Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

AIM: Introduction to UML – To create a UML diagram of ATM Application.

INTRODUCTION TO UML:

The **Unified Modeling Language™** (**UML®**) is a standard visual modeling language intended to be used for modeling business and similar processes, Analysis, design, and implementation of software-based systems.

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

UML can be applied to diverse **application domains** (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component **software development methods** and for various **implementation platforms** (e.g., J2EE, .NET).

UML is a standard modeling **language**, not a **software development process**. UML 1.4.2 specification explains that process:

- provides guidance as to the order of a team's activities,
- specifies what artifacts should be developed,
- directs the tasks of individual developers and the team as a whole, and
- Offers criteria for monitoring and measuring a project's products and activities.

UML is intentionally **process independent** and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is **Rational Unified Process** (RUP).

UML is not complete and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams.

For example, semantics of **multiplicity of actors** and **multiplicity of use cases** on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram.

UML 2.4 Diagrams Overview

A **UML diagram** is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains **graphical elements** (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The **kind of the diagram** is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram.

A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude **mixing** of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some **UML Tools** do restrict set of available graphical elements which could be used when working on specific type of diagram.

Classification of UML 2.4 Diagrams

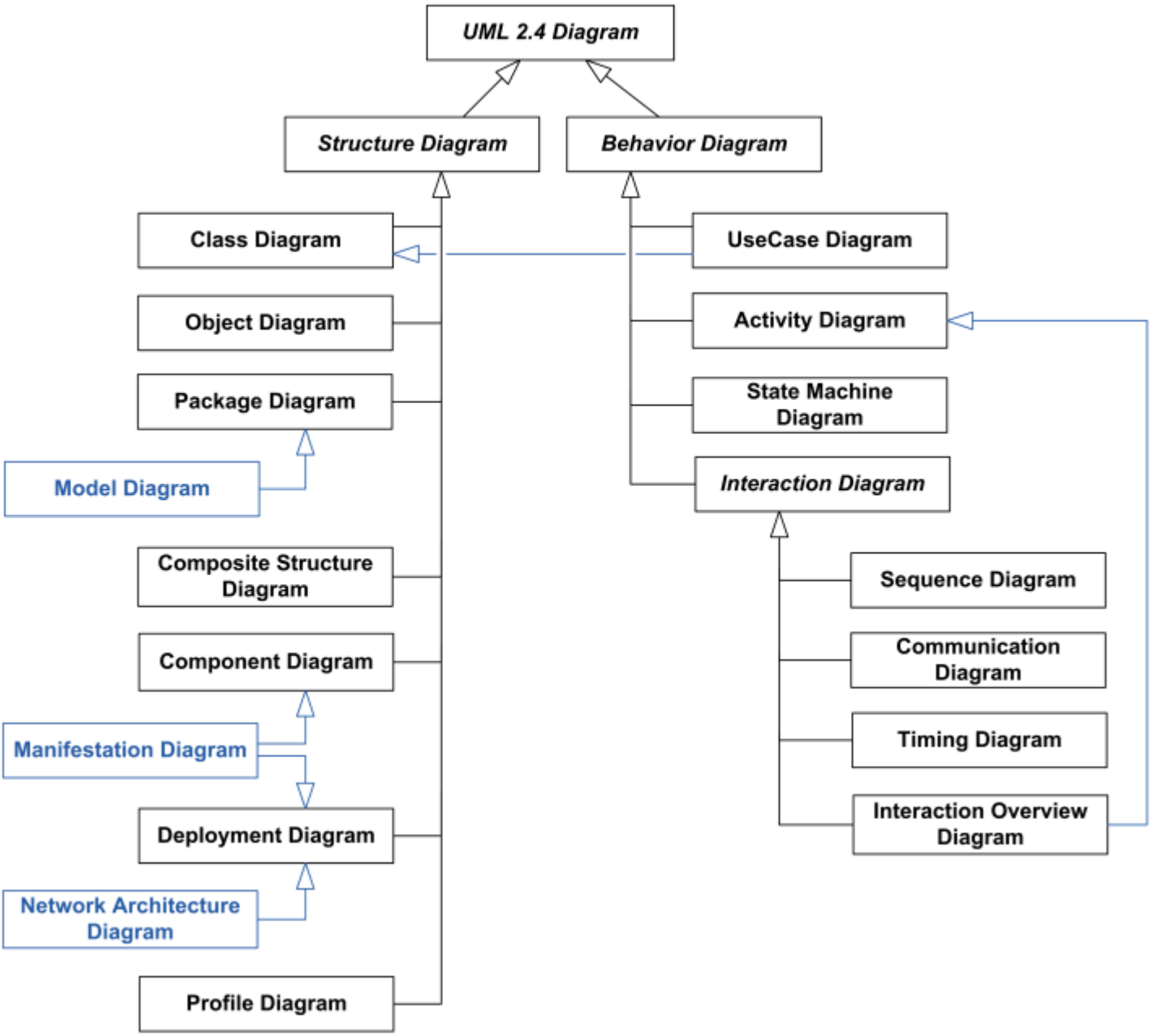
UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

UML 2.4 diagrams could be categorized hierarchically as shown below:



Structure Diagrams

Structure diagram shows **static structure** of the system and its parts on different abstraction and implementation levels and how those parts are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Structure diagrams are not utilizing **time** related concepts, do not show the details of dynamic behavior. However, they may show relationships to the behaviors of the classifiers exhibited in the structure diagrams.

Class diagram is a static structure diagram which describes structure of a system at the level of classifiers (classes, interfaces, etc.). It shows some classifiers of the system, subsystem or component, different relationships between classifiers, their attributes and operations, constraints.

Object diagram was defined in now obsolete **UML 1.4.2** specification as "a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time." It also stated that object diagram is "a class diagram with objects and no classes." **UML 2.4** specification simply provides no definition of **object diagram**.

Some major elements of object diagram are named and anonymous instance specifications for objects, slots with value specifications, and links (instances of association).

Package diagram shows packages and relationships between the packages.

Model diagram is UML auxiliary structure diagram which shows some abstraction or specific view of a system, to describe architectural, logical or behavioral aspects of the system. It could show, for example, architecture of a multi-layered (aka multi-tiered) application - multi-layered application model.

Composite structure diagram could be used to show:

- Internal structure of a classifier
- A behavior of a collaboration

Internal Structure diagrams show internal structure of a classifier - a decomposition of the classifier into its properties, parts and relationships.

Collaboration use diagram shows objects in a system cooperating with each other to produce some behavior of the system.

Component diagram shows components and dependencies between them. This type of diagrams is used for **Component-Based Development** (**CBD**), to describe systems with **Service-Oriented Architecture** (**SOA**).

Deployment diagram shows architecture of the system as deployment (distribution) of software artifacts to deployment targets.

Note, that components were directly deployed to nodes in UML 1.x deployment diagrams. In UML 2.x artifacts are deployed to nodes, and artifacts could manifest (implement) components. Components are deployed to nodes indirectly through artifacts.

Specification level deployment diagram (also called type level) shows some overview of deployment of artifacts to deployment targets, without referencing specific instances of artifacts or nodes.

Instance level deployment diagram shows deployment of instances of artifacts to specific instances of deployment targets. It could be used for example to show differences in deployments to development, staging or production environments with the names/ids of specific build or deployment servers or devices.

While component diagrams show components and relationships between components and classifiers, and deployment diagrams - deployments of artifacts to deployment targets, some missing intermediate diagram is manifestation diagram to be used to show manifestation (implementation) of components by artifacts and internal structure of artifacts.

Because manifestation diagrams are not defined by UML 2.4 specification, manifestation of components by artifacts could be shown using either component diagrams or deployment diagrams.

Deployment diagrams could also be used to show logical or physical **network architecture** of the system. This kind of deployment diagrams - not formally defined in UML 2.4 - could be called network architecture diagrams.

Profile diagram is auxiliary UML diagram which allows defining custom stereotypes, tagged values, and constraints. The Profile mechanism has been defined in UML for providing a **lightweight extension mechanism** to the UML standard. Profiles allow to adapt the UML metamodel for different

- **platforms** (such as J2EE or .NET), or
- **Domains** (such as real-time or business process modeling).

Profile diagrams were first introduced in UML 2.0.

Behavior Diagrams

Behavior diagrams show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

Use case diagrams are **behavior diagrams** used to describe a set of actions (use cases) that some system or systems (**subject**) should or can perform in collaboration with one or more external users of the system (actors) to provide some observable and valuable results to the actors or other stakeholders of the system(s).

Note, that **UML 2.4 specifications** also define **use case diagrams** as specialization of class diagrams (which are structure diagrams). Use case diagrams could be considered as a special case

of class diagrams where classifiers are restricted to be either **actors** or **use cases** and the most used relationship is association.

Activity diagram shows sequence and conditions for coordinating lower-level behaviors, rather than which classifiers own those behaviors. These are commonly called **control flow** and **object flow** models.

State machine diagram is used for modeling discrete behavior through finite state transitions. In addition to expressing the **behavior** of a part of the system, state machines can also be used to express the **usage protocol** of part of a system. These two kinds of state machines are referred to as **behavioral state machines** and **protocol state machines**.

Interaction diagrams include several different types of diagrams:

- sequence diagrams,
- interaction overview diagrams,
- communication diagrams, (known as **collaboration diagrams** in UML 1.x)
- timing diagrams.

Sequence diagram is the most common kind of interaction diagrams, which focuses on the message interchange between lifelines (objects).

Communication diagram (previously known as **Collaboration Diagram**) is a kind of **interaction diagram**, which focuses on the interaction between **lifelines** where the architecture of the internal structure and how this corresponds with the **message** passing is central. The sequencing of messages is given through a **sequence numbering** scheme.

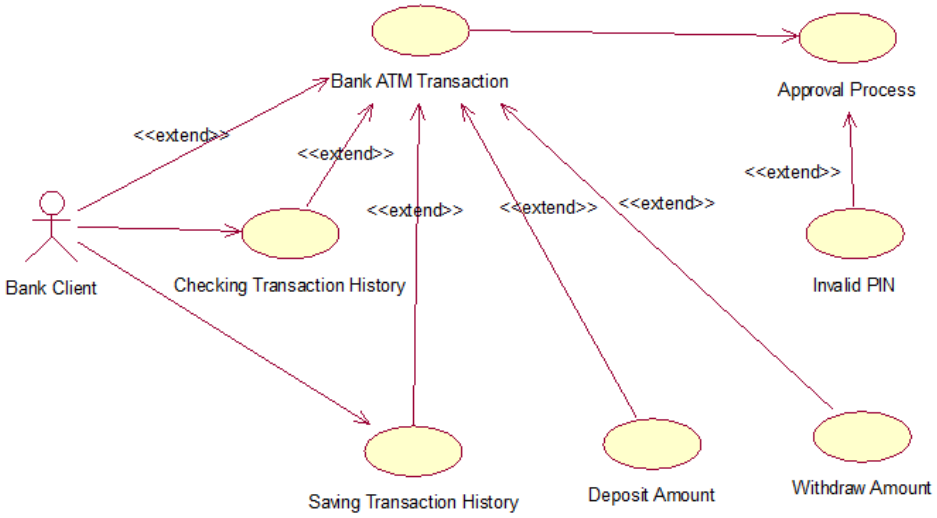
Interaction overview diagram defines interactions through a variant of activity diagrams in a way that promotes overview of the control flow. Interaction overview diagrams focus on the overview of the flow of control where the nodes are interactions or interaction uses. The lifelines and the messages do not appear at this overview level.

Timing diagrams are used to show interactions when a primary purpose of the diagram is to reason about time. Timing diagrams focus on conditions changing within and among Lifelines along a linear time axis.

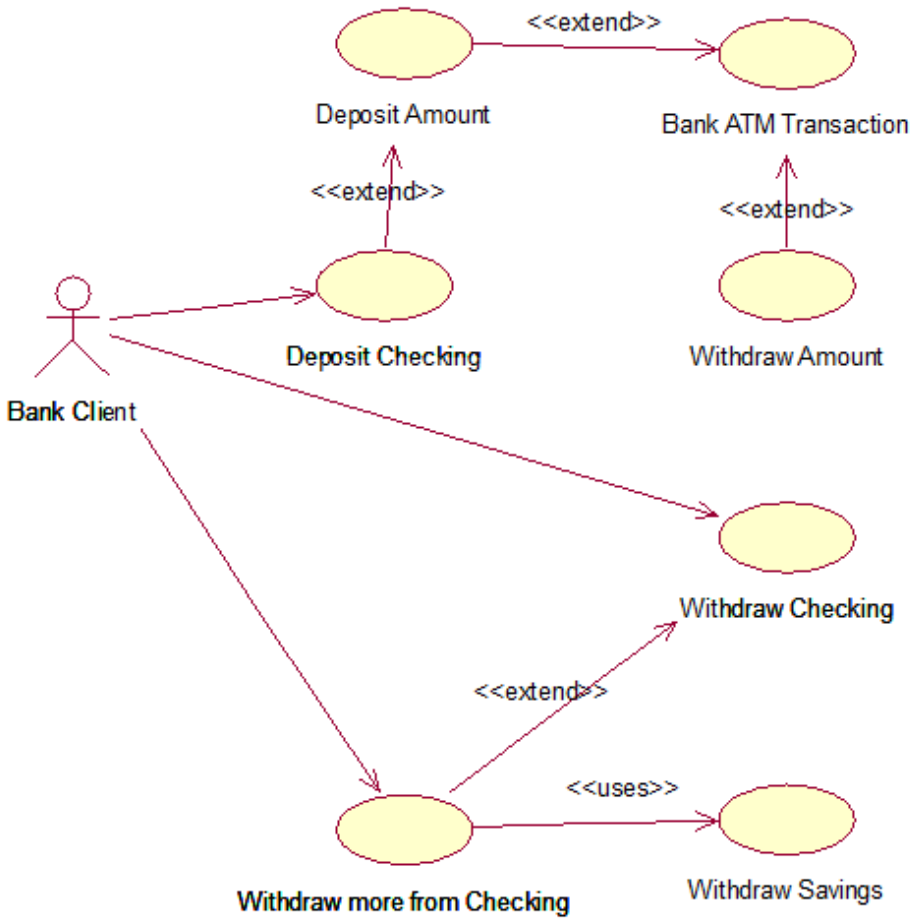
Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

1. UML DIAGRAMS FOR ATM APPLICATION:

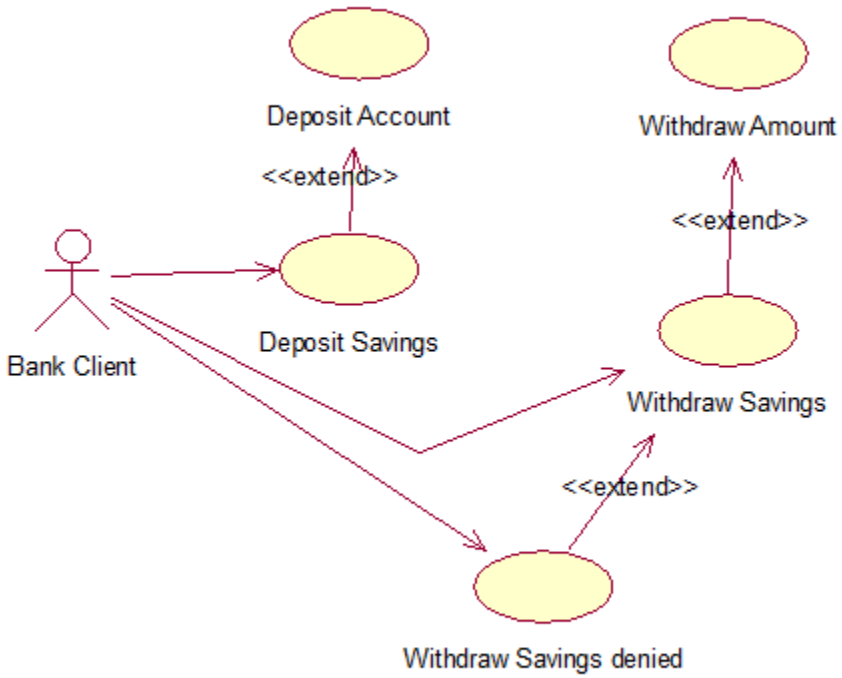
1.1 Use case diagrams for Bank ATM Transaction



1.2 Use case diagram for Checking Account:

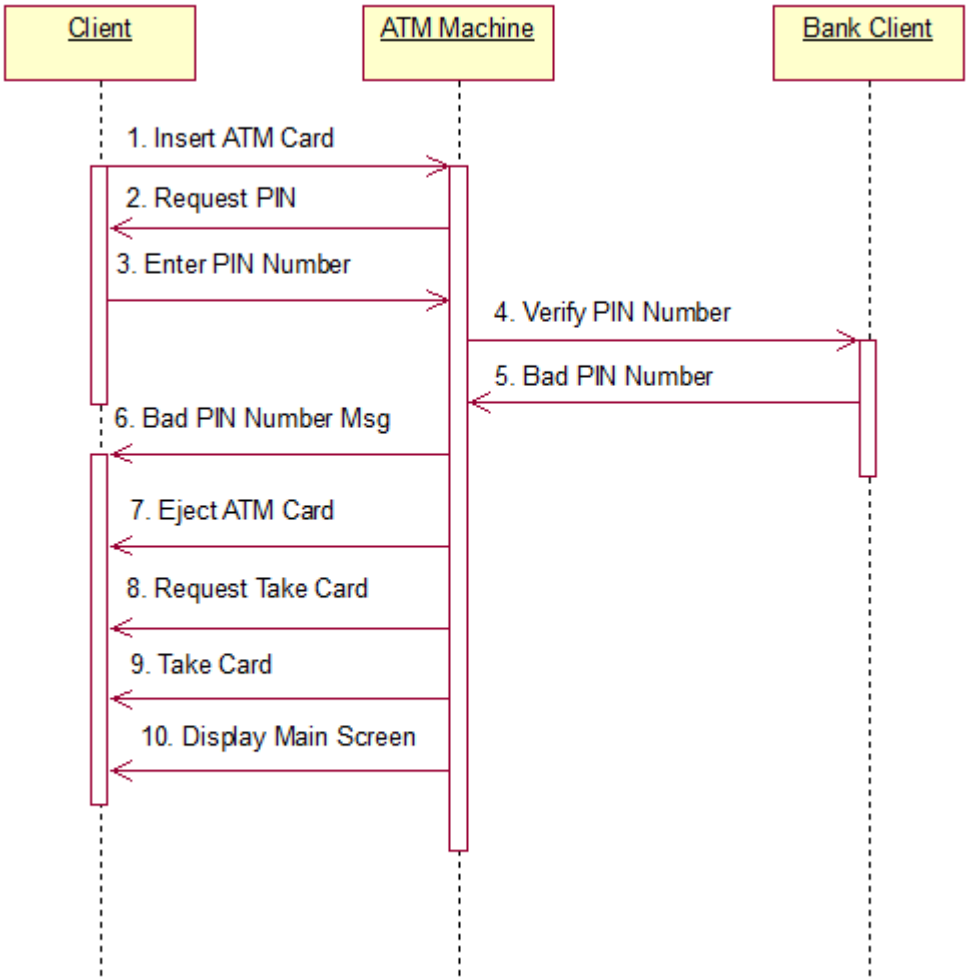


1.3 Use Case Diagram for Savings Account



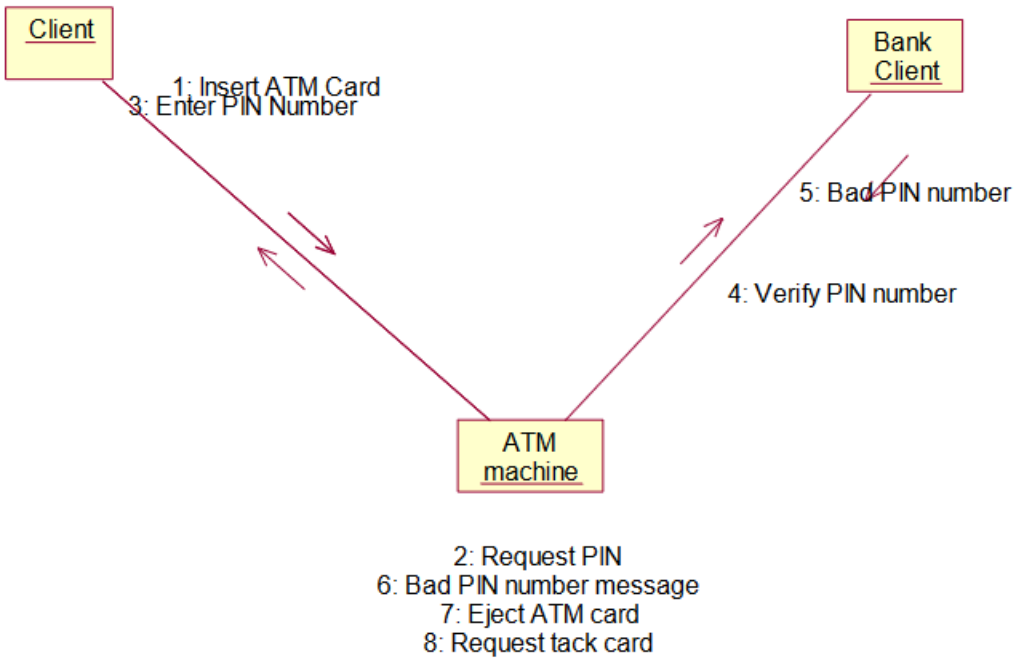
2. SEQUENCE DIAGRAMS

2.1 Sequence Diagram for Approval Process



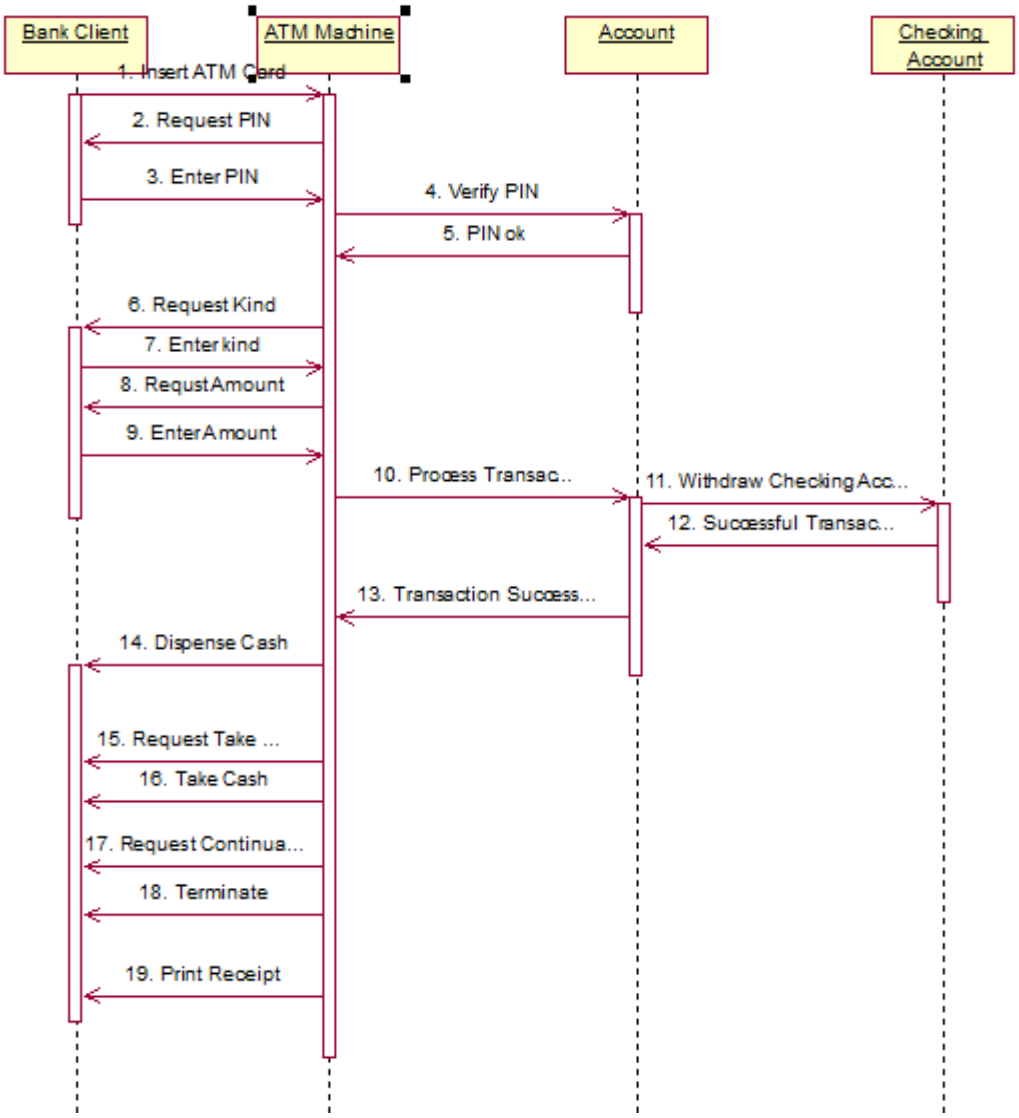
3. COLLABORATION DIAGRAM:

3.1 Collaboration Diagram for Approval Process:

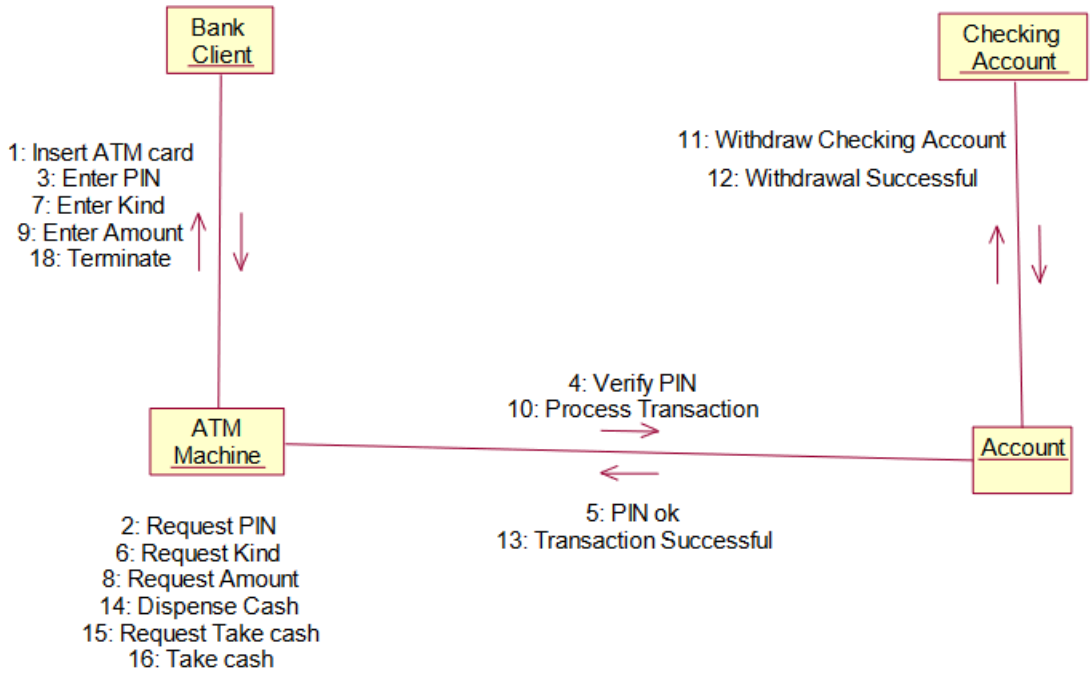


Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

2.2 Sequence Diagram for Checking Account

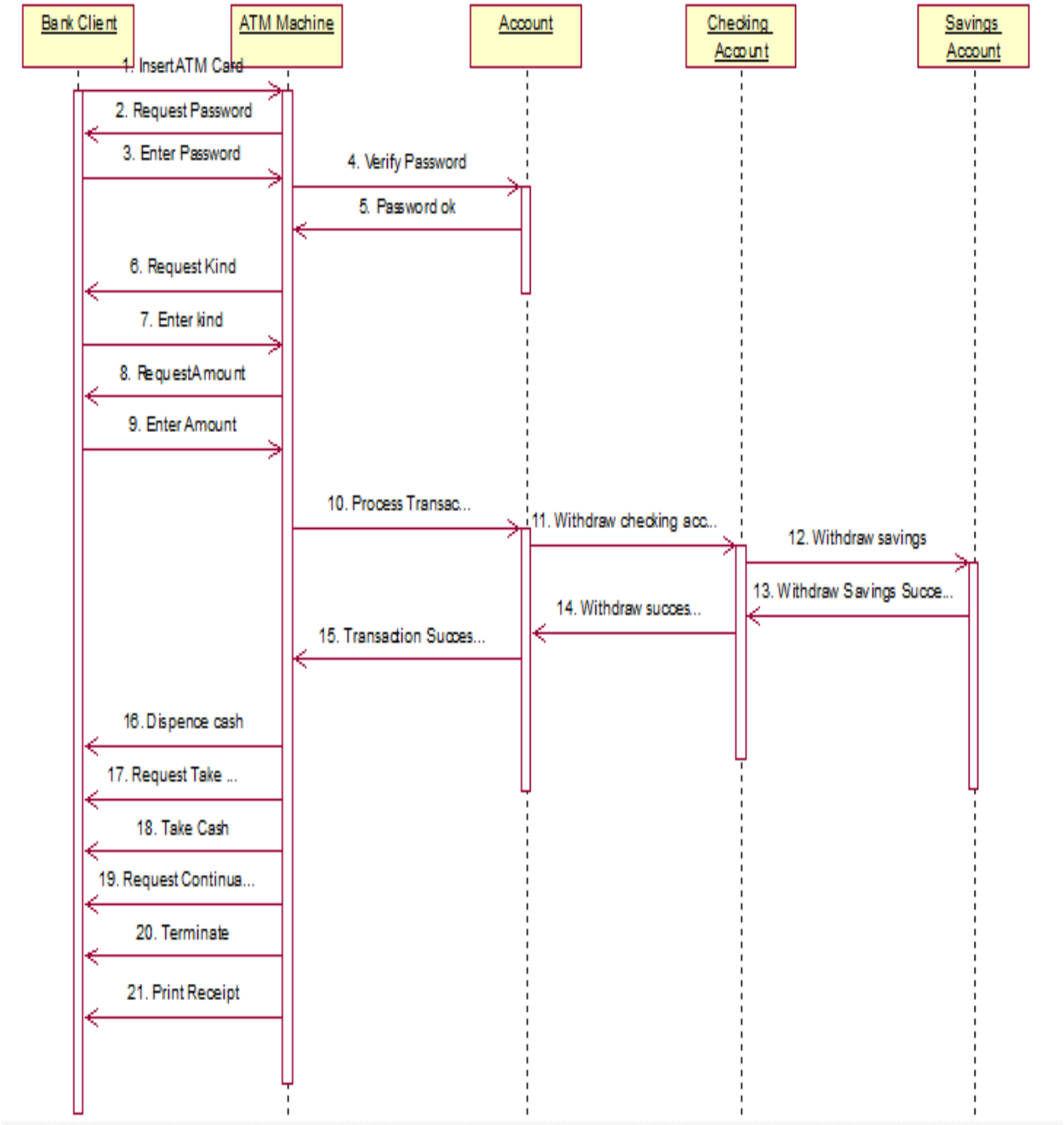


3.2 Collaboration Diagram for Checking Account:

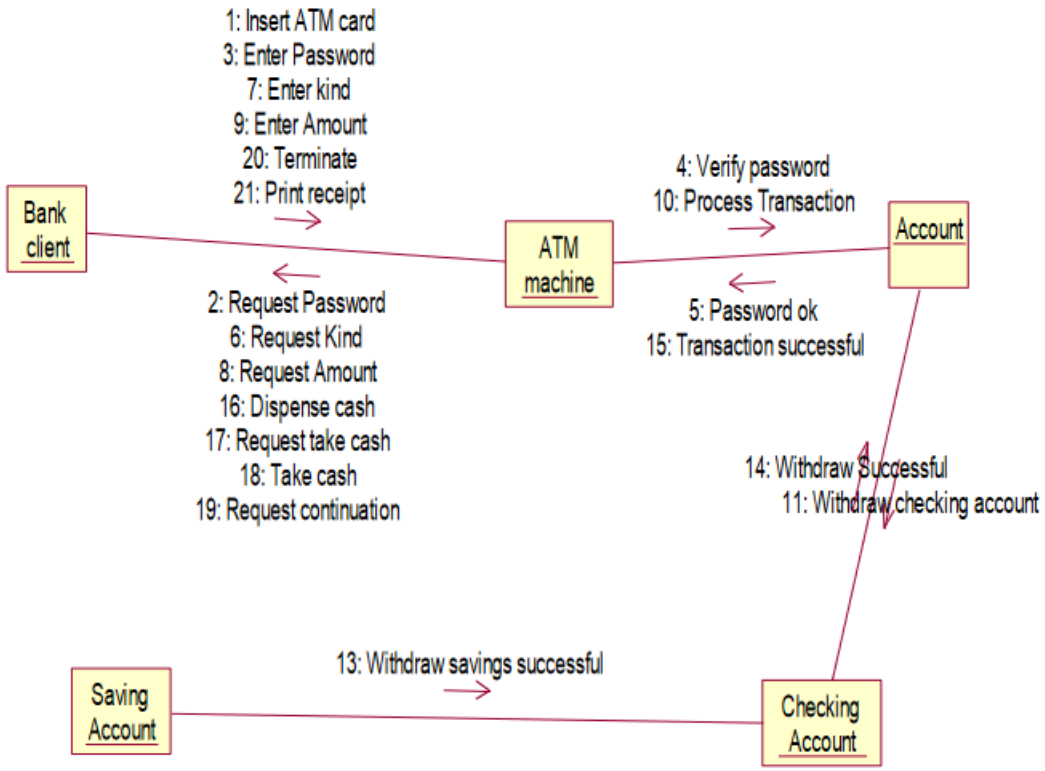


Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

2.3 Sequence Diagram for Withdraw more from Checking Account

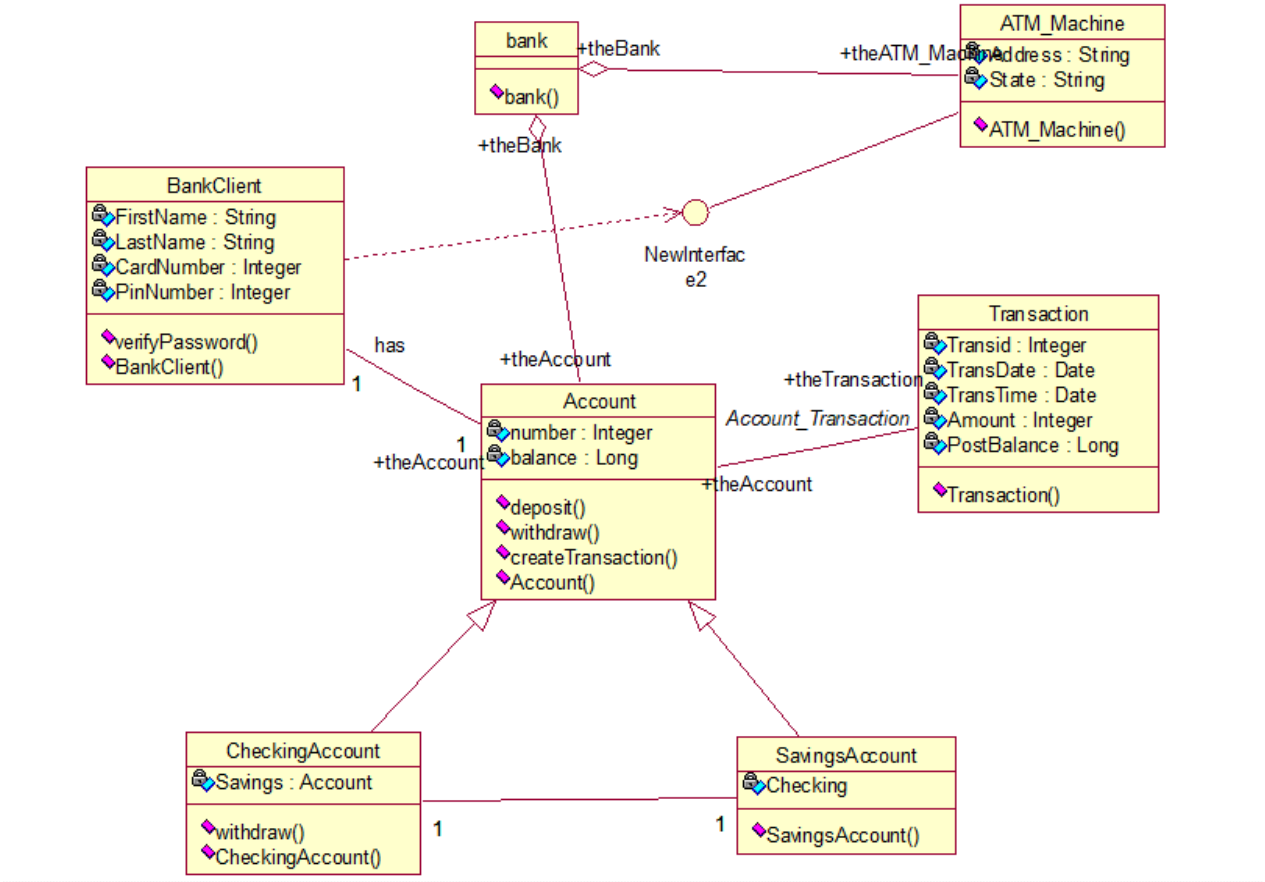


3.3 Collaboration Diagram for withdraw more from Checking Account:

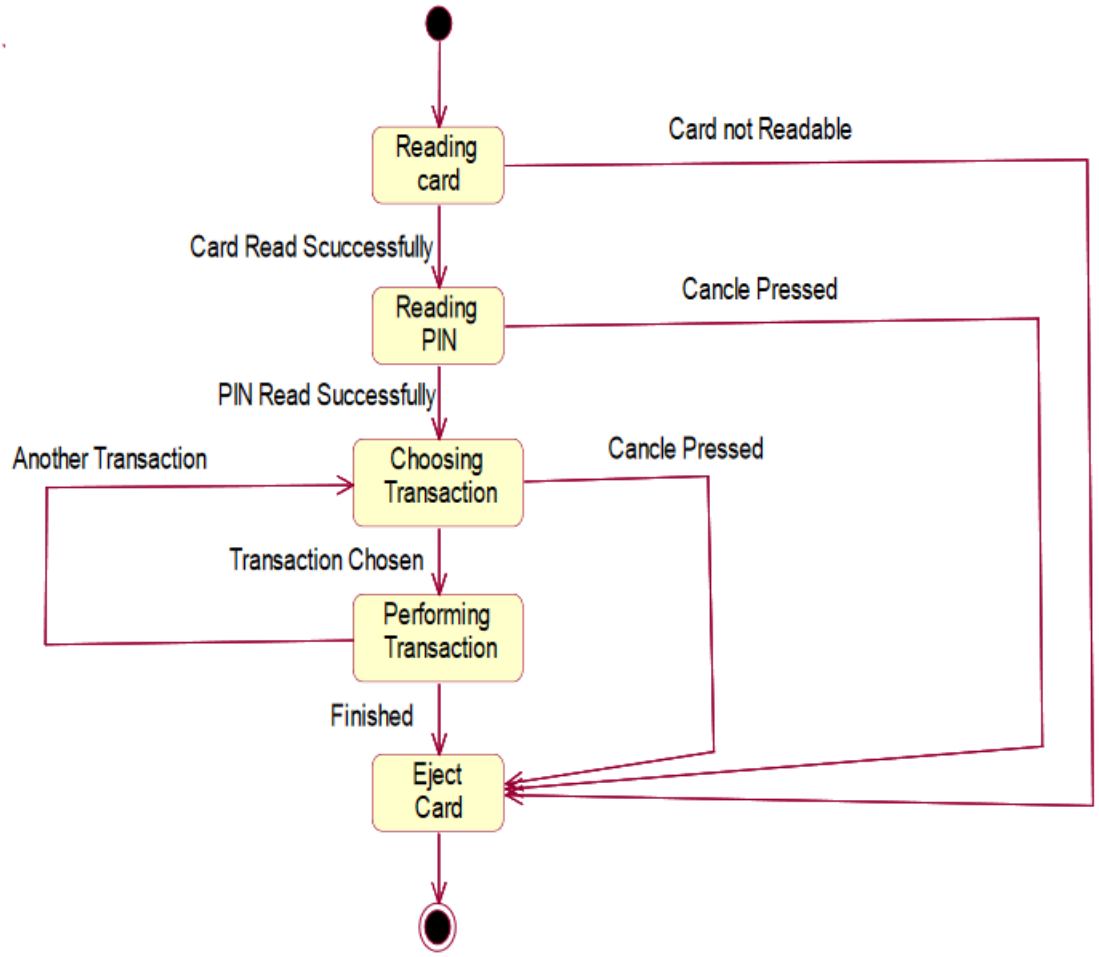


Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

4. CLASS DIAGRAM:

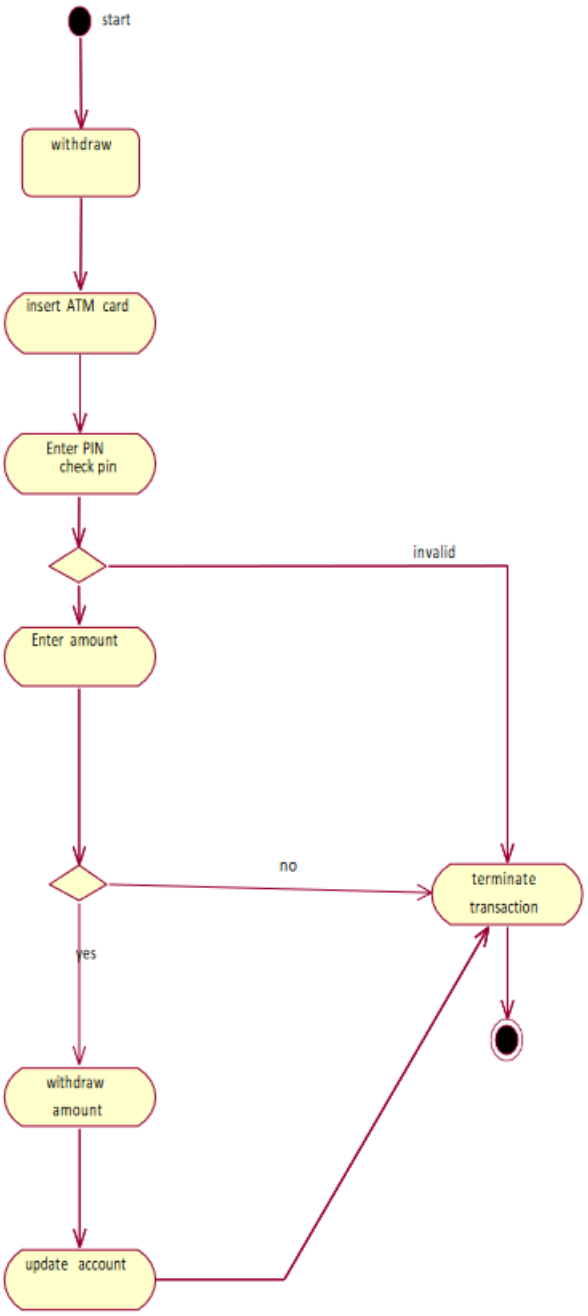


5. STATE CHART DIAGRAM:

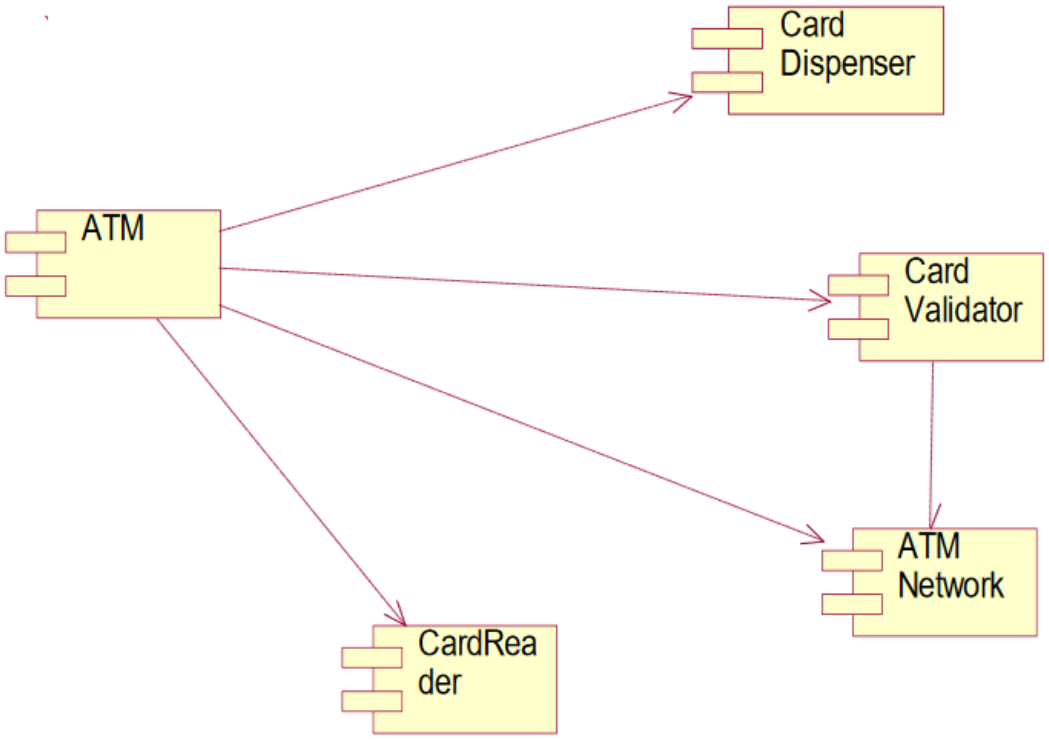


Exp No: 03	TITLE: UML DIAGRAM FOR ATM APPLICATION	Date: 13/05/2022

6. ACTIVITY DIAGRAM:



7. COMPONENT DIAGRAM:



8. DEPLOYMENT DIAGRAM: