Birla Institute of Technology & Science – Pilani
First Semester, 2016 – 2017

Date: 1st December, 2016 (Thursday)

Comprehensive Examination (Open Book)

Course No & Name	:	CS F213 (Object-Oriented Programming)
Max. Marks	:	120 Marks
Time	:	9:00 A.M – 12:00 AM (180 Minutes)

Note: This question paper has two parts Part-A and Part-B each of 60 Marks. You have to attempt each part on a separate main answer booklet. Write 'Part-A' or 'Part-B' on the top of the Cover Page of the main Answer Booklet.

PART-A [Open Book, Expected Time: 90 Minutes]

Q(1) [Expected Time : 30 Minutes]

Read the following description about a vehicle rental company and answer the questions asked at the end.

The company rents only four wheelers. A vehicle can be of three types, a car, a mini-bus or a luxury bus. The company has defined a rental policy, which is described by "per day charge" and "per day late charge". These charges are determined based on the type of the vehicle. Each customer has a separate membership with the company through which the customer is identified. The customer can rent a vehicle by initiating a rental transaction wherein he/she is required to specifying the date of travel and number of days for which the vehicle is rented. The due date is automatically calculated from the return date and return time once the customer returns from the trip. The payment is done towards the rental transaction. The customer is also required to pay overdue charges if any.

Q1 (a) Identify the various classes and their important attributes that are required to implement the above system. Just write the final list of classes and their final attributes.

Q1 (b) Identify various relationships that exist among the classes that you have identified in Q1 (a). Represent each identified relationship (along with multiplicity or cardinality) in UML.

[20 M]

Q(2) [Expected Time : 30 Minutes]

Professor Mark Solomon designs the following classes for maintaining the records of the students that he is currently teaching. Carefully study the design and answer the question asked at the end.

```
class Address
{
        private     String     city;
        private     int        pin;
        private     String     state;
        Address(String city, int pin, String state)     { // Assume code Given }
        public String           getCity()               { return city; }
        public int              getPin()                { return pin;  }
        public string           setCity(String city )   { this.city  = city;    return city; }
        public int              setPin(String pin )     { this.pin   = pin;     return pin;  }
        public string           setState(String state ) { this.state = state;   return state; }
}// End of class Address
class Student
{
        private String name;
        private String idno;
        private Address addr;
        // Assume a suitable parametrized constructor is already given
        public String           getName()   { return name; }
        public Address          getIdno()   { return idno; }
        public String            setName(String name)  { this.name = name ; return name; }
        public String           setidnumber(String idno){ this.idno = no;       return idno; }
}// End of class student
class StudentData
```

```
{
    ArrayList<student> currentStudentList;      // This is an instance field for holding elements of type Student
    StudentData(ArrayList StudentList)          {       currentStudentList=StudentList      }
    public      ArrayList<Student> getList( )   {       return currentStudentList;          }
    public void setStudentName(String newName, int index_position)
    {
        Student st = currentStudentList(index_position++);          // First Retrieve the student from index_position
        student std = st.setName(newName);
    }
    public Student removeStudent(int position)   {// Assume implementation is already provided }
    public Teacher getTeacher(Student s) throws NoSuchStudentException
                    {// Assume implementation is already provided}
    public float getCG(Student s) throws NoSuchStudentException
                    {// Assume implementation is already provided}
}// End of class StudentData
```

Find the flaw(s) in the design of above classes and suggest the measures how those flaw(s) can be removed.

[20 M]

Q(3) **[Expected Time : 30 Minutes]**

Read the following Java code carefully and answer the question asked at the end.

```
class Student
{
    private Name    name;   // name of student
    private int     age;    // age of student
    private Date    dob;    // date of birth of student
    Student(String n, int a, Date d)    { this.name = n; age = a; dob = d; }
    public String   getName()           { return name; }
    public String   getAge()            { return age;  }
    public String   getDOB(){ return dob;  }
    public String   toString()          { return " Name: " + name + " Age : " + age +" DOB : " + dob; }
}// End of class Student
class Test
{
    public static void main(String args[])
    {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        Date d1 = new Date();
        Date d2 = new Date();
        Student std;
        if (a > b)
        {
            while ( a - b != 0)
            {
                std = new Student("X",18,d1);
                System.out.println(std.getDOB());
                b++;
            }// End of loop
        }
        else
        {
            while ( b - a != 0)
            {
                std = new Student("Y",20,d2);
                System.out.println(std.getName());
                a++;
            }// End of while loop
        }// End of if
    }// End of Method
}// End of class
```

Draw the Sequence Diagram for the above Java Code.

[20 M]

PART-B [Open Book, Expected Time: 90 Minutes]

Q (1) [Expected Time : 40 Minutes]

Organizational structure of BITS is divisions and units based. A *unit chief* heads each unit and a *dean* heads division. Every employee of BITS (either a teaching faculty or a non-teaching one) has been assigned to some unit or division based upon his/her expertise and/or choice. Each unit and every division has mixed type of employees i.e. both teaching and non-teaching. There is a requirement to accomplish the following activities for every employee, unit, and division and for the whole institute before 31st March of each year.

(i) Yearly budget estimate is to be computed for every employee, for every unit, for every division and for the whole institute. For employees (either teaching or non-teaching) yearly budget is monthly salary * 12. For units and divisions this value (yearly budget) will be the sum of employee's salary plus expenditure to be spent on daily stationary requirements. The yearly budget for the whole institute will be grand total of budgets of all the units and divisions.

(ii) The yearly performance of each employee (teaching and non-teaching both), unit, division and the whole institute is to be measured. Performance measurement criteria for teaching faculty (an integer) will be based upon the research articles/papers published in conferences / journals. At the start of an academic year, this value is initialized to 0. When any faculty publishes a conference paper, then score of 5 is added to his/her performance value. Similarly, a score of 10 is added for every journal publishing. Performance for non-teaching staff will be measured upon the monthly performance rating (*****) given by his/her unit chief / division dean. Every unit chief and division dean rates the non-teaching staff working under his/her division every month on a scale of five stars (*****).The performance of units and divisions will be measured in terms of integer value (earned by teaching staff of that division/unit) and stars(earned by non-teaching staff of that unit/division). The performance of the whole institute will be measured and accumulated by the performance of its units and divisions.

Suppose you are selected as a designer to build a software to fulfill the above requirements.

Q1 (a) Which design pattern/or patterns will you select to solve this problem and why? The strength of this question is the justification that you will give. Blind guess will fetch zero marks.

Q1 (b) Draw the class diagram showing the participants involved in the choice of your design pattern. [Note: Draw a separate class diagram for each pattern if you think more than one design patterns are required. Even if one design pattern is required for multiple situations, you have to separately draw a class diagram for each such situation]

Q1 (c) For each class diagram that you have drawn in Q1(b), write as much java code (not more than five to ten lines) that you think is sufficient to justify your choice of design pattern or patterns.

[2+15+8=25 M]

Q (2) [Expected Time : 25 Minutes]

The following Java code prints out the sorted student list using various comparison mechanisms (such as by name, by age and by idno) for comparing students. Read the code carefully and answer the question asked at the end.

```
import java.util.*;
class Student
{
        private String  name;
        private int     age;
        private String  Idno;
        Student(String name, int age, String Idno) {   this.name = name; this.age = age; this.Idno = Idno; }
        public String   getName()                      {   return name;   }
        public int      getAge()                       {   return age;    }
        public String   getIdno()                      {   return Idno;   }
        public String   toString()
        {
                String s = "Name : "+name+" Age: "+age+" id No: "+ Idno;
```

```
                return s;
        }
}// End of class
class    StudentList
{
        private         ArrayList<Student>      studentList;
        StudentList()                           {       studentList = new ArrayList<Student>(); }
        public void addStudent(Student s)       {       studentList.add(s);     }
        public void printSortedList(Comparator<Student> sc)
        {
                Collections.sort(studentList, sc);
                System.out.println(" ********** ");
                System.out.println(sc.getClass().getName());
                for(int i =0; i < studentList.size(); i++) System.out.println(studentList.get(i));
                System.out.println(" ********** ");
        }
}// End of Class
class Driver
{
  public static void main(String args[])
  {
    Comparator<Student> c1 = new Comparator<Student>()
    {
        public int compare(Student s1, Student s2)
        {
            return s1.getName().compareTo(s2.getName()); // Compare Students by Name
        }
    };
    Comparator<Student> c2 = new Comparator<Student>()
    {
        public int compare(Student s1, Student s2)
        {
            // Compare Students by ID NO
            return s1.getIdno().compareTo(s2.getIdno());
        }
    };
    Comparator<Student> c3 = new Comparator<Student>()
    {
        public int compare(Student s1, Student s2)
        {
            // Compare Students by Age
            return ( s1.getAge() - s2.getAge());
        }
    };
        // Create a StudentList
        StudentList studentList = new StudentList();
        // Add Students
        studentList.addStudent(new Student("AKSHAY JHAMB",20,"2013B1A2235P"));
        studentList.addStudent(new Student("SAMIP JASANI",18,"2015A7PS127P"));
        studentList.addStudent(new Student("ANURADHA BANSAL",19,"2014B4A7800P"));
        studentList.addStudent(new Student("SHIVAM ARORA",20,"2013B2A3840P"));
        studentList.addStudent(new Student("VIGNESH N",18,"2015A7PS355P"));
        studentList.addStudent(new Student("ANANYASHREE GARG",18,"2015A7PS117P"));
        studentList.addStudent(new Student("VIPUL JINDAL",19,"2014B5A7310P"));
        studentList.printSortedList(c1);   // Prints Sorted List By Name
        studentList.printSortedList(c2);   // Prints Sorted List By Age
        studentList.printSortedList(c3);   // Prints Sorted List By IdNo
  } // End of Method
```

// End of Class

OUTPUT OF Q(2)

```
**********
Driver$1
Name : AKSHAY JHAMB Age: 20 id No: 2013B1A2235P
Name : ANANYASHREE GARG Age: 18 id No: 2015A7PS117P
Name : ANURADHA BANSAL Age: 19 id No: 2014B4A7800P
Name : SAMIP JASANI Age: 18 id No: 2015A7PS127P
Name : SHIVAM ARORA Age: 20 id No: 2013B2A3840P
Name : VIGNESH N Age: 18 id No: 2015A7PS355P
Name : VIPUL JINDAL Age: 19 id No: 2014B5A7310P
**********
**********
Driver$2
Name : AKSHAY JHAMB Age: 20 id No: 2013B1A2235P
Name : SHIVAM ARORA Age: 20 id No: 2013B2A3840P
Name : ANURADHA BANSAL Age: 19 id No: 2014B4A7800P
Name : VIPUL JINDAL Age: 19 id No: 2014B5A7310P
Name : ANANYASHREE GARG Age: 18 id No: 2015A7PS117P
Name : SAMIP JASANI Age: 18 id No: 2015A7PS127P
Name : VIGNESH N Age: 18 id No: 2015A7PS355P
**********
**********
Driver$3
Name : ANANYASHREE GARG Age: 18 id No: 2015A7PS117P
Name : SAMIP JASANI Age: 18 id No: 2015A7PS127P
Name : VIGNESH N Age: 18 id No: 2015A7PS355P
Name : ANURADHA BANSAL Age: 19 id No: 2014B4A7800P
Name : VIPUL JINDAL Age: 19 id No: 2014B5A7310P
Name : AKSHAY JHAMB Age: 20 id No: 2013B1A2235P
Name : SHIVAM ARORA Age: 20 id No: 2013B2A3840P
**********
```

Which design pattern/patterns is/are used in the above Java code. Draw the class diagram for your selected design pattern/patterns.

[15 M]

Q (3) [Expected Time : 25 Minutes]

The following given Java code simulates the working of a dummy emergency service (Assume Fire Service) of a city. Suppose you ring the emergency service then your phone number will be used as a basic input by the emergency service. The operator on duty uses the phone number (from where the call has come) to retrieve the details (such as location – where the phone is installed, owner_name, city etc) related with the phone number. When location associated with the phone number is retrieved, operator uses a mapping service similar to GoogleMap for displaying and printing the map (from the emergency service location to the location of the phone number). This map is then given to the van driver so that vehicle can be dispatched towards the desired location. Only one design pattern is used in this file. You have to identify the design pattern that is used and has to write the solution using the same answer-pattern style that we have discussed in class. [Note: Some features such as printing or displaying the map are not supported in reality but they are only available in simulated flavor].

[20 M]

```
import java.util.*;

// The class PhoneDataEntry encapsulates the values of various attributes associated with a phone
class PhoneDataEntry
{
        // Instance Fields
        private String phone_no;        // phone number
        private String location;        // location of phone
        private String owner_name;      // name of owner
```

```
        private String city;                    // Name of the City
        // Constructor Method
        PhoneDataEntry(String phone_no, String location, String owner_name, String city)
        {
            this.phone_no   =   phone_no;   this.location   =   location;
            this.owner_name =   owner_name; this.city       =   city;
        }// End of Constructor
        // Accessor Methods
        public String getPhone_no()     {   return this.phone_no;   }
        public String getLocation()     {   return this.location;   }
        public String getOwner_name()   {   return this.owner_name; }
        public String getCity()         {   return this.city;       }
}// End of class PhoneDataEntry

// The class PhoneDB encapsulates the telephone directory
class PhoneDB
{
        private static ArrayList<PhoneDataEntry>    phoneDB;    // list of phone numbers
        // Constructor Method
        PhoneDB()
        {
            phoneDB     = new ArrayList<PhoneDataEntry>();
            // Adding 15 Dummy Phone Numbers
            phoneDB.add(new PhoneDataEntry("3456628892","A","Mr. X1","Pilani"));
            phoneDB.add(new PhoneDataEntry("9460043456","B","Mr. X2","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643457","C","Mr. X3","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643458","D","Mr. X4","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643459","E","Mr. X5","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643460","F","Mr. X6","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643461","G","Mr. X7","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643462","H","Mr. X8","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643463","I","Mr. X9","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643464","J","Mr. X10","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643465","K","Mr. X11","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643466","L","Mr. X12","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643467","M","Mr. X13","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643468","N","Mr. X14","Pilani"));
            phoneDB.add(new PhoneDataEntry("3456643469","O","Mr. X15","Pilani"));
        }// End of Constructor Method
        // The following method returns the phone details of a given phone no if exists
        // Otherwise if phone no does not exists it returns null value
        public static PhoneDataEntry getDetails(String phone_no)
        {
            for(int i=0; i<phoneDB.size(); i++)
            {
                if(phoneDB.get(i).getPhone_no().equals(phone_no))
                return  phoneDB.get(i);
            }
            return null;
        }// End of Method
}// End of Class PhoneDB

// The following class encapsulates the working of an emergency service
class EmergencyService
{
  // The following class variable indicates the location where the office of the emergency service is located
        private static String emergencyServiceCurrentLocation = "S1";
        // Instance Fields
        private String phone_number;    //      Phone Number from where the ring has come
        // Construcor Method
```

```
        EmergencyService(String phone_number)
        {
                this.phone_number = phone_number;
        }// End of Constructor
        // Method to retrieve the current location of emergency service
        public String getEmergencyCurrentServiceLocation()      { return emergencyServiceCurrentLocation; }
        // Accessor Method for phone_no
        public String getPhone_number() {       return this.phone_number; }
        // The following method displays the details of a given phone no to System.out
        public void displayLocationDetails()
        {
                PhoneDataEntry pde =    PhoneDB.getDetails(phone_number);
                System.out.println("\n Displaying Details");
                System.out.println("\n Phone Number       : "+pde.getPhone_no());
                System.out.println("\n Location           : "+pde.getLocation());
                System.out.println("\n Ower Name: "+pde.getOwner_name());
                System.out.println("\n City               : "+pde.getCity());
        }// End of Method
        // The following method displays the details of a given phone no to printer
        public void printLocationDetails()
        {
                PhoneDataEntry pde =    PhoneDB.getDetails(phone_number);
                System.out.println("\n Printing Details to Printer");
                // Assume the following data items are printed on printer
                System.out.println("\n Printing Phone Number      : "+pde.getPhone_no());
                System.out.println("\n Printing Location          : "+pde.getLocation());
                System.out.println("\n Printing Ower Name         : "+pde.getOwner_name());
                System.out.println("\n Printing City                : "+pde.getCity());
        }// End of Method
}// End of class EmergencyService

// The following class displays and draws the map of a given location
class Mapper
{
  // The following method draws a map from originalLocation to destinationLocation in a given scale
  // You are given a dummy implementation of this method
  // The method only displays the map over System.out [Remember GoogleMap]
        public void drawMap(String originalLocation, String destinationLocation, int scale)
        {
        System.out.println("\n Displaying Map: "+ " From: "+ originalLocation + " To: "+ destinationLocation + " Scale: "+scale);
        } // End of Method
  // The following method prints a map from originalLocation to destinationLocation in a given scale
  // You are given a dummy implementation of this method
  // The method only prints the map over System.out [Remember GoogleMap]
        public void printMap(String originalLocation, String destinationLocation, int scale)
        {
        System.out.println("\n Printing Map: "+ " From: "+ originalLocation + " To: "+ destinationLocation + " Scale: "+scale);
        } // End of Method
}// End of Class Mapper

// The following class implements a dummy emergency service functionality
// It uses the features of both EmergencyService as well of Mapper

class DummyService extends EmergencyService
{
        private Mapper mapper;          // Holds a Mapper reference
        // Constructor Method
        DummyService(Mapper mapper, String phone_no)
        {
                super(phone_no);
                this.mapper = mapper;
        }
```

```
        // Overriding the displayLocationDetails() Method of EmergencyService Class
        public void displayLocationDetails()
        {
                super.displayLocationDetails();
                // Retrive the phone details first
                PhoneDataEntry          pde =   PhoneDB.getDetails(getPhone_number());

                // Displays the map over System.out from emergency service location to phone location
        mapper.drawMap(getEmergencyCurrentServiceLocation(),pde.getLocation(),1000);
        }

        // Overriding the printLocationDetails() Method of EmergencyService Class
        public void printLocationDetails()
        {
                super.printLocationDetails();
                // Retrive the phone details first
                PhoneDataEntry          pde =   PhoneDB.getDetails(getPhone_number());
                // Prints the map over Printer from emergency service location to phone location
                mapper.printMap(getEmergencyCurrentServiceLocation(),pde.getLocation(),1000);
        }
}// End of class

// Driver Class
class Driver
{
  public static void main(String args[])
  {
        PhoneDB phonedb = new PhoneDB();
        DummyService dms = new DummyService(new Mapper(),"9460043456");
        dms.displayLocationDetails();
        dms.printLocationDetails();
  }// End of Method
}// End of Class Driver
```

OUTPUT OF Q3

```
Displaying Details
Phone Number   : 9460043456
Location           : B
Ower Name     : Mr. X2
City         : Pilani
Displaying Map:  From: S1 To: B Scale: 1000
Printing Details to Printer
Printing Phone Number  : 9460043456
Printing Location          : B
Printing Ower Name         : Mr. X2
Printing City          : Pilani
Printing Map:  From: S1 To: B Scale: 1000
```

**********END**********