Veermata Jijabai Technological Institute, Mumbai 400019

Assignment No.: 03

Aim : Modeling UML Class Diagrams and Sequence diagrams

Name : Kiran K Patil

Enrollment No.: 211070904

Branch : Computer Engineering

Batch: IV

Modeling UML Class Diagrams and Sequence diagrams

UML (Unified Modeling Language) Class Diagrams and Sequence diagrams are used in software engineering to model and design software systems.

A UML Class Diagram represents the structure of a system by showing the classes, their attributes, methods, and the relationships between them. It is a static diagram that is used to visualize the object-oriented programming concepts such as inheritance, abstraction, and polymorphism.

UML Class Diagram:

A UML class diagram is composed of several elements, which include classes, attributes, operations, relationships, and various types of constraints. Here is a brief overview of each element:

- **Class**: A class is a blueprint for creating objects, and it defines the common attributes, behaviors, and relationships that its objects share. Each class is represented as a rectangle, which contains the name of the class.

- **Attribute**: An attribute is a property of a class or object, which describes its characteristics. Each attribute is represented as a name-value pair within the rectangle of a class.

- **Operation**: An operation is a behavior of a class or object, which describes its actions or functions. Each operation is represented as a name-value pair within the rectangle of a class, which is shown using brackets.

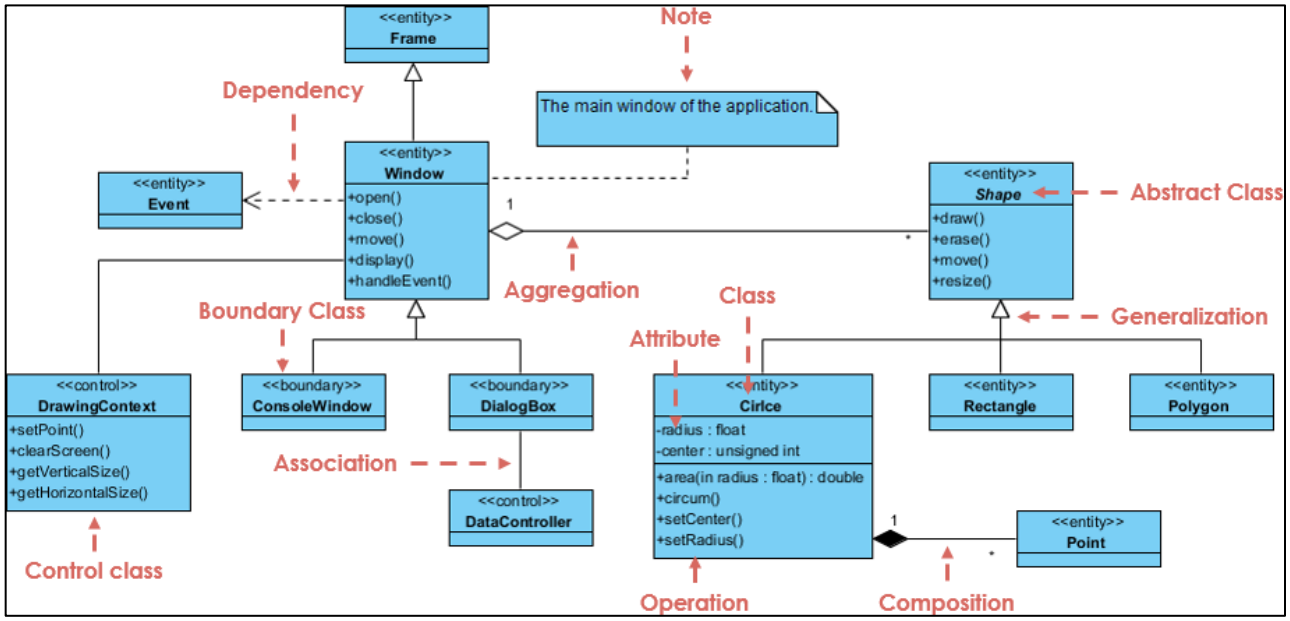
- **Relationship**: A relationship is a connection between two or more classes or objects, which defines how they are related to each other. There are several types of relationships, including association, aggregation, composition, inheritance, and realization.

- **Constraint**: A constraint is a limitation or rule that applies to a class or object, which defines its properties or behavior. Constraints can be represented using different symbols, such as <<constraint>> or <<invariant>>.

UML class diagrams are useful for visualizing the structure of a system, as well as identifying the relationships and interactions between its various components. They can also help to facilitate communication and collaboration between software developers, designers, and other stakeholders, thereby improving the overall efficiency and effectiveness of the software development process.

To create a UML Class Diagram, you start by identifying the classes that make up the system and their attributes and methods. Then, you can draw the classes as rectangles with the class name at the top, the attributes in the middle, and the methods at the bottom. You can use lines to connect the classes to show the relationships between them, such as inheritance (an arrow pointing to the base class), association (a line connecting two classes), and aggregation or composition (a diamond shape representing the whole-part relationship).

UML class Diagram sample :



UML Sequence Diagram:

In a UML sequence diagram, the objects are represented as vertical lifelines, and the interactions between them are shown as horizontal arrows. Each arrow represents a message that is sent from one object to another, and it can be synchronous (solid), asynchronous (dashed), or reply (dotted). The sequence of messages is shown from top to bottom, indicating the order in which they occur over time.

In addition to objects and messages, a UML sequence diagram can also include other elements such as actors, control flows, conditions, loops, and parallel interactions.

Here are some of the main components of a UML sequence diagram:

- **Lifeline**: A lifeline represents an object or participant in the system. It is depicted as a vertical line that starts from the top of the diagram and extends downwards.
- **Message**: A message represents a communication between two objects. It is shown as an arrow that goes from the sender lifeline to the receiver lifeline.
- **Activation**: An activation represents the time period during which an object is executing an operation. It is shown as a box that extends vertically from the lifeline.
- **Actor**: An actor represents an external entity that interacts with the system. It is depicted as a stick figure that sends messages to the objects in the system.
- **Control flow**: A control flow represents the flow of the sequence diagram. It is shown as a dashed arrow that connects two messages or activations.

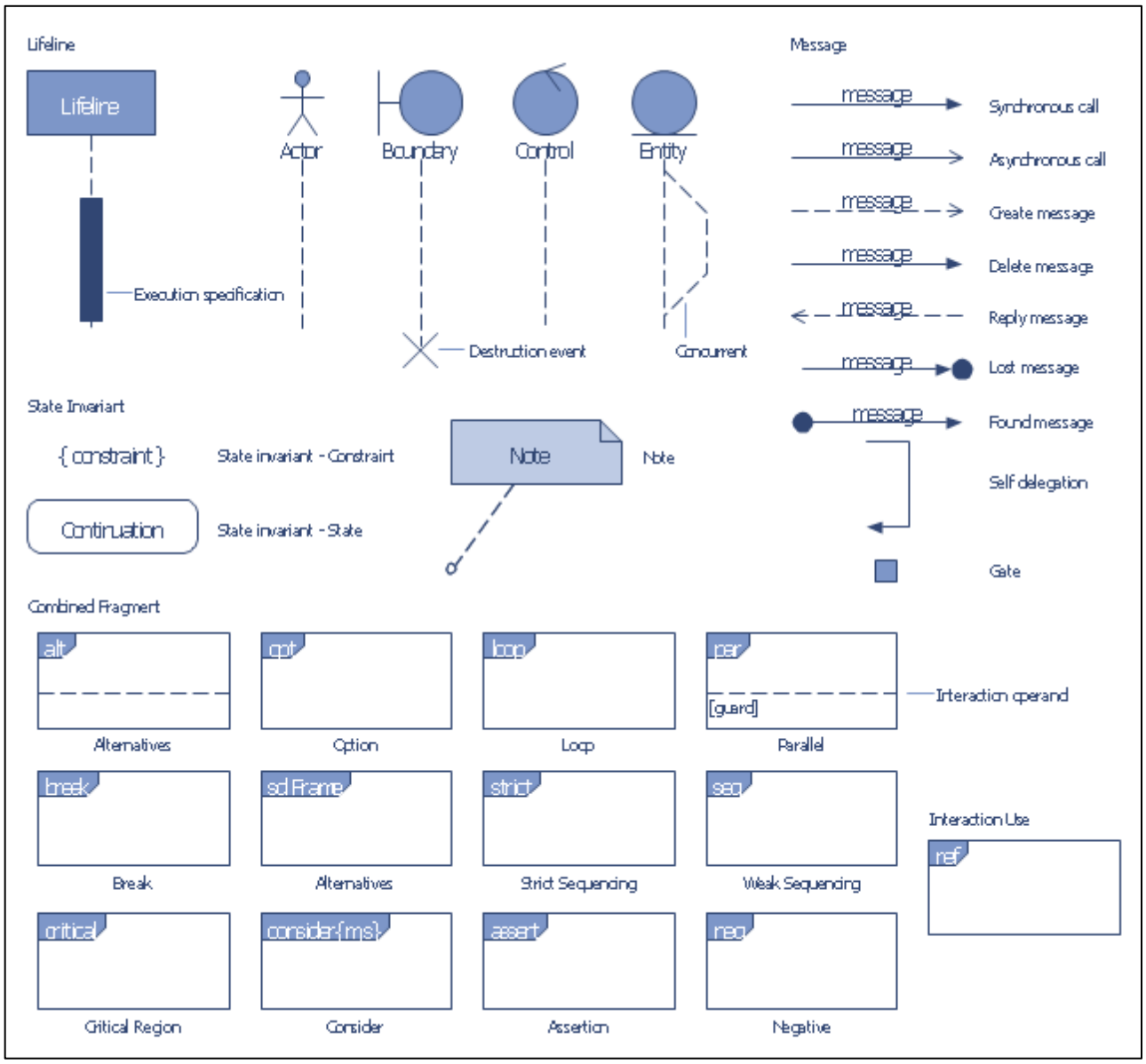
UML sequence diagrams are useful for understanding the interactions between objects in a system, as well as for identifying potential problems or inefficiencies in the system. They are widely used in software engineering and other related fields to model and design complex systems, and to communicate and collaborate among project team members.

An UML Sequence Diagram, represents the dynamic behavior of a system by showing the interactions between the objects over time. It is a type of interaction diagram that is used to visualize the messages exchanged between objects in a specific scenario.

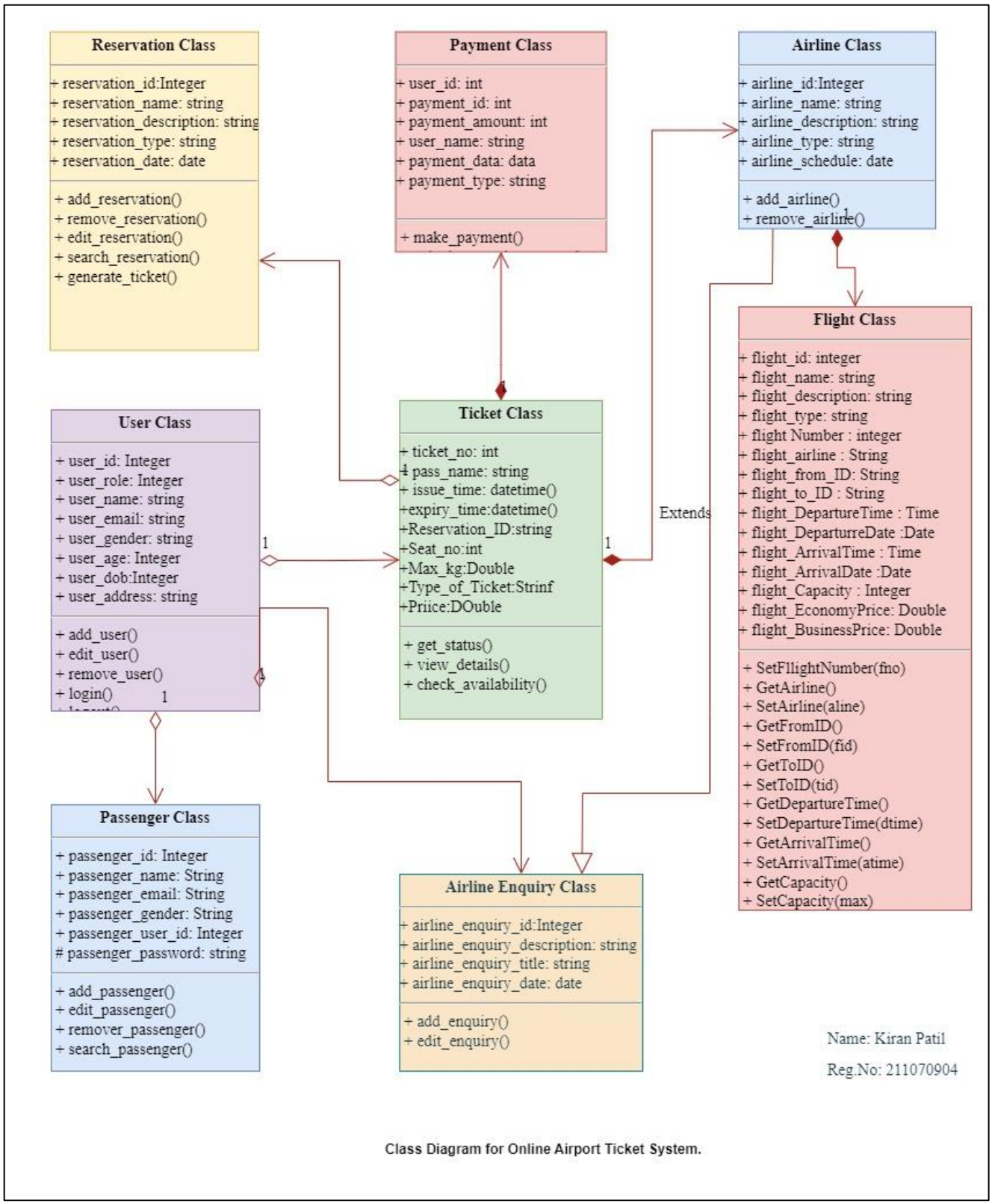
To create a UML Sequence Diagram, you start by identifying the objects involved in the scenario and their interactions. Then, you can draw the objects as vertical lifelines, and the messages as horizontal arrows between them. You can use different types of messages, such as synchronous (solid arrow), asynchronous (dashed arrow), and return messages (dotted arrow). You can also add conditions, loops, and other control structures to show the flow of the scenario.

Overall, UML Class Diagrams and Sequence diagrams are powerful tools for software engineers to model and design software systems.

Sequence Diagram Notations :

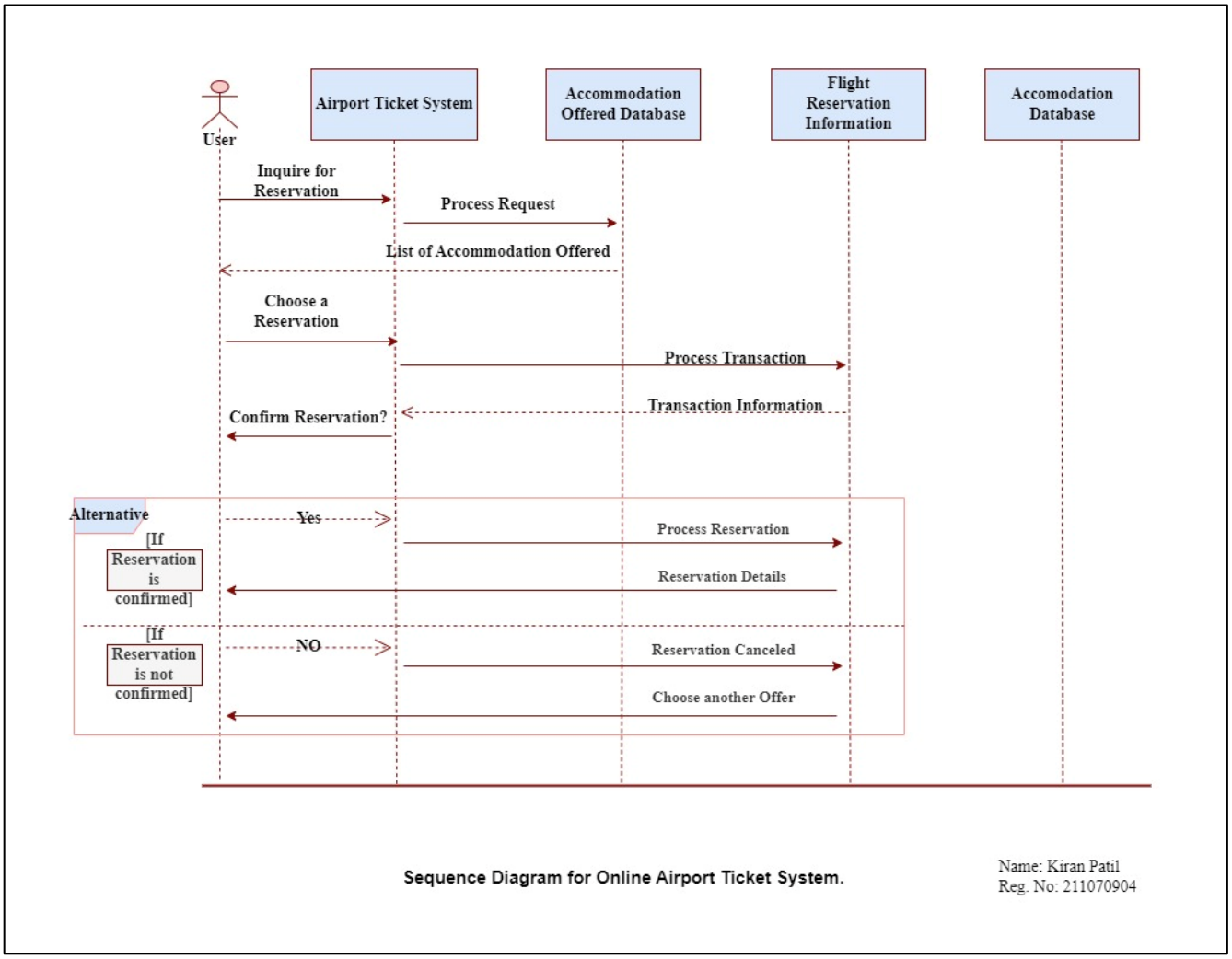


UML Class Diagram for Online Airport Ticket System



Class Diagram for Online Airport Ticket System.

UML Sequence Diagram for Online Airport Ticket System.



Conclusion:

In conclusion we can say that the UML Class Diagrams and Sequence diagrams are essential tools for modeling the structure and behavior of a software system. UML Class diagrams represent the structure of the system, while Sequence diagrams represent the dynamic behavior. Together, they help in efficient software development, management, and communication among stakeholders.