

NOIDA INSTITUTE OF ENGINEERING & TECHNOLOGY, GREATER NOIDA



SOFTWARE ENGINEERING (KCS-601) DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY:

Divya Kunwar

1901330100100 (CSE – B)

SUBMITTED TO:

Ashish Mathur

Assistant Professor

INDEX

S.NO	Name of Practical	Date	Remark
1	Prepare a SRS document in line with IEEE recommended standards.		
2	Draw the use case diagram and specify the role of each of the actors. Also state the precondition, postcondition, and function of each use case		
3	Draw the activity diagram.		
4	Identify the classes. Classify them as weak and strong classes and draw the class diagram.		
5	Draw the sequence diagram for any two scenarios.		
6	Draw the collaboration diagram.		
7	Draw the state chart diagram.		
8	Draw the component diagram.		
9	Perform forward engineering in java. (Model to code conversion)		
10	Perform reverse engineering in java. (Code to Model conversion)		
11	Draw the deployment diagram.		

Practical 1: Prepare a SRS document in line with IEEE recommended standards.

SRS for Automated Teller Machine System with IEEE standards

1 Introduction

1.1 Purpose

This document describes the software requirements and specification for an automated teller machine (ATM) network. The document is intended for the customer and the developer (designers, testers, maintainers).

1.2 Document Conventions

Account

A single account at a bank against which transactions can be applied. Accounts may be of various types with at least checking and savings. A customer can hold more than one account.

Max Daily WD

The maximum amount of cash that a customer can withdraw from an account in a day (from 00:00 AM to 23:59 PM) via ATMs. **PIN**

It refers to Personal Identification Number. Used to identify and validate the login of an ATM user.

1.3 Intended Audience and Reading Suggestions

- x Software designers
- x Systems engineers
- x Software developers
- x Software testers
- x Customers

1.4 Product Scope

The network enables customers to complete simple bank account services via automated teller machines (ATM). The ATM identifies a customer by a cash card and password. It collects information about a simple account. The bank owns software for their own computers transaction (e.g. deposit, withdrawal, transfer, bill payment).

1.5 Reference

References for above software are :

- i www.google.in
- ii www.wikipedia.com
- iii IEEE. Software Requirements Specification Std. 830-1993.
- iv www.onlinesbi.com
- v www.pnbindia.in

2. Overall Description

2.1 Product Perspective

An automated teller machine (ATM) is a computerized telecommunications device that provides the customers of a financial institution with access to financial transactions in a public space without the need for a human clerk or bank teller. On most modern ATMs, the customer is identified by inserting a plastic ATM card with a magnetic stripe or a plastic smartcard with a chip, that contains a unique card number and some security information, such as an expiration date or CVC (CVV). Security is provided by the customer entering a personal identification number (PIN).

2.2 Product Functions

Using an ATM, customers can access their bank accounts in order to make cash withdrawals (or credit card cash advances) and check their account balances.

The functions of the system are

1. Login
2. Get balance information
3. Withdraw cash
4. Transfer funds

2.3 User Classes and Characteristics

a) User

This actor is the person who uses the software.

b) BANK/ATM

This actor represents the financial institute that provides services to ATM. Responsible for verifying bank customers, authorizing transaction and recording completed transactions.

2.4 Operating Environment

The ATM is a network based mechanical device and shall operate in all environments, for a model we are taking Tidal 3600m.

2.5 Design and Implementation Constraints

Some of the constraints that have to be taken care of with respect to the software development could be the platform where the software is to be run should have an access to MS-ACCESS. The end user needs to have basic computer knowledge.

The user may fail to exactly explain his requirement however his requirements are analyzed and understood by the developer, as the software is a generic one

2.6 User Documentation

Online help is provided for each of the feature available with the ATM. Online help is provided for each and every feature provided by the system. The user manual should be available as a hard copy and also as online help. An installation document will be provided that includes the installation instructions and configuration guidelines, which is important to a full solution offering.

2.7 Assumptions and Dependencies

1. Hardware never fails
2. ATM casing is impenetrable
3. Limited number of transactions per day (sufficient paper for receipts)
4. Limited amount of money withdrawn per day (sufficient money)

3. External Interface Requirements

3.1 User Interfaces

The customer user interface should be intuitive, such that 99.9% of all new ATM users are able to complete their banking transactions without any assistance.

3.2 Hardware Interfaces

The hardware should have following specifications:

- x Ability to read the ATM card
- x Ability to count the currency notes
- x Touch screen for convenience
- x Keypad (in case touchpad fails)
- x Continuous power supply
- x Ability to connect to bank network
- x Ability to take input from user
- x Ability to validate user

3.3 Software Interfaces

The software interfaces are specific to the target banking software systems.

1. Languages supported: java (Front end)
2. Database: Microsoft Access (Back end)
3. MS-Office
4. ArgoUml

At present, two known banking systems will participate in the ATM network.

- x State Bank

- x Indian Overseas Bank

3.4 Communications Interfaces

These are protocols that are needed to directly interact with the customers. Apart from these protocols, to maintain a healthy relationship with the customer, both formal and informal meetings, group discussions and technical meetings will be conducted frequently.

4. System Features

This section of SRS should contain all of the software requirements and specification. At a minimum, it should include descriptions.

- x All interfaces to the system
 1. Every input to the system
 2. Every output from the system
- x All functions performed by the system are:
 1. Validity checks on inputs
 2. Relationship of outputs to inputs
 3. Responses to abnormal situations

Input and output definitions should be consistent among use cases, functional specifications and UI .

4.1 Use Cases

4.1.1 Login

This is a use case used to verify the authentication of the user. In this the user gives his allotted pin number as input, the system the verifies whether the card number and pin number stored in data base matches or not, if it matches then it allows the user to use the system else it asks to enter the pin number again.

4.1.2 Balance Enquiry

This use case is used to check the balance in the user account. After every transaction the balance in account balance is displayed to the user.

4.1.3 Withdrawal

This use case facilitates the user to withdraw money from his account. After the money is withdrawn it is deducted from the account.

4.1.4 Generate Receipt

This use case is used to generate receipt for the transaction made by the user.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

It must be able to perform in adverse conditions like high/low temperature etc.

Uninterrupted interrupted connections High
data transfer rate

5.2 Security Requirements

- x Users accessibility is censured in all the ways
- x Users are advised to change their PIN on first use
- x Users are advised not to tell their PIN to anyone
- x The maximum number of attempts to enter PIN will be three

5.3 Safety Requirements

- x Must be safe kept in physical aspects, say in a cabin
- x Must be bolted to floor to prevent any kind of theft.
- x Must have an emergency phone outside the cabin
- x There must be an emergency phone just outside the cabin
- x The cabin door must have an ATM card swipe slot
- x The cabin door will always be locked, which will open only when user swipes his/her ATM card
- x In the slot & is validated as genuine

5.4 Business Rules

- x Personal information should be protected
- x The ATM should comply with quality assurance standards
- x The Money circulation should be monitored and ATM should be filled regularly.

6. Other Requirements

The ATM should be implemented on computers with 50Mbytes free space on HDD for Database (80Gbytes for server) and 32Mbytes RAM for Database (256Mbytes for server)

The ATM should correctly interface if MS Access applications and MS SQL Server

Practical 2: Draw the use case diagram and specify the role of each of the actors. Also state the precondition, postcondition, and function of each use case.

UML Use Case Diagram:

A use case diagram is used to represent the dynamic behaviour of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purpose of Use Case Diagrams:

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Precondition:

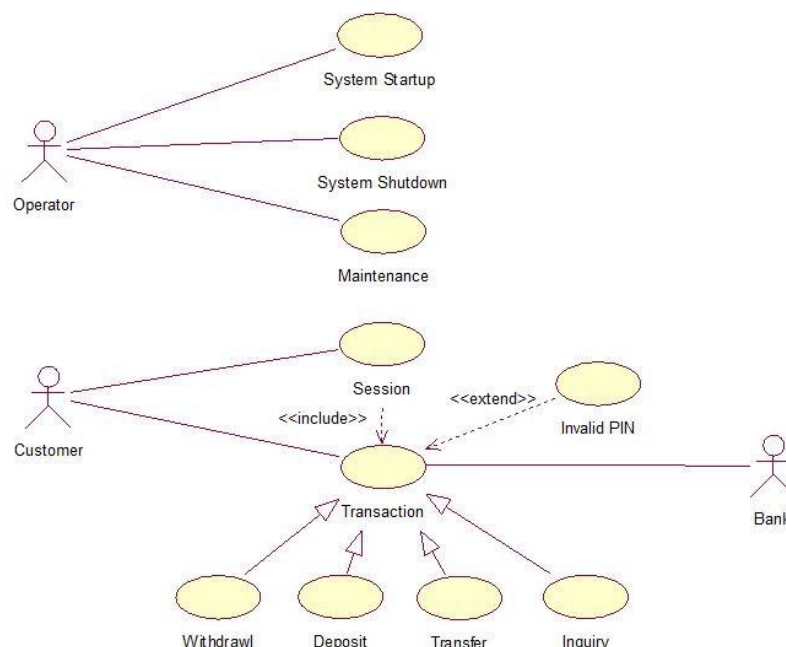
Customer should know about its credentials.

Postcondition:

Customer must logout from the system after completing its operation.

Example of a Use Case Diagram:

A use case diagram depicting the ATM is given below.



Practical 3: Draw the activity diagram.

UML Activity Diagram:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

Purpose of Activity Diagrams:

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another, but activity diagram is used to show message flow from one activity to another.

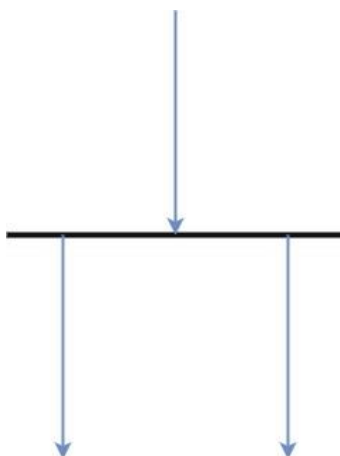
The purpose of an activity diagram can be described as – 1.

Draw the activity flow of a system.

2. Describe the sequence from one activity to another.
3. Describe the parallel, branched and concurrent flow of the system.

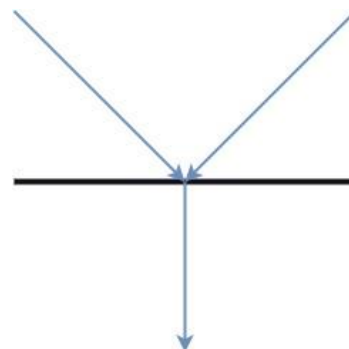
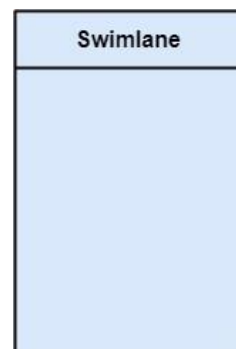
Components of an Activity Diagram:

Activities:



Forks:

Activity partition /Swimlane:



Join Nodes:

Notation of an Activity diagram:

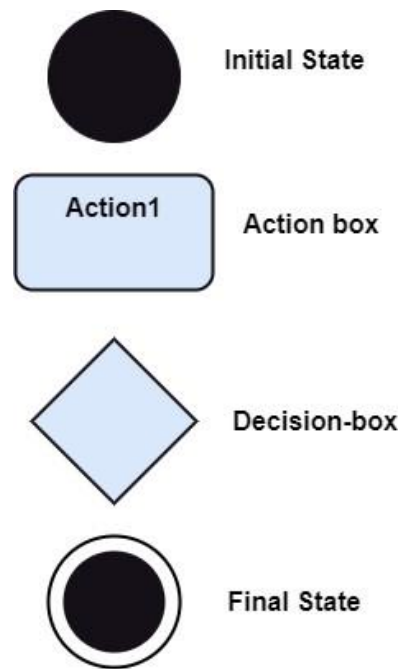
Activity diagram constitutes following notations:

Initial State: It depicts the initial stage or beginning of the set of actions.

Final State: It is the stage where all the control flows and object flows end.

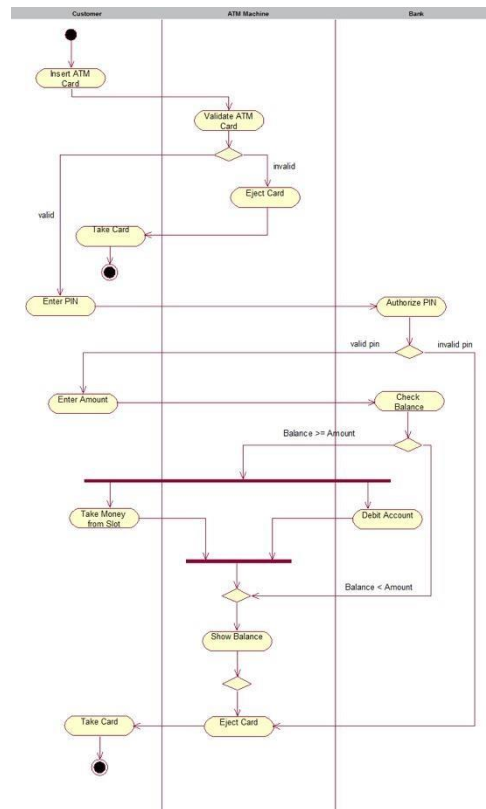
Decision Box: It makes sure that the control flow or object flow will follow only one path.

Action Box: It represents the set of actions that are to be performed.



Example of a Activity Diagram:

An activity diagram depicting the ATM is given below.



Practical 4: Identify the classes. Classify them as weak and strong classes and draw the class diagram.

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

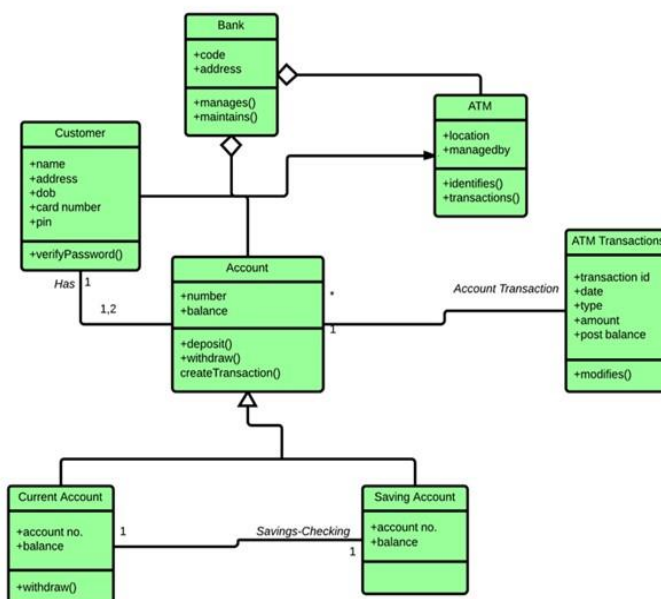
Purpose of Class Diagrams:

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

The purpose of the class diagram can be summarized as – 1.

- Analysis and design of the static view of an application.
2. Describe responsibilities of a system.
3. Base for component and deployment diagrams.
4. Forward and reverse engineering.

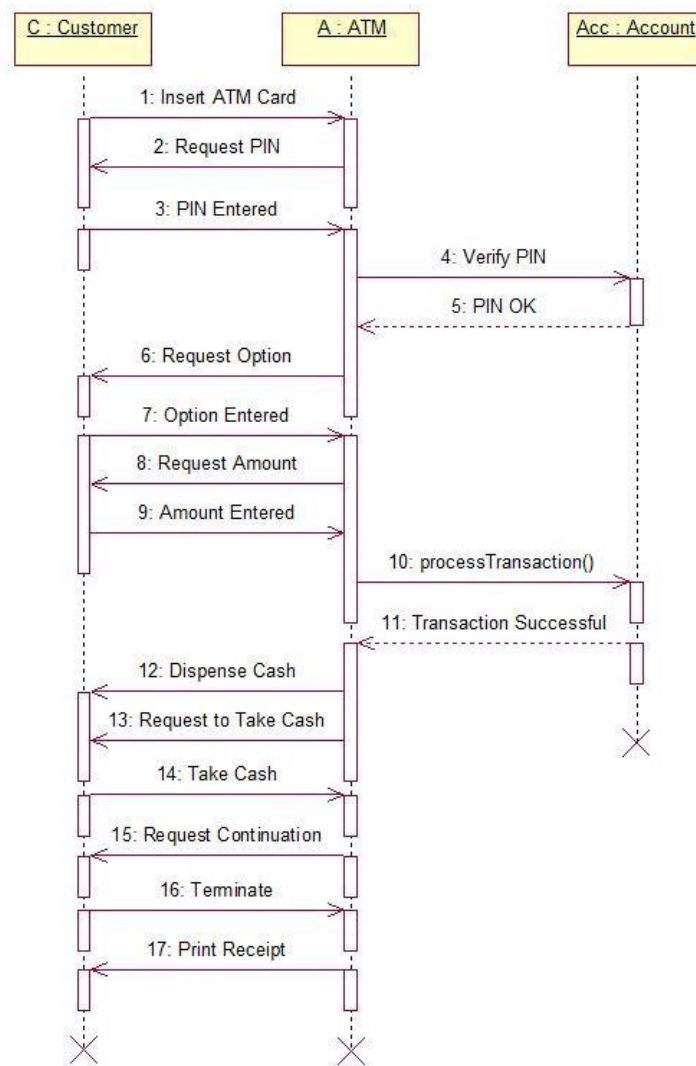
Example of a Class Diagram for ATM:



Practical 5: Draw the sequence diagram for any two scenarios.

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

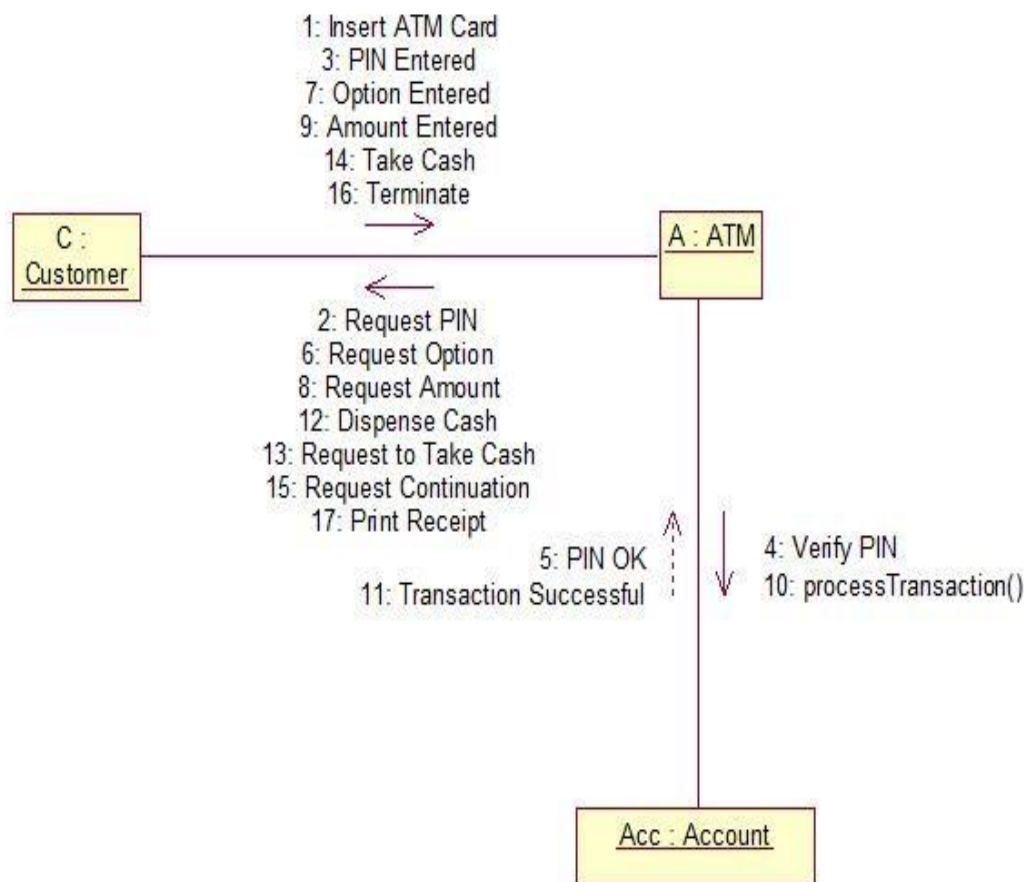
This is the sequence diagram of ATM.



Practical 6: Draw the collaboration diagram.

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

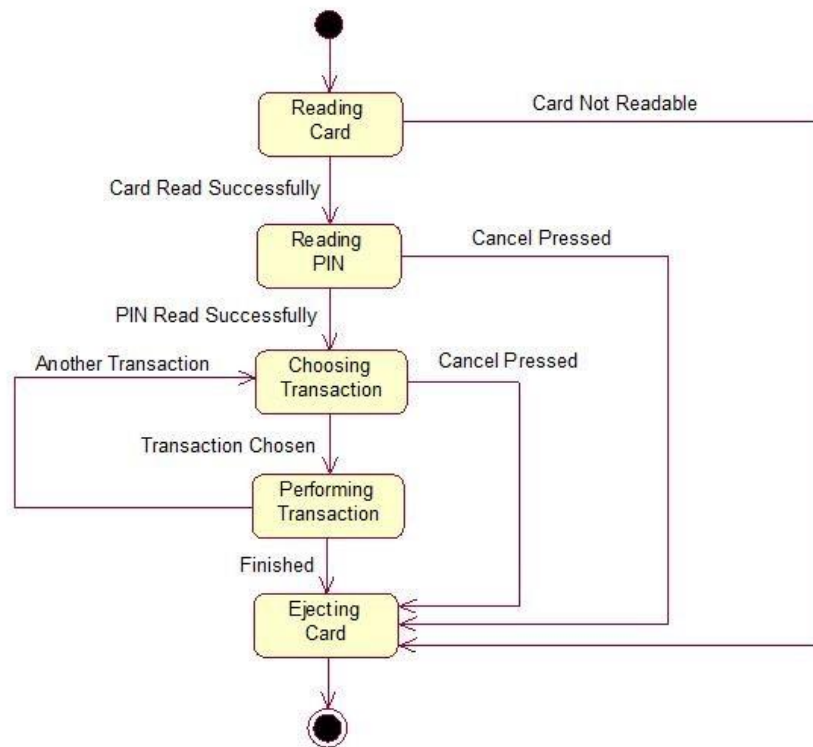
This is the collaboration diagram of ATM:



Practical 7: Draw the state chart diagram.

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems.

This is the State chart diagram of ATM.

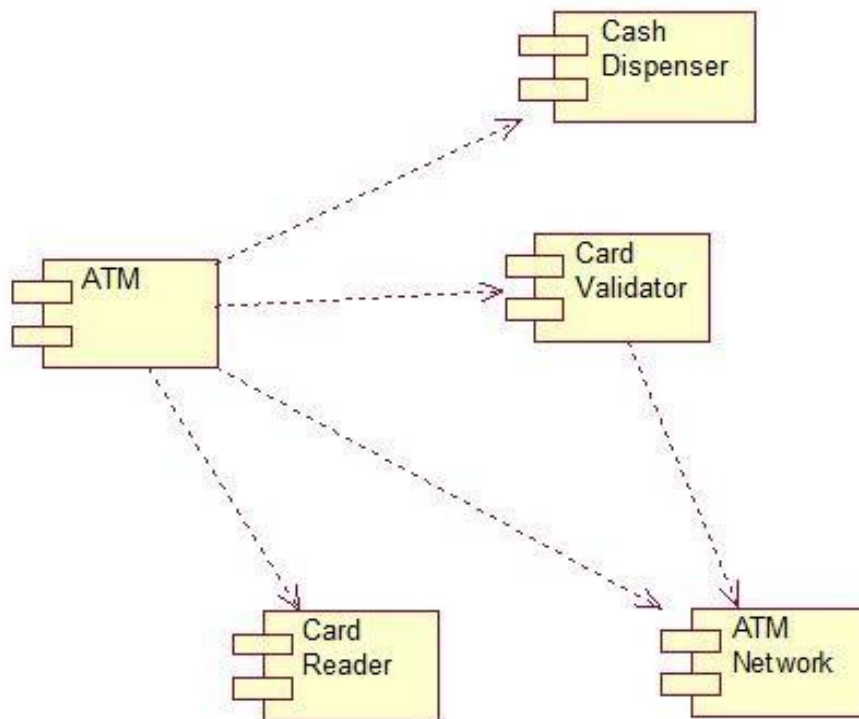


Practical 8: Draw the component diagram.

Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the

Organization and relationships among components in a system. These diagrams are also used to make executable systems.

This is the Component diagram of ATM.



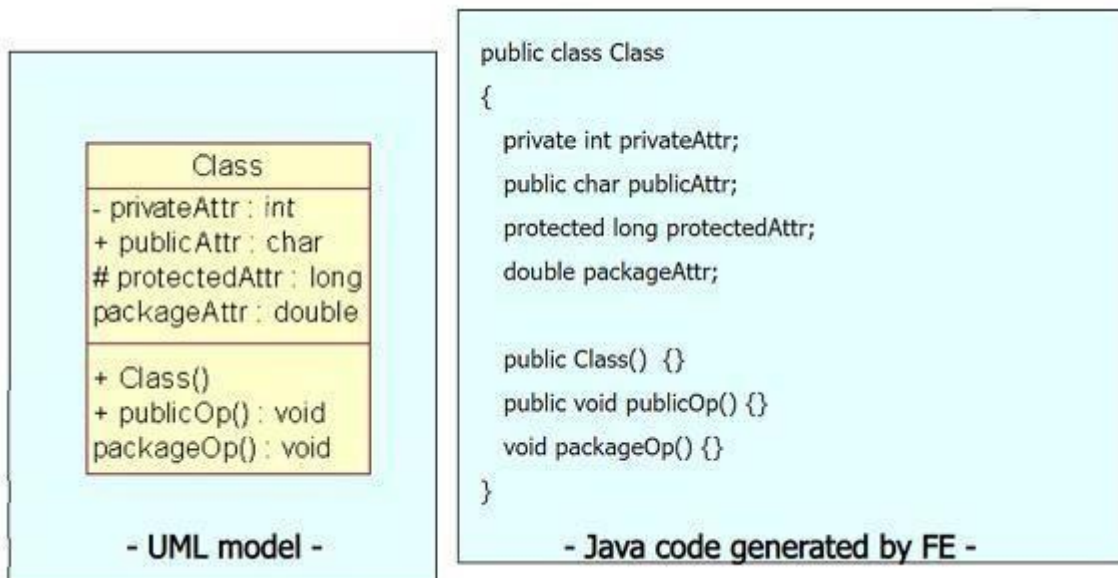
Practical 9: Perform forward engineering in java. (Model to code conversion)

Forward Engineering:

Forward Engineering is a method of creating or making an application with the help of the given requirements. Forward engineering is also known as Renovation and Reclamation. Forward engineering is required high proficiency skills. It takes more time to construct or develop an application.

Forward engineering is a technique of creating high-level models or designs to make in complexities and low-level information. Therefore, this kind of engineering has completely different principles in numerous package and information processes.

Forward Engineering applies of all the software engineering process which contains SDLC to recreate associate existing application. It is near to full fill new needs of the users into re-engineering.



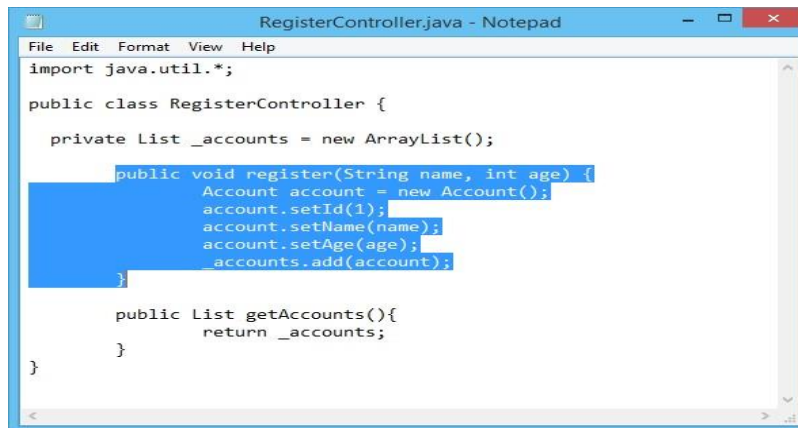
Practical 10: Perform reverse engineering in java. (Code to Model conversion)

Reverse Engineering is also known as backward engineering, is the process of forward engineering in reverse.

In this, the information is collected from the given or existing application.

It takes less time than forward engineering to develop an application. In reverse engineering, the application is broken to extract knowledge or its architecture.

1. Study the source code. Read the register method in RegisterController.java to see how it works.



```
RegisterController.java - Notepad
File Edit Format View Help
import java.util.*;

public class RegisterController {

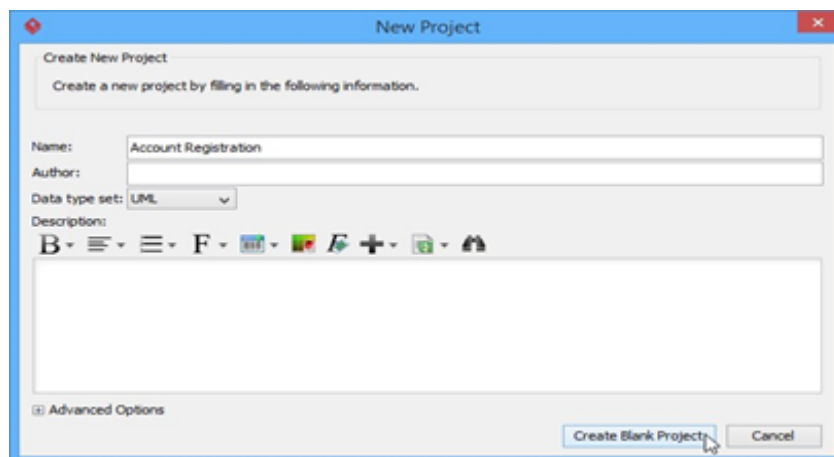
    private List _accounts = new ArrayList();

    public void register(String name, int age) {
        Account account = new Account();
        account.setId(1);
        account.setName(name);
        account.setAge(age);
        _accounts.add(account);
    }

    public List getAccounts(){
        return _accounts;
    }
}
```

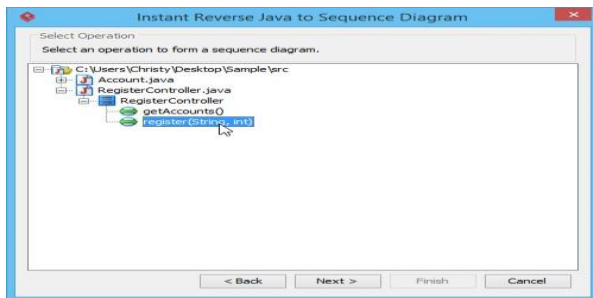
2. Start Visual Paradigm

3. Create a new project by selecting **Project > New** from the toolbar. In the **New Project** window, name the project as Account Registration and click **Create Blank Project** button.



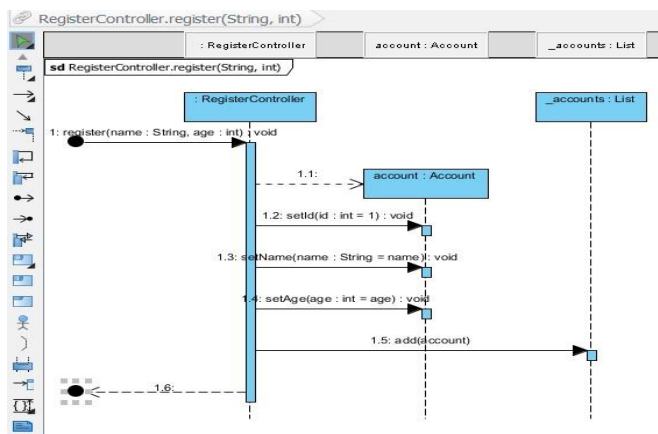
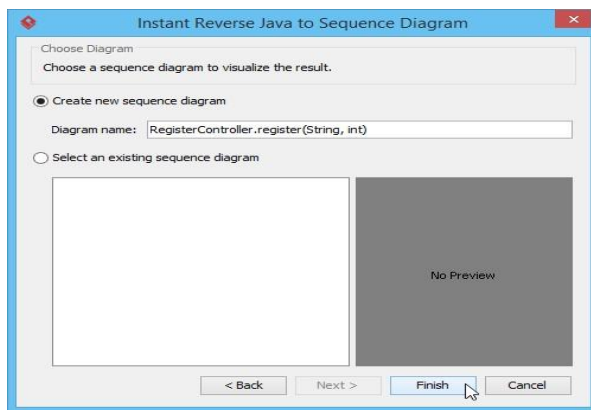
4. Select **Tools > Code > Instant Reverse Java to Sequence Diagram...** from the toolbar
5. In the **Instant Reverse Java to Sequence Diagram** window, click on **Add Source Folder...** button.
6. Select the extracted source folder src. Click **Next** button.

7. Select the method to visualize. Select **src > RegisterController.java > register (String,int)**. Click the Next button.



8. You need to select a diagram to visualize the interaction. The **Create new sequence diagram** option is selected and diagram name is entered by default. Click **Finish** button.

9. As a result, a sequence diagram is formed. x



When a person invokes Register Controller's register method (message: 1), it creates an account object (message: 1.1). After that, the controller sets the id, name and age to the account object (message 1.2, 1.3, 1.4) and adds itself to the account list (message: 1.5). The invocation ends with a return (message 1.6).

Practical 11: Draw the deployment diagram.

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

This is deployment diagram of ATM:

