Question 1

a) Compare and contrast procedural programming languages with object-oriented languages.

Procedural programming languages	Object-oriented languages
Mainly use subroutines/Procedures	Uses classes and objects
No inheritance possible	Makes it easy to maintain and modify existing code as new objects are created inheriting characteristics from existing ones
Top down design is used	Design is done using objects
Code reuse is limited	High reuse of code due to Inheritance and other concepts
Focus global data and unprotected	Focus private data which is encapsulated and protected
No usage of Object Oriented Concepts such as Encapsulation, Abstraction, Polymorphism	Can use the object oriented supporting concepts/techniques

b) Describe the following terms used in Java.

i. Object

- Object is a main component in an object-oriented program.
- An object has attributes which maintains the state of the object and method which represents the behavior of the object.
- Objects are created from classes and may represent a real world or a conceptual object.
- Eg: Real World Object – Student
- Eg. Conceptual Object – Course
- An object is an instance of a class

ii. Class

- Class is used as the template for creating an object.
- A class consists the definitions of attributes and methods which are created when the objects are created out of the classes.
- The attributes in a class do not have values. But when an object is created out of a class it has values for the attributes relevant for the class.

iii. Constructor

- A constructor of a class is a special method which has same name of the class which is used for creating the object of the class.

- If it is necessary to initialize an objects instance fields at the time when the object is created then it should be done inside the constructor.
- Constructors can be overloaded.

```
Class Student {
        //Constructor
        public Student()
        {
        //Code for initializing elements
        }
}
```

Using the constructor for creating an object

```
Student s1 = new Student();
```

iv. Polymorphism

- In java polymorphism is a way in which something behaves differently based on its call
- It is the ability to resolve the correct method at the runtime of the program.
- This may involve inheritance as well.
- Java is able to identify the relevant method to call based on the type of the object.
- There are two types of polymorphism.
  - They are compile time polymorphism and runtime polymorphism.
  - Compile time polymorphism is related to method overloading and runtime polymorphism is related to method overriding.

c) Using suitable examples, explain the difference between a *static method* and a *non-static method*.

What is an static method ? (Class methods)
A static method is a method that belongs to a class, but its not belongs to an instance of that class and this method can be called without the instance or object of that class.

What is an non-static method ? (Instance Methods)
Every method in java are non-static method, but the methods must not have static keyword before method name. non-static methods can access any static method and static variable also, without using the object of the class.

Refer to the link for finding all the differences:

https://www.geeksforgeeks.org/difference-between-static-and-non-static-method-in-java/

How to call a static method

```
class GFG {

    // static method
    public static int sum(int a, int b)
    {
        return a + b;
    }
}

class Main {
    public static void main(String[] args)
    {
        int n = 3, m = 6;

        // call the static method
        int s = GFG.sum(n, m);

        System.out.print("sum is = " + s);
    }
}
```

How to call a non static method

```
class GFG {

    // non-static method
    public int sum(int a, int b)
    {
        return a + b;
    }
}

class Main {
    public static void main(String[] args)
    {
        int n = 3, m = 6;
        GFG g = new GFG();
        int s = g.sum(n, m);
        // call the non-static method

        System.out.print("sum is = " + s);
    }
}
```

POINTS	STATIC METHOD	NON-STATIC METHOD
Definition	A **static method** is a method that belongs to a class, but its not belongs to an instance of that class and this method can be called without the instance or object of that class.	Every methods in java are non-static method, but the methods must not have **static** keyword before method name. **non-static** methods can access any **static** method and **static** variable also, without using the object of the class.
Accessing members and methods	In **static** method, the method can only access only static data members and static methods of another class or same class but cannot access non-static methods and variables.	In **non-static** method, the method can access static data members and static methods as well as non-static members and method of another class or same class.
Binding process	**Static** method uses compile time or early binding.	**Non-static** method uses runtime or dynamic binding.

POINTS	STATIC METHOD	NON-STATIC METHOD
Overriding	**Static** method cannot be overridden because of early binding.	**Non-static** method can be overridden because of runtime binding.
Memory allocation	In **static** method, less memory is use for execution because memory allocation happens only once, because the static keyword fixed a particular memory for that method in ram. .	In **non-static** method, much memory is used for execution because here memory allocation happens when the method is invoked and the memory is allocated every time when the method is called.

Question 2

a) Identify all the *classes*, *subclasses*, *super classes* and write Java codes to define them.

classes – C1, C11, C12, C13, C121, C122

Super classes – C1 (immediate super class of C11, C12, C13)

C12 (immediate super class of C121, C122)

Sub classes - C11, C12, C13 are sub classes of C1

C121, C122 are sub classes of C12

C1

```
class C1 {
        //variables and methods goes here
}
```

C11

```
class C11 extends C1 {
        //variables and methods goes here
}
```

C12

```
class C12 extends C1 {
        //variables and methods goes here
}
```

C13

```
class C13 extends C1 {
        //variables and methods goes here
}
```

C121

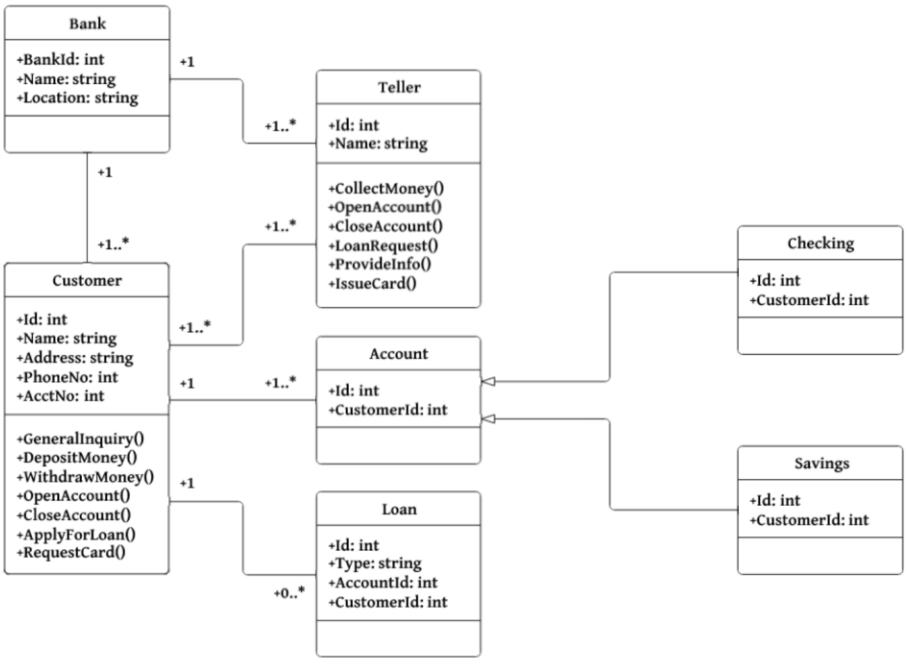
```
class C121 extends C12 {
        //variables and methods goes here
}
```

C122

```
class C122 extends C12 {
        //variables and methods goes here
}
```

b) The class diagrams are used to represent a particular feature in object-oriented systems. Identify and describe that feature providing an example from real-world applications.

- Static structure diagram that describes the structure of a system by showing the system's classes.



Class diagram above shows the structure of a system used inside a bank network.

- This shows how the classes are related to each other through different relationships such as association, aggregation, composition and inheritance.
    - E.g. "**Checking**" and "**Savings**" are subclasses of "**Account**" class. It shows the inheritance relationship between the classes.
    - Between the bank and the customer there is an association relationship
    - Also, the multiplicity of the relationships is given as well
        - Between the bank and the customer there is a one to many relationship
- This describes the members within the class
    - This shows the attributes (instance variables) within those classes.
    - Also, this shows the behavior (methods) within the classes
- It also shows the security level of each member within the class

Question 3

a) Discuss the difference between the *abstract class* and *final class*.

Abstract class

- Abstract class uses keyword abstract
- Abstract class can be extended by another class. Abstract class must be extended to be used by another class.
- Abstract class may have abstract method
- Abstract class cannot be final
- Cannot create objects

Final class

- Final class uses keyword final
- Final class cannot be extended (inherited by another class)
- Final class should not have any abstract methods
- Final class cannot be abstract
- Can create objects

b) A software engineer was asked to develop a lift control system using Java. The system analyst further asked him to reuse the already developed generic class "Lift". He developed the following java program to implement the system.

```
public class lift_control {

public static void main(String args[]){

Lift myLift = new Lift();

}

}
```

However, he could not successfully compile the above program. Identify the possible implementation errors and correct them.

Answer

- Assumption: Generic class is considered as an abstract class (But there is a type of a class called generic class that is not suitable for this example)
- First the Lift class should be extended and its methods should be implemented by another class as below.

```
public class LiftA extends Lift{
        //Implement the methods in the Lift class if any
}


//Now the below code should compile

public class lift_control {

public static void main(String args[]){

LiftA myLift = new LiftA();

}

}
```

Question 4

a) Briefly explain the advantages of using Unified Modeling Language (UML) in software modeling.

[6 marks]

Standardizes (Formalizes) the design to convey the same idea to any party
If we use UML which are standard notations for drawing design diagrams of static and dynamic designs of systems, they are able to convey the same idea for all the stakeholders of the system. UML removed ambiguity in design documents.

Can be used as a tool for generating code
Some tools available to take the UML diagram as input and generate the related classes and method stubs to help the developer to start developments faster.

Visual Representation
A UML diagram is a visual representation of the relationships between classes and entities in a computer program. A class is an object in programming that organizes similar variables and functions in one location. To understand a program, it is essential to understand what each class object does, the information it stores and how it relates to other classes in the program. By showing this information in a diagram, it is easy to understand and visualize a program's relationships.

Readability and Re-usability
A UML diagram is beneficial in that it is very readable. The diagram is meant to be understood by any type of programmer and helps to explain relationships in a program in a straightforward manner. Traditionally, to understand a program, a programmer would read the code directly. This could be thousands or millions of lines of code in very large programs. Having a UML diagram helps to quickly illustrate those relationships. Additionally, by using a diagram to show the code running in a program, a programmer is able to see redundant code and reuse portions of code that already exist rather than rewrite those functions.
Planning Tool

Standard
UML is the current standard for programming in object-oriented programming languages. When creating classes and other objects with relationships between each other, UML is what is used to visually describe these relationships. Because it is used as a standard, it is widely understood and well known. This makes it easy for a new programmer to step into a project and be productive from day one.

UML helps to plan a program before the programming takes place. In some tools used to model UML, the tool will generate code based on the classes set up in the model. This can help reduce overhead during the implementation stage of any program. Additionally, a UML model diagram is easy to change, whereas reprogramming a section of code can be tedious and time-consuming.

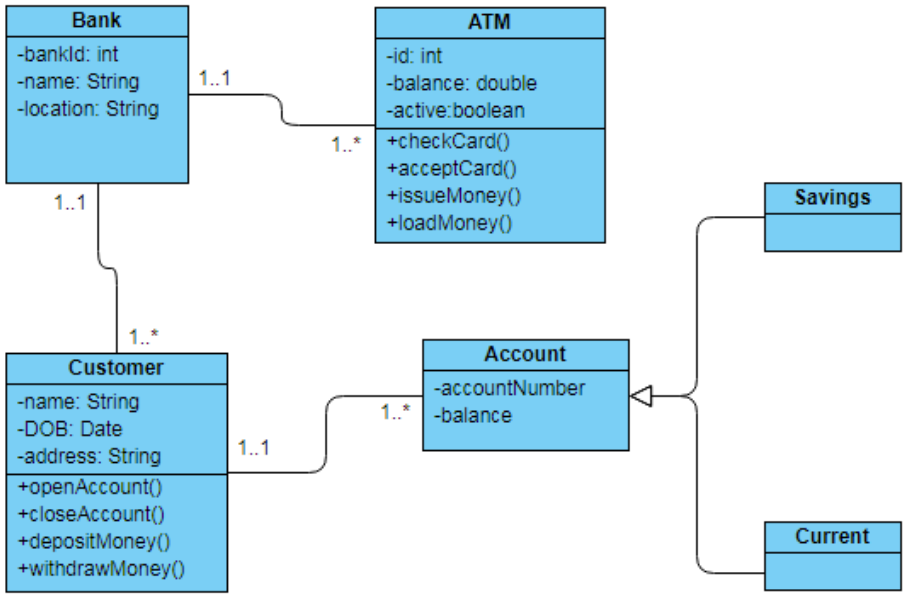
b) Consider the following requirements description for a bank's ATM.

Automated Teller Machines (ATM) are used by customers of a bank. Each branch of the bank has at least one ATM machine. The customers of the bank have two types of accounts namely, current accounts and savings accounts.

i. Do a non-verb analysis and identify the classes for the system. [4 marks] .

ATM
Customer
Bank
Current Account
Savings Account

ii. Draw the class diagram for the system. [10 marks]



Question 5

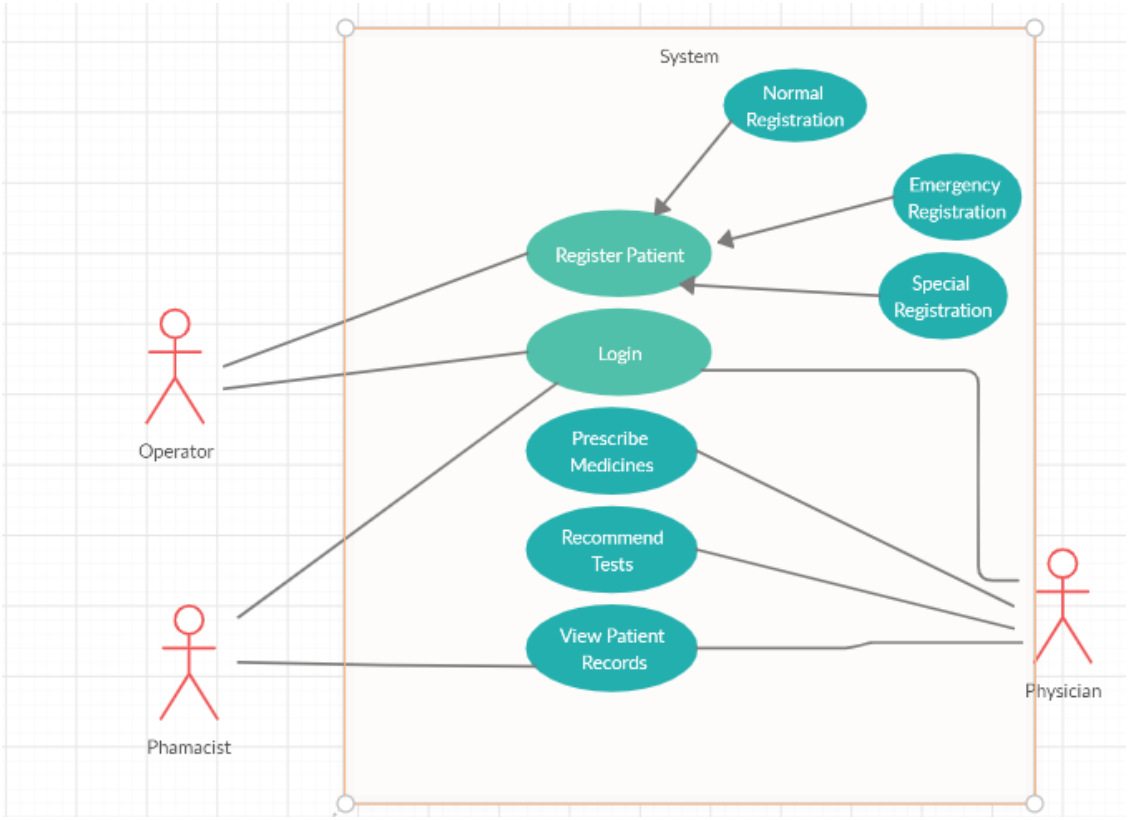
A patient care system needs to be developed and deployed in a hospital. The following high-level requirements have been identified.

A patient needed to be registered before any treatments. There are three registration types of cases, which are normal, emergency, and special. A patient is to be investigated by a medical officer before registration. If there is a need, the patient is directing to relevant physicians. After an investigation by a physician, medicines are prescribed, if needed medical tests recommended. Registered patients records, medical test reports, diagnostic cards are viewed through a computer system. The

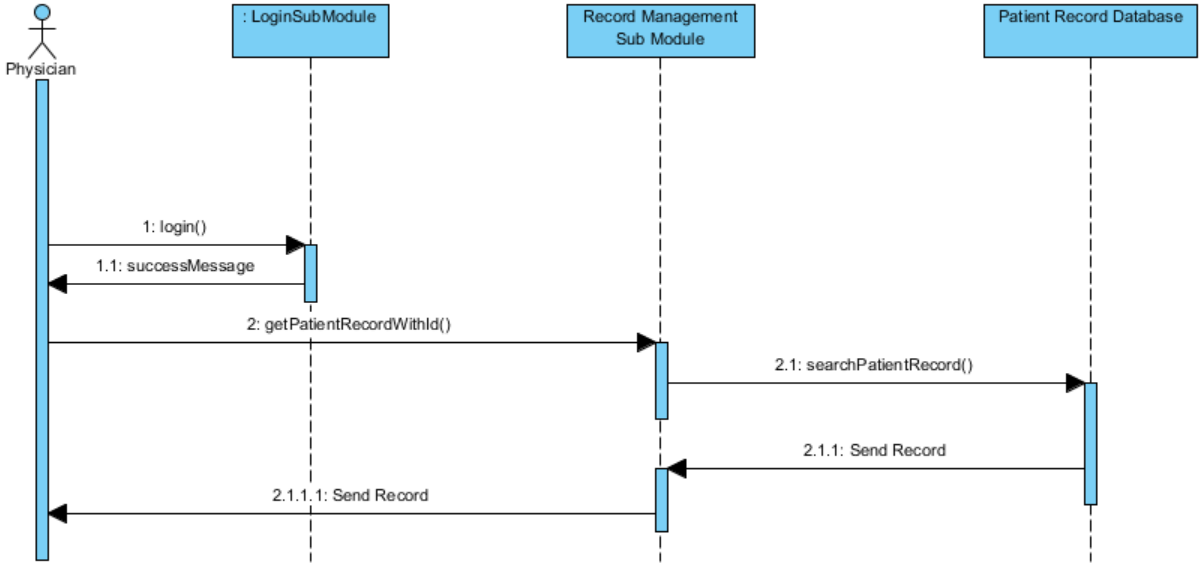
pharmacists and medical laboratory tests into electronic patient's records, respectively. After medical tests, if needed, further medicine is prescribed by physicians
Using UML notations,

a) Draw the use case diagram of the system.

[10 marks]



b) Draw a sequence diagram for the use case of obtaining a medical report of a registered patient.


Physician
: LoginSubModule
Record Management Sub Module
Patient Record Database
1: login()
1.1: successMessage
2: getPatientRecordWithId()
2.1: searchPatientRecord()
2.1.1: Send Record
2.1.1.1: Send Record