You will be able to resubmit before due date.

Due: 26 Jun 2022 15:30 IST
Time Left: 01:38:41

`Tourists t1` and `t2` visit a set of tourist places in India. `Tourist t2`, besides visiting all the tourist places visited by `t1`, also visits an additional place. Write a Java program that models this scenario using a copy constructor. The code should take as input the name of `Tourist t1` and the two tourist places visited by `t1`, followed by the name of `Tourist t2` and the additional place visited by `t2`.

• Class `Tourist` has `tName`, `tPlaces` as instance variables.

-- It should have two parameterized constructors, one to instantiate the instance variables and the other is a copy constructor.

-- Implement the copy constructor `public Tourist(Tourist t)` so that any later change in the visited places of `Tourist t2` does not affect the visited places of `t1`.

-- Override the method `toString()` to display the output as shown in the test cases.

• Class `Test` has the main method.

-- It accepts the names of the tourists, places visited by `t1` and the additional place visited by `t2` as input

Java documentation: https://docs.oracle.com/en/java/javase/11/docs/api/index.html

This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submitting using "Submit" button below. If you only test run the program, your assignment will not

This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submitting using "Submit" button below. If you only test run the program, your assignment will not be graded and you will not see your score after the deadline.

Java

TravellerTest.java

```
    public Tourist(String vName, ArrayList<String> vPlaces) {
        this.tName = vName;
        this.tPlaces = vPlaces;
    }
    public Tourist(Tourist t){
       // copy the values as shown in the test cases to create a new object
       this.tName=t.tName;
       this.tPlaces=t.tPlaces;
    }
    public String toString(){
                return (tName+":[]"+tPlaces+"[]");                    // Override the toString() method      retu
}
public class TravellerTest{
```

Test Run | Submit

Test Run Results

Summary: TravellerTest.java:16: error: missing return statement } => 1 error

Public Test: 0/0 Passed

Download All

Test Case 1

Input	Expected Output	Actual Output
Rocky Agra Goa Shafi Jaipur	Rocky:[Agra, Goa] Shafi:[Agra, Goa, Jaipur]	

Test Case 2

Input	Expected Output	Actual Output
Virat Kerala Varanasi Rohit Mussoorie	Virat:[Kerala, Varanasi] Rohit:[Kerala, Varanasi, Mussoorie]	

Java

TravellerTest.java

```java
import java.util.*;
class Tourist{
    String tName;
    ArrayList<String> tPlaces;
    public Tourist(String vName, ArrayList<String> vPlaces) {
        this.tName = vName;
        this.tPlaces = vPlaces;
    }
    public Tourist(Tourist t){
       // copy the values as shown in the test cases to create a new object
    }
    public String toString(){
      // Override the toString() method
  }
}
public class TravellerTest{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    ArrayList<String> vp = new ArrayList<String>();
    String n = sc.next();
    vp.add(sc.next());
    vp.add(sc.next());
    Tourist t1 = new Tourist(n,vp);
    Tourist t2 = new Tourist(t1);
    t2.tName = sc.next();
    t2.tPlaces.add(sc.next());
    System.out.println(t1 + "\n" + t2);
    sc.close();
  }
}
```

Java

TravellerTest.java

```java
import java.util.*;
class Tourist{
    String tName;
    ArrayList<String> tPlaces;
    public Tourist(String vName, ArrayList<String> vPlaces) {
        this.tName = vName;
        this.tPlaces = vPlaces;
    }
    public Tourist(Tourist t){
       // copy the values as shown in the test cases to create a new object
       this.tName=t.tName;
       ArrayList<String> vp1 = new ArrayList<String>();
       for(String s: t.tPlaces){
        vp1.add(s);
    }
    this.tPlaces=vp1;}
    public String toString(){
      // Override the toString() method
      return(tName + ":" + tPlaces);
  }
}
public class TravellerTest{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    ArrayList<String> vp = new ArrayList<String>();
    String n = sc.next();
    vp.add(sc.next());
    vp.add(sc.next());
    Tourist t1 = new Tourist(n,vp);
    Tourist t2 = new Tourist(t1);
    t2.tName = sc.next();
    t2.tPlaces.add(sc.next());
    System.out.println(t1 + "\n" + t2);
    sc.close();
  }
}
```

You will be able to resubmit before due date.

Due: 26 Jun 2022 15:30 IST
Time Left: 01:31:18

Complete the Java program to compute the area of a rectangle. The inputs for the program are the `x` and `y` coordinates of the points denoting the top-left corner and that of the bottom-right corner of the rectangle.

• Class `Rectangle` has/should have the following members:

-- Two instance variables `cp1` and `cp2` of type `Point`, where `Point` is a private class inside Class `Rectangle`.

-- Constructor `Rectangle(int, int, int, int)`, where the arguments are in the order: x-coordinate and y-coordinate of one corner, x-coordinate and y-coordinate of opposite corner of a rectangle.

-- Methods `int getWidth()` and `int getHeight()` compute the width and the height of the rectangle respectively. These should compute the absolute values (if needed, use method from `Math` class).

-- Class `Point` is an inner class with two instance variables `x` and `y`, and a constructor to initialize them.

• Class `PvtClass` has the main method that reads in all the four coordinates in the following order: x-coordinate and y-coordinate of one corner, x-coordinate and y- coordinate of opposite corner of a rectangle. It then creates a new `Rectangle` object using these coordinates, and finds its area.

Java documentation: https://docs.oracle.com/en/java/javase/11/docs/api/index.html

```
import java.util.*;
class Rectangle{
    private Point cp1;
    private Point cp2;
    public Rectangle(int x1, int y1, int x2, int y2){
//Complete the code
    }
    int getWidth(){
    //Complete the code
    }
    int getHeight(){
    //Complete the code
    }
```

Test Run | Submit

Sample Test Cases

Download All

Test Case 1

Input	Expected Output	Actual Output
2 2 4 1	Area = 2	

Test Case 2

Input	Expected Output	Actual Output
11 15 16 13	Area = 10	

Java

PvtClass.java

```java
import java.util.*;
class Rectangle{
    private Point cp1;
    private Point cp2;
    public Rectangle(int x1, int y1, int x2, int y2){
//Complete the code
   }
   int getWidth(){
   //Complete the code
   }
   int getHeight(){
   //Complete the code
   }
   private class Point{
   //Complete the code
    }
}
public class PvtClass{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int x1 = sc.nextInt();
        int y1 = sc.nextInt();
        int x2 = sc.nextInt();
        int y2 = sc.nextInt();
        Rectangle r = new Rectangle(x1, y1, x2, y2);
        System.out.println("Area = " + r.getHeight() * r.getWidth());
        sc.close();
    }
}
```

Java

PvtClass.java

```java
import java.util.*;
class Rectangle{
    private Point cp1;
    private Point cp2;
    public Rectangle(int x1, int y1, int x2, int y2){
//Complete the code
    cp1=new Point(x1,y1);
    cp2=new Point(x2,y2);
   }
   int getWidth(){
   //Complete the code
   return Math.abs(cp2.x-cp1.x);
   }
   int getHeight(){
   //Complete the code
   return Math.abs(cp2.y-cp1.y);
   }
   private class Point{
      int x;
      int y;
      public Point(int x1,int y1){
          x=x1;
          y=y1;
      }
   //Complete the code
    }
}
public class PvtClass{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int x1 = sc.nextInt();
        int y1 = sc.nextInt();
        int x2 = sc.nextInt();
        int y2 = sc.nextInt();
        Rectangle r = new Rectangle(x1, y1, x2, y2);
        System.out.println("Area = " + r.getHeight() * r.getWidth());
        sc.close();
    }
}
```

You will be able to resubmit before due date.

Due: 26 Jun 2022 15:30 IST
Time Left: 01:30:54

Complete the Java program that defines three classes `Person`, `Employee` and `Student`. The classes `Employee` and `Student` inherit from `Person`. The input to the program are the number of persons, followed by an integer indicating the type of person (1 for `Person`, 2 for `Employee` and 3 for `Student`), name and a number (if type is `Employee` or `Student`) that denotes salary for an `Employee` or semester for a `Student`. Your program should implement a method `printAll` that is capable of printing a list of objects each of which may be of type `Person`, `Employee` or `Student`.

- Class `Person` has been defined.
- Define class `Employee` which has an additional instance variable (as compared to class `Person`) `salary` of type double, and overrides the method `show` of class `Person`. Write the appropriate constructor for the same.
- Define class `Student` which has an additional instance variable (as compared to class `Person`) `semester` of type int, and overrides the method `show` of class `Person`. Write the appropriate constructor for the same.
- Class `TClass` has two methods - `main` and `printAll`. Complete the definition of method `printAll` that accepts a list of the following types only:

-- `List<Person>`, `List<Employee>`, and `List<Student>`

and prints the objects using method `show`.

Java documentation: https://docs.oracle.com/en/java/javase/11/docs/api/index.html

```
    }
}
//Define class Employee
//Define class Student
```

Test Run Submit

Sample Test Cases

Download All

Test Case 1

Input	Expected Output	Actual Output
5 1 jeevan 2 latha 20000 3 deba 3 2 jit 30000 3 riya 4	jeevan latha 20000.0 jit 30000.0 deba 3 riya 4	

Test Case 2

Input	Expected Output	Actual Output
7 2 dilip 20000 3 nitin 5 2 ananya 30000 1 susmitha 3 latha 6 1 gourav 2 amit 35000	susmitha gourav dilip 20000.0 ananya 30000.0 amit 35000.0 nitin 5 latha 6	

Java

TClass.java

```java
import java.util.*;
class Person{
    private String name;
    public Person(String n){
        name = n;
    }
    public void show(){
        System.out.print(name  + " ");
    }
}
//Define class Employee
//Define class Student

public class TClass{
    //Define method printAll
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        List<Person> lp = new ArrayList<Person>();
        List<Employee> le = new ArrayList<Employee>();
        List<Student> ls = new ArrayList<Student>();

        int n = sc.nextInt();    //number of inputs
        for(int i = 0; i < n; i++){
            int t = sc.nextInt();   //type of person
            String s1 = sc.next();    //name
            if(t == 2){
                double s2 = sc.nextDouble(); //salary
                le.add(new Employee(s1, s2));
            }
            else if(t == 3){
                int s3 = sc.nextInt();  //semester
                ls.add(new Student(s1, s3));
            }
            else {
                lp.add(new Person(s1));
            }
        }
        printAll(lp);
        System.out.println();
        printAll(le);
        System.out.println();
        printAll(ls);
    }
}
```

You will be able to resubmit before due date.

Due: 26 Jun 2022 15:30 IST
Time Left: 01:30:25

Complete the Java program that takes as input the names of four customers and the number of items bought by each of them, and prints the name of the customer who has bought the maximum number of items.

- Interface `Iterator` has the following members:
  - Two abstract methods: `has_next()` and `get_next()`
- Class `Customer` has the following members:
  - Two instance variables: `String custName` and `int numItems`
  - A constructor to initialize the instance variables
  - Accessor methods for the instance variables
  - Overridden `toString` method to display the customer name.

- Class `CustomerList` has the following members:
  - An instance variable: `Customer[] cArr`
  - A constructor to initialize the instance variable
  - Method `public Iterator getIterator()` which returns an inner class object.
  - Private inner class `CustIterator` which implements the `Iterator` interface and has the following members.
    - One instance variable: `private int index`

-- Method `public Iterator getIterator()` which returns an inner class object.

-- Private inner class `CustIterator` which implements the `Iterator` interface and has the following members.

* One instance variable: `private int index`
* A constructor to initialize the instance variable.
* Overridden `has_next()` and `get_next()` methods to traverse through the customer array.

• Class `IteratorTest` has the following methods.

-- The `main` method takes the name and the number of items bought by each customer, as input.

-- It also has method `getMaxCustomer(CustomerList cList)` to find the customer who has bought the maximum number of items. For ease of implementation, you may assume that there is exactly one customer who has bought the maximum number of items.

Java documentation:https://docs.oracle.com/en/java/javase/11/docs/api/index.html

This assignment has public test cases. Please click on "Test Run" button to see the status of public test cases. Assignment will be evaluated only after submitting using "Submit" button below. If you only test run the program, your assignment will not be graded and you will not see your score after the deadline.

Choose Language

1

/javase/11/docs/api/index.html

```
import java.util.*;
interface Iterator{
    public boolean has_next();
    public Object get_next();
}
class Customer{
    private String custName;
    private int numItems;
    public Customer(String custName, int num) {
        this.custName = custName;
        this.numItems = num;
    }
    public String getName() {
```

Test Run Submit

Sample Test Cases

Download All

Test Case 1

Input	Expected Output	Actual Output
Priya 12 Vidya 6 Reshma 13 Vignesh 4	Reshma	

Test Case 2

Input	Expected Output	Actual Output
Deepa 1000 Prakash 1150 Sukesh 1151 Mukesh 1152	Mukesh	

Java

IteratorTest.java

```java
import java.util.*;
interface Iterator{
    public boolean has_next();
    public Object get_next();
}
class Customer{
    private String custName;
    private int numItems;
    public Customer(String custName, int num) {
        this.custName = custName;
        this.numItems = num;
    }
    public String getName() {
        return custName;
    }
    public int getNumItems() {
        return numItems;
    }
    public String toString() {
        return custName;
    }
}
class CustomerList {
    private Customer[] cArr;
    public CustomerList(Customer[] ca) {
        cArr = ca;
    }
    public Iterator getIterator() {
        return new CustIterator();
    }
    private class CustIterator implements Iterator{
        private int index;
        public CustIterator() {
            index = -1;
        }
        public boolean has_next() {
            if(index < cArr.length - 1)
                return true;
            return false;
        }
        public Object get_next() {
            index++;
            return cArr[index];
        }
    }
}
public class IteratorTest {
    public static Customer getMaxCustomer(CustomerList cList) {

    //Complete the definition of this method
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Customer[] cA = new Customer[4];
        for(int i=0;i<4;i++) {
            cA[i] = new Customer(sc.next(), sc.nextInt());
        }
        CustomerList custList = new CustomerList(cA);
        System.out.println(getMaxCustomer(custList));
        sc.close();
    }
}
```

Java

IteratorTest.java

```java
import java.util.*;
interface Iterator{
    public boolean has_next();
    public Object get_next();
}
class Customer{
    private String custName;
    private int numItems;
    public Customer(String custName, int num) {
        this.custName = custName;
        this.numItems = num;
    }
    public String getName() {
        return custName;
    }
    public int getNumItems() {
        return numItems;
    }
    public String toString() {
        return custName;
    }
}
class CustomerList {
    private Customer[] cArr;
    public CustomerList(Customer[] ca) {
        cArr = ca;
    }
    public Iterator getIterator() {
        return new CustIterator();
    }
    private class CustIterator implements Iterator{
        private int index;
        public CustIterator() {
            index = -1;
        }
        public boolean has_next() {
            if(index < cArr.length - 1)
                return true;
            return false;
        }
        public Object get_next() {
            index++;
            return cArr[index];
        }
    }
}
public class IteratorTest {
    public static Customer getMaxCustomer(CustomerList cList) {

    //Complete the definition of this method
    Iterator it=cList.getIterator();
    int val=0;
    Customer cu=null;
    Customer cu1=null;
    while(it.has_next()){
         cu1=(Customer)it.get_next();
         if(cu1.getNumItems()>val){
             val=cu1.getNumItems();
             cu=cu1;
         }
        // System.out.println(cu.getName());
    }
    return cu;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Customer[] cA = new Customer[4];
        for(int i=0;i<4;i++) {
            cA[i] = new Customer(sc.next(), sc.nextInt());
        }
        CustomerList custList = new CustomerList(cA);
        System.out.println(getMaxCustomer(custList));
        sc.close();
    }
}
```