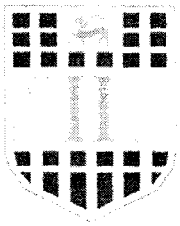

Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology

Specialized in Information Technology

Final Examination
Year 2, Semester I (2019)

IT2030 – Object Oriented Programming

Paper B

Duration: 3 Hours

October 2019

Instructions to Candidates:

- This paper contains **Four** questions. **Answer All** Questions.
- Fill **Student Details** in the last page.
- Marks for each question are given in the paper.
- Total Marks is 100.
- Create a separate Project for each question. The name of the project is provided. Save each Java program using the class name given.
- Store all your program files in the Desktop Folder provided.
- This paper contains 11 pages with the Cover Page.

Question 1 (30 marks)

This question is based on the **Object-Oriented Programming (OOP) concepts**. You are going to control two types of sensors called Humidity Sensor and Rain Fall Sensor using one Remote Controller device.

a) You can refer the output is given in **SensorDemo** class and adjust your code accordingly

```
public class SensorDemo {

    public static void main(String[] args) {
        ISensor humiditySensor = new HumiditySensor("Humidity");
        IMotionTracker humidityTracker = new SensorLocation("Colombo");
        ISensor rainFallSensor = new RainFallSensor("RainFall");
        IMotionTracker rainFallTracker = new SensorLocation("Kandy");

        ISensor [] sensorArray = new ISensor[]{humiditySensor, rainFallSensor};
        IMotionTracker [] trackerArray = new IMotionTracker[]{humidityTracker, rainFallTracker};

        RemoteController remoteController = new RemoteController(0, sensorArray,trackerArray);
        remoteController.startService();
        remoteController.stopService();
        remoteController.locationService();

        RemoteController remoteController2 = new RemoteController(1, sensorArray,trackerArray);
        remoteController2.startService();
        remoteController2.stopService();
        remoteController2.locationService();
    }
}
```

```
Problems  Console  Servers  Data Source Explorer
<terminated> SensorDemo (1) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Sep 3, 2019, 9:25:05 AM)
Humidity sensor switch on
Humidity sensor switch off
Sensor Location is = Colombo

RainFall sensor switch on
RainFall sensor switch off
Sensor Location is = Kandy
```

i). First implement the **ISensor** interface and declare **on()** and **off()** methods.

(03 marks)

ii). Then implement the **IMotionTracker** interface and declare the method called **displayLocation()**

(02 marks)

iii). Create two classes called **HumiditySensor** and **RainFallSensor** and implement the **ISensor** interface in each class and override necessary methods in each. You should overload the constructor to pass the name of the sensor in both classes.

(4 x 2 = 08 marks)

iv). Similarly create a class called **SensorLocation** and implement the **IMotionTracker** interface with in the class and **override the displayLocation()** method. Then overload the constructor to pass the location of the satellite.

(03 marks)

b) Remote controller maintains multiple sensors and multiple motion trackers. To on and off sensors and trackers switches will be used.

i). Create the **RemoteController** class and implement the properties **Switch(int),** and array of **ISensor (ISensor [ ])** and the array of **IMotionTracker (IMotionTracker [ ])** tracker.

(02 marks)

ii). Overload the constructor of the same class and initialize the above properties.

(03 marks)

iii). Implement the method called **startService()** and you should invoke the **on()** method of the sensor class by using the switch. [E.g.: - if Switch = 0 it will turn off the Humidity Sensor and if Switch = 1 it will turn on the humidity sensor]

(02 marks)

iv). Implement the method called **stopService()** and you should invoke the **off()** method.

(02 marks)

v). Then develop the **locationService()** method and based on the given option tracker should invoke the **displayLocation()** method

(02 marks)

vi). Extends the **SensorDemo** class by adding another Rain Fall Sensor and the tracker. Display your modified output again in the console

(03 marks)

Save the project as **Paper01B**

Question 2 (20 marks)

This question is based on the **Threads** implementation.

a) You are going to implement two threads to add numbers and multiply numbers called **AddNumbers ("ADD")** and **MultiplyNumbers("MUL")** respectively. **DemoThread** class is given as below and both Threads should execute one after the other for the given range and check the given output to make your implementation ease.
[**Assumption: - Thread synchronization is essential and both threads should print the output as synchronized manner. Correct implementation of *wait()*, *notify()* methods are compulsory to obtain full marks**]

```
public class DemoThreads {

    public static void main(String[] args) {

        DemoThreads demoThread = new DemoThreads();
        Thread addNumbers = new Thread(new AddNumbers(demoThread, 10, 20), "ADD");
        Thread multiplyNumbers = new Thread(new MultiplyNumbers(demoThread, 10, 20), "MUL");
        addNumbers.start();
        multiplyNumbers.start();
    }
}
```

Console

```
<terminated> DemoThreads [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (Sep 3, 2019, 11:13:04 AM)
MUL => 10 (*) 10 = 100
ADD => 10 (+) 10 = 20
MUL => 11 (*) 11 = 121
ADD => 11 (+) 11 = 22
MUL => 12 (*) 12 = 144
ADD => 12 (+) 12 = 24
MUL => 13 (*) 13 = 169
ADD => 13 (+) 13 = 26
MUL => 14 (*) 14 = 196
ADD => 14 (+) 14 = 28
MUL => 15 (*) 15 = 225
ADD => 15 (+) 15 = 30
MUL => 16 (*) 16 = 256
ADD => 16 (+) 16 = 32
MUL => 17 (*) 17 = 289
ADD => 17 (+) 17 = 34
MUL => 18 (*) 18 = 324
ADD => 18 (+) 18 = 36
MUL => 19 (*) 19 = 361
ADD => 19 (+) 19 = 38
MUL => 20 (*) 20 = 400
ADD => 20 (+) 20 = 40
```

i). You have to overload the **AddNumbers** constructor with a **DemoThread** object (for synchronization), **begin** and **end** parameters.

(01 mark)

ii). Implement a method called **addNumbers(DemoThreads demoThread, int begin, int end)** and pass parameters which are passed through the overloaded constructor.

(05 marks)

iii). In each iteration the Thread should **sleep 1 second** of time interval and it should print the thread name and given values as per the given output.

(02 marks)

iv). Override the **run()** method and call the **addNumbers** method within that.

(02 marks)

b) **MultiplyNumbers** should print the values as per the given console output and use iterator to limit the begin and the end to be printed with displaying the name of currently running thread. [**Hint: - Thread.currentThread.getName()**]

i). You have to overload the **MultiplyNumbers** constructor with a lock (for synchronization), **begin** and **end** parameters.

(01 mark)

ii). Implement a method called **multiplyNumbers(DemoThreads demoThread, int begin, int end)** and pass parameters which are passed through the overloaded constructor.

(05 marks)

iii). In each iteration the Thread should **sleep 1 second** of time interval and it should print the thread name and given values as per the given output.

(02 marks)

iv). Override the **run()** method and call the **multiplyNumbers** method within that.

(02 marks)

Save the project as **Paper02B**

Question 3 (20 marks)

This question is based on the **Collection Framework and Generics**.

a) You should implement an array list of Engineers and Managers and use one Generic class called **GenericEmployee** to display elements in both array lists. Please refer the **GenericEmployeeDemo** Test class and its execution output to fine-tune your results.

```
15 public class GenericEmployeeDemo {
16
17     public static void main(String[] args) {
18         ArrayList<Engineer> engineers = new ArrayList<>();
19         engineers.add(new Engineer("E0000", "IFS"));
20         engineers.add(new Engineer("E1111", "Virtusa"));
21         engineers.add(new Engineer("E2222", "99x"));
22         engineers.add(new Engineer("E3333", "Cambio"));
23         engineers.add(new Engineer("E4444", "CodeGen"));
24
25         ArrayList<Manager> managers = new ArrayList<>();
26         managers.add(new Manager("MGD5555", 250000.00));
27         managers.add(new Manager("MG4444", 225000.00));
28         managers.add(new Manager("MG3333", 175000.00));
29         managers.add(new Manager("MG2222", 200000.00));
30         managers.add(new Manager("MG1111", 150000.00));
31
32         GenericEmployee genericEmployee = new GenericEmployee();
33         genericEmployee.showElements(managers);
34         genericEmployee.showElements(engineers);
35     }
36 }
```

Console

```
<terminated> GenericEmployeeDemo [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw
Manager ID = MGD5555, Salary = 250000.0
Manager ID = MG4444, Salary = 225000.0
Manager ID = MG3333, Salary = 175000.0
Manager ID = MG2222, Salary = 200000.0
Manager ID = MG1111, Salary = 150000.0

Engineer = E0000, Company = IFS
Engineer = E1111, Company = Virtusa
Engineer = E2222, Company = 99x
Engineer = E3333, Company = Cambio
Engineer = E4444, Company = CodeGen
```

i). Implement an interface **IEmployee** and declare the method **showEmployeeDetails()** should return the output in **String** type.

(02 marks)

ii). Create a class called **Manager** and implement the two properties called **managerID** (String) and **salary** (double) and values should be assigned through the **overloaded constructor**.

(02 marks)

iii). Implement the **IEmployee** interface in the **Manager** class and override the method **showEmployeeDetails ()** to print the manager ID and the salary.

(02 marks)

iv). Create a class called **Engineer** and implement the two properties called **employeeID** (String) and **company** (String) and the values should be assigned through the **overloaded constructor**.

(02 marks)

v). Implement the **IEmployee** interface in the **Engineer** class and override the method **showEmployeeDetails ()** to print the employee ID and the company.

(02 marks)

vi). Now create the generic class called **GenericEmployee** and implement the method **showElements** should support passing **generic array list** (either Engineers array list or Managers array list). The **showElements ()** method should have an iteration and within the iteration, the each element should call the **showEmployeeDetails**() method to print the Engineer and Manager details as per the given output.

(05 marks)

b) You should create a class called **AscendingList** and that should store list of elements. Elements should be stored according to the **Ascending order** and it should **remove all duplicate elements** as well.

i). Implement the method called **displayMyList()** it should print elements according to the ascending order. Refer the **GenericTest** class and the console output to adjust your results accordingly

(05 marks)

```
public class GenericTest {

    public static void main(String[] args) {
        AscendingList<Integer> ascendingList = new AscendingList<>();
        ascendingList.add(80);
        ascendingList.add(80);
        ascendingList.add(70);
        ascendingList.add(50);
        ascendingList.add(10);
        ascendingList.add(20);
        ascendingList.add(10);
        ascendingList.add(50);

        AscendingList<String> ascendingList2 = new AscendingList<>();
        ascendingList2.add("aaa");
        ascendingList2.add("bbb");
        ascendingList2.add("ddd");
        ascendingList2.add("bbb");
        ascendingList2.add("ddd");
        ascendingList2.add("ccc");

        ascendingList.displayMyList(ascendingList);
        ascendingList2.displayMyList(ascendingList2);
    }
}
```

```
<terminated> GenericTest [
10
20
50
70
80

aaa
bbb
ccc
ddd
```

Save the project as **Paper03B**

Question 4 (30 marks)

This question is based on the **Design Patterns** implementation.

a) You are going to implement the Strategy Design Pattern based on the university degree programs (**PhDPrograms, MScPrograms, and BScPrograms**) with cost of **(6000000.00/=, 500000.00/=, and 120000.00/=**).

i). Implement two interfaces **IFestival** and **IPrograms**. Each interface you should declare methods (in **IFestival** interface declare the method **void performEvent**() and **double getBudget()** and in **IPrograms** interface declare methods **void offerPrograms(), double getCost())**

(02 marks)

ii). Then create 3 classes **RoboFest**, **GameFest**, and **CodeFest** and those classes should implement the **IFestival** interface and override all methods with in the class.

(06 marks)

iii). Similarly create another 3 classes **PhDPrograms**, **MScPrograms**. and **BScPrograms** and those classes should implement the **IPrograms** interface and override the methods as well.

(06 marks)

iv). Create an **Abstract** class **Students** and aggregate two interfaces (**IFestival,** and **IPrograms),** you should set those two behaviors with using two set methods **setFestival ( )** and **setPrograms( )**. (Those "set" methods are used to dynamically add festivals and degree programs features to Students)

(06 marks)

b) Now for the above two student types you can add different events such as **RoboFest, CodeFest**, and **GameFest** and the budget for the each event respectively RoboFest - 800000.00/=, CodeFest – 300000.00/=, and GameFest – 400000.00/= rupees. Based on the event budget should be different and **assume you can't add more than one event for Student**.

i). Then implement another two methods called **offerPrograms ()**, and **conductEvents ()** and you should call relevant **offerPrograms ()** and **performEvent ()** method respectively through the declared interfaces of the Student class

(02 marks)

ii). Apart from that within the **Students** class you should add two **abstract** methods **displayStudents() and displayCost()**

(01 mark)

iii). Now **extends** the **Students** class in the **UndergraduateStudents, PostGraduateStudents** classes and implement all abstract methods. Within the **displayStudents()** method you should call for the **offerPrograms (), conductEvents (), and displayCost () methods.**

(07 marks)

iv). Please refer the **output of the test class** when you run. Make sure you got the same output.

```
public class TestStratergy {

    public static void main(String [] args){

        Students poStudents = new PostGraduateStudents();
        poStudents.setFestival(new CodeFest());
        poStudents.setPrograms(new MScPrograms());
        poStudents.displayStudents();

        System.out.println();

        Students unStudents = new UndergraduateStudents();
        unStudents.setFestival(new RoboFest());
        unStudents.setPrograms(new BScPrograms());
        unStudents.displayStudents();
    }
}
```

```
Offer MSc Programs
Perform CodeFest Event for 300000.0
Cost for the postgraduate program is = 500000.0
Display Post gratuate students

Offer BSc degree programs
Perform Robo Fest Event for 600000.0
Cost for the undergraduate program is = 120000.0
Display under gratuate students
```

Save the project as **Paper04B**

COMPULSORY TO FILL BEFORE STARTING THE EXAM

Student ID :

Student Name:-

Machine No :-

Machine IP Address :-

Location :-

Question Number	Marks
Q1	
Q2	
Q3	
Q4	
TOTAL	

Evaluated Lecturer :- .....

Signature:- .....

End of The Examination Paper