INDEX

S.No.	Experiment Name	Experiment Date	Submitted Date	Remarks	Signature
1	Prepare a SRS document in line with the IEEE recommended standards.				
2	Draw the use case diagram and specify the role of each of the actors. Also state the precondition, post condition 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)				

Experiment No:-1

OBJECTIVE:- Prepare a SRS document in line with the IEEE recommended standards.

Theory:-

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

The software requirements specification document enlists enough and necessary requirements that are required for the project development. To derive the requirements we need to have clear and thorough understanding of the products to be developed or being developed. This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

Implementation:

SRS For Railway Reservation System:

1.Introduction

1.1.Purpose

The purpose of this source is to describe the railway reservation system which provides the train timing details, reservation, billing and cancellation on various types of reservation namely,

- Confirm Reservation for confirm Seat.
- Reservation against Cancellation.
- Waiting list Reservation.
- Online Reservation.
- Tatkal Reservation.

1.2.SCOPE

“Railways Reservation System” is an attempt to simulate the basic concepts of an online Reservation system. The system enables to perform the following functions:

- SEARCH FOR TRAIN
- BOOKING OF A SELECTED TRAIN
- PAYMENT
- CANCELLATION
- PASSENGER AND REVENUE ENHANCEMENT

- IMPROVED AND OPTIMIZED SERVICE

1.3.OBJECTIVE

General Objective:

Software has to be developed for automating the manual Railway Reservation System.

Specific Objectives:

Some of the Specific Objectives of the system are listed below:

- RESERVE SEATS – Reservation form has to be filled by passenger. If seats are available entries like train name, number, destination are made.
- CANCEL RESERVATION- The clerk deletes the entry in the System and changes in the Reservation Status.
- VIEW RESERVATION STATUS-The user need to enter the PIN number printed on ticket

1.4.OVERVIEW

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. General description of the project is discussed in section 2 of this document. Section 3 gives the functional requirements, data requirements and constraints and assumptions made while designing the E-Store. It also gives the user viewpoint of product. Section 3 also gives the specific requirements of the product. Section 3 also discusses the external interface requirements and gives detailed description of functional requirements. Section 4 is for supporting information.

1.5.REFERENCES

Some of the references used for preparing the vision document include:

1. IEEE document for Software Requirements Specifications
2. Wikipedia

2. FUNCTIONAL REQUIREMENTS

1. Login and registration: This function allows a new user to register himself on the railway reservation system and lets him login on the system using the username and password supplied during registration.

2. History of booking: This function allows a registered user to view all the booking and transactions allotted to him during booking the ticket.

3. Check PNR status: It allows the user to check the status of his reserved birth using the PNR no. allotted to him during booking the ticket.

4. Seat Availability: This function lets the user check the availability of seats in the train selected by user.

5. Search: It allows user to view all the trains that are running from the source station to the destination on the given date as entered by user during form filling.

6. Live station: This function list all the trains leaving or going to the specified station as supplied by the user.

7. Booking: The user supplies all the information in the railway reservation form and can book his ticket.

8. Ticket cancellation: This function allows user to cancel his reservation whether the user status is confirmed or waiting.

9. Running status: It is used to spot our train. The user enters a train number and we get to know details like the current station where the train is, delay, and expected time of arrival.

3.INTERFACES REQUIREMENTS:

3.1.USER INTERFACE

Keyboard and mouse

3.2.HARDWARE INTERFACE

For the hardware requirements the SRS specifies the logical characteristics of each interface b/w the software product and the hardware components. It specifies the hardware requirements like memory restrictions, cache size, the processor, RAM size etc... those are required for the software to run.

Minimum Hardware Requirements

Processor Pentium III

Hard disk drive 40 GB

RAM 128 MB

Cache 512 kb

Preferred Hardware Requirements

Processor Pentium IV

Hard disk drive 80 GB

RAM 256 MB

Cache 512 kb

3.3.SOFTWARE INTERFACE

- Any window based operating system with DOS support are primary requirements for software development. Windows XP, FrontPage and dumps are required. The systems must be connected via LAN and connection to internet is mandatory.

4. Nonfunctional Requirements

Security:
The system use SSL (secured socket layer) in all transactions that include any confidential customer information. The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's password. The system's back-end servers shall only be accessible to authenticated management.

Reliability:
The reliability of the overall project depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. Also the system will be functioning inside a container. Thus the overall stability of the system depends on the stability of container and its underlying operating system.

Availability:
The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. A customer friendly system which is in access of people around the world should work 24 hours. In case of a of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the Organizer. Then the service will be restarted. It means 24 x 7 availability.

5.Conclusion

This SRS document is used to give details regarding Railway Reservation System. In this all the functional and non-functional requirements are specified inorder to get a clear cut idea to develop a project.

Output / Conclusion:- SRS is developed as per requirements

Experiment No:-2

OBJECTIVE: - Draw the use case diagram and specify the role of each of the actors. Also state the precondition, post condition and function of each of the use case.

Theory:-

Concepts:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.
So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.
Functionalities to be represented as an use case Actors.
Relationships among the use cases and actors.

Implementation:

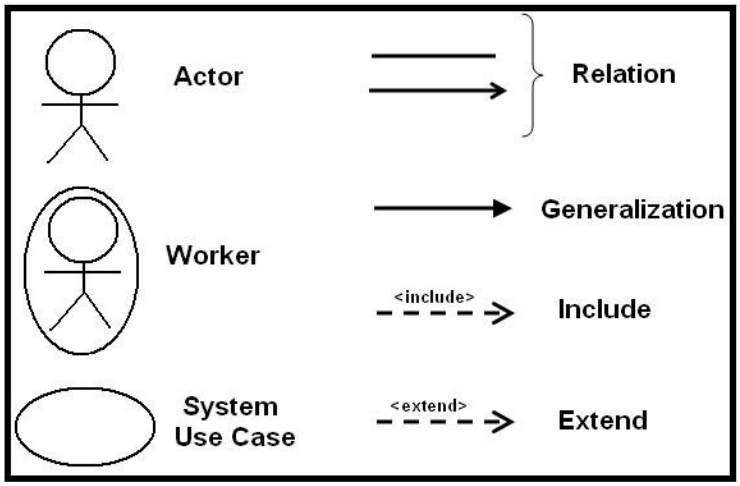
Use case diagram For Railway Reservation System:

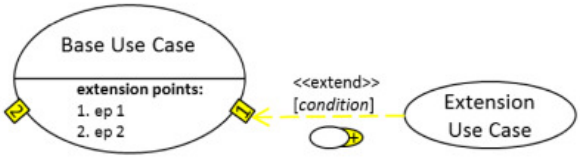
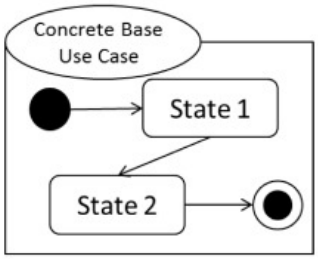
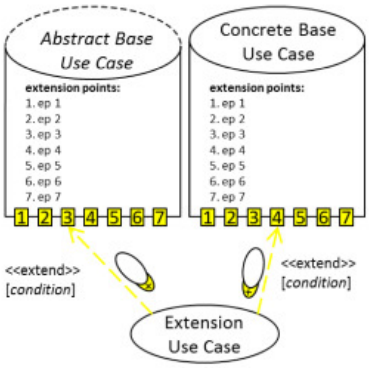
Role of each of the actors:
Clerk: He is responsible to take printout and cancel the ticket.
Customer: He able to perform reservation of train ticket through online system
Railway website: It is taken as actor to show data flows needed by web site to perform online reservation functionalities.

Precondition, Post condition:
Use case: Enquiry Ticket Availability
Precondition: Schedule date, Source and Destination
Post condition : None

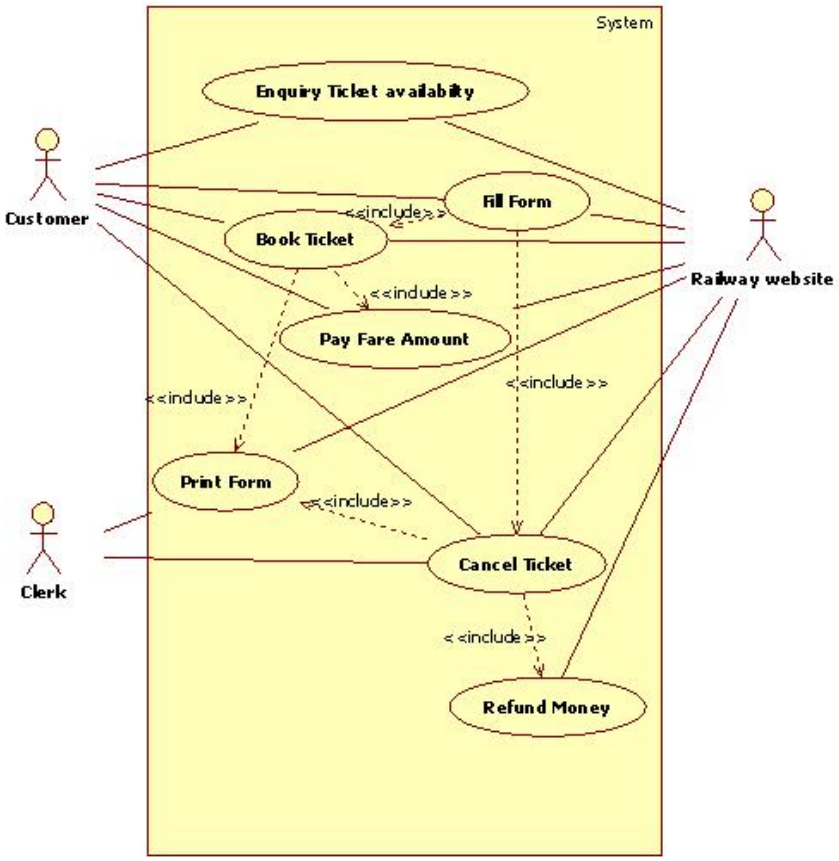
Use case: Book Ticket
Precondition: Schedule date, source, destination, no. of persons
Post condition: Update the log file

Notations used:-



Visual Symbol	Semantic Construct
	The include relationship.
	The extend relationship.
	The extend relationship with a specified extension point and condition.
	An abstract use case.
	Concrete use case as a classifier
	Concrete and abstract use cases with many extension points.

Class Diagram Relationship Type	Notation
Association	
Inheritance	
Realization/ Implementation	
Dependency	
Aggregation	
Composition	



Use case: Cancel Ticket
Precondition: PNR no.
Post Condition: Free the seats and available for booking

Include Relationship:
The "include" relationship denotes that one use case includes the functionality of another use case. It represents a common behaviour that is shared by multiple use cases. The included use case is always executed whenever the base use case is executed. The arrow in the diagram points from the base use case to the included use case.
Example: In an online shopping system, the "Checkout" use case may include the "Login" and "View Cart" use cases. This means that whenever a customer proceeds to checkout, they must first log in and view their shopping cart.

Exclude Relationship:
The "exclude" relationship denotes that one use case excludes the functionality of another use case under certain conditions. It represents a situation where executing one use case prevents or cancels the execution of another use case. The excluded use case is not executed when the conditions specified by the excluding use case are met. The arrow in the diagram points from the excluding use case to the excluded use case.
Example: In a social media platform, the "Delete Account" use case may exclude the "Post Content" use case. This means that when a user decides to delete their account, all their posts will be deleted, so they cannot post any new content.

Output/Conclusion:- The use case diagram is made successfully.

Experiment No:-3

OBJECTIVE: - Draw the activity diagram.

Theory:-

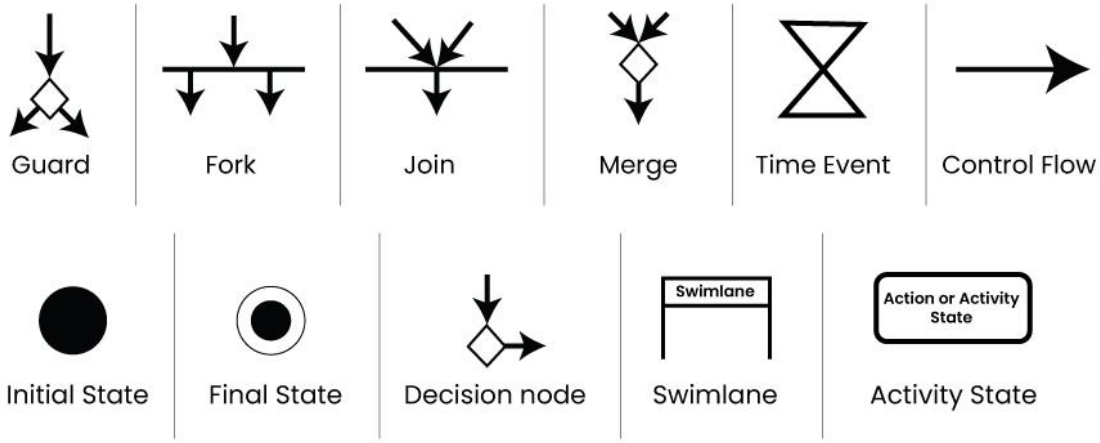
Concepts:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

Notations used:-



Initial State

The starting state before an activity takes place is depicted using the initial state.
A process can have only one initial state unless we are depicting nested activities. We use a black filled circle to depict the initial state of a system. For objects, this is the state when they are instantiated. The Initial State from the UML Activity Diagram marks the entry point and the initial Activity State.

Action or Activity State

An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

Action Flow or Control flows

Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another activity state.

An activity state can have multiple incoming and outgoing action flows. We use a line with an arrow head to depict a Control Flow. If there is a constraint to be adhered to while making the transition it is mentioned on the arrow.

Decision node and Branching

When we need to make a decision before deciding the flow of control, we use the decision node. The outgoing arrows from the decision node can be labelled with conditions or guard expressions. It always includes two or more output arrows.

Guard

A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.

The statement must be true for the control to shift along a particular direction. Guards help us know the constraints and conditions which determine the flow of a process.

Fork

Fork nodes are used to support concurrent activities. When we use a fork node when both the activities get executed concurrently i.e. no decision is made before splitting the activity into two parts. Both parts need to be executed in case of a fork statement. We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent activity state and outgoing arrows towards the newly created activities.

Join

Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.

Merge or Merge Event

Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.

Swimlanes

We use Swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activity diagram. It is not mandatory to use swimlanes. They usually give more clarity to the activity diagram. It’s similar to creating a function in a program. It’s not mandatory to do so, but, it is a recommended practice.

We use a rectangular column to represent a swimlane as shown in the figure above.

Time Event

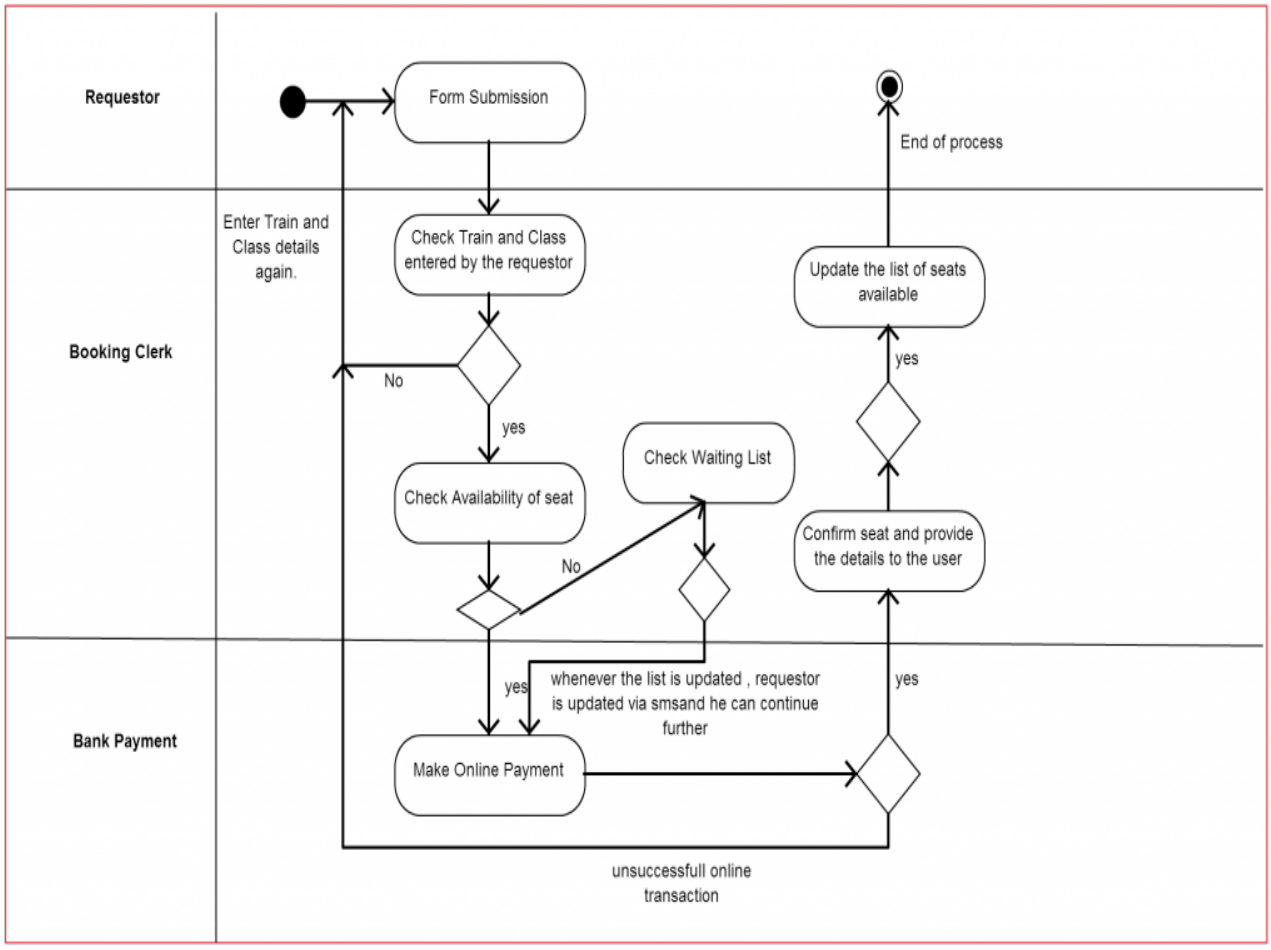
This refers to an event that stops the flow for a time; an hourglass depicts it. We can have a scenario where an event takes some time to completed.

Final State or End State

The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

Implementation:

Activity diagram For Railway Reservation System:



Output/Conclusion:- The activity diagram was made successfully.

Experiment No:- 4

OBJECTIVE:- Identify the classes. Classify them as weak and strong classes and draw the class diagram.

Theory:-

Concepts:

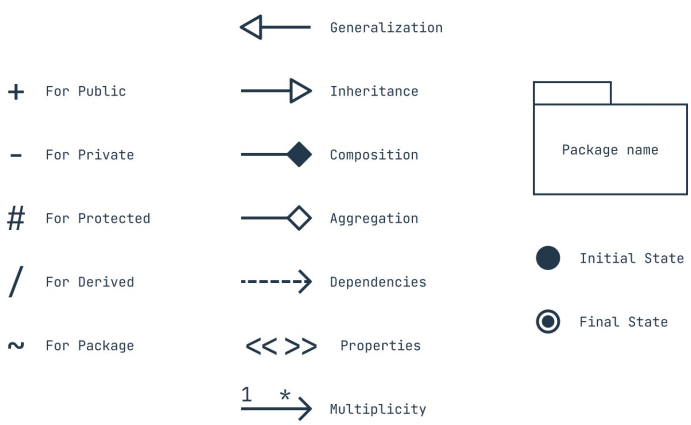
A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

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

The class diagram describes the attributes and operations of a class and also 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.The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.
To specify the visibility of a class member (i.e. any attribute or method), these notations must be placed before the member's name.

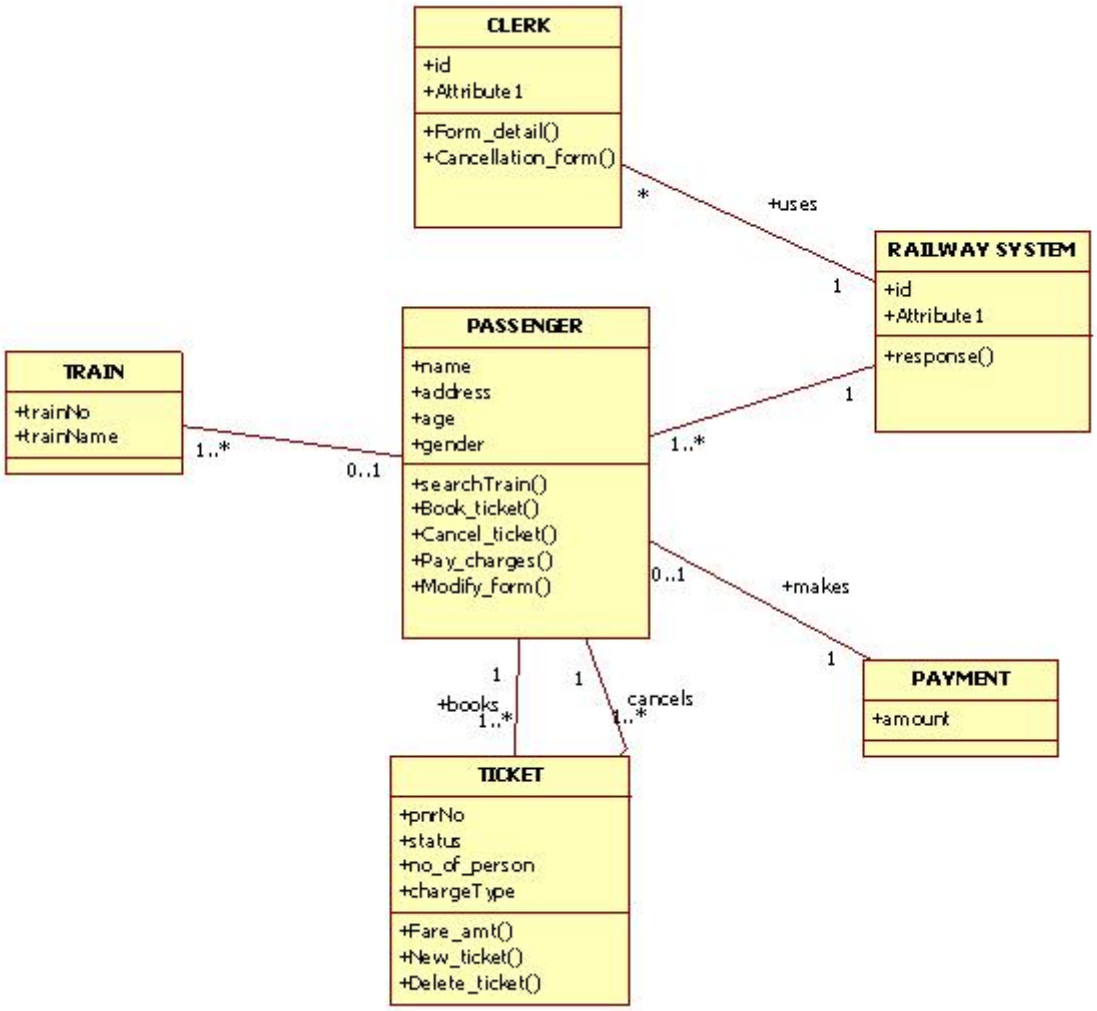
+	Public
-	Private
#	Protected
/	Derived (can be combined with one of the others)
~	Package

Notations used:-



Notation	Visibility
+	Public
-	Private
#	Protected
~	Package

Class diagram For Railway Reservation System



Output/Conclusion:- The class diagram was made successfully.

Experiment No:- 5

OBJECTIVE:- Draw the sequence diagram for any two scenarios.

Theory:-

Concepts:

Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange.

Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom.

Implementation:

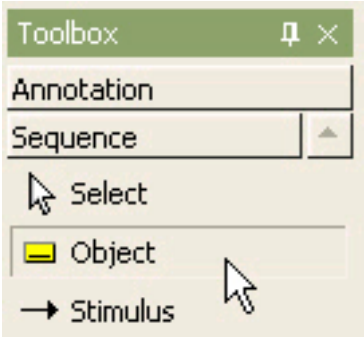
There are many elements in a sequence diagram, let's see how to create an **Object** using StarUML.

Object:

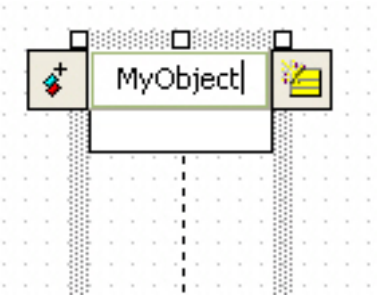
Procedure for creating object:

In order to create object,

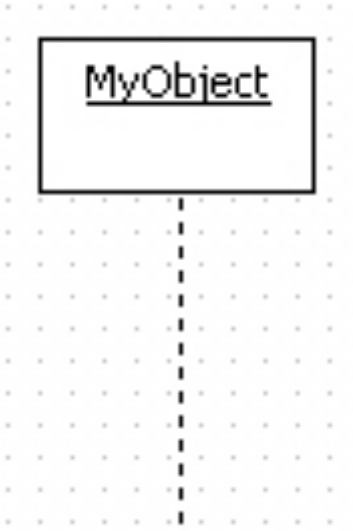
- Click [Toolbox] -> [Sequence] -> [Object] button.



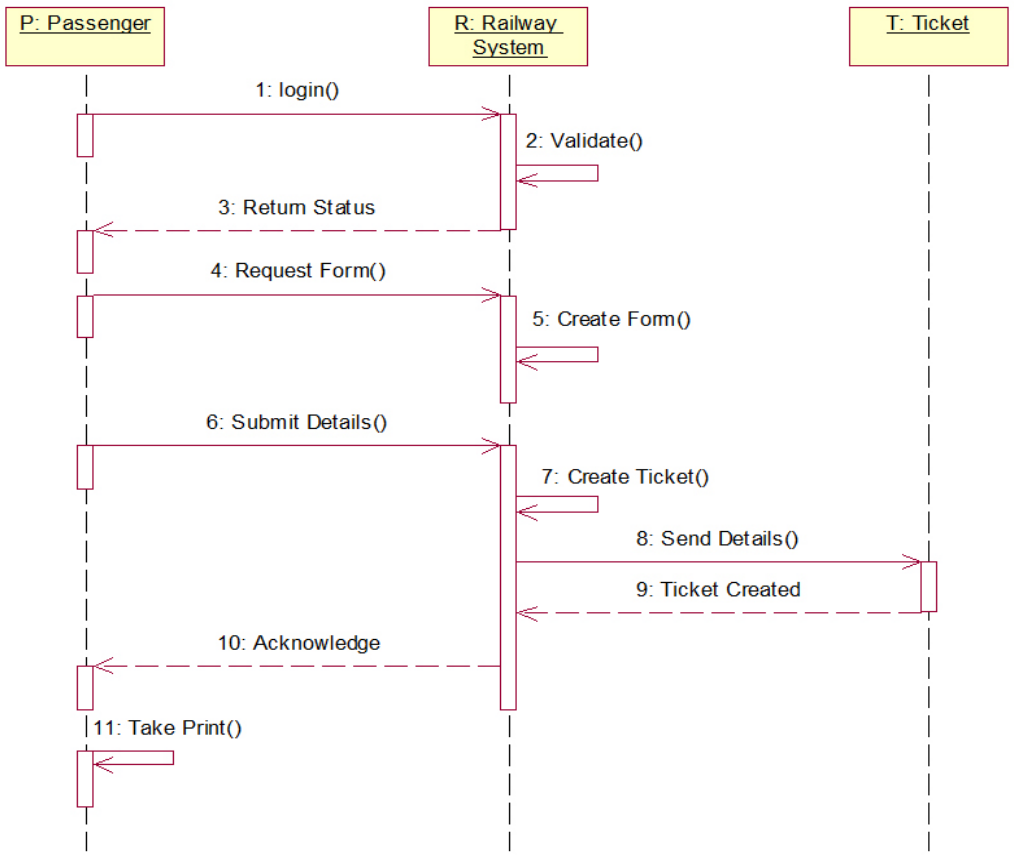
- And click at the position where object will be placed in the [main window]. Object quick dialog is shown. At the quick dialog, enter the object name.



- Press [Enter] key.

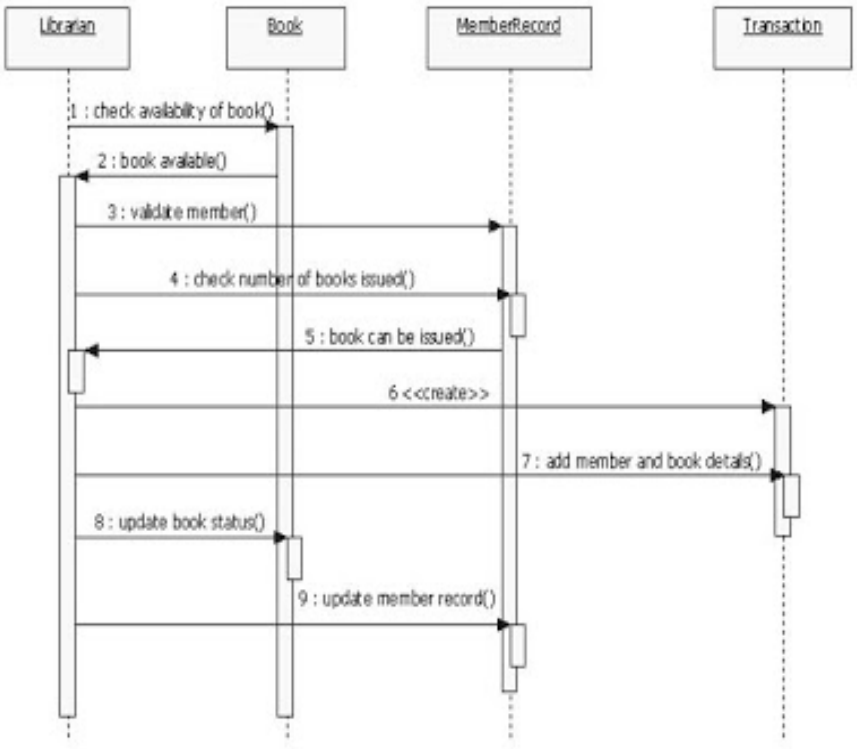


1. **Sequence Diagram for Railway Reservation System**

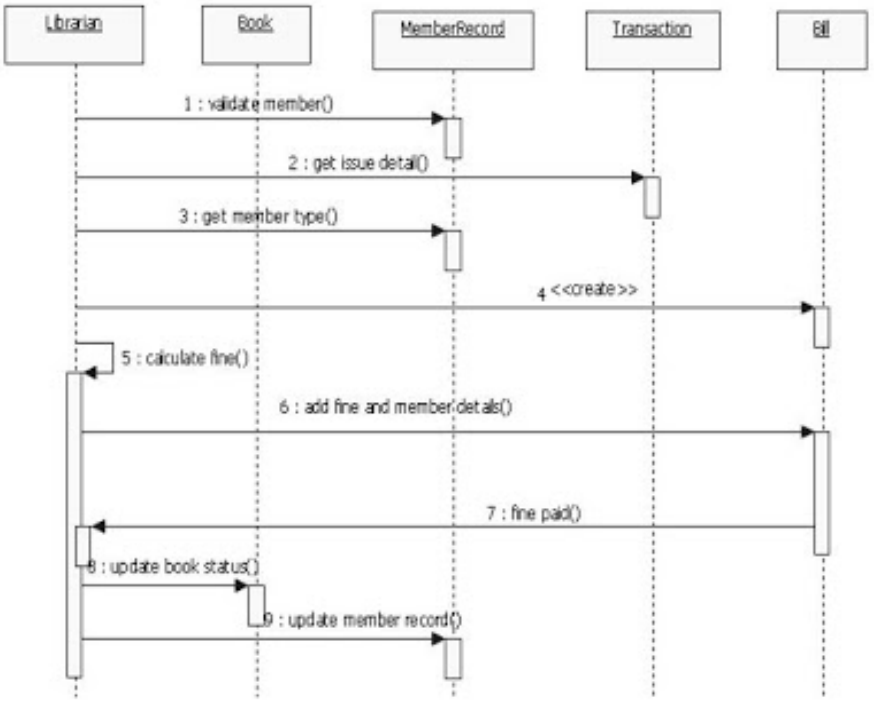


2. Sequence Diagram for Library Management system

a. Book Issue



b. Book Return



Output/Conclusion:- The sequence diagram was made successfully.

Experiment No:- 6

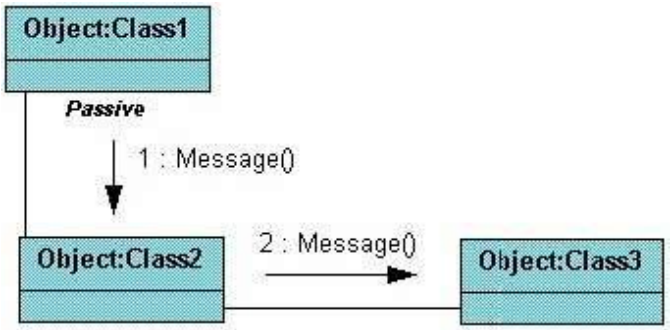
OBJECTIVE:- Draw the collaboration diagram.

Theory:-

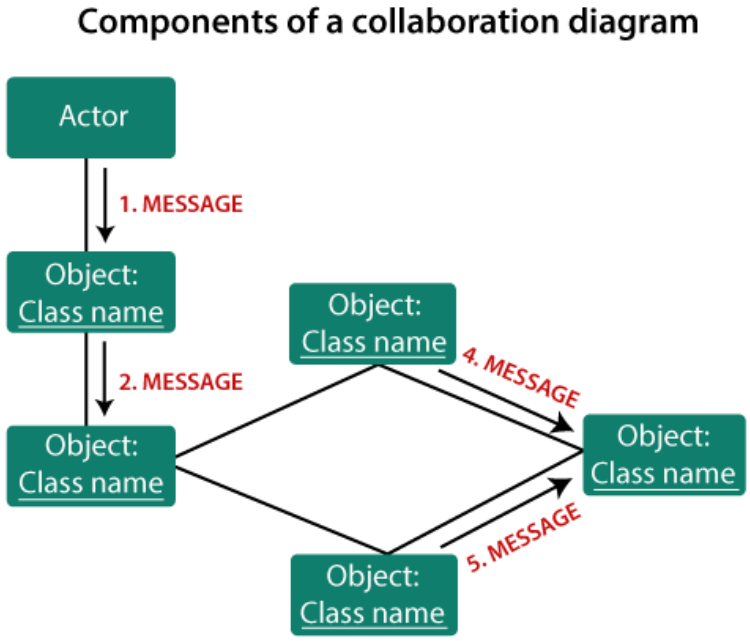
Concepts:

The second interaction diagram is collaboration diagram. It shows the object organization as shown below. In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. A collaboration diagram is also called a communication diagram or interaction diagram.

The method calls are similar to that of a sequence diagram. Sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



Notations of a Collaboration Diagram:



Following are the components of a component diagram that are enlisted below:
Objects: The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.

In the collaboration diagram, objects are utilized in the following ways:

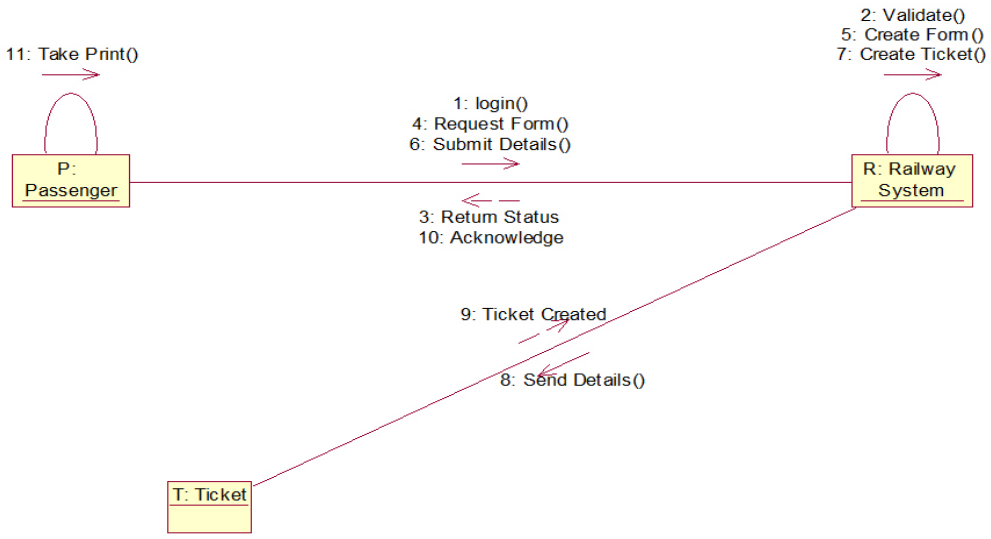
- The object is represented by specifying their name and class.
- It is not mandatory for every class to appear.
- A class may constitute more than one object.
- In the collaboration diagram, firstly, the object is created, and then its class is specified.
- To differentiate one object from another object, it is necessary to name them.

Actors: In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.

Links: The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.
Messages: It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

Implementation:

Collaboration Diagram for Railway Reservation System:



Output/Conclusion:- The collaboration diagram was made successfully.

Experiment No:- 7

OBJECTIVE:- Draw the state chart diagram.

Theory:-

Concepts:

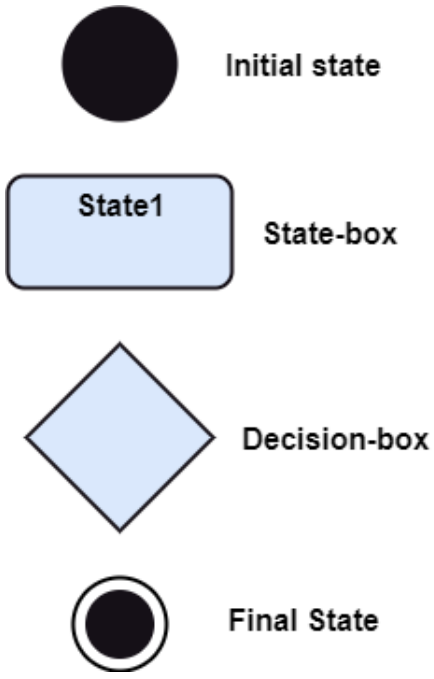
State chart diagrams are used for describing the states. States can be identified as the condition of objects when a particular event occurs.
Before drawing a Statechart diagram we should:

- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

During the life cycle of an object (here order object) it goes through the following states and there may be some abnormal exists also. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete it is considered as the complete transaction as mentioned below.

Notation of a State Machine Diagram:

Following are the notations of a state machine diagram enlisted below:



Initial state: It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
Final state: It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
Decision box: It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
Transition: A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change

has ensued.
State box: It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.

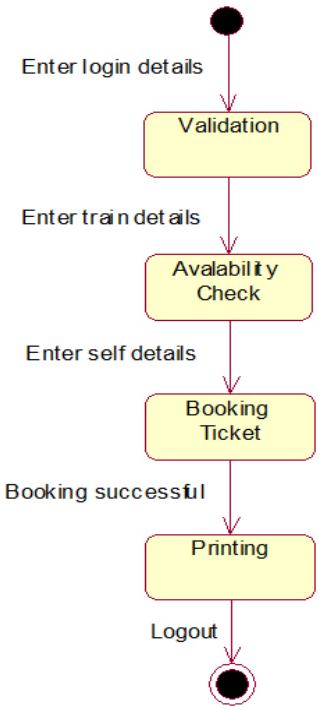
Types of State

The UML consist of three states:
Simple state: It does not constitute any substructure.
Composite state: It consists of nested states (substates), such that it does not contain more than one initial state and one final state. It can be nested to any level.
Submachine state: The submachine state is semantically identical to the composite state, but it can be reused.

Implementation:

State Chart diagram For Railway Reservation System:

The initial and final state of an object is also shown below.



Output/Conclusion:- The state chart diagram was made successfully.

Experiment No:- 8

OBJECTIVE:- Draw the component diagram.

Theory:-

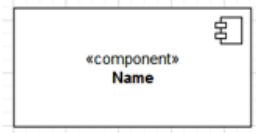
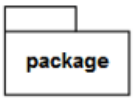
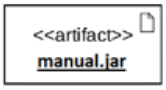
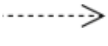
Concepts:

Component diagram shows components, provided and required interfaces, ports, and relationships between them. This type of diagrams is used in Component-Based Development (CBD) to describe systems with Service-Oriented Architecture (SOA).

Components in UML could represent

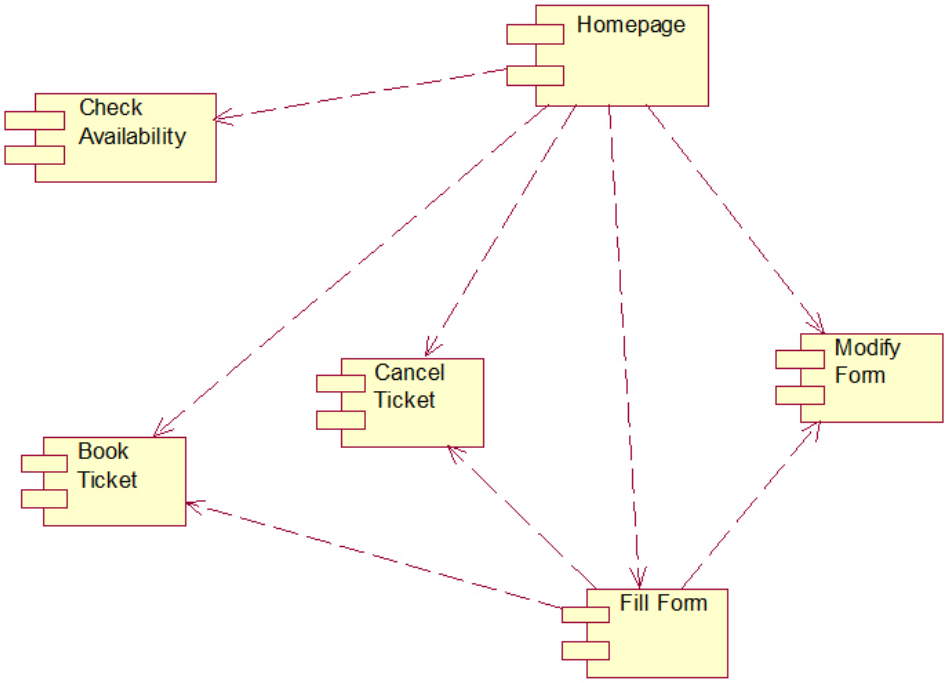
- ✓ **Logical components** (e.g., business components, process components), and
- ✓ **Physical components** (e.g., CORBA components, EJB components, COM+ and .NET components, WSDL components, etc.),

Notation of a Component Diagram:

Element	Symbol/Notation	Explanation
Component		Symbol for modules in a system (interaction and communication occur via interfaces).
Package		A package combines multiple elements in a system (e.g. classes, components, interfaces) into a group.
Artifact		Artifacts are physical units of information (e.g. source code, *.exe* files, scripts, documents) that are created or required during a system's development process or at runtime.
Provided interface		Symbol for one or more clearly defined interfaces that provide functions, services or data to the outside (the open semi-circle is also called a socket).
Required interface		Symbol for a required interface that receives functions, services or data from the outside (the circle-with-stick notation is also referred to as a lollipop).
Port		This symbol indicates a separate point of interaction between a component and its environment.
Relationship		Lines act as connectors and show relationships between components.
Dependency		This special connector indicates a dependency between two parts of a system (not always explicitly shown).

Implementation:

Component diagram For Railway Reservation System:



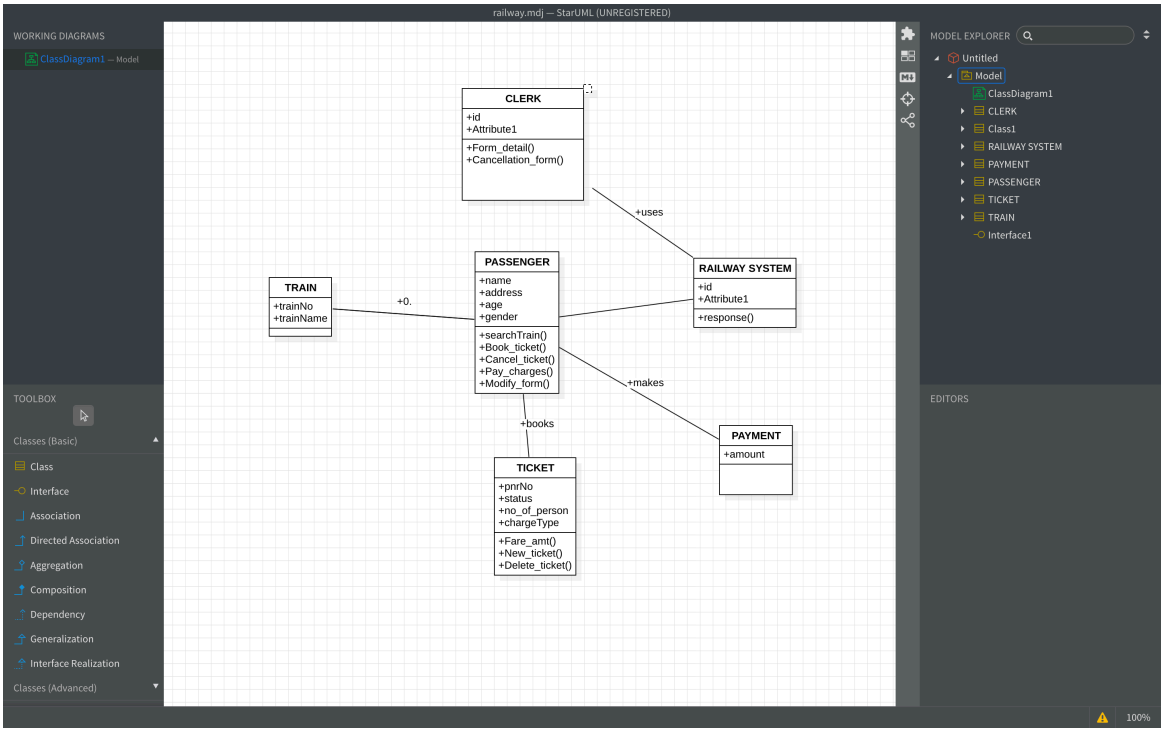
Output/Conclusion:- The state chart diagram was made successfully.

Experiment No:- 9

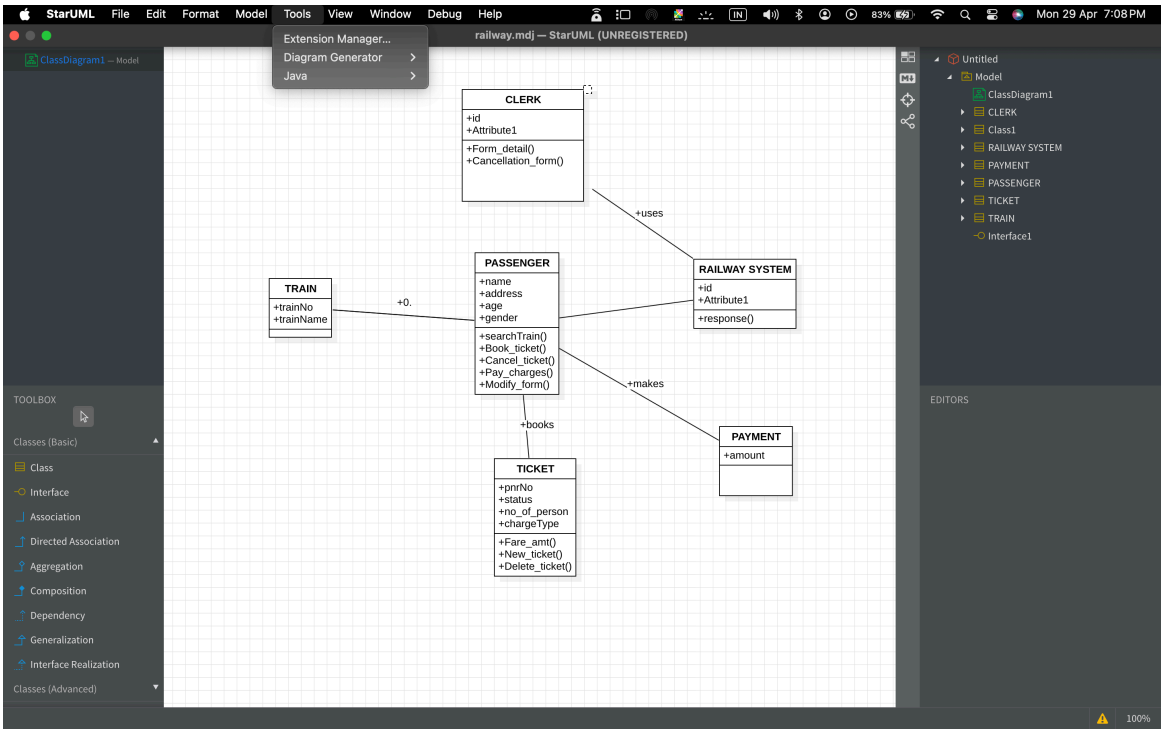
OBJECTIVE:- Perform forward engineering in JAVA. (Model to code conversion)

Implementation:-

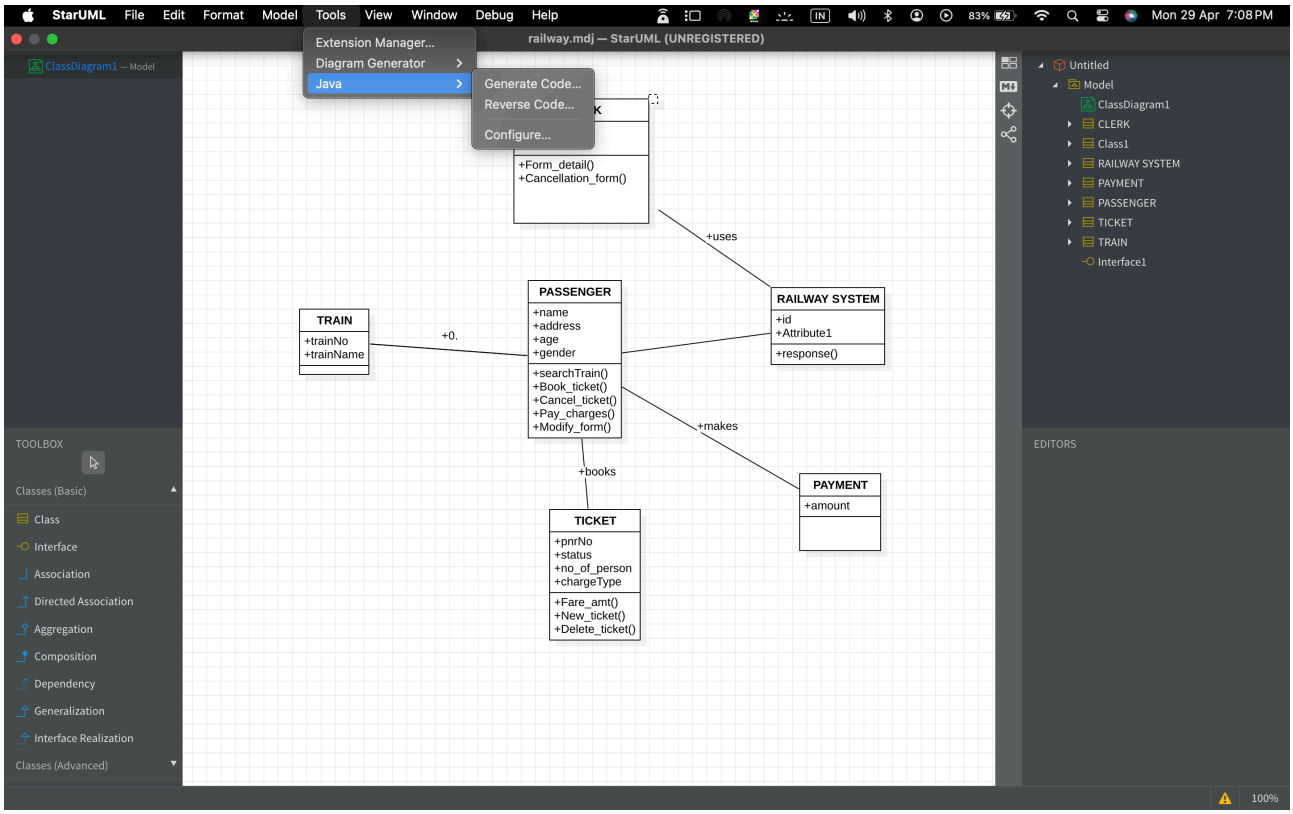
Step 1:- Create/Open the UML model on the StarUML.



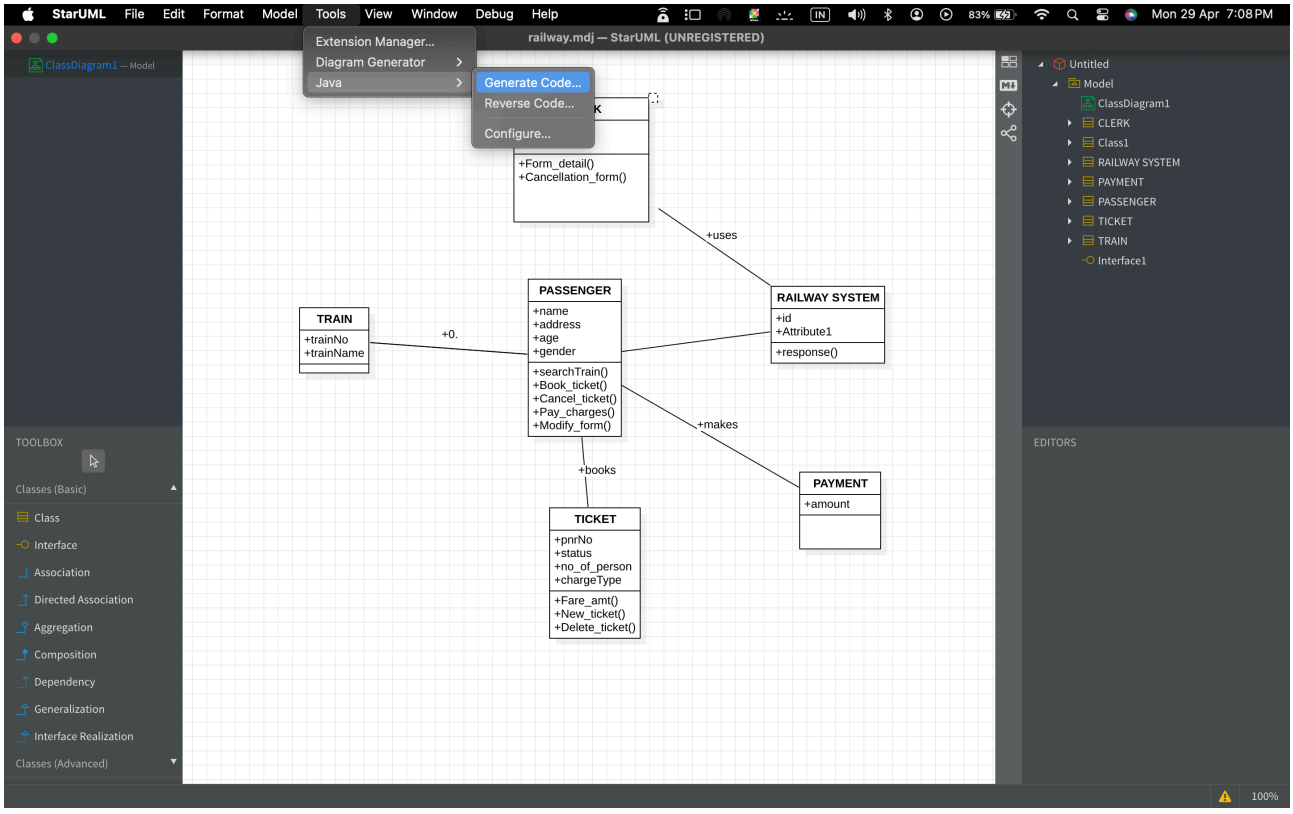
Step 2:- Open Tools option on the Toolbar.



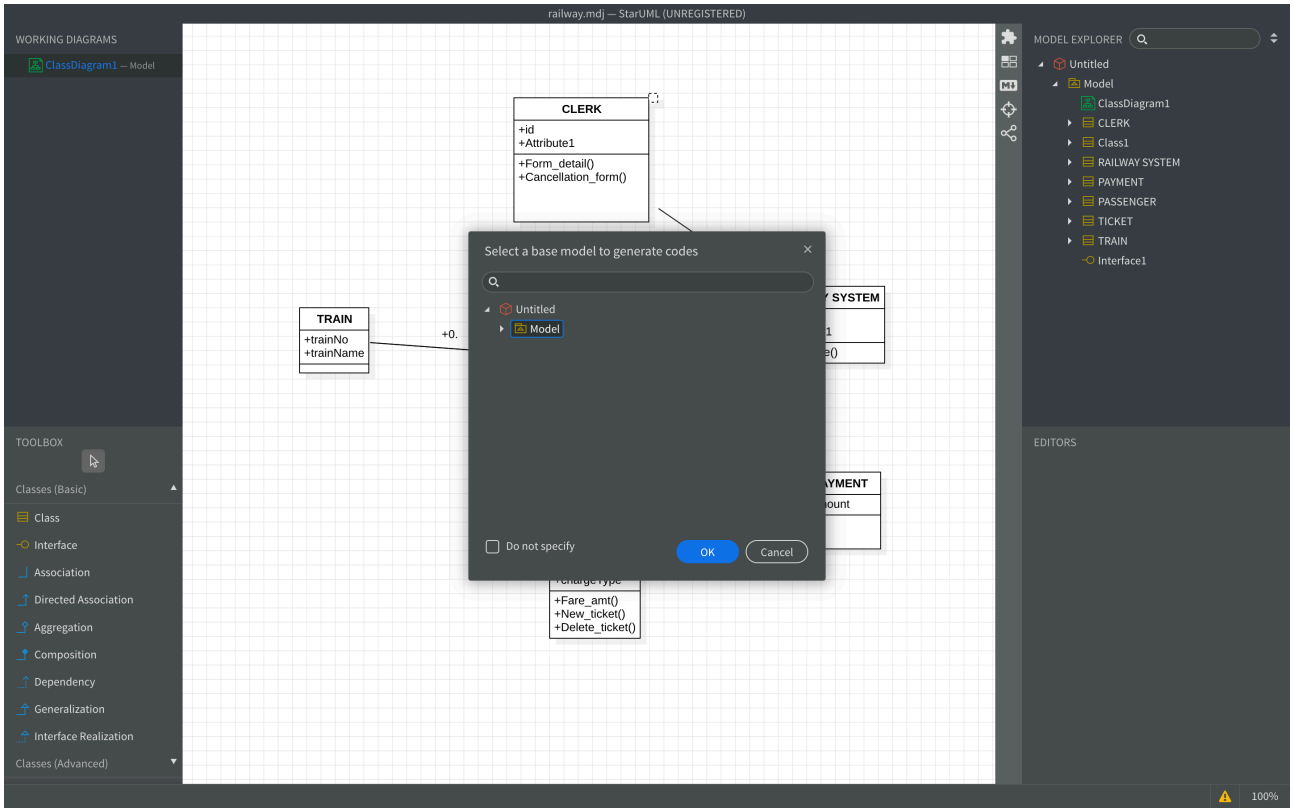
Step 3:- Select Java option in the Toolbar.



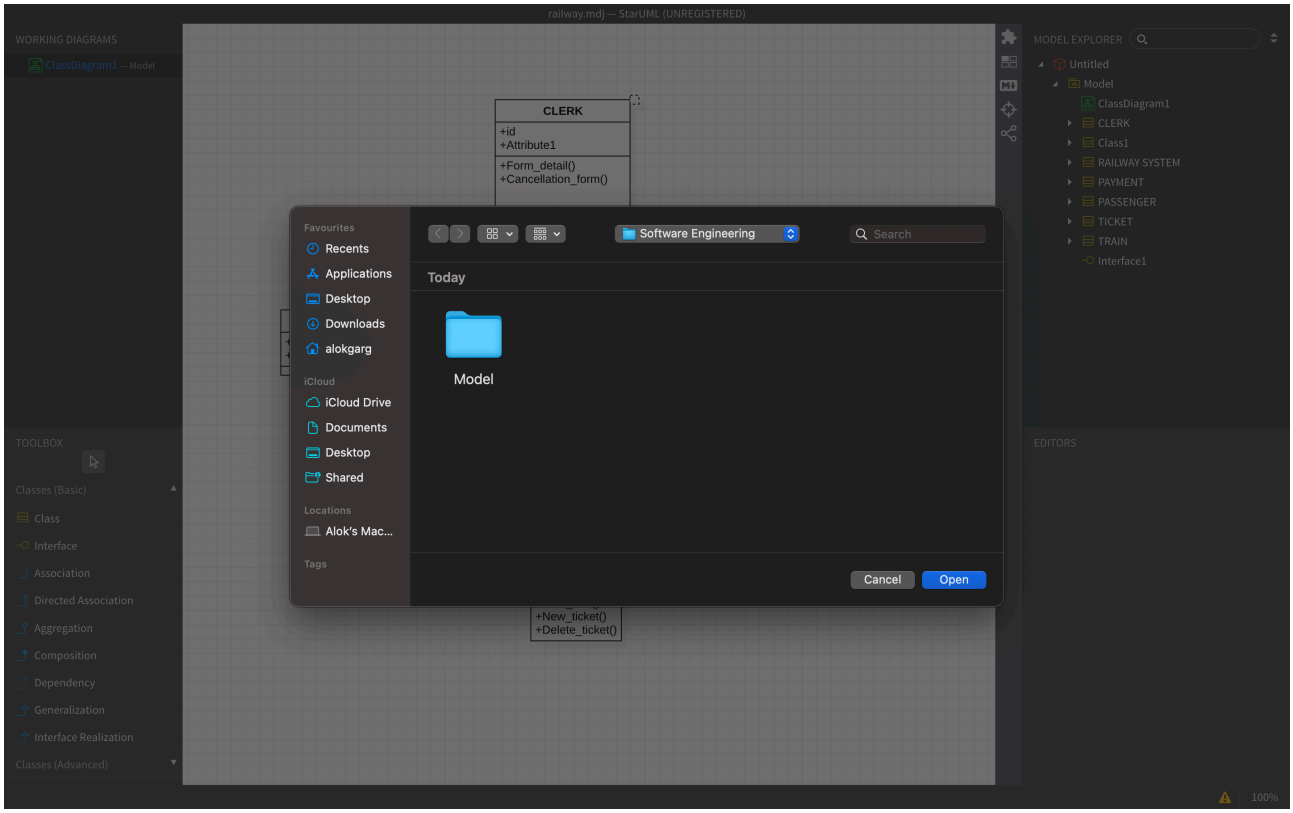
Step 4:- Select the Generate Code Option in the Java tool.



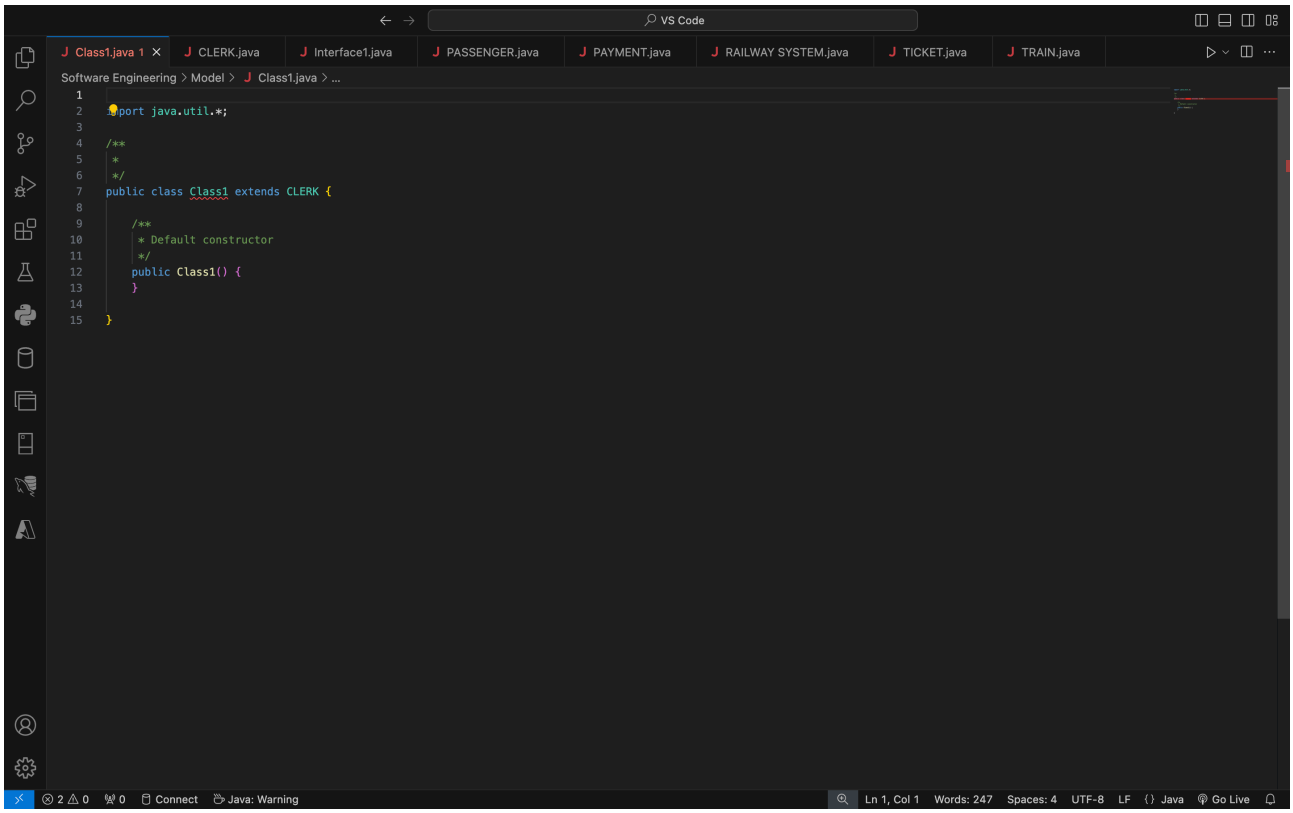
Step 5:- A new dialogue box appears, select the base model to generate codes and then click OK.



Step 6:- After that select the desirable location to save the files and then click Open.



Step 7:- Open the file and now you can see all the java files of each component.



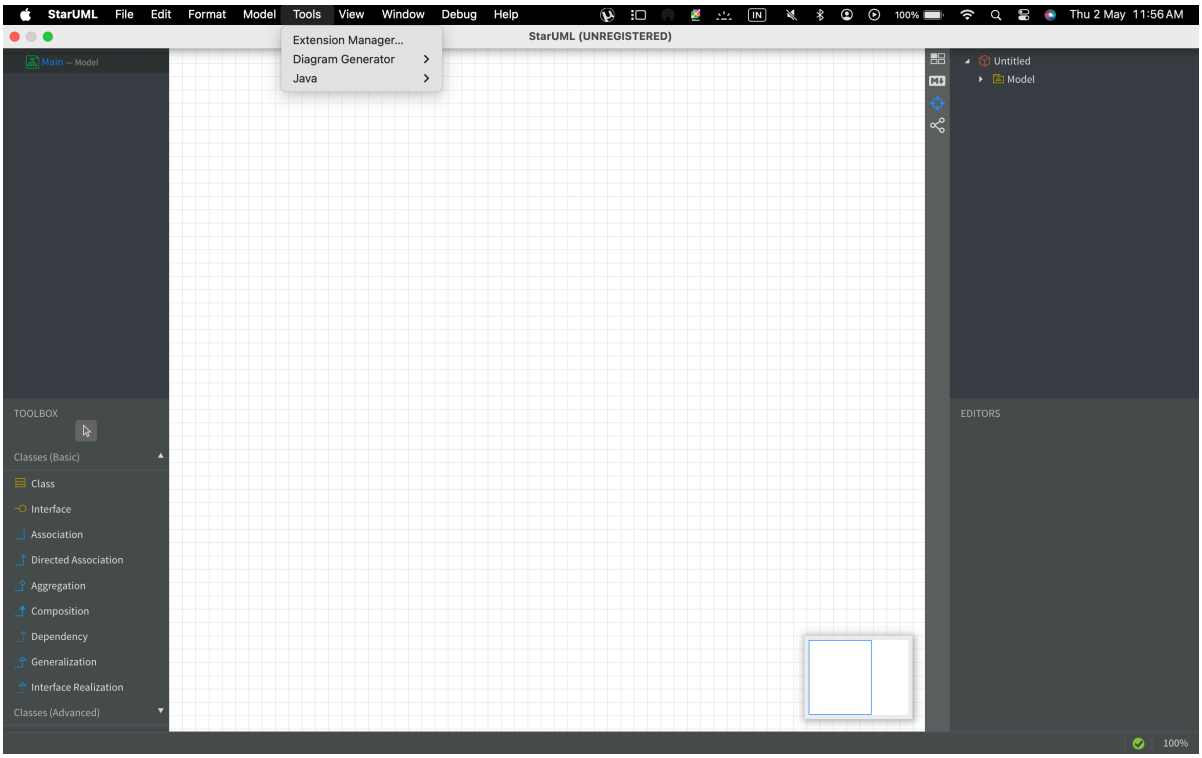
Output/Conclusion:- Using forward engineering method, Java code is generated.

Experiment No:- 10

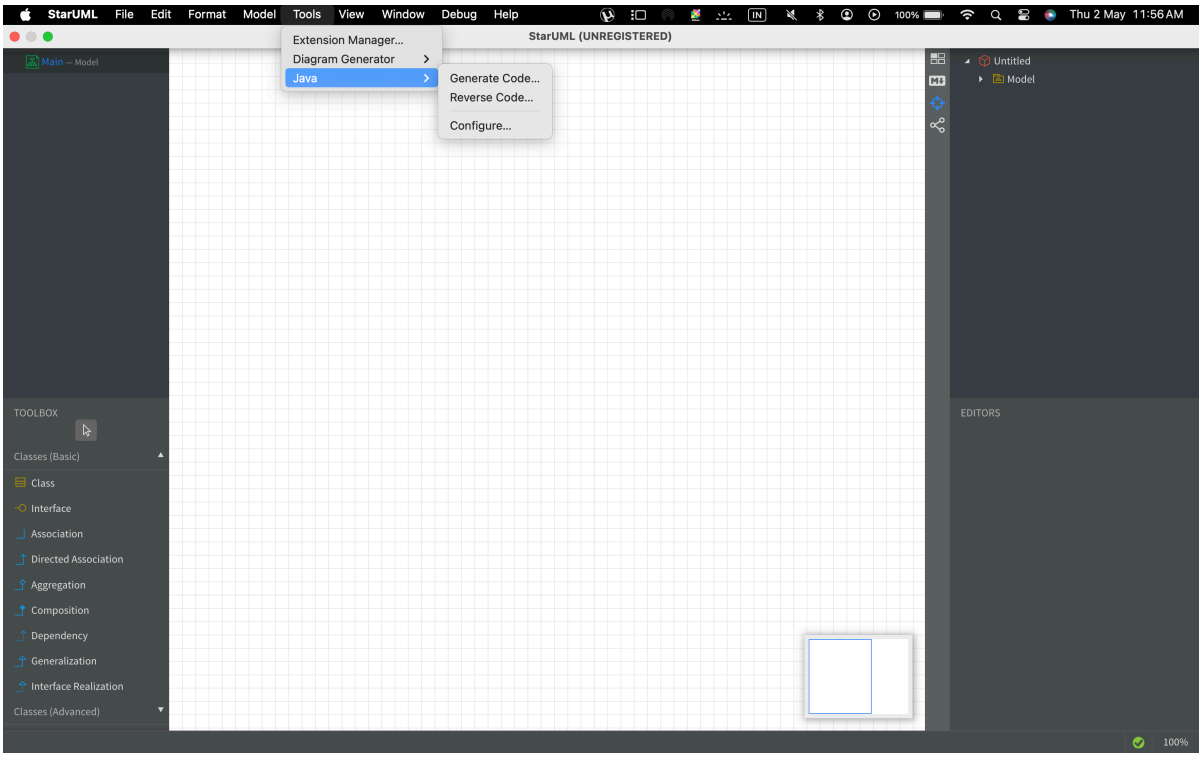
OBJECTIVE:- Perform reverse engineering in JAVA. (Code to Model conversion)

Implementation:-

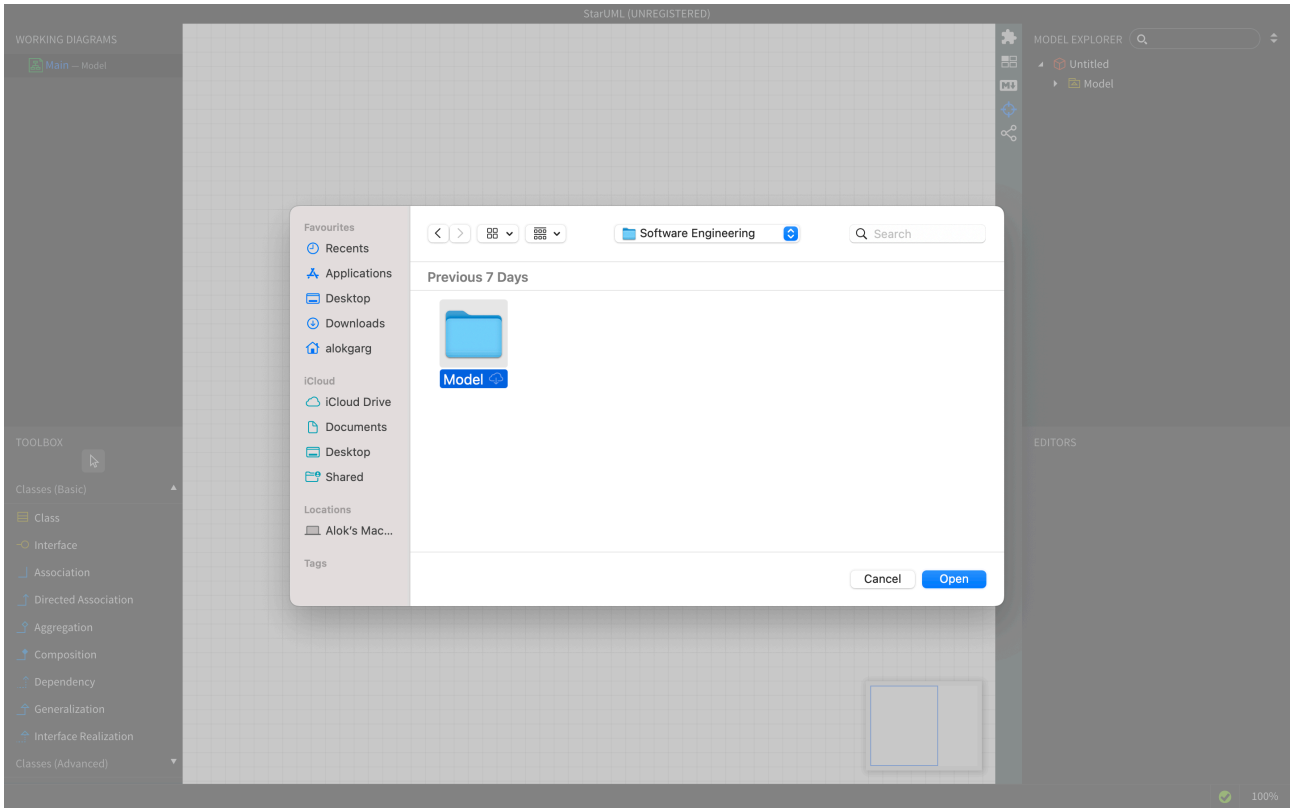
Step 1:- Open blank StarUML document. Hover to Tools Option on the Toolbar.



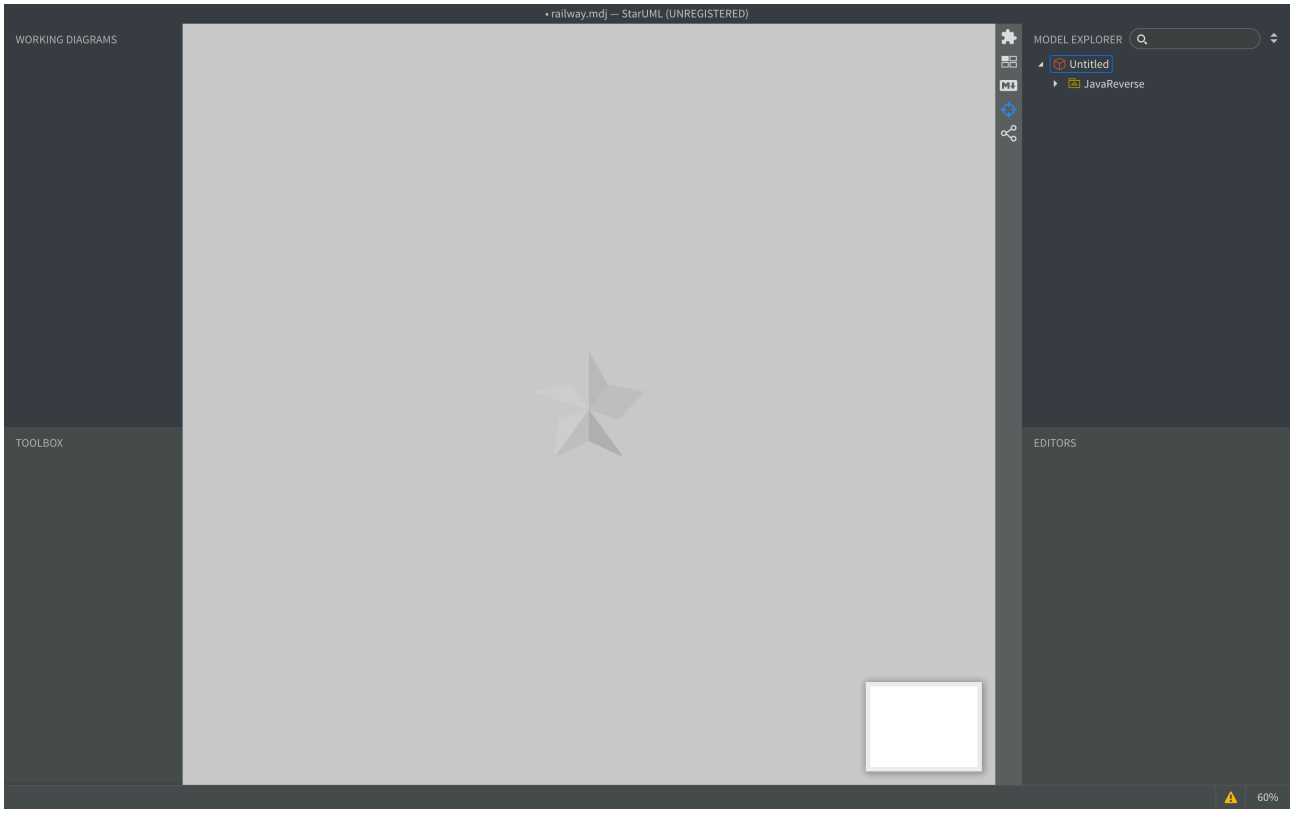
Step 2:- Select Java option in the Toolbar and Select the Reverse Code Option in the Java Tool.



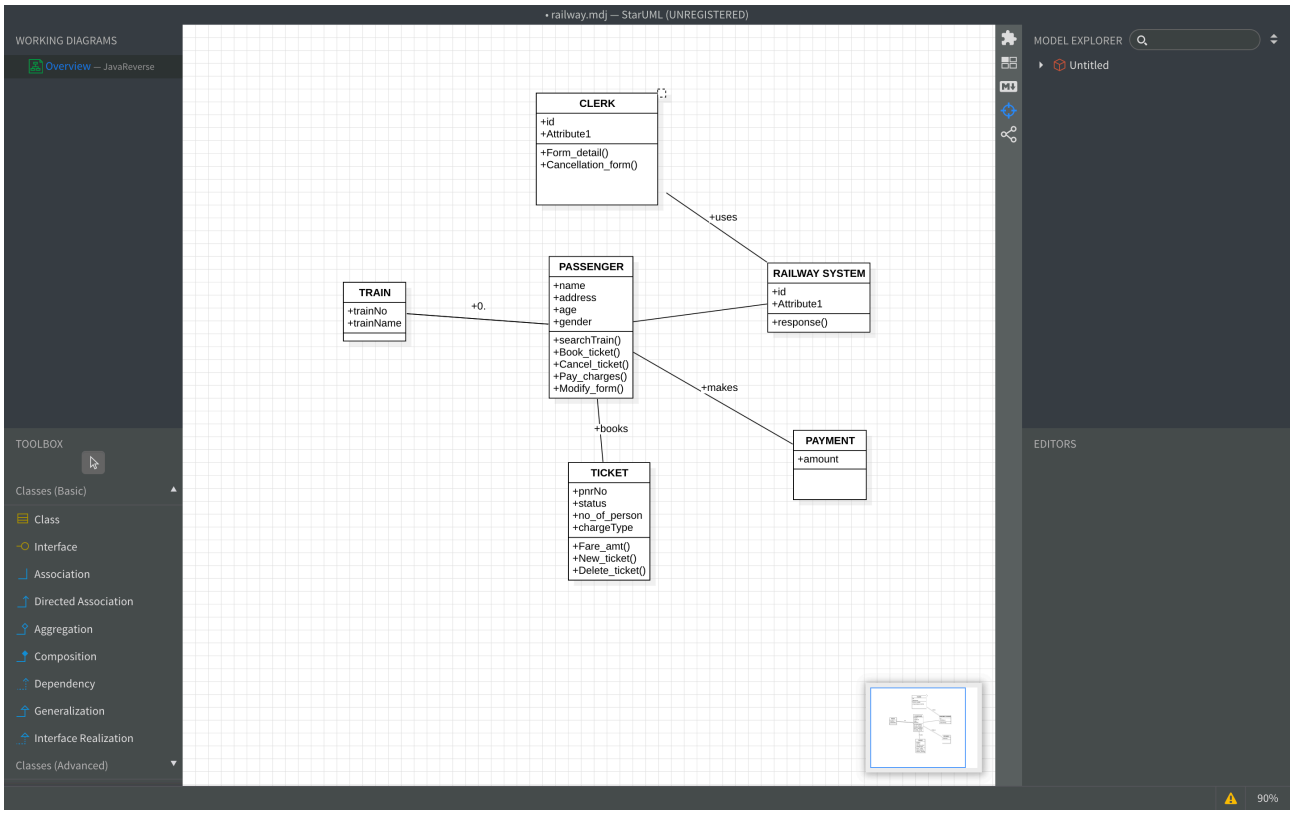
Step 3:- Select the the file which has the Java Code for which the Model must be created using Reverse Engineering and click Open.



Step 4:- A new File is created in Model Explorer which is JavaReverse.



Step 5:- Select the diagram in JavaReverse and double click on it. The required model is generated using reverse engineering. After that we can find different Classes in the Model Explorer and join them with the Associations.



Output/Conclusion:- Using reverse engineering method, Java code generated the Model.