Software Engineering-BSCE-301L

LAB DA-3:

State Transition, Class Diagram, and Use Case

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

CASE TOOLs – UML Class Diagram

- The class diagram depicts a static view of an application.
- It represents the types of objects residing in the system and the relationships between them.
- A class consists of its objects, and also it may inherit from other classes.
- A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.
- It shows the attributes, classes, functions, and relationships to give an overview of the software system.
- It constitutes class names, attributes, and functions in a separate compartment that helps in software development.
- Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

CASE TOOLs – UML Class Diagram

- The **main purpose of class diagrams** is to build a static view of an application.
- It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages.
- It is one of the most popular UML diagrams.
- Following are the purpose of class diagrams given below:

1. It analyses and designs a static view of an application.
2. It describes the major responsibilities of a system.
3. It is a base for component and deployment diagrams.
4. It incorporates forward and reverse engineering.

CASE TOOLs – UML Class Diagram

❑Benefits of Class Diagrams

1. It can represent the object model for complex systems.
2. It reduces the maintenance time by providing an overview of how an application is structured before coding.
3. It provides a general schematic of an application for better understanding.
4. It represents a detailed chart by highlighting the desired code, which is to be programmed.
5. It is helpful for the stakeholders and the developers.

CASE TOOLs – UML Class Diagram

- **Vital components of a Class Diagram**: The class diagram is made up of three sections:
- **Upper Section**
- **Middle Section**
- **Lower Section**

CASE TOOLs – UML Class Diagram

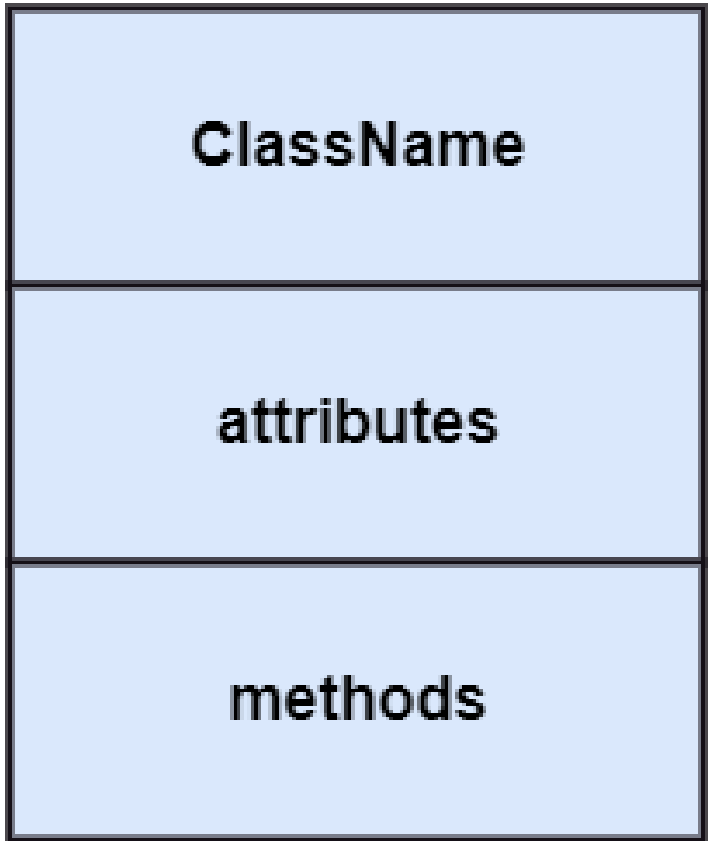
- **Upper Section:** The upper section encompasses the name of the class.
- A class is a representation of similar objects that shares the same relationships, attributes, operations, and semantics.
- Some of the following rules that should be taken into account while representing a class are given below:
  - Capitalize the initial letter of the class name.
  - Place the class name in the center of the upper section.
  - A class name must be written in bold format.
  - The name of the abstract class should be written in italics format.

CASE TOOLs – UML Class Diagram

- **Middle Section:** The middle section constitutes the attributes, which describe the quality of the class.
- The attributes have the following characteristics:
  - The attributes are written along with its visibility factors, which are public (+), private (-), protected (#), and package (~).
  - The accessibility of an attribute class is illustrated by the visibility factors.
  - A meaningful name should be assigned to the attribute, which will explain its usage inside the class.

CASE TOOLs – UML Class Diagram

- **Lower Section:** The lower section contain methods or operations.
- The methods are represented in the form of a list, where each method is written in a single line.
- It demonstrates how a class interacts with data.



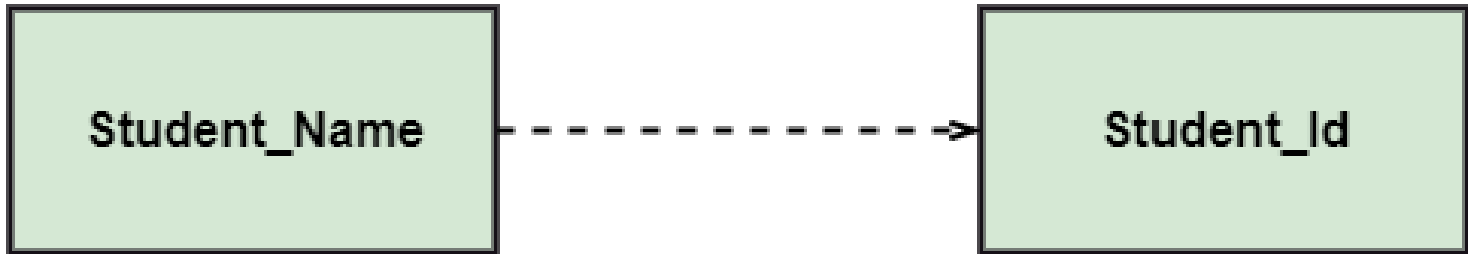
CASE TOOLs – UML Class Diagram

- **Relationships:** Following are the relationship in UML Class Diagram.
- Dependency:
- Generalization:
- Association:
- Multiplicity:
- Aggregation:
- Composition:

CASE TOOLs – UML Class Diagram

❑Relationships:

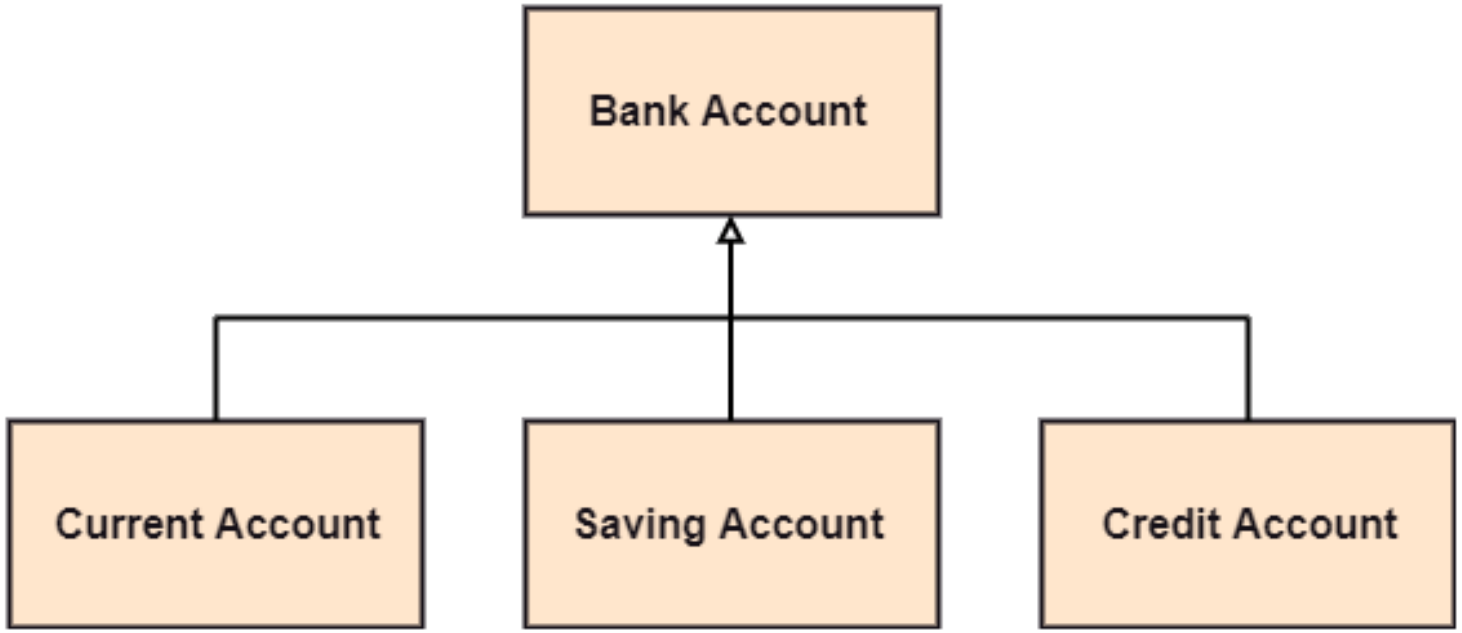
❑**Dependency:** A dependency is a semantic relationship between two or more classes where a change in one class cause changes in another class. It forms a weaker relationship. In the following example, Student_Name is dependent on the Student_Id.



CASE TOOLs – UML Class Diagram

❑**Relationships:**

❑**Generalization:** A generalization is a relationship between a parent class (superclass) and a child class (subclass). In this, the child class is inherited from the parent class. For example, The Current Account, Saving Account, and Credit Account are the generalized form of Bank Account.

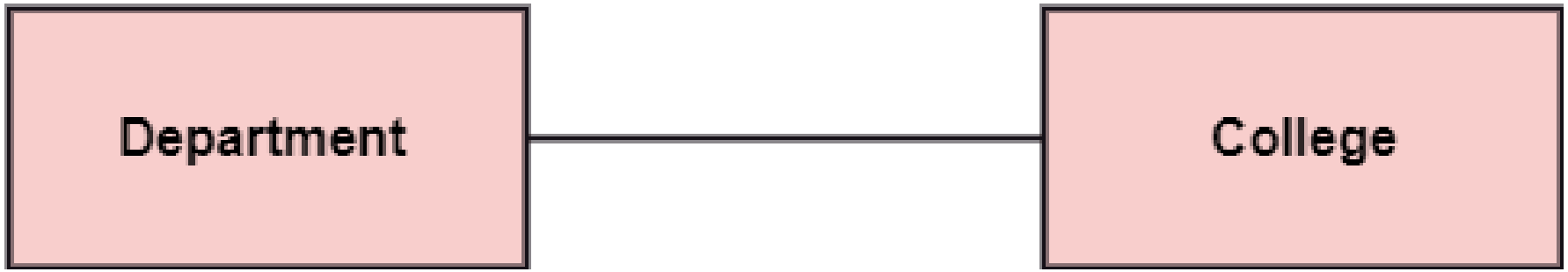


CASE TOOLs – UML Class Diagram

❑**Relationships:**

❑**Association:** It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.

❑For example, a department is associated with the college.

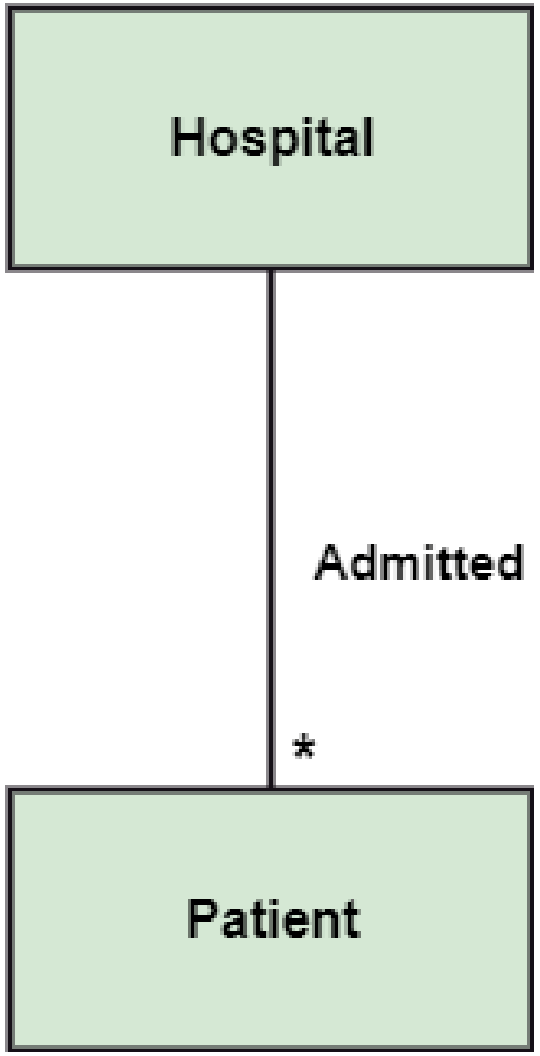


CASE TOOLs – UML Class Diagram

❑**Relationships:**

❑**Multiplicity:** It defines a specific range of allowable instances of attributes. In case if a range is not specified, one is considered as a default multiplicity.

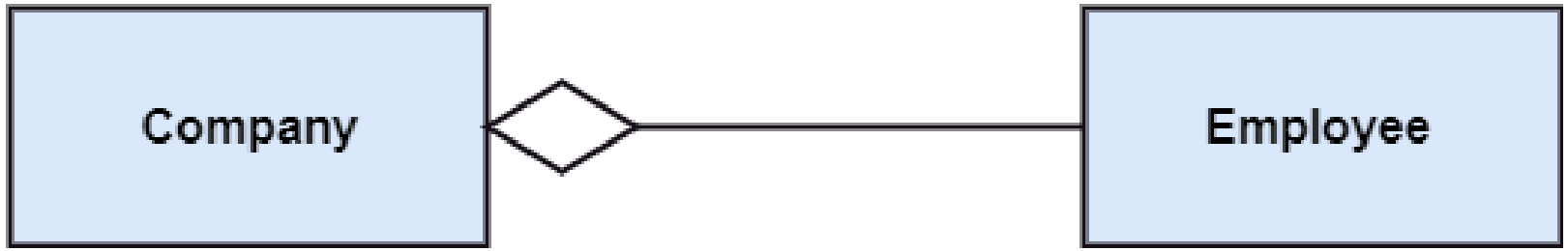
❑For example, multiple patients are admitted to one hospital.



❑Relationships:

❑**Aggregation:** An aggregation is a subset of association, which represents has a relationship. It is more specific then association. It defines a part-whole or part-of relationship. In this kind of relationship, the child class can exist independently of its parent class.

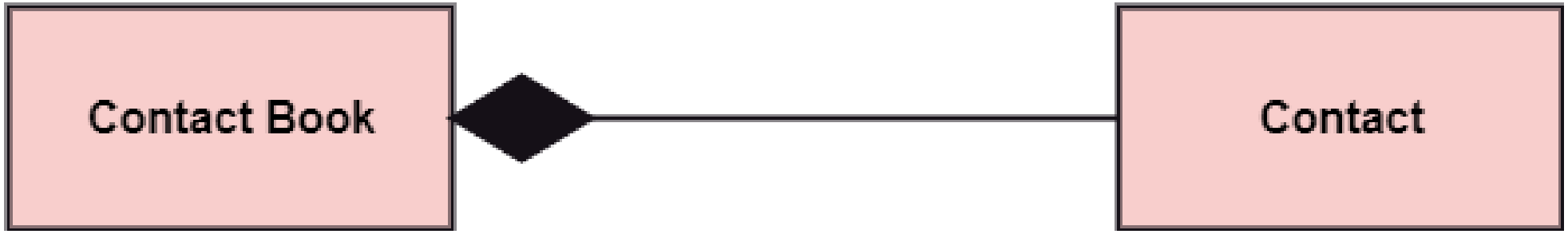
❑The company encompasses a number of employees, and even if one employee resigns, the company still exists.



❑Relationships:

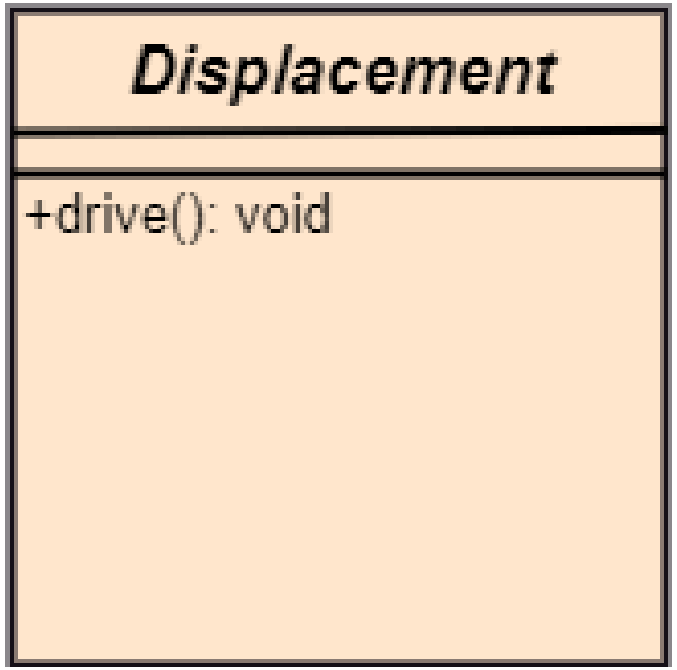
❑**Composition:** The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.

❑A contact book consists of multiple contacts, and if you delete the contact book, all the contacts will be lost.



CASE TOOLs – UML Class Diagram

❑**Abstract Classes:** In the abstract class, no objects can be a direct entity of the abstract class. The abstract class can neither be declared nor be instantiated. It is used to find the functionalities across the classes. The notation of the abstract class is similar to that of class; the only difference is that the name of the class is written in italics. Since it does not involve any implementation for a given function, it is best to use the abstract class with multiple objects.



❑Let us assume that we have an abstract class named **displacement** with a method declared inside it, and that method will be called as a **drive ()**. Now, this abstract class method can be implemented by any object, for example, car, bike, scooter, cycle, etc.

CASE TOOLs – UML Class Diagram

❑How to draw a Class Diagram?

❑The class diagram is used most widely to construct software applications.

❑It not only represents a static view of the system but also all the major aspects of an application.

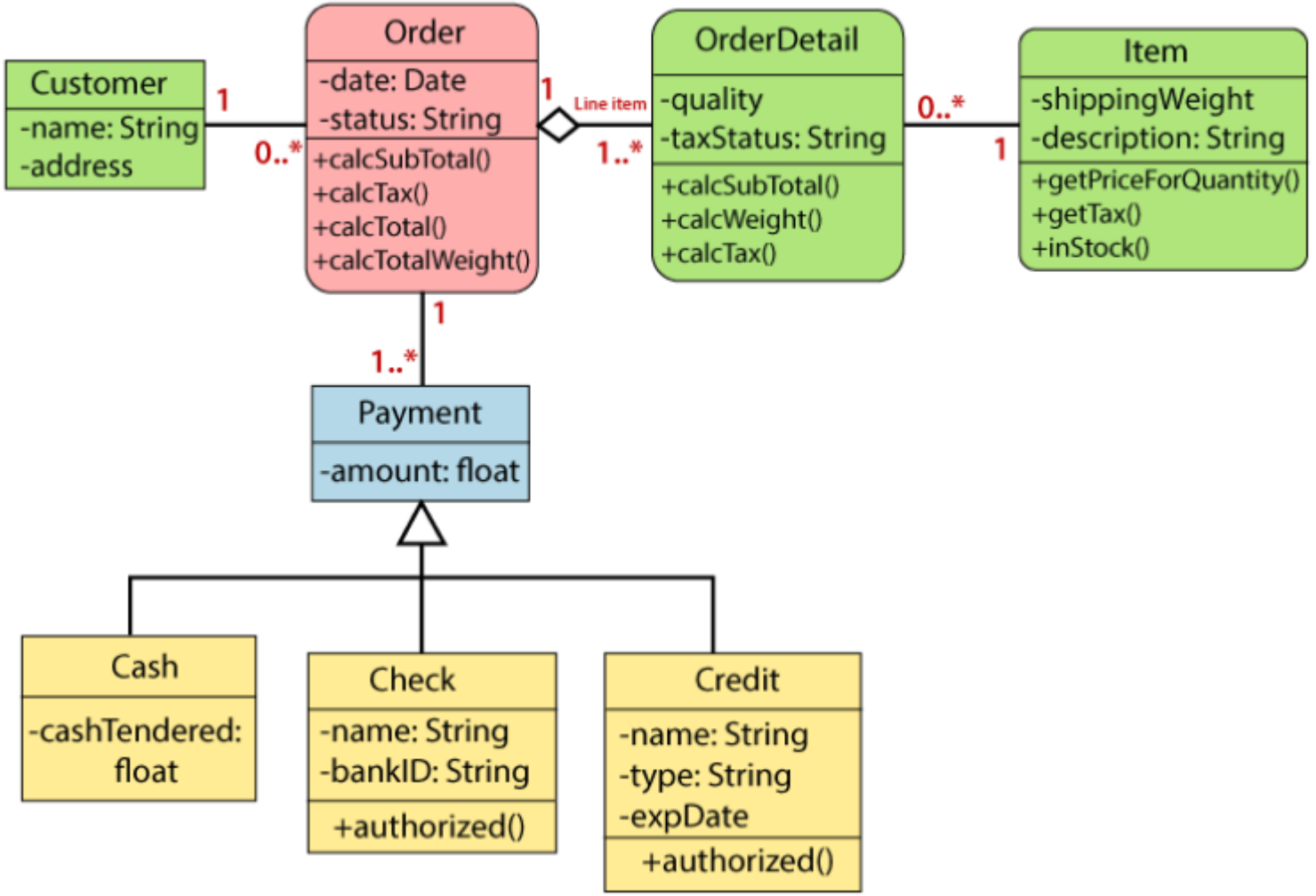
❑A collection of class diagrams as a whole represents a system.

❑Some key points that are needed to keep in mind while drawing a class diagram are given below:

1. **To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.**
2. **The objects and their relationships should be acknowledged in advance.**
3. **The attributes and methods (responsibilities) of each class must be known.**
4. **A minimum number of desired properties should be specified as more number of the unwanted property will lead to a complex diagram.**
5. **Notes can be used as and when required by the developer to describe the aspects of a diagram.**
6. **The diagrams should be redrawn and reworked as many times to make it correct before producing its final version.**

CASE TOOLs – UML Class Diagram

❑**Class Diagram Example:** A class diagram describing the sales order system is given below.



CASE TOOLs – UML Use Case Diagram

- A use case diagram is used to represent the dynamic behavior 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.

CASE TOOLs – UML Use Case Diagram

❑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.

CASE TOOLs – UML Use Case Diagram

❑How to draw a Use Case diagram?

❑It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found.

❑And once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram.

❑After that, we will enlist the actors that will interact with the system.

❑The actors are the person or a thing that invokes the functionality of a system.

❑It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact.

❑Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected.

❑It identifies the no of times an actor communicates with the system. Basically, an actor can interact multiple times with a use case or system at a particular instance of time.

CASE TOOLs – UML Use Case Diagram

❑How to draw a Use Case diagram?

❑Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

CASE TOOLs – UML Use Case Diagram

❑**Example of a Use Case Diagram:** A use case diagram depicting the Online Shopping website is given below.

❑Here the Web Customer actor makes use of any online shopping website to purchase online.

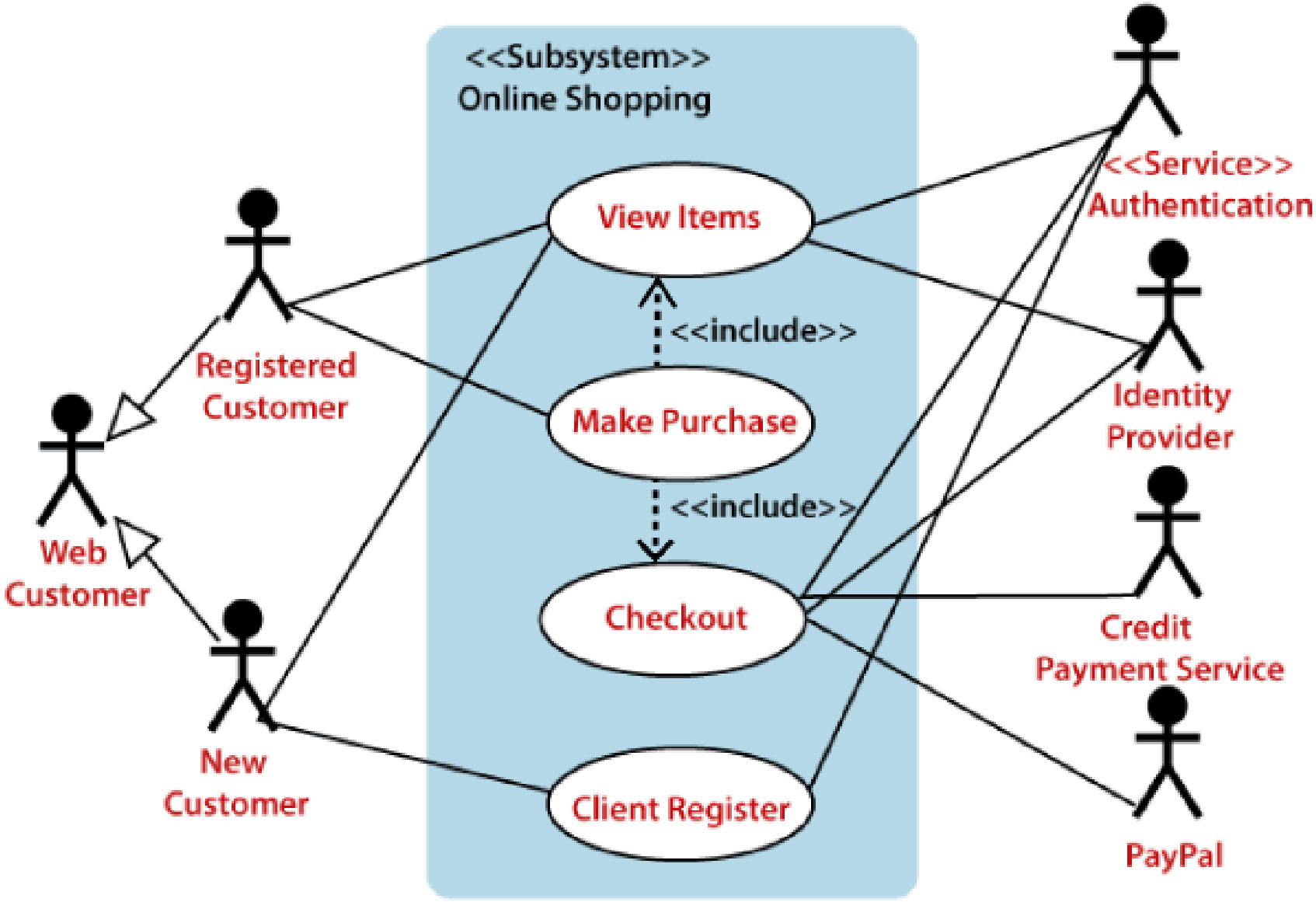
❑The top-level uses are as follows; View Items, Make Purchase, Checkout, Client Register.

❑The **View Items** use case is utilized by the customer who searches and view products.

❑The **Client Register** use case allows the customer to register itself with the website for availing gift vouchers, coupons, or getting a private sale invitation.

❑It is to be noted that the **Checkout** is an included use case, which is part of **Making Purchase,** and it is not available by itself.

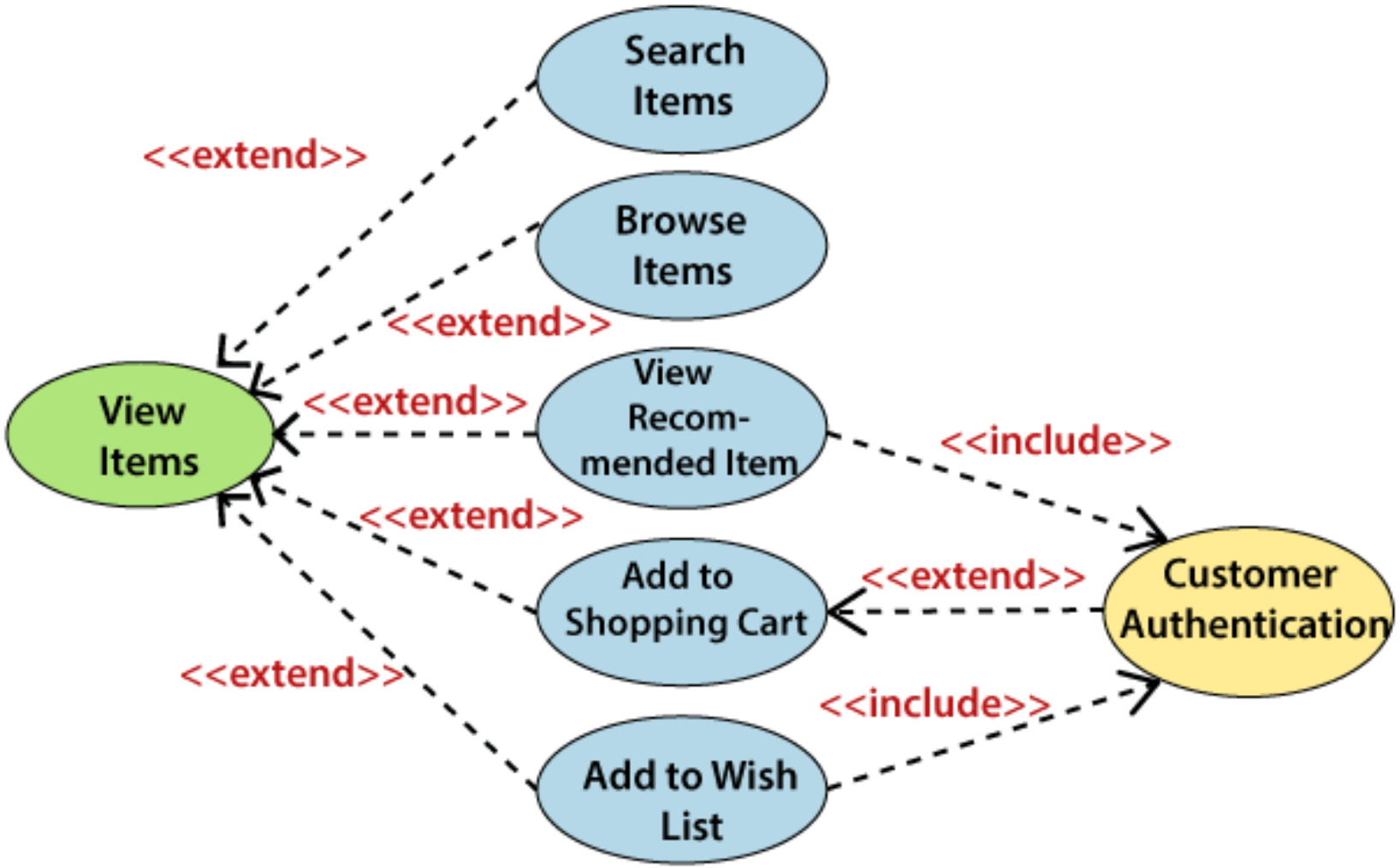
CASE TOOLs – UML Use Case Diagram



CASE TOOLs – UML Use Case Diagram

- The **View Items** is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list.
- All of these extended use cases provide some functions to customers, which allows them to search for an item.
- The View Items is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list.
- All of these extended use cases provide some functions to customers, which allows them to search for an item.
- Both **View Recommended Item** and **Add to Wish List** include the Customer Authentication use case, as they necessitate authenticated customers, and simultaneously item can be added to the shopping cart without any user authentication.

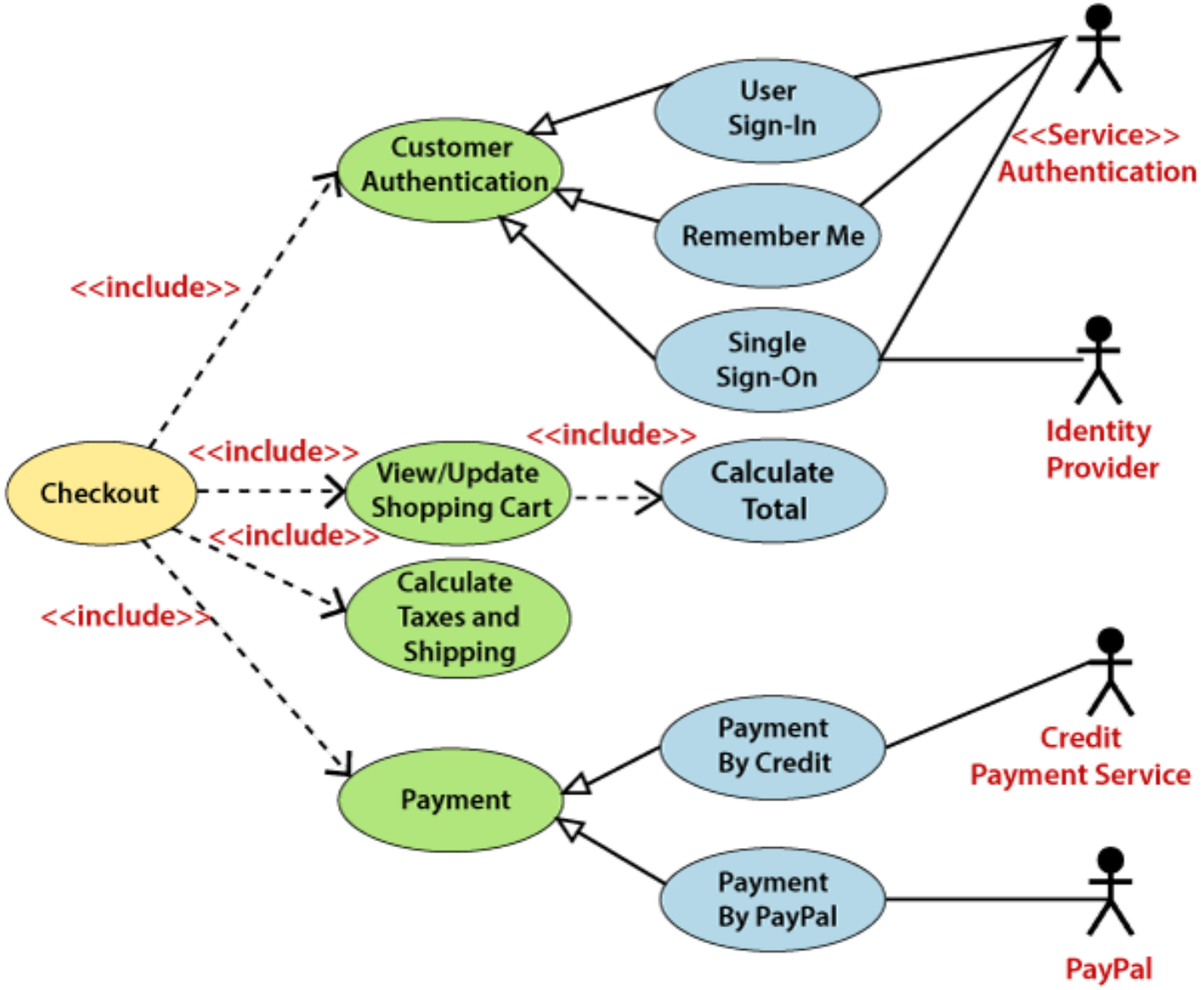
CASE TOOLs – UML Use Case Diagram



CASE TOOLs – UML Use Case Diagram

- Similarly, the **Checkout** use case also includes the following use cases, as shown below.
- It requires an authenticated Web Customer, which can be done by login page, user authentication cookie ("Remember me"), or Single Sign-On (SSO).
- SSO needs an external identity provider's participation, while Web site authentication service is utilized in all these use cases.
- The Checkout use case involves Payment use case that can be done either by the credit card and external credit payment services or with PayPal.

CASE TOOLs – UML Use Case Diagram



CASE TOOLs – UML Use Case Diagram

❑Tips for drawing a Use Case diagram: Following are some important tips that are to be kept in mind while drawing a use case diagram:

1. **A simple and complete use case diagram should be articulated.**
2. **A use case diagram should represent the most significant interaction among the multiple interactions.**
3. **At least one module of a system should be represented by the use case diagram.**
4. **If the use case diagram is large and more complex, then it should be drawn more generalized.**

CASE TOOLs – Collaboration Diagram

❑Steps for creating a Collaboration Diagram

1. Determine the behavior for which the realization and implementation are specified.
2. Discover the structural elements that are class roles, objects, and subsystems for performing the functionality of collaboration.
   - Choose the context of an interaction: system, subsystem, use case, and operation.
3. Think through alternative situations that may be involved.
   - Implementation of a collaboration diagram at an instance level, if needed.
   - A specification level diagram may be made in the instance level sequence diagram for summarizing alternative situations.

CASE TOOLs – State (Machine) Diagram

- The state machine diagram is also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system.
- It captures the software system's behavior. It models the behavior of a class, a subsystem, a package, and a complete system.
- It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system.
- It models event-based systems to handle the state of an object. It also defines several distinct states of a component within the system.
- Each object/component has a specific state.

CASE TOOLs – State (Machine) Diagram

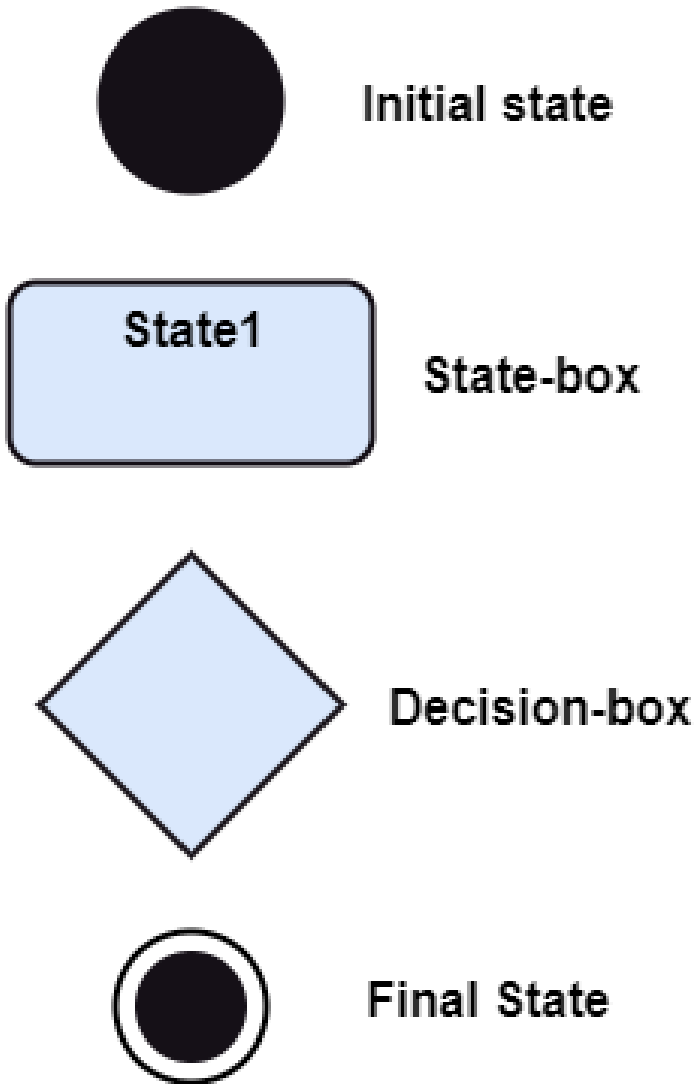
- Following are the types of a state machine diagram that are given below:
- **Behavioral state machine:** The behavioral state machine diagram records the behavior of an object within the system. It depicts an implementation of a particular entity. It models the behavior of the system.
- **Protocol state machine:** It captures the behavior of the protocol. The protocol state machine depicts the change in the state of the protocol and parallel changes within the system. But it does not portray the implementation of a particular component.

CASE TOOLs – State (Machine) Diagram

- **Why State Machine Diagram?**
- Since it records the dynamic view of a system, it portrays the behavior of a software application.
- During a lifespan, an object underwent several states, such that the lifespan exist until the program is executing.
- Each state depicts some useful information about the object.
- It blueprints an interactive system that response back to either the internal events or the external ones.
- The execution flow from one state to another is represented by a state machine diagram.
- It visualizes an object state from its creation to its termination.
- The main purpose is to depict each state of an individual object.
- It represents an interactive system and the entities inside the system.
- It records the dynamic behavior of the system.

CASE TOOLs – State (Machine) Diagram

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



CASE TOOLs – State (Machine) Diagram

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

1. **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
2. **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
3. **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
4. **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.
5. **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.

CASE TOOLs – State (Machine) Diagram

❑Types of State: The UML consist of three states:

1. **Simple state:** It does not constitute any substructure.
2. **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.
3. **Submachine state:** The submachine state is semantically identical to the composite state, but it can be reused.

CASE TOOLs – State (Machine) Diagram

❑How to Draw a State Machine Diagram?

- ❑The state machine diagram is used to portray various states underwent by an object.
- ❑The change in one state to another is due to the occurrence of some event.
- ❑All of the possible states of a particular component must be identified before drawing a state machine diagram.
- ❑The primary focus of the state machine diagram is to depict the states of a system.
- ❑These states are essential while drawing a state transition diagram.
- ❑The objects, states, and events due to which the state transition occurs must be acknowledged before the implementation of a state machine diagram.

CASE TOOLs – State (Machine) Diagram

❑How to Draw a State Machine Diagram? Steps

1. A unique and understandable name should be assigned to the state transition that describes the behavior of the system.
2. Out of multiple objects, only the essential objects are implemented.
3. A proper name should be given to the events and the transitions.

CASE TOOLs – State (Machine) Diagram

- State machine diagram is used for:

1. For modeling the object states of a system.
2. For modeling the reactive system as it consists of reactive objects.
3. For pinpointing the events responsible for state transitions.
4. For implementing forward and reverse engineering.

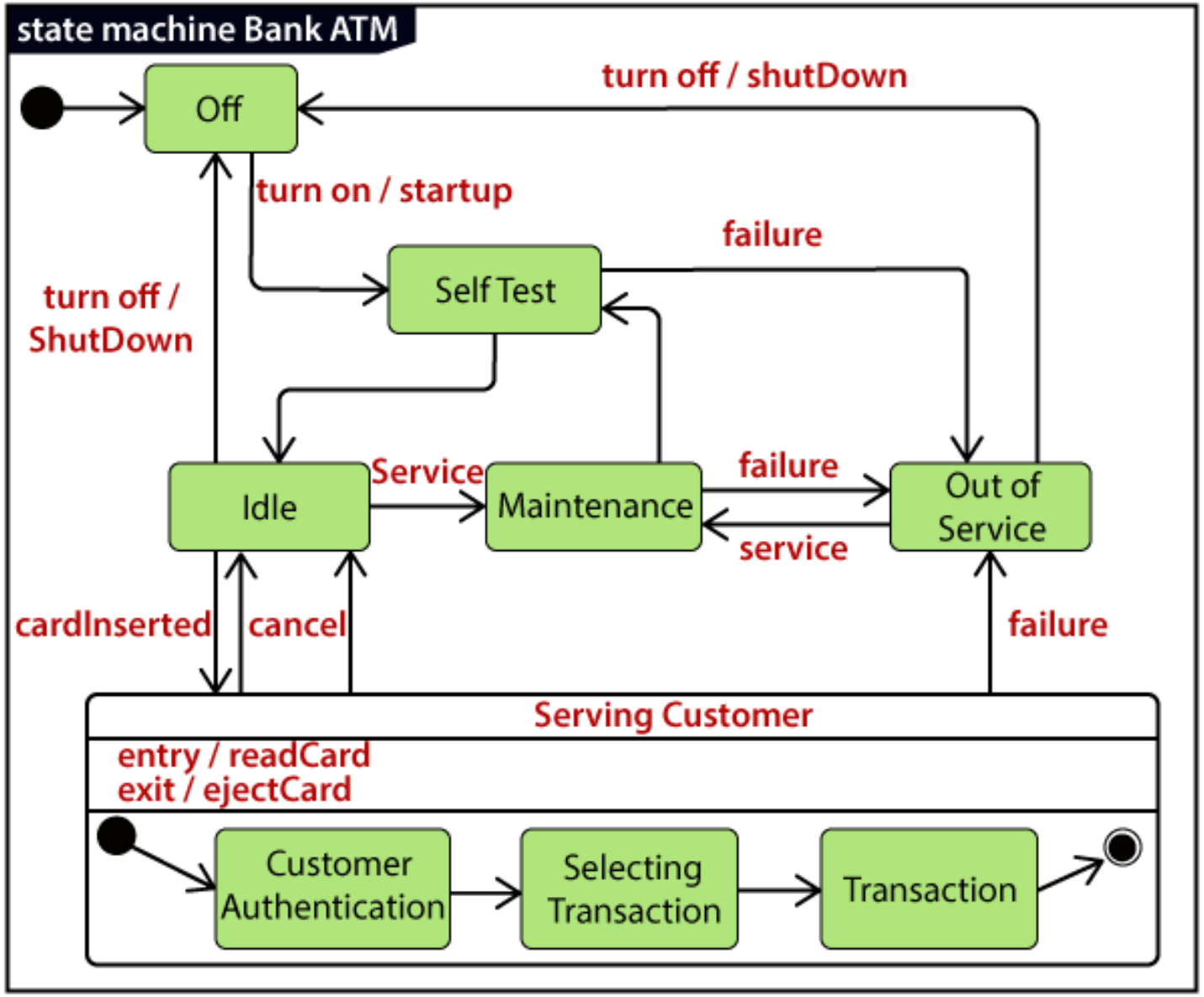
CASE TOOLs – State (Machine) Diagram

❑Example:

1. An example of a top-level state machine diagram showing Bank Automated Teller Machine (ATM) is given below.
2. Initially, the ATM is turned off. After the power supply is turned on, the ATM starts performing the startup action and enters into the **Self Test** state.
3. If the test fails, the ATM will enter into the **Out Of Service** state, or it will undergo **a triggerless transition** to the **Idle** state. This is the state where the customer waits for the interaction.
4. Whenever the customer inserts the bank or credit card in the ATM's card reader, the ATM state changes from **Idle** to **Serving Customer**, the entry action **readCard** is performed after entering into **Serving Customer** state.
5. Since the customer can cancel the transaction at any instant, so the transition from **Serving Customer** state back to the **Idle** state could be triggered by **cancel** event.

CASE TOOLs – State (Machine) Diagram

❑Example:



CASE TOOLs – State (Machine) Diagram

❑Example:

6. Here the **Serving Customer** is a composite state with sequential substates that are **Customer Authentication, Selecting Transaction,** and **Transaction**.
7. **Customer Authentication** and **Transaction** are the composite states itself is displayed by a hidden decomposition indication icon.
8. After the transaction is finished, the **Serving Customer** encompasses a triggerless transition back to the **Idle** state.
9. On leaving the state, it undergoes the exit action **ejectCard** that discharges the customer card.

Note for Students

- This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.