# Question I

Based on the following description of a computer, draw a class diagram by specifying associations and cardinalities (no need to put attributes or methods).

A computer consists of one or more monitors, a case, an optional mouse, and a keyboard. A case has a metal chassis, a motherboard, several memory modules (RAM, ROM and cache), an optional fan, storage media (floppy disk, hard disk, CD-ROM, DVD-ROM ...), and peripheral cards (sound, network, graphics ...). A computer always has at least one floppy drive or a hard drive.

# Question II

Prepare a class diagram and an object diagram for a system for a real estate company lists property for sale.

Business Rules:

The firm has a number of sales offices in different locations. One or more employees are assigned to each sale office. Employees have an ID number and a name. An employee can be assigned to only one sales office. For each sales office, there is always one employee to manage this sales office. An employee can manage only the sales office to which he/she is assigned. The firm lists property for sale. Property listings include the name and the location of the property. Each unit of property must be listed with one (and only one) of the sales offices. A sales office may have any number of properties listed, or may have no properties listed. Each of property has one or more owners. Owners have names and addresses. An owner may own one or more units of property. Properties are own with different percentages of by different owners (e.g., 30% owner A, 70% owner B). How could you model this?

State all assumptions that you make.

# Question III

Consider the following program written in JAVA. Analyze the code and deduce the corresponding class diagram by defining the association’s types between classes and multiplicities if necessary.

public class University {

private List<Department> departments;

public void destroy(){

if(departments!=null)

for(Department d : departments) d.destroy();

departments.clean();

departments = null;

}

}

////////////////////END OF CLASS University

public class Department {

private List<Professor> professors;

private University university;

Department(University univ){

this.university = univ;

}

public void destroy(){

for(Professor p:professors)

p.fire(this);

professors.clean();

professors = null;

}

} ////////////////////END OF CLASS Department

public class Professor extends Person{

private List<Department> attachedDepartments;

public void destroy(){

}

public void fire(Department d){

attachedDepartments.remove(d);

}

}////////////////////END OF CLASS Professor

public class Person {

private double salary;

private String name;

private Birthday bday;

public Person(int y,int m,int d,String name){

bday=new Birthday(y, m, d);

this.name=name;

}

public double getSalary() {

return salary;

}

public String getName() {

return name;

}

public Birthday getBday() {

return bday;

}

///////////////////////////////inner class///////////////////////

private class Birthday {

int year,month,day;

public Birthday(int y,int m,int d){

year=y;

month=m;

day=d;

}

public String toString(){

return String.format("%s-%s-%s", year,month,day);

}

}////////////////////END OF CLASS Birthday

}////////////////////END OF CLASS Person

public class Test {

public static void main(String[] args) {

Person person=new Person(2001, 11, 29, "Thilina");

System.out.println("Name : "+person.getName());

System.out.println("Birthday : "+person.getBday());

}

}

# Question IV

Convert the following code to a **UML Class** Diagram

public class **Point2D** {

private int x, y;

public Point2D() {

this(0, 0);

}

public Point2D(int x, int y) {

this.x = x;

this.y = y;

}

public int getX() {

return this.x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return this.y;

}

public void setY(int y) {

this.y = y;

}

public String toString() {

return "(" + this.x + "," + this.y + ")";

}

}

public interface **Movable** {

public void move(int deltaX, int deltaY);

}

public class **MovablePoint2D extends Point2D implements Movable** {

public MovablePoint2D() {

this(0, 0);

}

public MovablePoint2D(int x, int y) {

super(x, y);

}

public void move(int deltaX, int delt) {

setX(getX() + delatX);

setY(getY() + delatY);

}

}

abstract public class **Shape implements Movable** {

private String color;

private Point2D center;

public Shape (String color) {

this.color = color;

this.center = new MovablePoint2D(0, 0);

}

public String toString() {

return "Shape of color=\"" + color + "\"";

}

public void move(int deltaX, int delt) {

this.center.move(deltaX, deltaY);

}

abstract public double getArea();

}

public class **Rectangle extends Shape** {

private int length;

private int width;

public Rectangle(String color, int length, int width) {

super(color);

this.length = length;

this.width = width;

}

public String toString() {

return "Rectangle[length=" + length +

",width=" + width + "," + super.toString() + "]";

}

public double getArea() {

return length\*width;

}

}

public class **Triangle extends Shape** {

private int base;

private int height;

public Triangle(String color, int base, int height) {

super(color);

this.base = base;

this.height = height;

}

public String toString() {

return "Triangle[base=" + base +

",height=" + height + "," + super.toString() + "]";

}

public double getArea() {

return 0.5\*base\*height;

}

}

# Question V

A shoppingCart object is associated with only one creditCard and customer and to items in itemToBuy object. Persistent customer information such as name, billing address, delivery address, e-mail address and credit rating are stored in the customer object. The credit card object is associated with a frequentShopper discount object, if the credit rating for the customer is good. The customer can make or cancel orders as well as add and delete items to the shopping cart product. The credit card object contains the secure method for checking that the charge is authentic.

# Question VI

A software is intended to help the organizers of the Festival of Cannes to manage data related to the films that come in competition during the festival. In this software, each film is designated by its title, date of publication, duration in minutes and a kind that can take a value from a predefined list of values (Romance, Drama, Fiction, horror, etc.).

Each film has a director, one or more authors and a team of actors. All these people are considered artists and can be candidates to win prizes. An artist has a name, first name, date of birth, a number of years of experience. An artist can play the role of director, author and/or actor at a time in the same film and/or in different films. Each actor has a specific rank in each film (Hero, main actor, invites, etc.). A prize is given to an artist for a specific role in a given film. The prize is issued by a different artist. An artist can get more than one prize in one or several films. For that, a function will give the total number of prizes that an artist has obtained.

On the other hand, several artists of a same film can obtain prices for their roles in the film. For this, a function will give the total number of prizes of a movie.

Give a UML diagram which allows modeling the classes of this software.

# Question VII

Consider the following set of requirements for a university information system that is used to keep track of student’s transcripts.

The first requirement is broken down into five parts.

* The university keeps track of each student's name, student number, social security number, current address and phone, permanent address and phone, birthdate, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and zip of the student's permanent address, and to the students' last name. Both social security number and student number have unique values for each student.
* Each department is described by a name, office number, and office phone number.
* Each course has a course name, description, code number, number of semester hours, level, and offering department. The value of code number is unique for each course.
* Each section has an instructor, semester, year, course, and section number. The section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ...
* A grade report has a student, section, and grade.

The other requirements for this application are:

* An administrator can update the courses to be taught by instructors, and enter the list of students taking a course.
* An instructor can enter and update the grades of the course(s) taught by this instructor.
* A student can request a grade report from the information system.

The additional requirements for this application to be considered:

* A student can fill out an online add/drop form.
* A student may request an instructor to approve the add/drop form.
* The add/drop form must be approved by the administrator.
* The add/drop information is entered into the database.

Draw a class diagram of the system. Based on your class diagram, draw a sequence diagram showing how a student can add/drop a course.

# Question IIX

In the following class diagram:

1. Briefly describe the “Bank” class.
2. How many accounts a single customer may have?
3. How many customers may own a given account?
4. Explain the link between a bank and its branches.
5. Explain the link between a branch and its accounts.

How can the accounts classes be further factorized? Give the fragment of the diagram.

