Question 01

a)

Structured Programming	Object Oriented Programming
Less abstraction and less flexibility.	More abstraction and more flexibility.
Less secure and no way to data hiding.	More secure and have data hiding features.
Programs are divided into small self-contained functions.	Programs are divided into small entities called objects.

b)

Object – An object is a self-contained component which consists of methods and properties to make particular type of data useful.

Object determines the behavior of the class.

An object can be a data structure, a variable or a function. It has memory location allocated.

Ex : If we take 'Vehicles' as a class, 'Car' is an object of that class.

Encapsulation – The wrapping up of data and functions together, into a single unit is called encapsulation.

An act of combining properties and methods, related to the same object, is known as Encapsulation. Encapsulation hides data for the purpose of data protection.

Ex :

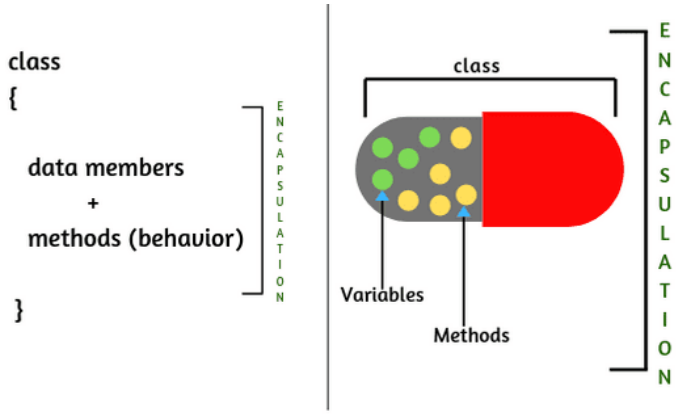


Fig: Encapsulation

Polymorphism – The ability to appear in many forms. More specifically, it is the ability to redefine methods for derived classes.

Ex : If there's a base class called 'Shape' , polymorphism enables the programmer to define different 'area' methods for any number of derived classes, such as 'circles', 'rectangles' and 'triangles.

c)

Easy to understand.

Reusability.

Data hiding.

Ability to simulate real world event much more effectively.

Improved software maintainability.

d)

Vehicle, Hire, Customer, Driver,.

e)

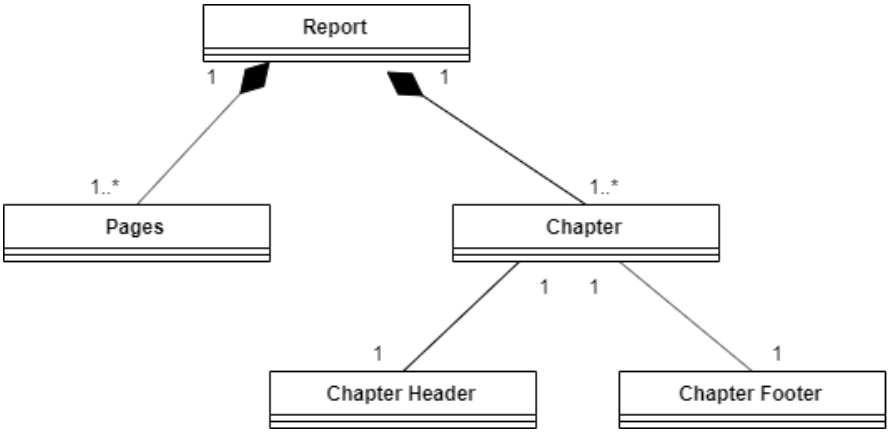
Student
- studentNo : int - name[20] : char + marks[3] : int
+ Student(pStNo : int, *pName : char) + calcAvg() : float + print() : void

f)

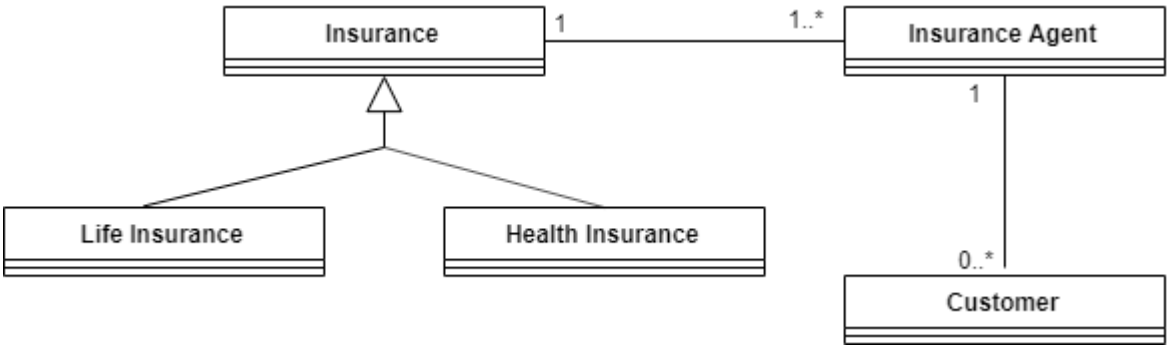
```
int main(){
        Rectangle rec1(14,7);
        Triangle tri1(12,10), tri2(18,18);
        double finalarea;

        finalarea = rec1.area() + tri1.area() + tri2.area();
        cout << finalarea << endl;

        return 0;
}
```

g)

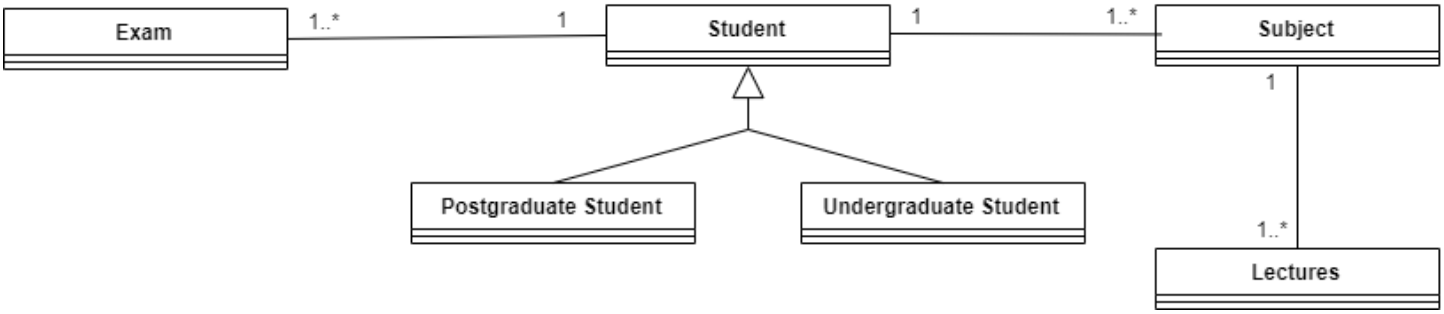
i)


Report
1
1
1..*
1..*
Pages
Chapter
1
1
1
1
Chapter Header
Chapter Footer

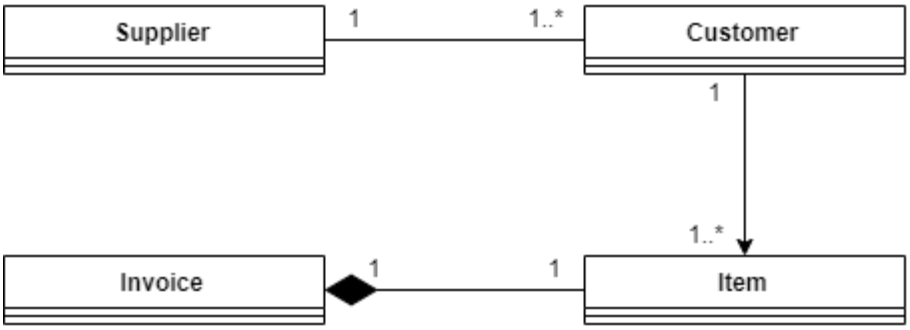
ii)


Insurance
1
1..*
Insurance Agent
1
Life Insurance
Health Insurance
0..*
Customer

iii)


Exam
1..*
1
Student
1
1..*
Subject
1
Postgraduate Student
Undergraduate Student
1..*
Lectures

iv)


Supplier
1
1..*
Customer
1
1..*
Invoice
1
1
Item

Question 2

Gate Keeper	
Responsibilities	Collaborations
logs to the system	
Produce temporary card	Temporary User
Produce a receipt	Payment

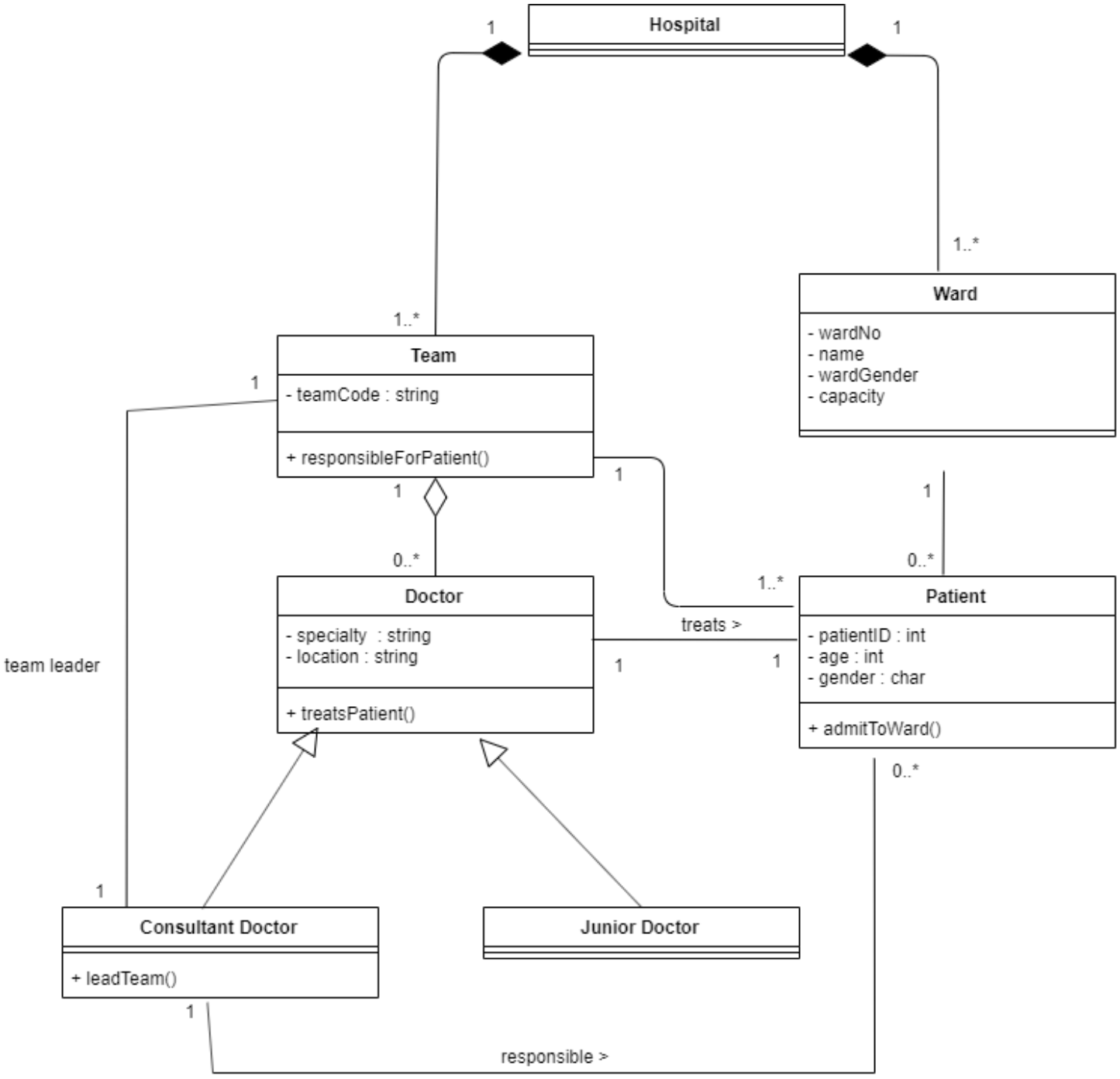
Registered User	
Responsibilities	Collaborations
Enter the parking lot	
Park the vehicle	
Leaves the parking lot	
Pay for parking	
Swipe the card	

RFID User	
Responsibilities	Collaborations
Enter	
Leaves	
Register	
Pay a fixed amount	
Reload the money	Gatekeeper

Temporary User	
Responsibilities	Collaborations
Obtain a parking card	Gatekeeper
Enter the parking lot	
Use the card	
Exit the parking lot	

Manager	
Responsibilities	Collaborations
Generates report summarizing	
Generates report profitability	

Question 3


Hospital
1
1
1..*
Ward
- wardNo
- name
- wardGender
- capacity
1..*
Team
- teamCode : string
+ responsibleForPatient()
1
1
1
0..*
Doctor
- specialty : string
- location : string
+ treatsPatient()
1..*
Patient
- patientID : int
- age : int
- gender : char
+ admitToWard()
1
0..*
treats >
1
1
team leader
1
Consultant Doctor
+ leadTeam()
Junior Doctor
1
0..*
responsible >

Question 4

```
Class Faculty{
        Private:
                String name;
                Institute *inst[SIZE];
                Employee *emp;
}

Class Institute{
        Private:
                String name;
                String address;
                ResearchAssociate *reas;

}

Class Employee{
        Protected:
                Int ssNo;
                String name;
                String email;
                Int counter;
                Faculty *fal;
        Public:
                virtual void EmployeeAbstract() = 0;
}

Class AdministrativeEmployee : public Employee{
}
```

```
Class ResearchAssociate : public Employee{
        Private:
                Int fieldOfStudy;
                Project *pro;
}

Class Lecturer : public ResearchAsssociate{
        Private:
                String name;
                Course *course;
}

Class Course{
        Private:
                String name;
                Int id;
                Float hours;
                Lecturer *lecturer;
}
```

```
Class Project(
        Private:
                String name;
                Date start;
                Date end;
                ResearchAssociate *ra;
}

Class Participation{
        Private:
                Int hours;
}
```

h.g.d.sandakalum@gmail.com