IRC_SKCT_Java2_COD_thread

Test Summary

- No. of Sections: 1
- No. of Questions: 8
- Total Duration: 120 min

Section 1 - Coding

Section Summary

- No. of Questions: 8
- Duration: 120 min

Additional Instructions:

None

Q1. Write a program which starts two threads *marked* "even" and "odd". The threads cooperate to print the numbers from 1 to 20 in sequence with the "even" thread printing only even numbers and odd the thread printing only odd numbers.
Ignore output line order.

Output Format

The numbers from 1 to 20 are printed in sequence alternatively by the odd and even threads with the even thread printing only even numbers and the odd thread printing only odd numbers.

Sample Input

Sample Output

```
Thread Odd: 1
Thread even: 2
Thread Odd: 3
Thread even: 4
```

Time Limit: 100 ms Memory Limit: 256 kb Code Size: 1024 kb

Q2. Maruti Suzuki gives a special bonus to its employees with 100% attendance in a year with a certificate of appreciation. Assume there are N employees. Get the attendance percentage of N employees and store them in an array.

Create **two threads** so that thread1 determines the total count of employees eligible for a certificate in the first half of the array and thread2 in the second half of the array.

The main( ) has to wait till both the threads complete their task and arrive at a final count indicating the total number of employees eligible for the certificate of appreciation.

Input Format

The first line of the input consists of the number of employees.
The second line has the average attendance percentage of employees.

Output Format

The output prints the winners count.

Sample Input

```
10
89 100 75 98 90 100 100 98 83 99
```

Sample Output

```
Winners : 3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **Stall Revenue**

The Accounting department of the fair committee wants an console application that can estimate the total revenue by rent from an exhibition. So write a program that accepts the stall details of an exhibition that includes the stallArea which is used for computing the stallCost. Using threads, calculate the stallCost of each stalls and in the main method, print the consolidated data.

Create a class **Stall** which implements Runnable interface with the following private attributes,

Create default constructor and a parameterized constructor with arguments in order Stall(String stallName, String details, Double stallArea, String owner). Create appropriate getters and setters.
Override the following methods in the **Stall** class,

Get the number of stalls and stall details and calculate the total revenue of all the stalls. Calculate the stall cost for each stall, each cost will be calculated by seperate thread.
Create a driver class **Main** to test the above requirements.

Input Format

First line of the input consist of the number of inputs to be given
Next input is the stall details
Refer sample input

Output Format

Output prints the stall cost of each stall
Refer sample output

Sample Input

```
3
Book stall
Stall for books
25
```

Sample Output

```
3750.0
4500.0
9000.0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. **Profit or Loss**

Now we are going to create a console application that can estimate whether the booking is a profit or loss, thereby enabling hall owners to reduce or increase expenses depending on the status. Hence if the several Booking details are given, compute whether the bookings are profitable or not. Use Threads to compute for each booking, Finally display the profit/loss status.
Create a class HallBooking which implements Runnable interface with following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor with arguments in order HallBooking(String hallName, Double cost, Integer hallCapacity,Integer seatsBooked).
Override run() method which display the status (i.e) Profit or Loss.If SeatsBooked*100 > cost then it is a profit else loss .
Create a driver class Main. The status for each hall is calculated by separate threads. The Threads print the status of the events.
Ignore output line order

Input Format

The first line of input corresponds to the number of events 'n'.
Next input is the hall details.
Refer sample input for formatting specifications.

Output Format

The output consists of the status of the events. (Profit or Loss).
Refer sample output for formatting specifications.

Sample Input

```
4
Le Meridian
12000
400
```

Sample Output

```
Profit
Loss
Profit
Profit
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. Write a program for matrix multiplication in java using threads.

Input Format

Row and column value of Matrix 1 in first line separated by space
Row and column value of Matrix 2 in second line separated by space
Matrix 1 elements
Matrix 2 elements

Output Format

Display the matrix after multiplication.

Constraints

Integers only.

Sample Input

Sample Output

```
2 2
2 2
1 2
3 4
```

```
19 22
43 50
```

Time Limit: 10 ms Memory Limit: 256 kb Code Size: 1024 kb

Q6. **Interest Calculation**

Now we are going to calculate the interest based on the balance for a bank application. Use threads to calculate the interest and final amount. Finally print the interest and the final balance.
Create a class **Account** that extends thread class with the following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor.
Override the run() method to calculate and display the interest and balance.
If the balance is greater than or equal to 10000 then the rate of interest is 8% else 5%.
Create a driver class **Main.** The interest and balance for each account is calculated by separate threads. The Threads print the interest and final balance of the accounts.
Ignore Output line order.

Input Format

The first line of the input consists of the input n.
Next input is the account details.

Output Format

The output prints the interest and balance of the accounts in next lines.

Sample Input

```
2
3256858548
50000
mahesh
```

Sample Output

```
250.00
4000.00
5250.00
54000.00
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. **ItemType-Amount Calculation**

Now we are going to calculate the total amount of the items purchased based on the number of items and cost per item. Use threads to calculate the total amount.
Create a class **ItemType** that extends thread class with the following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor.
Override the run() method to calculate and display the total amount.
Round off the double values to two decimal places.
Create a driver class **Main.** The total amount is calculated by separate threads. The Threads print the total amount of the items purchased.

Input Format

First line of the input consist of number of inputs to be given
Next input is the item details
Refer sample input

Output Format

Output prints the total amount of the items purchased.
Refer sample output

Sample Input

```
2
Laptop
40000
30000
```

Sample Output

```
300000.00
400000.00
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. Your English literature friend is very happy with the code you gave him. Now for his research, he used your application to find character frequency in many novels. For larger novels, the application takes a lot of time for computation. So he called you on a fine Sunday to discuss this with you. He wanted to know whether you can improve the speed of the application.

You decided to modify the application by using multiple threads to reduce the computation time. For this, accept the number of counters or threads at the beginning of the problem and get the string for each counter or thread. Create a thread by extending the Thread class and take the user entered the string as input. Each thread calculates the character frequency for the word assigned to that thread. All the counts are stored locally in the thread and once all the threads are completed print the character frequency for each of the thread.

Ignore output line order.

Input Format

Number of String (N)
N number of Strings in each line

Output Format

Frequency of characters as shown in sample output

Sample Input

```
2
welcome
java
```

Sample Output

```
w1
e2
l1
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Coding

Q1

Test Case

Input	Output
	Thread Odd: 1 Thread even: 2 Thread Odd: 3 Thread even: 4

Weightage - 100

Sample Input	Sample Output
	Thread Odd: 1 Thread even: 2 Thread Odd: 3 Thread even: 4

Solution

```
import java.util.*;
import java.lang.*;
import java.io.*;
import java.util.concurrent.*;

class Q01Complex_Thread
{
    private static final Object objectToSync = new Object();
    static CyclicBarrier barrier = new CyclicBarrier(2);
    public static class MyRunnable implements Runnable {
        private boolean printEven;
        public MyRunnable(boolean b) {
            this.printEven = b;
        }
        public void run() {
            try {
                for (int i = 1; i < 21; i++) {
                    if (printEven && i % 2 == 0)
                        System.out.println("Thread even: " + i);
                    else if (!printEven && i % 2 == 1)
                        System.out.println("Thread Odd: " + i);
                    barrier.await();
                }
            } catch (BrokenBarrierException e) {
            } catch (InterruptedException e) {
            }
        }
    }
    public static void main (String[] args) throws java.lang.Exception {
       Thread tEven = new Thread(new MyRunnable(true));
       tEven.start();
       Thread tOdd = new Thread(new MyRunnable(false));
       tOdd.start();
    }
}
```

Q2

Test Case

Input

```
30
84 99 75 100 89 98 97 99 93 98 96 98 93 98 82 1
```

Output

```
Winners : 5
```

Weightage - 10

Input

```
60
79 99 87 99 95 98 75 100 82 98 97 99 81 100 83
```

Output

```
Winners : 11
```

Weightage - 10

Input

```
100
92 100 82 99 83 98 88 99 94 99 95 98 80 100 81
```

Output

```
Winners : 24
```

Weightage - 10

Input

```
150
78 98 80 99 94 100 92 99 91 98 78 99 80 98 78 9
```

Output

```
Winners : 28
```

Weightage - 15

Input

```
300
84 100 93 99 96 99 75 98 88 99 80 100 81 99 76
```

Output

```
Winners : 45
```

Weightage - 15

Input

```
250
78 100 99 100 92 99 95 99 98 100 90 98 80 98 87
```

Output

```
Winners : 47
```

Weightage - 20

Input

```
397
79 100 86 100 85 99 80 98 80 99 88 99 79 98 89
```

Output

```
Winners : 76
```

Weightage - 20

Sample Input

Sample Output

```
10
89  100  75  98  90  100  100  98  83  99
```

```
Winners : 3
```

Solution

```
import java.util.Scanner;
import java.util.Vector;

class Attendance{
    static int winners = 0;
    static int i=0;
    static Vector<Integer> arr = new Vector<>();

    static synchronized void counting(int n){
        //System.out.println(Thread.currentThread().getName() + " is counting");
        for( ; i < n; i++){
            if(arr.get(i) == 100) {
                //System.out.println(i);
                winners++;
            }
        }
    }
}

class count extends Thread{
    int n;

    count(int n){
        this.n = n;
    }
    public void run(){
        Attendance.counting(n);
    }
}

class Honors {
    public static void main(String[] args) {

        //int max = 100;
       // int min = 75;
        Scanner myObj =new Scanner(System.in);
 int m = Integer.parseInt(myObj.nextLine());
 int tc1=m/2;
// int tc2=m-tc1;

 for(int i = 0; i < m; i++){
     int n = myObj.nextInt();
     Attendance.arr.add(n);
 }

 Thread t1 = new count(tc1);
 Thread t2 = new count(m);
 t1.start();
 t2.start();

 try{
     t1.join();
     t2.join();
 }catch(InterruptedException e) {
     System.out.println(e);
 }

// System.out.println(Attendance.arr);
```

```
        System.out.println("Winners : " + Attendance.winners);

    }
}
```

Q3

Test Case

Input

```
3
Book stall
Stall for books
25
```

Output

```
3750.0
4500.0
9000.0
```

Weightage - 20

Input

```
3
Book stall
Stall for books
30
```

Output

```
4500.0
6000.0
7500.0
```

Weightage - 20

Input

```
4
Book stall
Stall for books
30
```

Output

```
4500.0
9750.0
6000.0
7500.0
```

Weightage - 20

Input

```
5
Book stall
Stall for books
30
```

Output

```
4500.0
5700.0
9750.0
6000.0
```

Weightage - 20

Input

```
5
Book stall
Stall for books
30
```

Output

```
4500.0
5700.0
9750.0
6000.0
```

Weightage - 20

Sample Input

```
3
Book stall
Stall for books
25
```

Sample Output

```
3750.0
4500.0
9000.0
```

Solution

```
import java.io.*;
import java.util.*;
class Stall implements Runnable  {
```

```
private String stallName;
private String details;
private double stallArea;
private String owner;
private double stallCost;

public Stall(String stallName, String details, double stallArea, String owner) {
    this.stallName = stallName;
    this.details = details;
    this.stallArea = stallArea;
    this.owner = owner;
}
public Stall() {
    this.stallName = null;
    this.details = null;
    this.stallArea = 0;
    this.owner = null;
}
@Override
public void run() {
    this.stallCost = this.stallArea * 150.0;
    System.out.println(this.stallCost);
}

public String getStallName() {
    return stallName;
}

public void setStallName(String stallName) {
    this.stallName = stallName;
}

public String getDetails() {
    return details;
}

public void setDetails(String details) {
    this.details = details;
}

public double getStallArea() {
    return stallArea;
}

public void setStallArea(double stallArea) {
    this.stallArea = stallArea;
}

public String getOwner() {
    return owner;
}

public void setOwner(String owner) {
    this.owner = owner;
}

public double getStallCost() {
    return stallCost;
}

public void setStallCost(double stallCost) {
    this.stallCost = stallCost;
}
```

```
}
class Main {
public static void main(String [] args) {
    int n,i;
    Scanner sc = new Scanner(System.in);
    n = Integer.parseInt(sc.nextLine());
    for(i=0;i<n;i++) {
        Stall st = new Stall();
        st.setStallName(sc.nextLine());
        st.setDetails(sc.nextLine());
        st.setStallArea(Double.parseDouble(sc.nextLine()));
        st.setOwner(sc.nextLine());
        Thread s = new Thread(st);
        s.start();

    }
}
}
```

Q4

Test Case

Input

```
4
Le Meridian
12000
400
```

Output

```
Profit
Loss
Profit
Profit
```

Weightage - 20

Input

```
3
Le Meridian
10000
4000
```

Output

```
Profit
Profit
Profit
```

Weightage - 20

Input

```
5
Le Meridian
12000
400
```

Output

```
Profit
Loss
Profit
Profit
```

Weightage - 20

Input

```
2
Vivanta taj
25000
1000
```

Output

```
Loss
Loss
```

Weightage - 20

Input

```
3
Orbis
20000
1000
```

Output

```
Profit
Profit
Profit
```

Weightage - 20

Sample Input

```
4
Le  Meridian
12000
400
```

Sample Output

```
Profit
Loss
Profit
Profit
```

Solution

```
import java.io.*;
import java.util.*;
class HallBooking implements Runnable {
    public HallBooking(String hallName, double cost, int hallCapacity, int seatsBooked) {

        HallName = hallName;
        this.cost = cost;
        HallCapacity = hallCapacity;
        SeatsBooked = seatsBooked;
    }
    public HallBooking() {
        this.HallName = null;
        this.cost = 0;
        this.HallCapacity = 0;
        this.SeatsBooked = 0;
    }
    private String HallName;
    private double cost;
    private int HallCapacity;
    private int SeatsBooked;
    public String getHallName() {
        return HallName;
    }
    public void setHallName(String hallName) {
        HallName = hallName;
    }
    public double getCost() {
        return cost;
    }
    public void setCost(double cost) {
        this.cost = cost;
    }
    public int getHallCapacity() {
        return HallCapacity;
    }
    public void setHallCapacity(int hallCapacity) {
        HallCapacity = hallCapacity;
    }
    public int getSeatsBooked() {
        return SeatsBooked;
    }
    public void setSeatsBooked(int seatsBooked) {
        SeatsBooked = seatsBooked;
    }
    @Override
    public void run() {
        int res = this.SeatsBooked* 100;
        if(res > cost) {
            System.out.println("Profit");
        }
        else
            System.out.println("Loss");
```

```
    }
}
class Main {
public static void main(String [] args) {
    int i,n;
    Scanner sc = new Scanner(System.in);
    n = Integer.parseInt(sc.nextLine());
    for(i=0;i<n;i++) {
        HallBooking h = new HallBooking();
        h.setHallName(sc.nextLine());
        h.setCost(Double.parseDouble(sc.nextLine()));
        h.setHallCapacity(Integer.parseInt(sc.nextLine()));
        h.setSeatsBooked(Integer.parseInt(sc.nextLine()));
        Thread t = new Thread(h);
        t.start();
    }
}
}
```

Q5

Test Case

Input

```
3 3
3 3
1 2 3
4 5 6
```

Output

```
10 23 48
22 62 117
34 101 186
```

Weightage - 20

Input

```
2 2
2 2
1 2
3 4
```

Output

```
19 22
43 50
```

Weightage - 20

Input

```
2 2
11 2
3 41
```

Output

```
89 226
712 3298
```

Weightage - 20

Input

```
2 2
2 2
90 2
3 14
```

Output

```
464 5416
113 292
```

Weightage - 20

Input

```
2 2
2 2
14 34
38 49
```

Output

```
3148 3690
5428 6423
```

Weightage - 20

Sample Input

```
2 2
2 2
1 2
3 4
```

Sample Output

```
19 22
43 50
```

Solution

```
import java.util.Scanner;
class Main {
    public Main()
    {
       Scanner cin=new Scanner(System.in);
       A_row=cin.nextInt();
       A_col=cin.nextInt();
       B_row= cin.nextInt();
       B_col=cin.nextInt();
if(A_col==B_row)
 {
     matrix_A=new int[A_row][A_col];
     matrix_B=new int[B_row][B_col];
     mult_ans=new int[A_row][B_col];

      for(int i=0;i<A_row;i++)
          for(int j=0;j<A_col;j++)
          {
             matrix_A[i][j]=cin.nextInt();
          }

     for(int i=0;i<B_row;i++)
          for(int j=0;j<B_col;j++)
          {
             matrix_B[i][j]=cin.nextInt();
          }
      thread_pool=new mythread[A_row];

     for(int i=0;i<A_row;i++)
     {
         thread_pool[i]=new mythread(i);
             thread_pool[i].start();
           }

            printer_thread=new threadprint[3];

               printer_thread[2]=new threadprint(mult_ans);
               printer_thread[2].start();

}
        else
        {
           System.out.print("Invalid size");
        }

    }

    public static void main(String[] args) {

new Main();
  }
private class mythread extends Thread
    {
```

```
    int index;

     mythread(int index)
     {
         this.index=index;
     }
    public void run()
    {
        for(int i=0;i<B_col;i++)
        {
            for(int j=0;j<B_col;j++)
            {
                mult_ans[index][i]+=matrix_A[index][j]*matrix_B[j][i];
            }
        }
    }
}
private class threadprint extends Thread
{
    threadprint (int[][] matrix)
    {
        this.matrix=matrix;
    }
    public synchronized  void run()
    {
        for(int i=0;i<matrix.length;i++)
        {
            //System.out.print("|");
            for(int j=0;j<matrix[0].length;j++)
            {
              System.out.print(matrix[i][j]+" ");
            }
            System.out.println();
            //.out.println("|");
        }
    }
    int[][] matrix;
}

private int A_row;
private int A_col;
private int B_row;
private int B_col;
private int[][] matrix_A;
 private int[][] matrix_B;
 private int[][] mult_ans;
 private mythread[] thread_pool;
 private threadprint[] printer_thread;
}
```

Q6

Test Case

Input	Output
2 3256858548 50000 mahesh	250.00 4000.00 5250.00 54000.00

Weightage - 20

Input	Output

3 5325856532 25000 Alice	125.00 3840.00 2000.00 51840.00

Weightage - 20

Input **Output**

4 2545256585 65000 Alice	6880.00 325.00 5200.00 400.00

Weightage - 20

Input **Output**

5 2545256585 65000 Alice	50.00 6900.00 321.40 424.20

Weightage - 20

Input **Output**

6 2545256585 65000 Alice	5200.00 50.00 4678.56 6900.00

Weightage - 20

Sample Input **Sample Output**

2 3256858548 50000 mahesh	250.00 4000.00 5250.00 54000.00

Solution

```
import java.io.*;
import java.text.DecimalFormat;
import java.util.*;
class Account extends Thread {
    private String accountNumber;
    private double balance;
    private String accountHoldername;
    public String getAccountNumber() {
        return accountNumber;
    }
    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }
    public double getBalance() {
        return balance;
    }
    public void setBalance(double balance) {
        this.balance = balance;
    }
    public String getAccountHoldername() {
        return accountHoldername;
    }
```

```
    public void setAccountHoldername(String accountHoldername) {
        this.accountHoldername = accountHoldername;
    }
    public void run() {
        double interest,amount;
        DecimalFormat d = new DecimalFormat("0.00");
        if(this.balance >= 10000) {
            interest = balance*0.08;
            amount = balance+interest;
        }
        else {
            interest = balance*0.05;
            amount = balance+interest;
        }
        System.out.println(d.format(interest));
        System.out.println(d.format(amount));
    }
}
class Main {
    public static void main(String [] args) {
        int i,n;
        Scanner sc = new Scanner(System.in);
        n = Integer.parseInt(sc.nextLine());
        for(i=0;i<n;i++) {
            Account a = new Account();
            a.setAccountNumber(sc.nextLine());
            a.setBalance(Double.parseDouble(sc.nextLine()));
            a.setAccountHoldername(sc.nextLine());
            Thread t = new Thread(a);
            a.start();
        }
    }
}
```

Q7

Test Case

Input	Output
2 Laptop 40000 30000	300000.00 400000.00

Weightage - 20

Input	Output
3 Laptop 40000 30000	300000.00 22500.00 400000.00

Weightage - 20

Input	Output
4 Laptop 40000 30000	300000.00 3500.00 22500.00 400000.00

Weightage - 20

Input

```
5
Laptop
40000
30000
```

Output

```
3500.00
12500.00
300000.00
22500.00
```

Weightage - 20

Input

```
6
Laptop
40000
30000
```

Output

```
22500.00
400000.00
12500.00
300000.00
```

Weightage - 20

Sample Input

```
2
Laptop
40000
30000
```

Sample Output

```
300000.00
400000.00
```

Solution

```
import java.io.*;
import java.text.DecimalFormat;
import java.util.*;
class ItemType extends Thread{
    private String name1;
    private double deposit;
    private double costPerItem;
    private int noOfItems;
    public String getName1() {
        return name1;
    }
    public void setName1(String name1) {
        this.name1 = name1;
    }
    public double getDeposit() {
        return deposit;
    }
public void setDeposit(double deposit) {
    this.deposit = deposit;
}
public double getCostPerItem() {
    return costPerItem;
}
public void setCostPerItem(double costPerItem) {
    this.costPerItem = costPerItem;
}
public ItemType() {
    this.name1 = null;
    this.deposit =0;
    this.costPerItem = 0;
}
public ItemType(String name1,double deposit,double costPerItem) {
    this.name1 = name1;
    this.deposit= deposit;
    this.costPerItem = costPerItem;
}
public int getNoOfItems() {
```

```
        return noOfItems;
    }
    public void setNoOfItems(int noOfItems) {
        this.noOfItems = noOfItems;
    }
    public void run() {
        double res = this.costPerItem*this.noOfItems;
        DecimalFormat d = new DecimalFormat("0.00");
        System.out.println(d.format(res));
    }
}
class Main {
public static void main(String [] args) {
    int i,n;
    Scanner sc = new Scanner(System.in);
    n = Integer.parseInt(sc.nextLine());
    for(i=0;i<n;i++) {
        ItemType it = new ItemType();
        it.setName1(sc.nextLine());
        it.setDeposit(Double.parseDouble(sc.nextLine()));
        it.setCostPerItem(Double.parseDouble(sc.nextLine()));
        it.setNoOfItems(Integer.parseInt(sc.nextLine()));
        Thread t = new Thread(it);
        t.start();
     }
}
}
```

Q8

Test Case

Input

```
2
test
abc
```

Output

```
t2
e1
s1
```

Weightage - 100

Sample Input

```
2
welcome
java
```

Sample Output

```
w1
e2
l1
```

Solution

```
import java.util.Scanner;

class Mythread extends Thread{
    String str;
    static int n;
    public Mythread(String str) {
        super();
        this.str = str;
    }

    public void run() {
         n = this.str.length();
        int[] freq = new int[26];

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

```
            freq[this.str.charAt(i) - 'a']++;

        for (int i = 0; i < n; i++) {
            if (freq[this.str.charAt(i) - 'a'] != 0) {
                System.out.print(this.str.charAt(i));
                System.out.print(freq[this.str.charAt(i) - 'a']);
                System.out.println();
              freq[this.str.charAt(i) - 'a'] = 0;
            }
    }

}
}
class MainThread{
    public static void main(String args[]) {
        Scanner sc =new Scanner(System.in);
        int n=Integer.parseInt(sc.nextLine());
        for(int i=0;i<n;i++) {
            String str= sc.nextLine();
            Mythread t2 =new Mythread(str);
            t2.start();
            System.out.println();
        }


    }
}
```