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 A

Duration: 3 Hours

October 2019

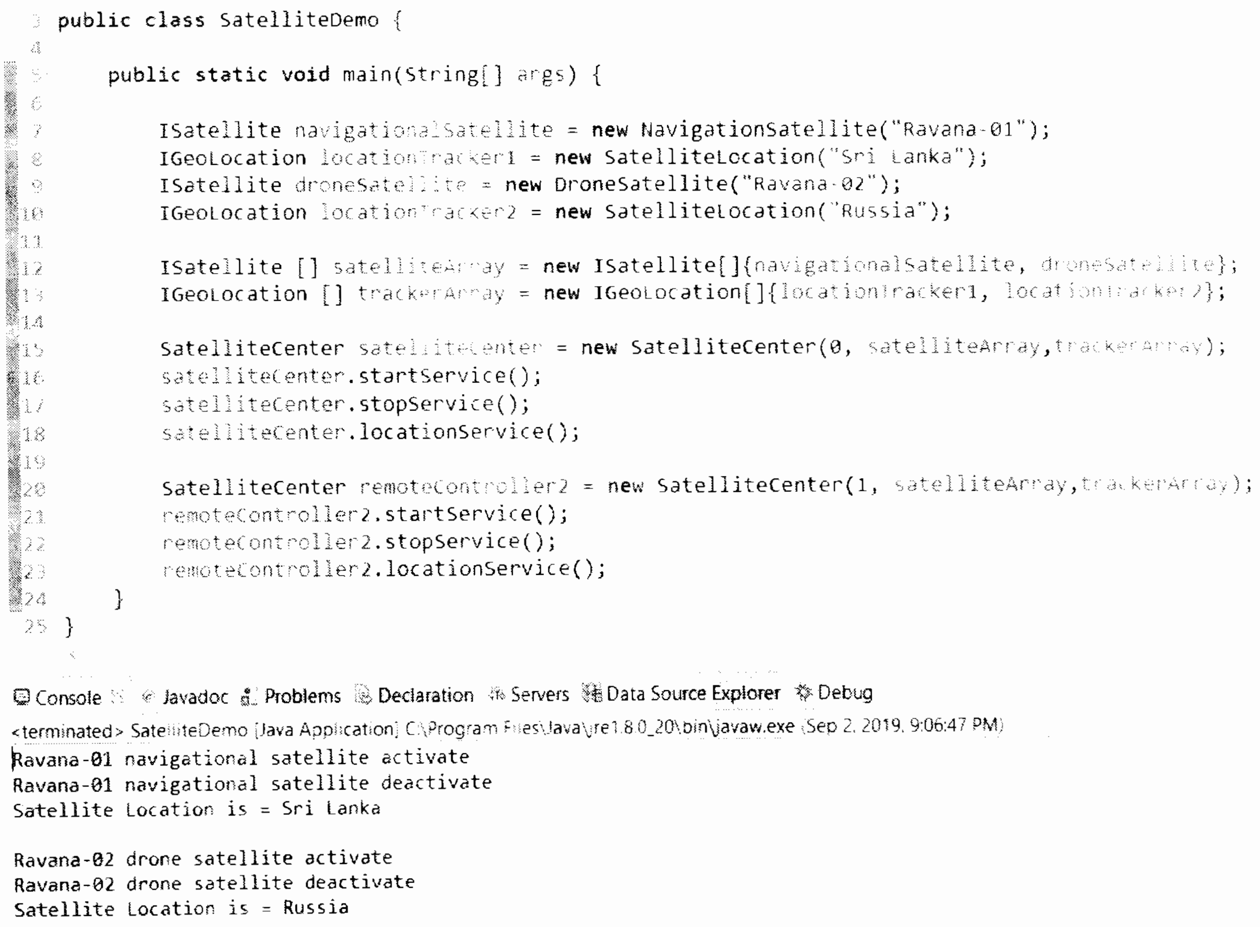
Instructions to Candidates:

- 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 satellites called Drone Satellite and Navigational Satellite from one location called Satellite Center.

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



```
public class SatelliteDemo {

    public static void main(String[] args) {

        ISatellite navigationalSatellite = new NavigationSatellite("Ravana-01");
        IGeoLocation locationTracker1 = new SatelliteLocation("Sri Lanka");
        ISatellite droneSatellite = new DroneSatellite("Ravana-02");
        IGeoLocation locationTracker2 = new SatelliteLocation("Russia");

        ISatellite [] satelliteArray = new ISatellite[]{navigationalSatellite, droneSatellite};
        IGeoLocation [] trackerArray = new IGeoLocation[]{locationTracker1, locationTracker2};

        SatelliteCenter satelliteCenter = new SatelliteCenter(0, satelliteArray,trackerArray);
        satelliteCenter.startService();
        satelliteCenter.stopService();
        satelliteCenter.locationService();

        SatelliteCenter remoteController2 = new SatelliteCenter(1, satelliteArray,trackerArray);
        remoteController2.startService();
        remoteController2.stopService();
        remoteController2.locationService();
    }
}
```

```
<terminated> SatelliteDemo [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Sep 2, 2019, 9:06:47 PM)
Ravana-01 navigational satellite activate
Ravana-01 navigational satellite deactivate
Satellite Location is = Sri Lanka

Ravana-02 drone satellite activate
Ravana-02 drone satellite deactivate
Satellite Location is = Russia
```

i). First implement the **ISatellite** interface and declare **activate()** and **deactivate()** methods.
(03 marks)

ii). Then implement the **IGeoLocation** interface and declare the method called **displayLocation()**
(02 marks)

iii). Create two classes called **DroneSatellite** and **NavigationSatellite** and implement the **ISatellite** interface in each class and override necessary methods in each. You should overload the constructor to pass the name of the satellite in both classes.
(4 x 2 = 08 marks)

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

(03 marks)

b) Satellite center maintain multiple satellites and multiple Geo Location trackers. To activate each satellite and the tracker the option can be used as a switch.

i). Create the **SatelliteCenter** class and implement the properties **option(int),** and array of **ISatellite (ISatellite [ ])** and the array of **IGeoLocation (IGeoLocation [ ])** 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 **activate()** method of the satellite class by using the option as switch. [E.g.: - if option = 0 activate Navigation Satellite if option = 1 activate drone satellite]

(02 marks)

iv). Implement the method called **stopService()** and you should invoke the **deactivate()** 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 **SatelliteDemo** class by adding another Drone Satellite and the tracker. Display your modified output again in the console

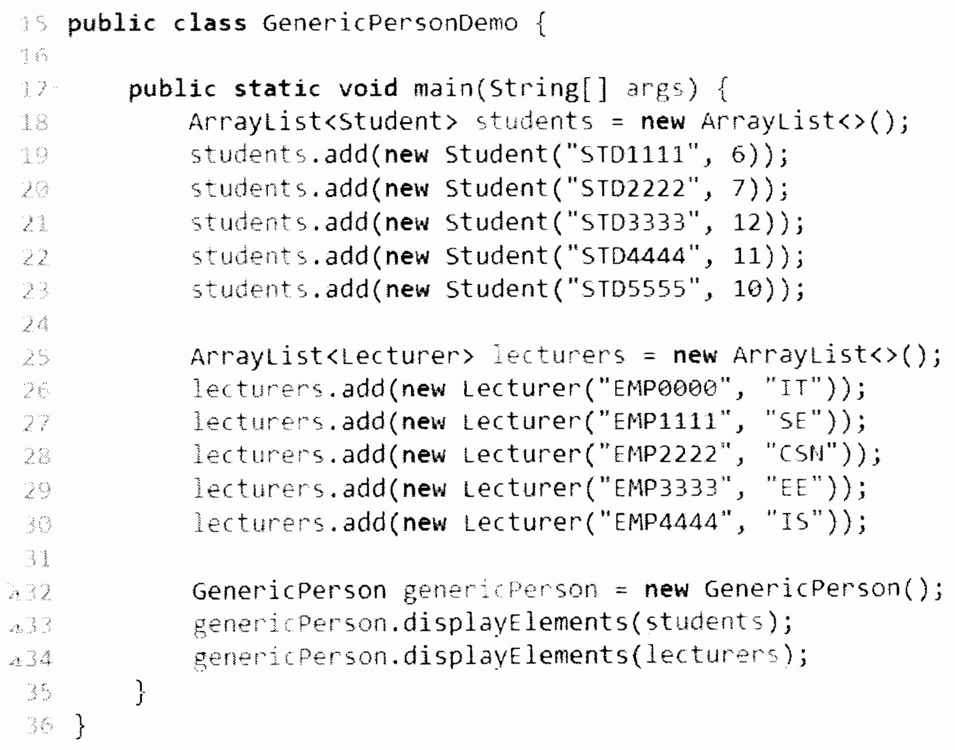
(03 marks)

Save the project as **Paper01A**

Question 2 (20 marks)

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

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

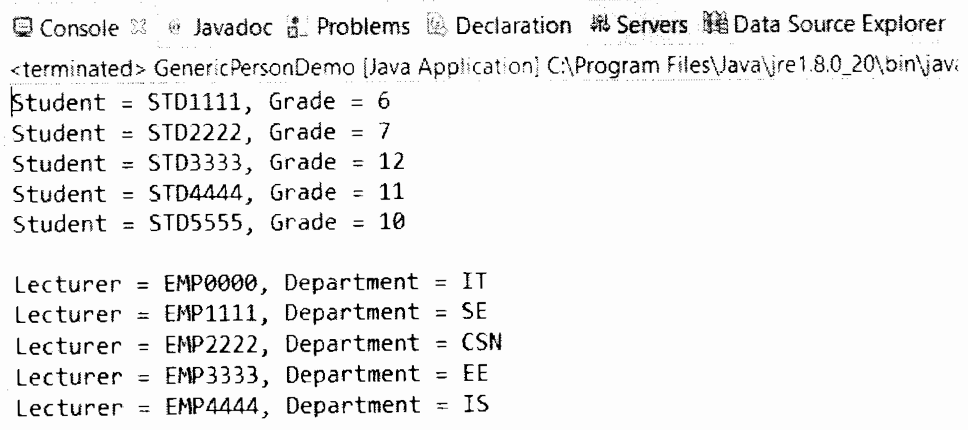


```java
public class GenericPersonDemo {

    public static void main(String[] args) {
        ArrayList<Student> students = new ArrayList<>();
        students.add(new Student("STD1111", 6));
        students.add(new Student("STD2222", 7));
        students.add(new Student("STD3333", 12));
        students.add(new Student("STD4444", 11));
        students.add(new Student("STD5555", 10));

        ArrayList<Lecturer> lecturers = new ArrayList<>();
        lecturers.add(new Lecturer("EMP0000", "IT"));
        lecturers.add(new Lecturer("EMP1111", "SE"));
        lecturers.add(new Lecturer("EMP2222", "CSN"));
        lecturers.add(new Lecturer("EMP3333", "EE"));
        lecturers.add(new Lecturer("EMP4444", "IS"));

        GenericPerson genericPerson = new GenericPerson();
        genericPerson.displayElements(students);
        genericPerson.displayElements(lecturers);
    }
}
```



```
<terminated> GenericPersonDemo [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\jav
Student = STD1111, Grade = 6
Student = STD2222, Grade = 7
Student = STD3333, Grade = 12
Student = STD4444, Grade = 11
Student = STD5555, Grade = 10

Lecturer = EMP0000, Department = IT
Lecturer = EMP1111, Department = SE
Lecturer = EMP2222, Department = CSN
Lecturer = EMP3333, Department = EE
Lecturer = EMP4444, Department = IS
```

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

(02 marks)

ii). Create a class called **Student** and implement the two properties called **studetID** (String) and **grade** (int) and values should be assigned through the **overloaded constructor**.

(02 marks)

iii). Implement the **IPerson** interface in the **Student** class and override the method **displayDetails()** to print the student ID and the grade.

(02 marks)

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

(02 marks)

v). Implement the **IPerson** interface in the **Lecturer** class and override the method **displayDetails()** to print the **employeeID** and the **department**.

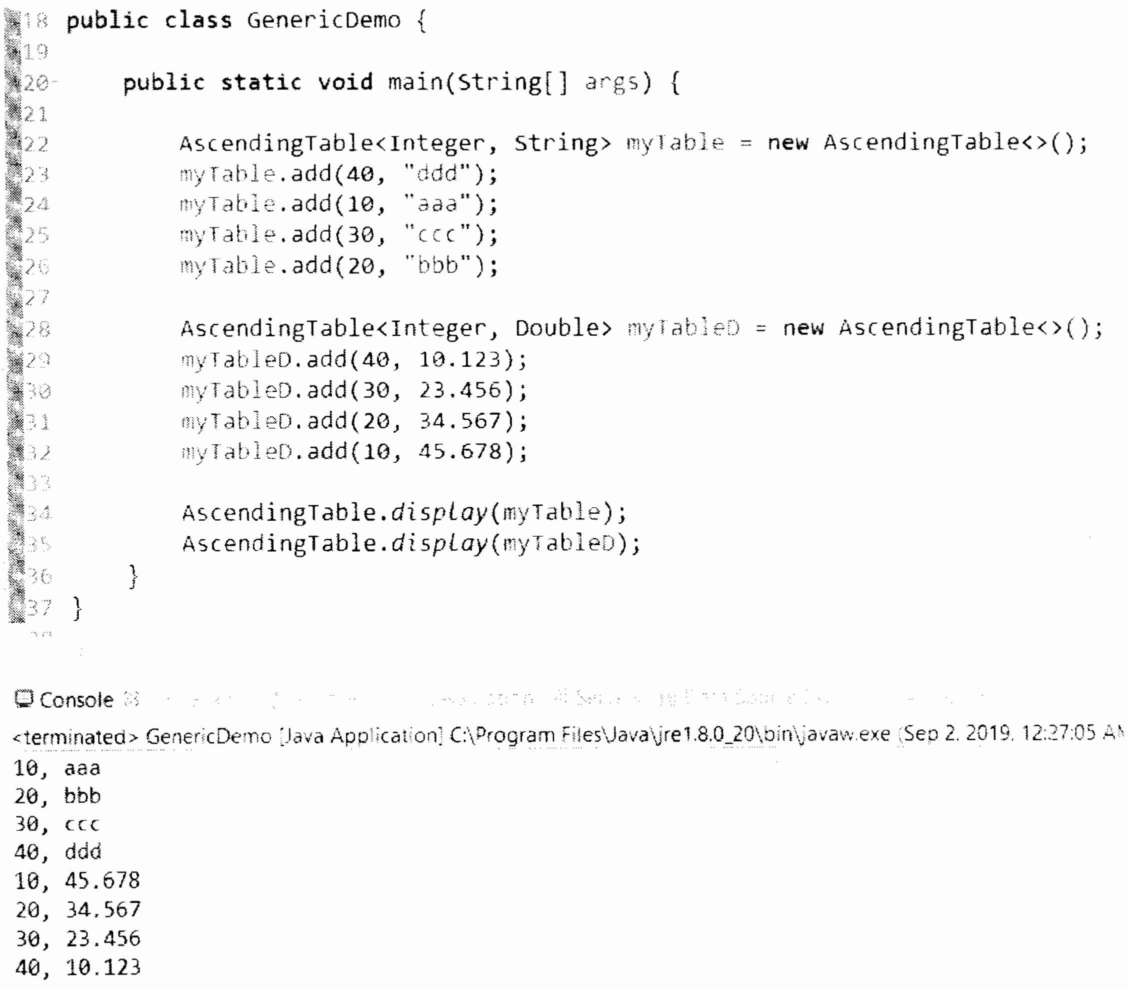
(02 marks)

vi). Now create the generic class called **GenericPerson** and implement the method **displayElements** should support passing **generic array list** (either Lecturers array list or Students array list). The **displayElements()** method should have an iteration and within the iteration, the each element should call the **displayDetails()** method to print the Lecturer and Student details as per the given output.

(05 marks)

b) You should create a class called **AscendingTable** and that should store elements as key, value pairs. Keys should be stored according to the Ascending order. Implement the **display()** method that should print keys and values according to the ascending order. Refer the **GenericDemo** Test class and console output to adjust your results accordingly

(05 marks)



```java
public class GenericDemo {

    public static void main(String[] args) {

        AscendingTable<Integer, String> myTable = new AscendingTable<>();
        myTable.add(40, "ddd");
        myTable.add(10, "aaa");
        myTable.add(30, "ccc");
        myTable.add(20, "bbb");

        AscendingTable<Integer, Double> myTableD = new AscendingTable<>();
        myTableD.add(40, 10.123);
        myTableD.add(30, 23.456);
        myTableD.add(20, 34.567);
        myTableD.add(10, 45.678);

        AscendingTable.display(myTable);
        AscendingTable.display(myTableD);
    }
}
```

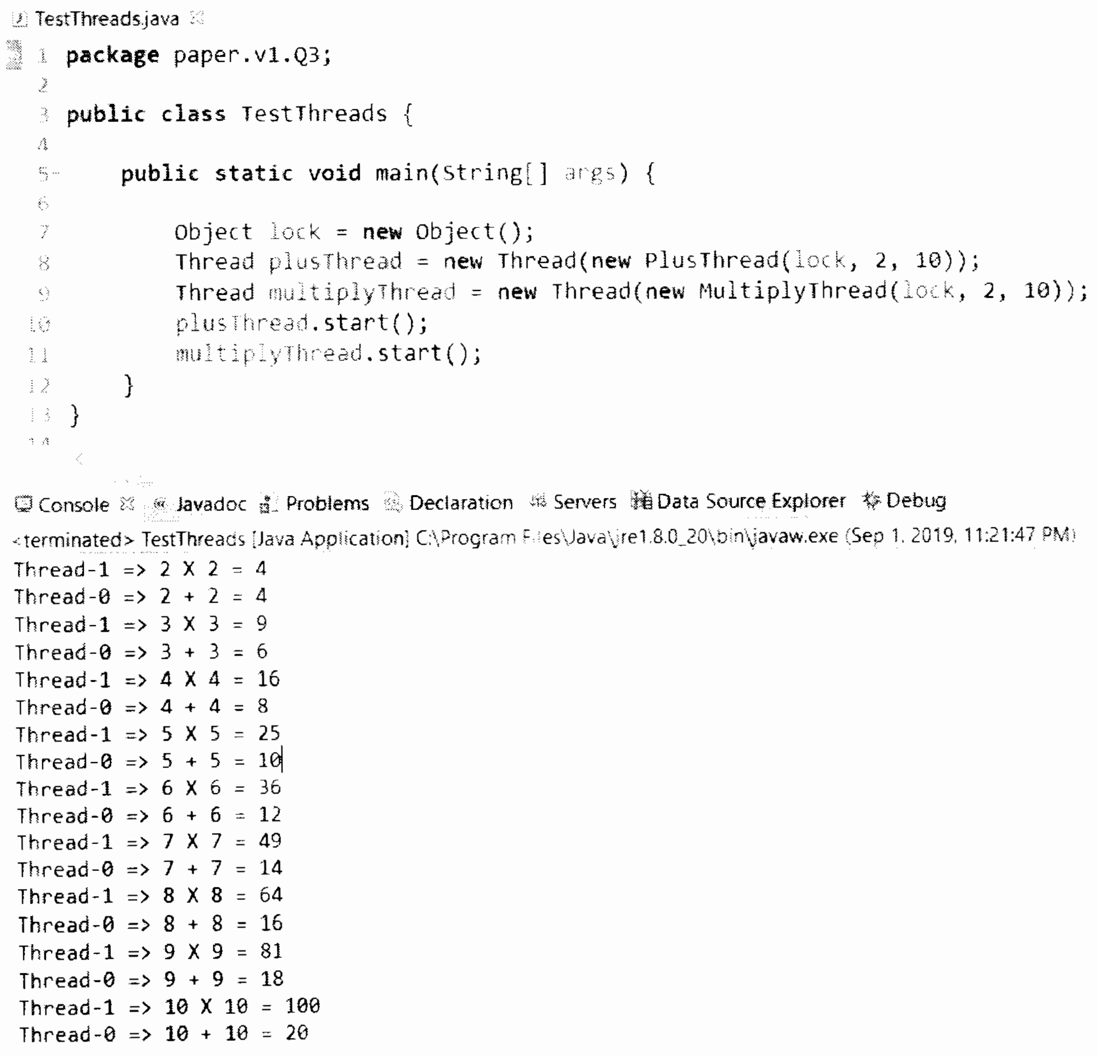
```
<terminated> GenericDemo [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Sep 2, 2019, 12:27:05 A
10, aaa
20, bbb
30, ccc
40, ddd
10, 45.678
20, 34.567
30, 23.456
40, 10.123
```

Save the project as **Paper02A**

Question 3 **(20 marks)**

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

a) You are going to implement two threads to multiply numbers and add numbers called **MultiplyThread (Thread-1)** and **PlusThread (Thread-0)** respectively. **TestThread** 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 is compulsory to obtain full marks]



```java
package paper.v1.Q3;

public class TestThreads {

    public static void main(String[] args) {

        Object lock = new Object();
        Thread plusThread = new Thread(new PlusThread(lock, 2, 10));
        Thread multiplyThread = new Thread(new MultiplyThread(lock, 2, 10));
        plusThread.start();
        multiplyThread.start();
    }
}
```

```
<terminated> TestThreads [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Sep 1, 2019, 11:21:47 PM)
Thread-1 => 2 X 2 = 4
Thread-0 => 2 + 2 = 4
Thread-1 => 3 X 3 = 9
Thread-0 => 3 + 3 = 6
Thread-1 => 4 X 4 = 16
Thread-0 => 4 + 4 = 8
Thread-1 => 5 X 5 = 25
Thread-0 => 5 + 5 = 10
Thread-1 => 6 X 6 = 36
Thread-0 => 6 + 6 = 12
Thread-1 => 7 X 7 = 49
Thread-0 => 7 + 7 = 14
Thread-1 => 8 X 8 = 64
Thread-0 => 8 + 8 = 16
Thread-1 => 9 X 9 = 81
Thread-0 => 9 + 9 = 18
Thread-1 => 10 X 10 = 100
Thread-0 => 10 + 10 = 20
```

i). You have to overload the **PlusThread** constructor with a lock (for synchronization), **start** and **range** parameters.

(01 mark)

ii). Implement a method called **addNumbers(Object lock, int start, int range)** 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) **MultiplyThread** should print the values as per the given console output and use iterator to limit the start and the range to be printed with displaying the name of currently running thread. [**Hint: - Thread.currentThread.getName()**]

i). You have to overload the **MultiplyThread** constructor with a lock (for synchronization). **start** and **range** parameters.

(01 mark)

ii). Implement a method called **multiplyNumbers(Object lock, int start, int range)** 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 **Paper03A**

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 scenario for meal preparation of a Restaurant. You can prepare three meals for the day (**Breakfast, Dinner, and Lunch**) with time duration of (**60 minutes, 45 minutes, and 30 minutes**).

i). Implement two interfaces **IPrepareQuickly** and **IPrepareDeliciously**. Each interface you should declare methods (in **IPrepareQuickly** interface declare the method **void deliveryTime()** and in **IPrepareDeliciously** interface declare methods **void addFlavour()** and **double getCost()**)

(02 marks)

ii). Then create 3 classes **ChickenFlavour**. **FishFlavour**, and **EggFlavour** and those classes should implement the **IPrepareDeliciously** interface and override all methods with in the class.

(06 marks)

iii). Similarly create another 3 classes **OneHour**, **ThirtyMinutes**, and **FourtyFiveMinutes** and those classes should implement the **IPrepareQuickly** interface and override the method as well.

(03 marks)

iv). Create an **Abstract** class **Meal** and aggregate two interfaces (**IPrepareQuickly**, and **IPrepareDeliciously), ** you should set those two behaviors with using two set methods **setFlavour()** and **setDuration()**. (Those "set" methods are used to dynamically add prepare **quickly** and **prepare deliciously** features to meal)

(06 marks)

b) Now for the above three meals you can add different flavors such as **chicken, fish**, and **egg** and the cost of each flavored meal respectively chicken - 100/=, Fish – 80/=, and egg – 60/= rupees. Based on the flavor cost should be different and **assume you can't add more than one flavor per meal.**

i). Then implement another two methods called **mealWithFlavour(),** and **mealInDuration()** and you should call relevant **addFlavour()** and **deliveryTime()** method respectively through the declared interfaces of the Meal class

(02 marks)

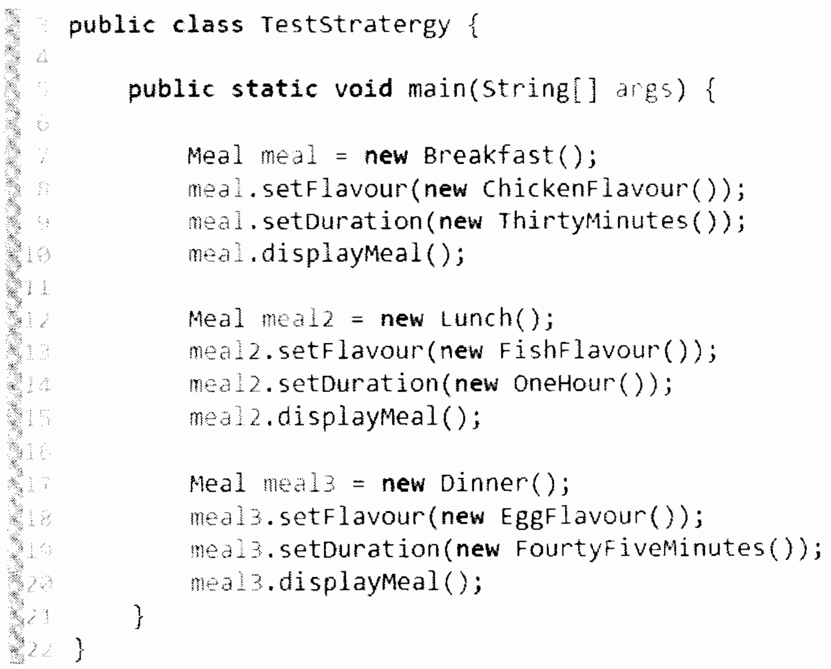
ii). Apart from that with in the **Meal** class you should add two **abstract** methods **displayMeal() and displayCost()**

(01 mark)

iii). Now **extends** the **Meal** class in the **Breakfast, Lunch and Dinner** classes. Implement all abstract methods. Within the **displayMeal()** method you should call for the **mealWithFlavour(), mealInDuration(), and displayCost() methods.**

(10 marks)

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



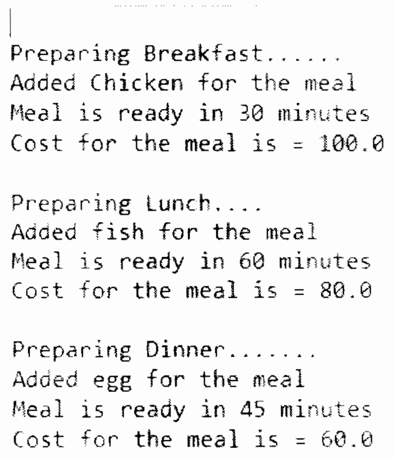
```java
public class TestStratergy {

    public static void main(String[] args) {

        Meal meal = new Breakfast();
        meal.setFlavour(new ChickenFlavour());
        meal.setDuration(new ThirtyMinutes());
        meal.displayMeal();

        Meal meal2 = new Lunch();
        meal2.setFlavour(new FishFlavour());
        meal2.setDuration(new OneHour());
        meal2.displayMeal();

        Meal meal3 = new Dinner();
        meal3.setFlavour(new EggFlavour());
        meal3.setDuration(new FourtyFiveMinutes());
        meal3.displayMeal();
    }
}
```



```
Preparing Breakfast......
Added Chicken for the meal
Meal is ready in 30 minutes
Cost for the meal is = 100.0

Preparing Lunch....
Added fish for the meal
Meal is ready in 60 minutes
Cost for the meal is = 80.0

Preparing Dinner.......
Added egg for the meal
Meal is ready in 45 minutes
Cost for the meal is = 60.0
```

Save the project as **Paper04A**

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