on the exam portal.
The challenging questions are not graded. The purpose of these questions is to engage students who finish the other questions early.

1 Session 2

1.1 Session 2 Type 1 Copy Constructor

Problem Statement

In a college, `Student s1` chooses a set of courses. `Student s2` also chooses all the courses chosen by `s1` except the second course, in place of which `s2` chooses another course. Write a program that defines two classes `Student` and `Admission`. Define copy constructor to create `s2` from `s1` such that changing the values of instance variables of either `s2` or `s1` does not affect the other one. The code takes `name` of student `s2` and the new course chosen by `s2` as input.

- Class `Student` has/should have the following members.
  - Private instance variables `String name` and `String[] courses` to store name and courses chosen respectively
  - Define required constructor(s)
  - Accessor methods `getName( )` and `getCourses(int)` to get the name of the student and the course at a specific index respectively.
  - Mutator methods `setName(String)` and `setCourses(int,String)` to set the name of the student and the course at a specific index respectively.
- Class `Admission` has method `main` that does the following.
  - Two objects of `Student s1` and `s2` are created. `s2` is created using `s1`
  - `name` of `Student s2` and second course chosen by `s2` are updated by taking the input
  - Finally, name of `s1`, `s2` and second course chosen by `s1` and `s2` are printed

What you have to do

- Define constructor(s) in class `Student`

Template Code

```
import java.util.*;
class Student{
    String name;
    String[] courses;
    //***** Define constructor(s) here
    public void setName(String n) {
        name = n;
    }
    public void setCourses(int indx, String c) {
        courses[indx] = c;
    }
    public String getName() {
        return name;
    }
    public String getCourses(int indx) {
        return courses[indx];
    }
}
public class Admission {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] courses = {"Maths", "DL","DSA","DC"};
        Student s1 = new Student("Nandu", courses);
        Student s2 = new Student(s1);
        s2.setName(sc.next());
        s2.setCourses(1,sc.next());
        System.out.println(s1.getName() + ": "+ s1.getCourses(1));
        System.out.println(s2.getName() + ": " + s2.getCourses(1));
    }
}
```

<u>**Test cases:**</u>

Public test case 1:
Input:

```
Suba COA
```

Output:

```
Nandu: DL
Suba: COA
```

Public test case 2:
Input:

```
Pai CV
```

Output:

```
Nandu: DL
Pai: CV
```

Private test case 1:
Input:

```
Neha DS
```

Output:

```
Nandu: DL
Neha: DS
```

Solution:

```
import java.util.*;
class Student{
    String name;
    String[] courses;
    public Student(String n, String[] c) {
        name = n;
        courses=c;
    }
    public Student(Student s) {
        this.name = s.name;
        this.courses = new String[s.courses.length];
        for(int i = 0; i < courses.length; i++) {
            this.courses[i] = s.courses[i];
        }
    }
    public void setName(String n) {
        name = n;
    }
    public void setCourses(int indx, String c) {
        courses[indx] = c;
    }
    public String getName() {
        return name;
    }
    public String getCourses(int indx) {
        return courses[indx];
    }
}
public class Admission {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] courses = {"Maths", "DL","DSA","DC"};
        Student s1 = new Student("Nandu", courses);
        Student s2 = new Student(s1);
        s2.setName(sc.next());
        s2.setCourses(1,sc.next());
        System.out.println(s1.getName() + ": "+ s1.getCourses(1));
        System.out.println(s2.getName() + ": " + s2.getCourses(1));
    }
}
```

Question:Abstract Classes and Interfaces

Complete the Java program to demonstrate the use of abstract classes and interfaces. You have to complete the definition of classes JuniorRS and SeniorRS to obtain the output as given in the public test cases.

- Interface `IResearchScholar` has two methods: `public void teaches(String str)` and `public void studies(String str)`.
- Define classes `JuniorRS` and `SeniorRS` such that `JuniorRS` implements `IResearchScholar` and `SeniorRS` extends `JuniorRS`.
- Class `InterAbstrTest` extends `SeniorRS`, and has the main method. An object of `JuniorRS` invokes the method `studies`, and an object of `SeniorRS` invokes methods `studies` and `teaches`.

Public Test Cases:
Test Case 1:

```
Input 1:
Python
Java
Output 1:
TA studies Python
TA studies Java
TA teaches Java
```

Test Case 2:

```
Input 2:
Cloud computing
Data Mining
Output 2:
TA studies Cloud computing
TA studies Data Mining
TA teaches Data Mining
```

Private Test Cases:
Test Case 1:

```
Input 1:
Machine Learning
Machine Learning
Output 1:
TA studies Machine Learning
TA studies Machine Learning
TA teaches Machine Learning
```

Test Case 2:

```
Input 2:
Cloud computing
Data Mining
Output 2:
TA studies Cloud computing
TA studies Data Mining
TA teaches Data Mining
```

Solution:

```
import java.util.Scanner;

interface IResearchScholar {
    public void teaches(String str);
    public void studies(String str);
}

abstract class JuniorRS implements IResearchScholar {
    public void studies(String str1) {
        System.out.println("TA studies " + str1);
    }
}

class SeniorRS extends JuniorRS {
    public void teaches(String str) {
        System.out.println("TA teaches " + str);
    }
}

public class InterAbstrTest extends SeniorRS {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str1 = sc.nextLine();
        String str2 = sc.nextLine();
        JuniorRS jrs = new InterAbstrTest();
        SeniorRS srs = new InterAbstrTest();
        jrs.studies(str1);
        srs.studies(str2);
        srs.teaches(str2);
        sc.close();
    }
}
```

Dynamic Dispatch

Problem Statement

Complete the Java code that uses the concept of inheritance to demonstrate dynamic method dispatching.

- Create a class `Vehicle` with the following members:
  - Private instance variable `name`.
  - Constructor to initialize `name`.
  - Accessor method for `name`.
  - Method `display` to display the text: "This is a generic vehicle."

  Classes `Car` and `Bicycle` should be defined in such a way that any object of `Car` or `Bycycle` can be assigned to a reference variable of type `Vehicle`. See the `main` method to understand the context.

  - For `Car`, the method `display` should print: "This is a car named name ."
  - For `Bicycle`, the method `display` should print: "This is a bicycle named name ."

In the `main` method of the `DispatchExample` class, create an array of `Vehicle` objects with size 3.

- Initialize the first element with a generic vehicle (you can use an empty string for its name).
- Initialize the second and third elements with a `Car` and a `Bicycle`, respectively, by taking the vehicle's name as input from the user.

Iterate over the array and call the `display` method for each vehicle.

Sample Input/Output

Input:

```
BMW
Giant
```

Output:

```
This is a generic vehicle.
This is a car named BMW.
This is a bicycle named Giant.
```

Template Code

```
import java.util.Scanner;

class Vehicle {
    private String name;

    public Vehicle(String n) {
        name = n;
    }
    // Define method display
    // Define an accessor method

}

//Define class Car
//Define class Bicycle

public class DispatchExample {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Vehicle[] vehicles = new Vehicle[3];

        vehicles[0] = new Vehicle("");
        vehicles[1] = new Car(sc.nextLine());
        vehicles[2] = new Bicycle(sc.nextLine());

        for (Vehicle v : vehicles) {
            v.display();
        }

        sc.close();
    }
}
```

Solution

```
import java.util.Scanner;

class Vehicle {
    private String name;

    public Vehicle(String n) {
        name = n;
    }

    public String getName() {
        return name;
    }

    public void display() {
        System.out.println("This is a generic vehicle.");
    }
}

class Car extends Vehicle {
    public Car(String n) {
        super(n);
    }

    public void display() {
        System.out.println("This is a car named " + getName() + ".");
    }
}

class Bicycle extends Vehicle {
    public Bicycle(String n) {
        super(n);
    }

    public void display() {
        System.out.println("This is a bicycle named " + getName() + ".");
    }
}

public class DispatchExample {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Vehicle[] vehicles = new Vehicle[3];

        vehicles[0] = new Vehicle("");
        vehicles[1] = new Car(sc.nextLine());
        vehicles[2] = new Bicycle(sc.nextLine());

        for (Vehicle v : vehicles) {
            v.display();
        }
        sc.close();
    }
}
```

1.2 Maps

Problem Statement

The Java program below takes as input the names of cricket players in a team and the runs scored by each of them in 3 consecutive matches. The program is supposed to print the names of those players who have scored at least 80 runs in all the matches. Complete the code to obtain the specified output.
Class `Team` has the following members:

- Instance variable `Map<String, ArrayList<Integer>> playerMap` (maps the player name to the list of runs scored by him/her in each match).
- A constructor to initialize the instance variable.
- An accessor method to access the instance variable.

Class `FClass` has the following members:

- A method `getFinalList( )` that accepts an object of class `Team` as input and returns the list of player names who has/have scored at least 80 runs in all the matches.
- `main( )` method does the following:
  - accepts inputs to instantiate an object of `Team`. The input is accepted in the order - player name followed by the list of his/her runs.
  - invokes method `getFinalList( )` by passing an object of `Team` as input, to return the list of player names who has/have scored at least 80 runs in all the matches.
  - prints the list returned by the method `getFinalList( )`

What you have to do

- Define method `getFinalList( )` of class `FClass`

Test Cases

Public test case 1 (Input):

```
Ravi 76 76 76
sonu 80 80 89
viral 98 47 99
```

Output:

```
[sonu]
```

Test Cases

Public test case 2 (Input):

```
P1 79 80 45
P2 88 46 90
P3 89 56 21
```

Output:

```
[]
```

Private test case 1 (Input):

```
P1 82 97 120
P2 80 90 99
P3 87 112 145
```

Output:

```
[P1, P2, P3]
```

Private test case 2 (Input):

```
P1 23 90 92
P2 88 65 78
P3 80 80 80
```

Output:

```
[P3]
```

Template Code

```
import java.util.*;

class Team{
    private Map<String, ArrayList<Integer>> playerMap;
    public Team( Map<String, ArrayList<Integer>> m) {
        playerMap = m;
    }
    public Map<String, ArrayList<Integer>> getPlayerMap(){
        return playerMap;
    }
}

public class FClass{
    public static ArrayList<String> getFinalList(Team t) {
        // Define the method getFinalList( ) here
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Map<String, ArrayList<Integer>> pmap =
                    new LinkedHashMap<String, ArrayList<Integer>>();

        for(int i = 0; i < 3; i++) {
            ArrayList<Integer> pruns = new ArrayList<Integer>();
            String name = sc.next();
            for(int j = 0; j < 3; j++) {
                pruns.add(sc.nextInt());
            }
            pmap.put(name, pruns);
        }

        Team t = new Team(pmap);
        System.out.println(getFinalList(t));
    }
}
```

Solution:

```
import java.util.*;

class Team{
    private Map<String, ArrayList<Integer>> playerMap;
    public Team( Map<String, ArrayList<Integer>> m) {
        playerMap = m;
    }
    public Map<String, ArrayList<Integer>> getPlayerMap(){
        return playerMap;
    }
}

public class FClass{
    public static ArrayList<String> getFinalList(Team t) {
        ArrayList<String> pList = new ArrayList<String>();
        Map<String, ArrayList<Integer>> pmap = t.getPlayerMap();
        for(String p : pmap.keySet()) {
            boolean flag = true;
            for(Integer i: pmap.get(p)) {
                if(i < 80) {
                    flag = false;
                    break;
                }
            }
            if(flag)
                pList.add(p);
        }
        return pList;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Map<String, ArrayList<Integer>> pmap =
                    new LinkedHashMap<String, ArrayList<Integer>>();

        for(int i = 0; i < 3; i++) {
            ArrayList<Integer> pruns = new ArrayList<Integer>();
            String name = sc.next();
            for(int j = 0; j < 3; j++) {
                pruns.add(sc.nextInt());
            }
            pmap.put(name, pruns);
        }

        Team t = new Team(pmap);
        System.out.println(getFinalList(t));
    }
}
```

1.3 Generics

Problem Statement

Write a Java program to find the sum of two complex numbers. You are given as input two integers `n1, n2` and two double values `d1, d2,` from which two complex numbers `c1 and c2` are obtained as described below.

- The real parts of `c1 and c2` are `n1` and `d1` respectively, whereas their imaginary parts are `n2 and d2`, respectively.

  Class `ComplexNum` is a generic class with the following members.

- Instance variables `r` and `i`
- A constructor to initialize `rand` `i`
- Method `add` to return the sum of two instances of generic type `ComplexNum`
- Method `toString()` to format the output as is given in the test cases.

  Class `FClass` has method main that does the following:

- Accepts two integers `n1` and `n2`, and two doubles `d1` and `d2` as input
- Creates two objects of `ComplexNum` using the input values
- Invokes method `add` to obtain the `sum`
- prints the `sum`

  What you have to do

- Define class `ComplexNum`

Template code

```
import java.util.*;
//DEFINE class ComplexNum
class FClass{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1, n2;
        double d1, d2;
        n1 = sc.nextInt();
        n2 = sc.nextInt();
        d1 = sc.nextDouble();
        d2 = sc.nextDouble();
        ComplexNum<Integer> c1 = new ComplexNum<Integer>(n1, n2);
        ComplexNum<Double> c2 = new ComplexNum<Double>(d1, d2);
        ComplexNum<Double> c3 = c1.add(c2);
        System.out.println(c1 + " + " + c2 + " = " + c3);
    }
}
```

Public test case 1:
Input:

```
6 10
10.3 15.6
```

Output:

```
6.0 + 10.0i + 10.3 + 15.6i = 16.3 + 25.6i
```

Public test case 2:
Input:

```
10 15
5.4 1.6
```

Output:

```
10.0 + 15.0i + 5.4 + 1.6i = 15.4 + 16.6i
```

Private test case 1:
Input:

```
3 15
5.4 2.8
```

Output:

```
3.0 + 15.0i + 5.4 + 2.8i = 8.4 + 17.8i
```

Private test case 1:
Input:

```
10 20
13.3 5.12
```

Output:

```
10.0 + 20.0i + 13.3 + 5.12i = 23.3 + 25.12i
```

Solution:

```
import java.util.*;
class ComplexNum<T extends Number>{
    private T r, i;
    public ComplexNum(T r, T i) {
        this.r = r;
        this.i = i;
    }
    public ComplexNum<Double> add(ComplexNum<?> c){
        ComplexNum<Double> dc = new ComplexNum<Double>(0.0, 0.0);
        dc.r = this.r.doubleValue() + c.r.doubleValue();
        dc.i = this.i.doubleValue() + c.i.doubleValue();
        return dc;
    }
    public String toString() {
        return r.doubleValue() + " + " + i.doubleValue() + "i";
    }
}
class FClass{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1, n2;
        double d1, d2;
        n1 = sc.nextInt();
        n2 = sc.nextInt();
        d1 = sc.nextDouble();
        d2 = sc.nextDouble();
        ComplexNum<Integer> c1 = new ComplexNum<Integer>(n1, n2);
        ComplexNum<Double> c2 = new ComplexNum<Double>(d1, d2);
        ComplexNum<Double> c3 = c1.add(c2);
        System.out.println(c1 + " + " + c2 + " = " + c3);
    }
}
```

Problem Statement

Write a Java program that, given as input name and salary of a number of employees, creates an array of `Employee` objects and prints the bonus of employees. Complete the program as specified below.

- Abstract class `Employee` has the following members:
  - Private instance variables String `name` and double `salary`

- Constructor to initialize the instance variables
- Accessor method for `salary`, `name`
- Abstract method `public abstract void printBonus()`

- Classes `Manager` and `Director` should be defined in such a way that any object of `Manager` or `Director` can be assigned to a reference variable of type `Employee`.
  - Class `Manager` should have constructor to initialize the instance variables. For an object of `Manager`, method `printBonus` should print the name of employee followed by his/her bonus.The bonus for a Manager is 10% of the salary.The accessor method `getSalary()` in `Employee` class returns the current salary.
  - Class `Director` should have constructor to initialize the instance variables. For an object of `Director`, method `printBonus` should print the name of employee followed by his/her bonus.The bonus for a Director is 15% of the salary.The accessor method `getSalary()` in `Employee` class returns the current salary.
- Class `Organisation` has the following members:
  - Method `main` accepts the `name`, `salary` of a `Manager` object followed by that of a `Director` object and stores the objects in an Employee array, and then invokes the method `printBonus` for each object in the array

What you have to do

- Define subclasses `Manager` and `Director`

[Abstract Class:Sonam]

Template Code

```
import java.util.*;
abstract class Employee {
    private String name;
    private double salary;

    public Employee(String n, double s) {
        name = n;
        salary = s;
    }

    public double getSalary() {
        return salary;
    }
    public String getName() {
        return name;
    }

    public abstract void printBonus();
}

//********* DEFINE class Manager

//********* DEFINE class Director

public class Organisation {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee[] eArr = new Employee[2];
        eArr[0] = new Manager(sc.nextLine(), sc.nextDouble());
        eArr[1] = new Director(sc.nextLine(), sc.nextDouble());
        eArr[0].printBonus();
        eArr[1].printBonus();
        sc.close();
    }
}
```

Public test case 1:
Input:

```
Ashok 30000.00
Swaraj 40000.00
```

Output:

```
Ashok : 3000.0
Swaraj : 6000.0
```

Public test case 2:
Input:

```
Srinivas 50000.00
Sureka 453200.00
```

Output:

```
Srinivas : 5000.0
Sureka : 67980.0
```

Private test case 1:
Input:

```
Rahul 40000.00
Usha 234000.00
```

Output:

```
Rahul : 4000.0
Usha : 35100.0
```

Private test case 2:
Input:

```
Saurab 56000.00
Harsha 23000.00
```

Output:

```
Saurab : 5600.0
Harsha : 3450.0
```

Solution:

```
import java.util.*;
abstract class Employee {
    private String name;
    private double salary;

    public Employee(String n, double s) {
        name = n;
        salary = s;
    }

    public double getSalary() {
        return salary;
    }

    public String getName() {
        return name;
    }

    public abstract void printBonus();
}

public class Manager extends Employee {
    public Manager(String name, double salary) {
        super(name, salary);
    }

    public void printBonus() {
         System.out.println(getName() + " : " + getSalary() * 0.1);
    }
}

public class Director extends Employee {
    public Director(String name, double salary) {
        super(name, salary);
    }

    public void printBonus() {
         System.out.println(getName() + " : " + getSalary() * 0.15);
    }
}

public class Organisation {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Employee[] eArr = new Employee[2];
        eArr[0] = new Manager(sc.nextLine(), sc.nextDouble());
        eArr[1] = new Director(sc.nextLine(), sc.nextDouble());
        eArr[0].printBonus();
        eArr[1].printBonus();
        sc.close();
    }
}
```

1.4 Session 1 Type 4 Interfaces

Problem Statement

A smartphone is both a mobile device and a camera. Complete the Java program below in order to print customized messages based on which among the two roles - mobile device or camera - of a smartphone is being given as input.

- Interface `MobileDevice` has the abstract method `makeCall()`
- Interface `Camera` has the abstract method `takePicture()`
- Class `Smartphone` has properties of both a `MobileDevice` and a `Camera`, and has the following members:
  - method `makeCall()` that returns the string: `<Name of the device> makes a call`
  - method `takePicture()` that returns the string: `<Name of the device> takes a picture`
- Class `InterfaceTest` has the main method that does the following:
  - Accepts as input the types and names of 3 devices
  - If the type is `M`, then invoke method `makeCall()`, else invoke method `takePicture()`
  - Print the messages

What you have to do

- Define interface `MobileDevice`
- Define interface `Camera`
- Define class `Smartphone`

Test Cases

Public test case 1 (Input):

```
M Samsung
C Canon
M iPhone
```

Output:

```
Samsung makes a call
Canon takes a picture
iPhone makes a call
```

Test Cases

Public test case 2 (Input):

```
C Nikon
C Sony
C GoPro
```

Output:

```
Nikon takes a picture
Sony takes a picture
GoPro takes a picture
```

Private test case 1 (Input):

```
C Huawei
M Huawei
M Huawei
```

Output:

```
Huawei takes a picture
Huawei makes a call
Huawei makes a call
```

Template Code

```
import java.util.*;

//DEFINE interface MobileDevice
//DEFINE interface Camera
//DEFINE class Smartphone

class InterfaceTest{
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
ArrayList<String> messagesList = new ArrayList<>();
    for(int i=0;i<3;i++){
        String type = sc.next();
        if (type.equals("M")){
            MobileDevice m = new Smartphone(sc.next());
            messagesList.add(m.makeCall());
        }
        else if (type.equals("C")){
            Camera c = new Smartphone(sc.next());
            messagesList.add(c.takePicture());
        }
    }
    for (String s:messagesList){
        System.out.println(s);
    }
    sc.close();
}
}
```

Solution:

```
import java.util.*;
interface MobileDevice{
String makeCall();
}
interface Camera{
String takePicture();
}
class Smartphone implements MobileDevice, Camera{
String name;
public Smartphone(String n){
name = n;
}
public String makeCall(){
return name+" makes a call";
}
public String takePicture(){
return name+" takes a picture";
}
}
class InterfaceTest{
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
ArrayList<String> messagesList = new ArrayList<>();
    for(int i=0;i<3;i++){
        String type = sc.next();
        if (type.equals("M")){
            MobileDevice m = new Smartphone(sc.next());
            messagesList.add(m.makeCall());
        }
        else if (type.equals("C")){
            Camera c = new Smartphone(sc.next());
            messagesList.add(c.takePicture());
        }
    }
    for (String s:messagesList){
        System.out.println(s);
    }
    sc.close();
}
}
```

1.5 Session 2 Type 2 Copy 1 and 2 Maps

Problem Statement

Complete the 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 a key and the number of items as a value to the map object. After adding all objects to the map, display the shop name which has sold the 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 the maximum number of items.
  - `main` method has the following functionality
    * Creates a list of 4 `Shop` objects.
    * Invokes method `printShopName(list)` to print the shop name which has sold the maximum number of items.

What you have to do:

- Define method `printShopName` inside class `MapTest`

Template Code

```
import java.util.*;

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;

        // Write your code here

        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);
    }
}
```

<u>Test Cases</u>

Public test case 1:
Input:

```
SuperBazar 30
More 40
Shopsy 30
More 30
```

Output:

```
More : 70
```

Public test case 2:
Input:

```
Lulu 40
Lulu 34
DLF 54
DLF 67
```

Output:

```
DLF : 121
```

Private test case 1:
Input:

```
HiLITE 56
Sarath-City 40
Z-Square 54
World-Trade 43
```

Output:

```
HiLITE : 56
```

Private test case 2:
Input:

```
Mantri-Square 56
Mantri-Square 76
Mantri-Square 11
Phoenix 590
```

Output:

```
Phoenix : 590
```

Solution:

```
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);
    }
}
```