4. SYSTEM MODELING

Contents:

System Modeling

System modeling

- System modeling is **the process of developing abstract models of a system,** with each model **presenting a different view** or perspective of that system.
- System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System modelling **helps the analyst to understand the functionality of the system** and models are used to communicate with customers.

Different perspective for System modeling

1. **An external perspective**,
   - where you model the **context** or **environment** of the system.
2. **An interaction perspective**,
   - where you model the **interactions between a system and its environment**, or between the components of a system.
3. **A structural perspective,**
   - where you model **the organization of a system** or the **structure of the data** that is processed by the system.
4. **A behavioral perspective,**
   - where you model the **dynamic behavior of the system** and how it responds to events.

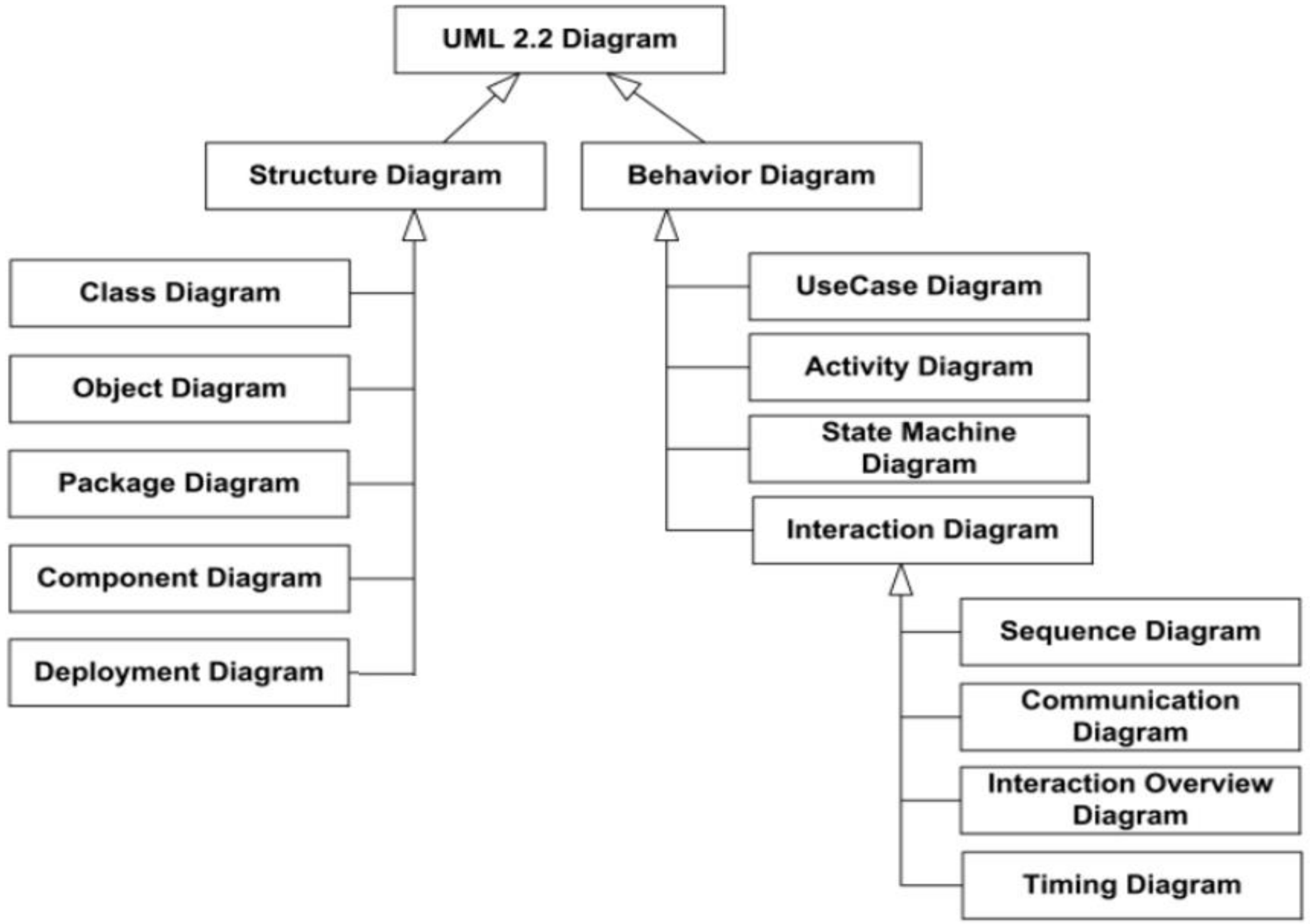
The Unified Modeling Language

- The Unified Modeling Language (UML) is a set of 13 different diagram types that may be **used to model software systems**.
- It emerged from work in the 1990s on object-oriented modeling, where similar object-oriented notations were integrated to create the UML.
- A major revision (UML 2) was finalized in 2004.
- The UML **is universally accepted as the standard approach for developing models of software systems.**
- “The UML **is a graphical language** for specifying, visualizing, constructing and documenting the **artifacts** of the software system”.
- Variants, such as SysML, have been proposed for more general system modeling.

Use of UML

- **Specifying** –provides the means to model the system precisely, unambiguously and completely.
- **Visualizing** – UML provides graphical notation which articulates and unambiguously communicates the overall view of the system.
- **Constructing** - UML provides the ‘design’ dimension to the models built. These are language independent and can be implemented in any programming language.
- **Documenting** – every software project involves a lot of documentation from the inception phase to the deliverables.

Classification of UML Diagrams



a) Structural Diagrams

- Structural diagrams are used to represent a static view of a system.
- It represents a part of a system that makes up the structure of a system.
- A structural diagram shows various objects within the system.
- Following are the various structural diagrams in UML:
    - Class diagram
    - Object diagram
    - Package diagram
    - Component diagram
    - Deployment diagram

b) Behavioral Diagrams

- Any real-world system can be represented in either a static form or a dynamic form. A system is said to be complete if it is expressed in both the static and dynamic ways.
- The behavioral diagram **represents the functioning of a system.**
- UML diagrams that **deals with the moving or dynamic parts of the system** are called behavioral diagrams.
- Following are the various behavioral diagrams in UML:
  - Activity diagram
  - Use case diagram
  - State machine diagram
  - Interaction diagram

UML diagram types

- **Activity diagrams**, which show the activities involved in a process or in data processing .
- **Use case diagrams,** which show the interactions between a system and its environment.
- **Sequence diagrams**, which show interactions between actors and the system and between system components.
- **Class diagrams,** which show the object classes in the system and the associations between these classes.
- **State diagrams,** which show how the system reacts to internal and external events.

Use of graphical models

- **As a means of facilitating discussion about an existing or proposed system**
  - Incomplete and incorrect models are OK as their role is to support discussion.
- **As a way of documenting an existing system**
  - Models should be an accurate representation of the system but need not be complete.
- **As a detailed system description** that can be used to generate a system implementation
  - Models have to be both correct and complete.

4.1 Context models

Context models

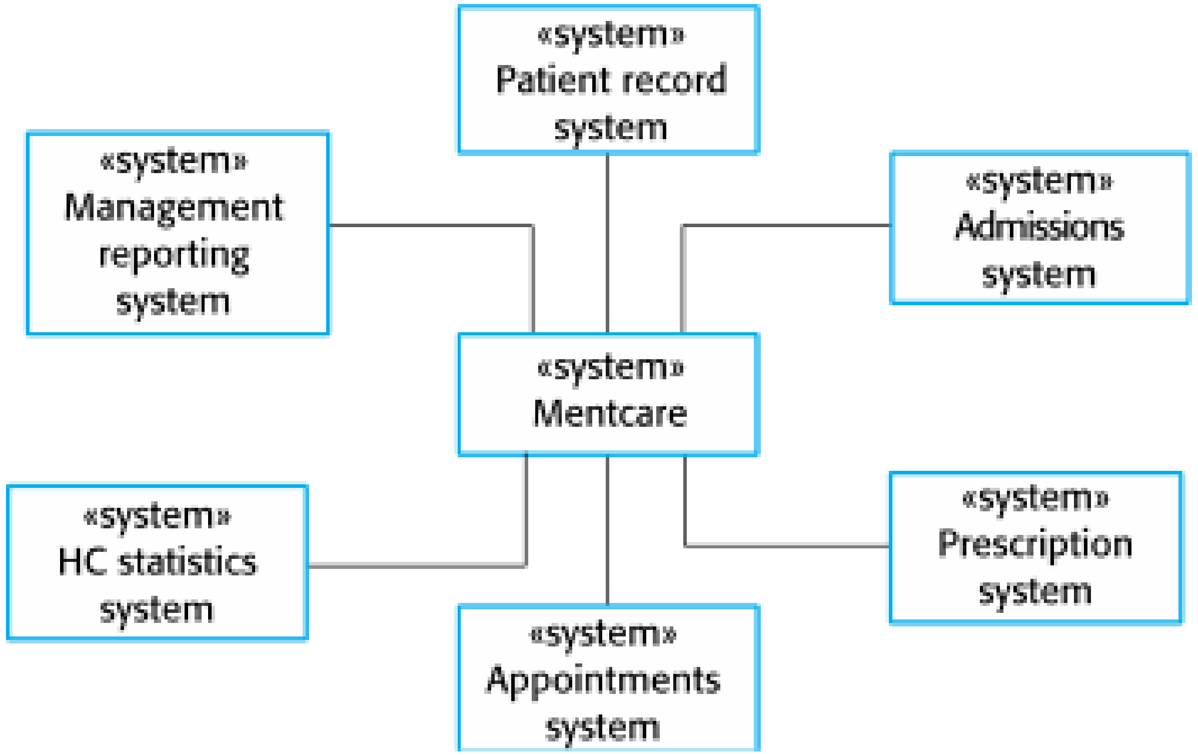
- Context models are **used to illustrate the operational context of a system** - they show what lies outside the system boundaries.
- Social and organizational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.
- System boundaries are established to define what is inside and what is outside the system.
    - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
    - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

Context Diagram

- The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world.
- It identifies the entities outside the proposed system that interact with the system.

The context of the Mentcare system

- Mental Health Care – Patient Management Systems
- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.


«system» Patient record system
«system» Management reporting system
«system» Admissions system
«system» Mentcare
«system» HC statistics system
«system» Prescription system
«system» Appointments system

MENTCARE: A PATIENT INFORMATION SYSTEM FOR MENTAL HEALTH CARE

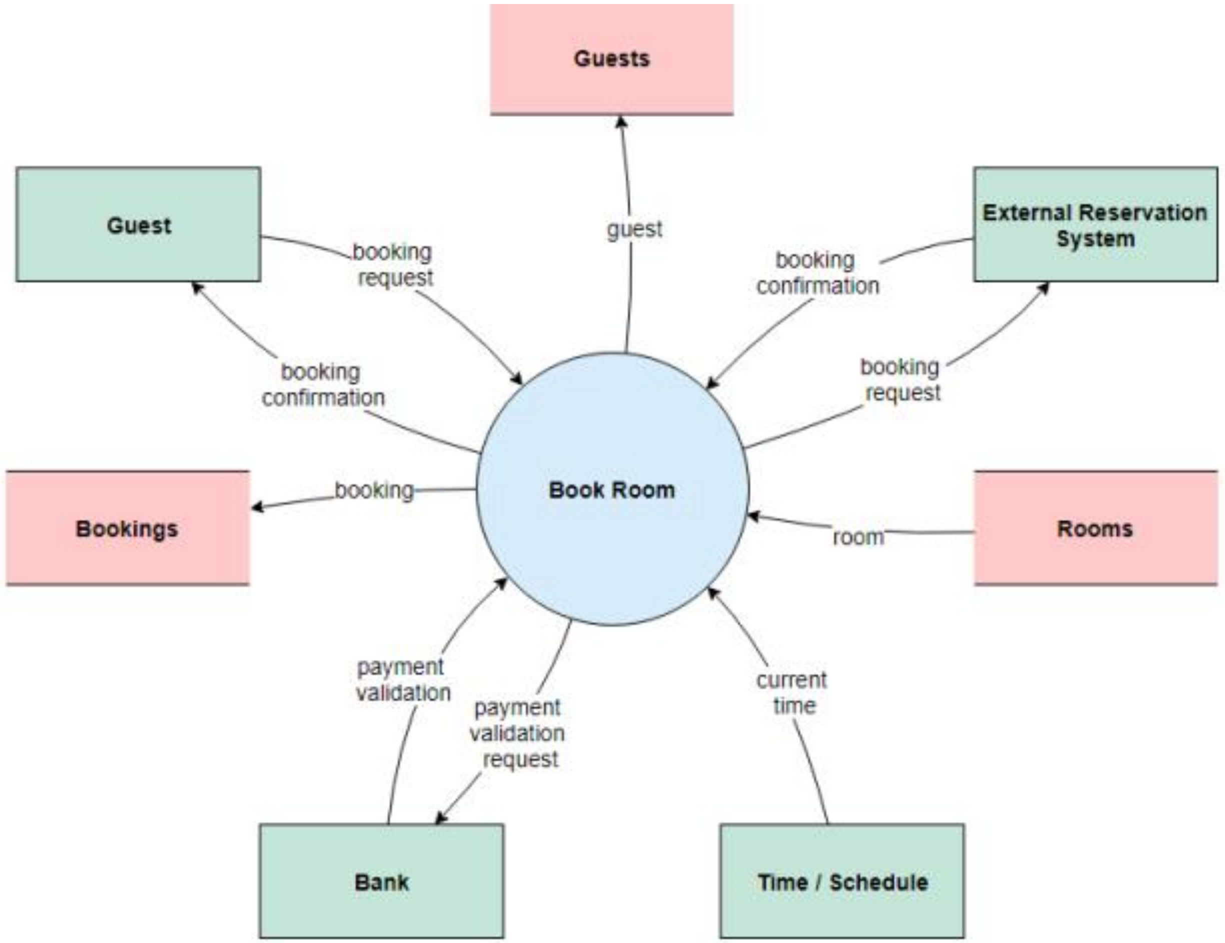
- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received. • Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems. • To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.
- Mentcare is an information system that is intended for use in clinics. • It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity. • When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

Context Diagram

- A system context diagram (SCD) in engineering is **a diagram that defines the boundary between the system, or part of a system, and its environment**, showing the entities that interact with it.
- This diagram is **a high level view of a system.**
- System context diagrams show a system, as a whole and its inputs and outputs from/to external factors.
- System Context Diagrams ... **represent all external entities that may interact with a system** ... Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments and activities.
- The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints.
- System context diagrams **are used early in a project** to get agreement on the scope under investigation.
- Context diagrams are typically **included in a requirements document.**
- These diagrams must be read by all project stakeholders and thus should be written in plain language, so the stakeholders can understand items within the document.

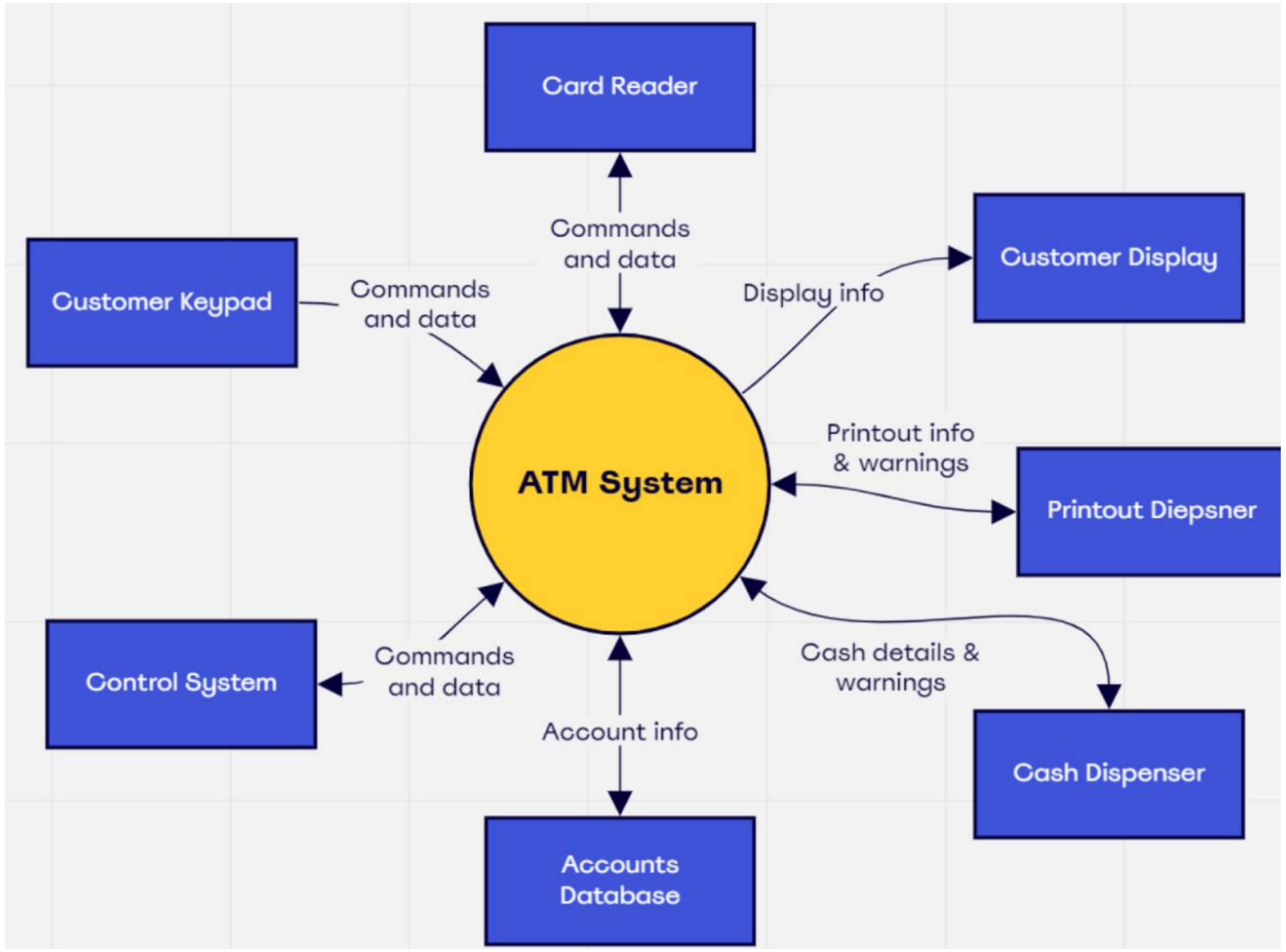
Context Diagram: Online Reservation System

- Context diagram for Online Reservation System.
- Main system in the center circle.
- Using squares, rectangles, or any other shapes you want, you can list external entities that interact with the system.
- arrows are used to represent the flow of data between the system and each external element.



Context Diagram : ATM

- ATM context diagram
- outlines how your banking ATM process will work.
- The diagram shows all the external units that interact with the ATM system and how data flows between them.



When to use Context Diagram

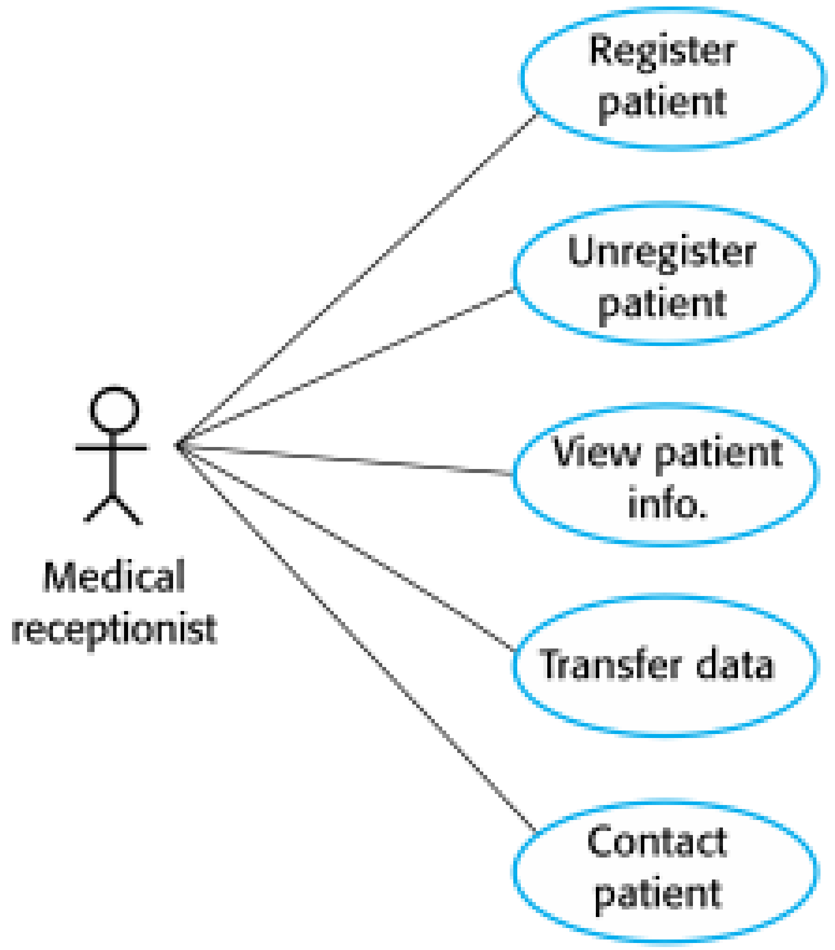
- A context diagram is helpful in a variety of situations. Here are some of the common scenarios:
  1. Implementing or updating software
  2. Reviewing a business process
  3. Tackling resource management

4.2 Interaction models

- Modeling user interaction is important as it **helps to identify user requirements.**
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Interaction models describe the relationship between a user and the system, outlining the flow and sequence of actions and responses.
- Interaction model uses:
  - **Use case diagrams**
  - **sequence diagrams**
  - **Activity diagram**
  - **Collaborative diagram**

Use case modeling

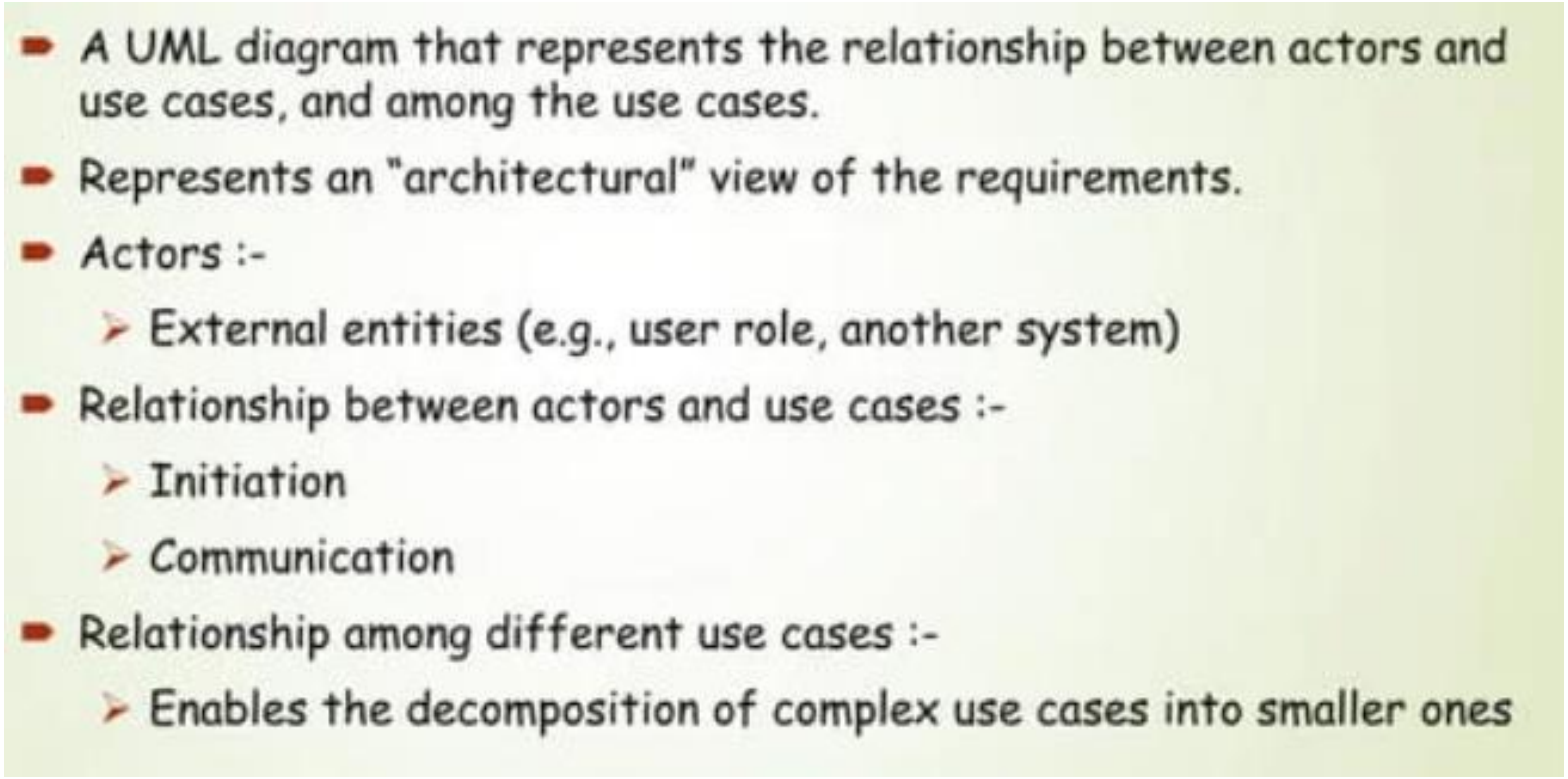
- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagrammatically to provide an overview of the use case and in a more detailed textual form.
- **Use cases in the Mentcare system involving the role 'Medical Receptionist'**



Use-case Diagram

- **Use Case Diagram** captures the system’s functionality and requirements by using actors and use cases.
- Use Cases model the services, tasks, function that a system needs to perform.
- Use cases represent high-level functionalities and how a user will handle the system.
- It models how an external entity interacts with the system to make it work.
- Use case diagrams are responsible for visualizing the external things that interact with the part of the system.
- It models the system from the end-users point of views. So, the actor should be identified first and their roles are also defined

Use-case Diagram



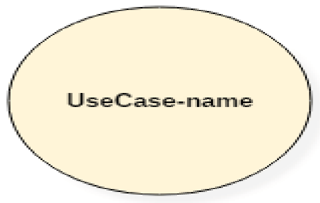
- A UML diagram that represents the relationship between actors and use cases, and among the use cases.
- Represents an "architectural" view of the requirements.
- Actors :-
  - External entities (e.g., user role, another system)
- Relationship between actors and use cases :-
  - Initiation
  - Communication
- Relationship among different use cases :-
  - Enables the decomposition of complex use cases into smaller ones

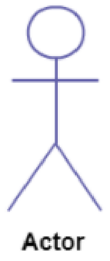
Use-case Diagram

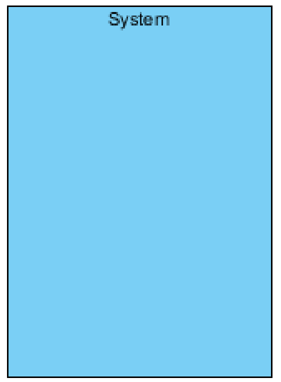
- Purposes of Use-case diagrams:
  - To gather requirements of a system
  - To present an outside view of a system
  - Identify external and internal factor influencing the system.
  - Show the interaction among requirements and actors

Notations in Use-case diagrams:

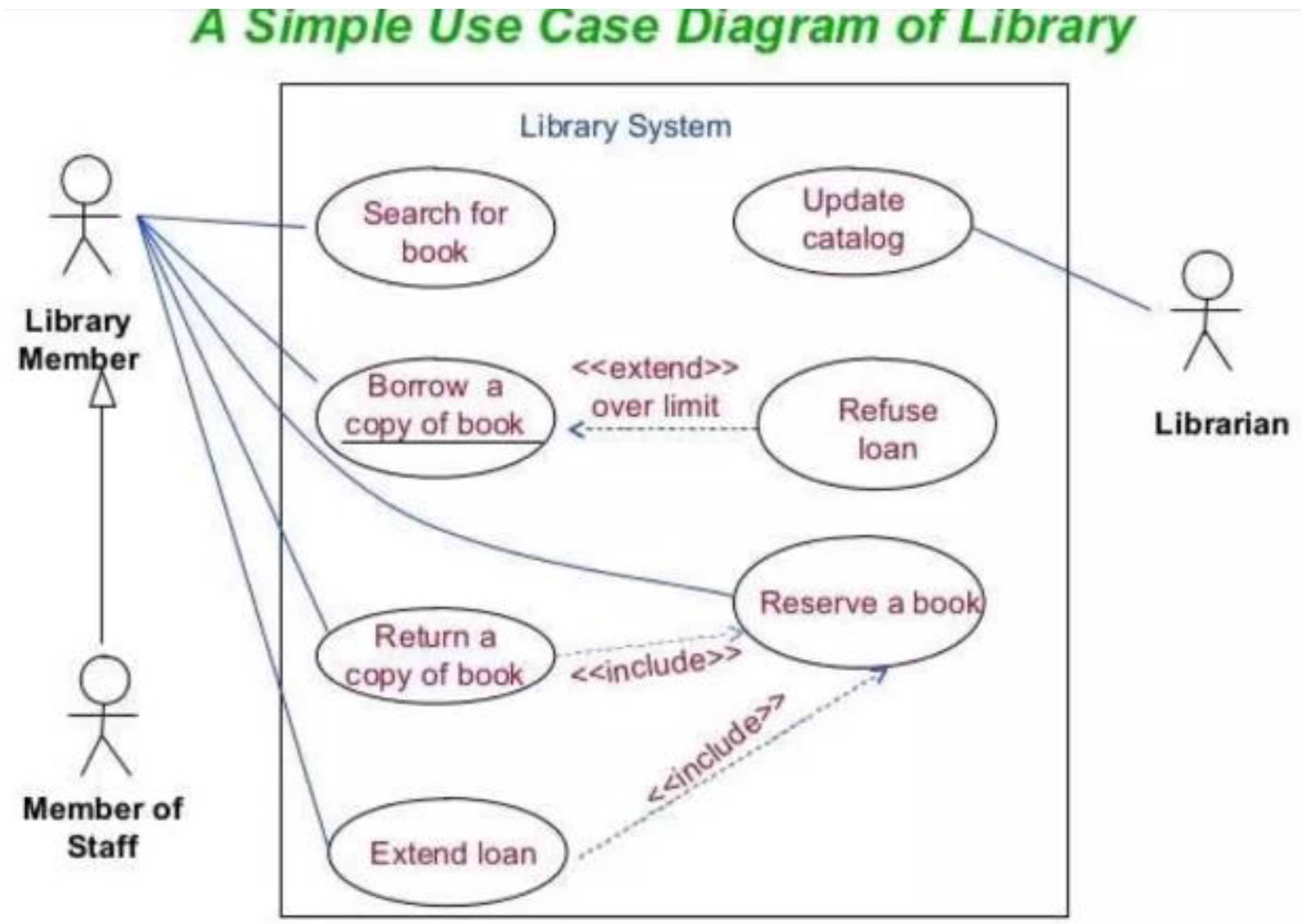
- **Use-case:**
  - Use cases are used to represent high-level functionalities and how the user will handle the system.
  - It is denoted by an oval shape with the name of a use case written inside the oval shape.
- **Actor:**
  - The actor is an entity that interacts with the system. A user is the best example of an actor.
  - An actor initiates the use case from outside the scope of a use case.
  - It can be any element that can trigger an interaction with the use case.
  - One actor can be associated with multiple use cases in the system.
- **Communication Link**
  - The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link.
- **Boundary of system**
  - The system boundary is potentially the entire system as defined in the requirements document.
  - For large and complex systems, each module may be the system boundary.







Examples: Use-case diagrams



Examples: Use-case diagrams

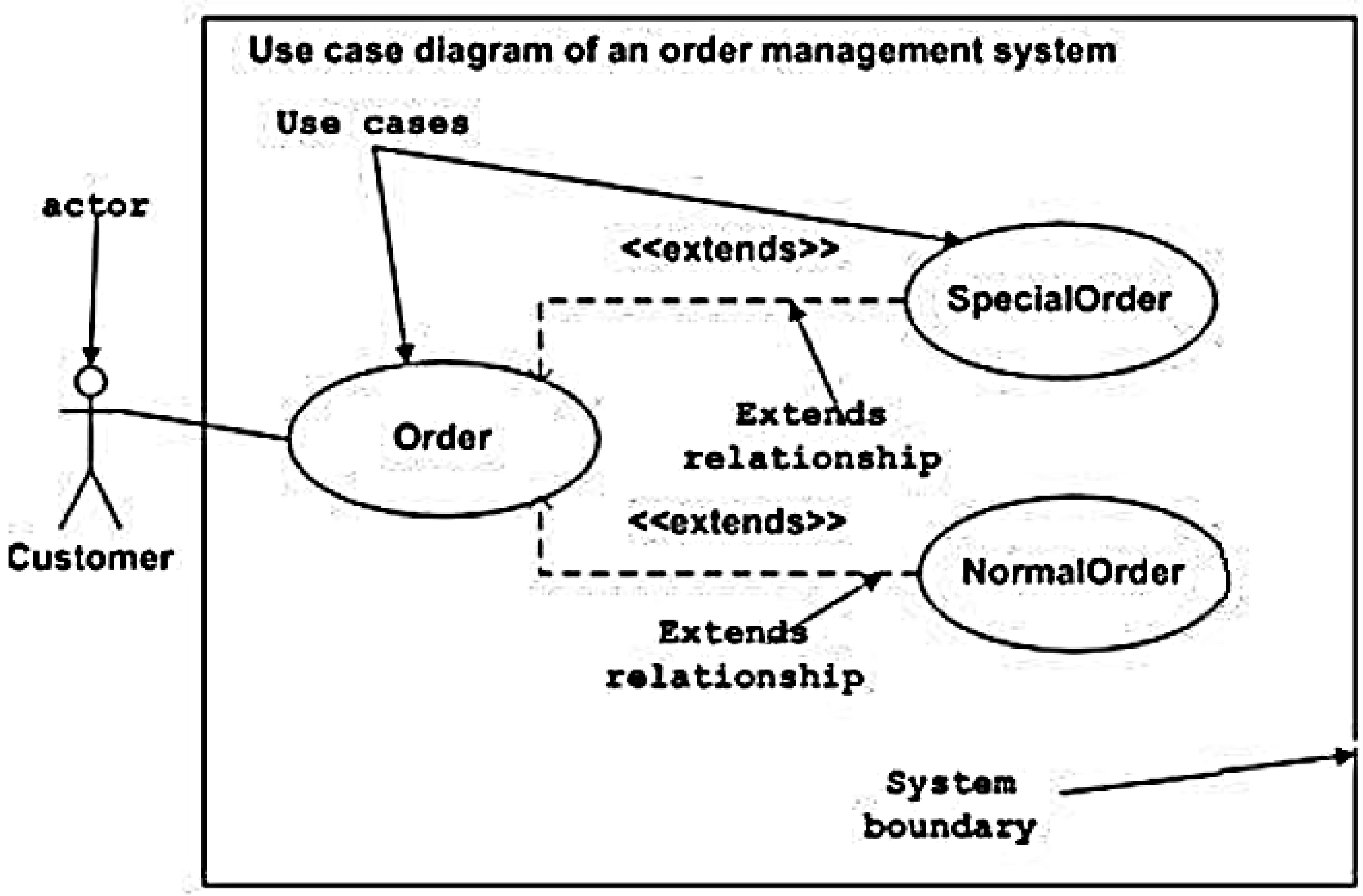
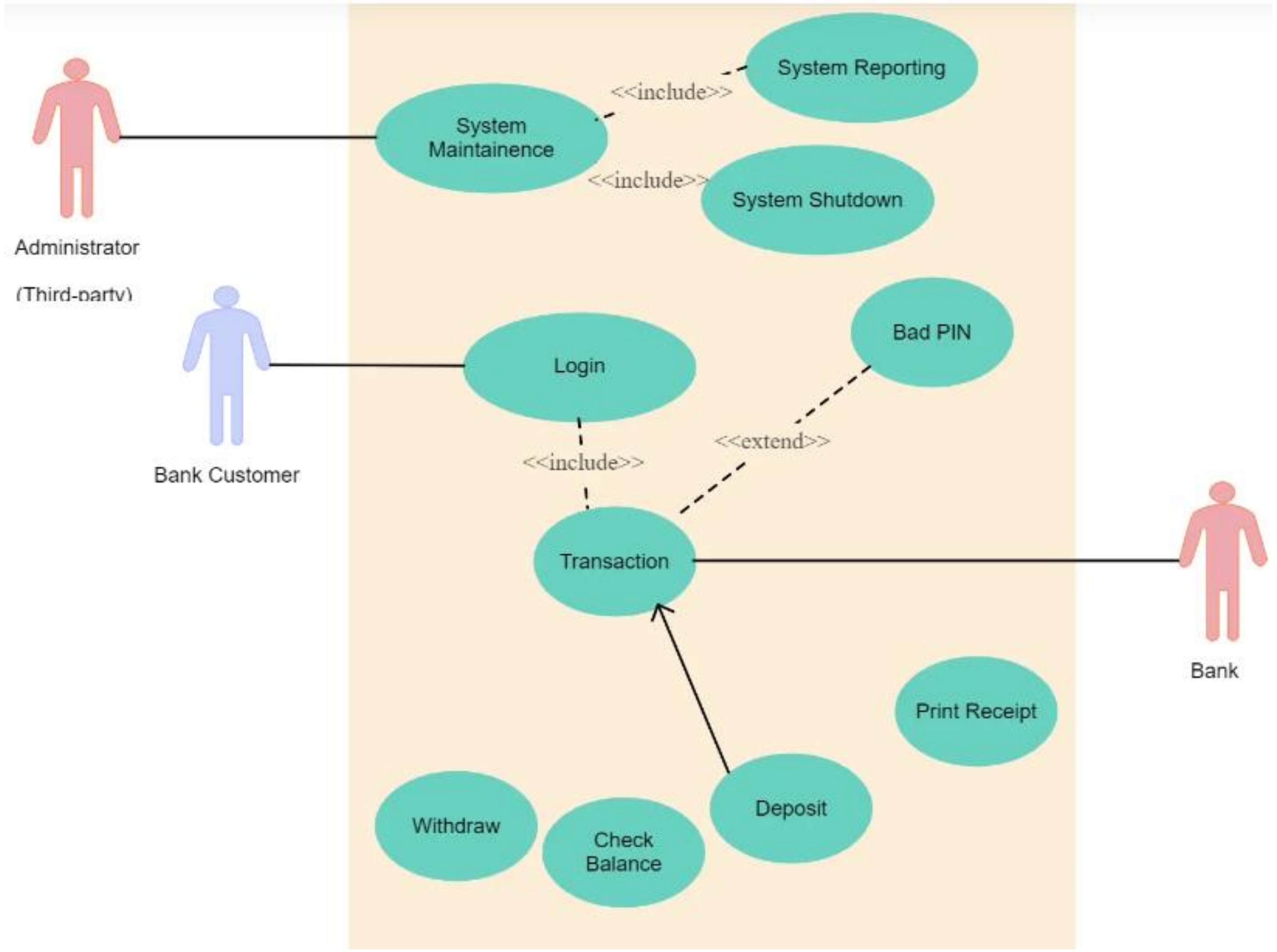
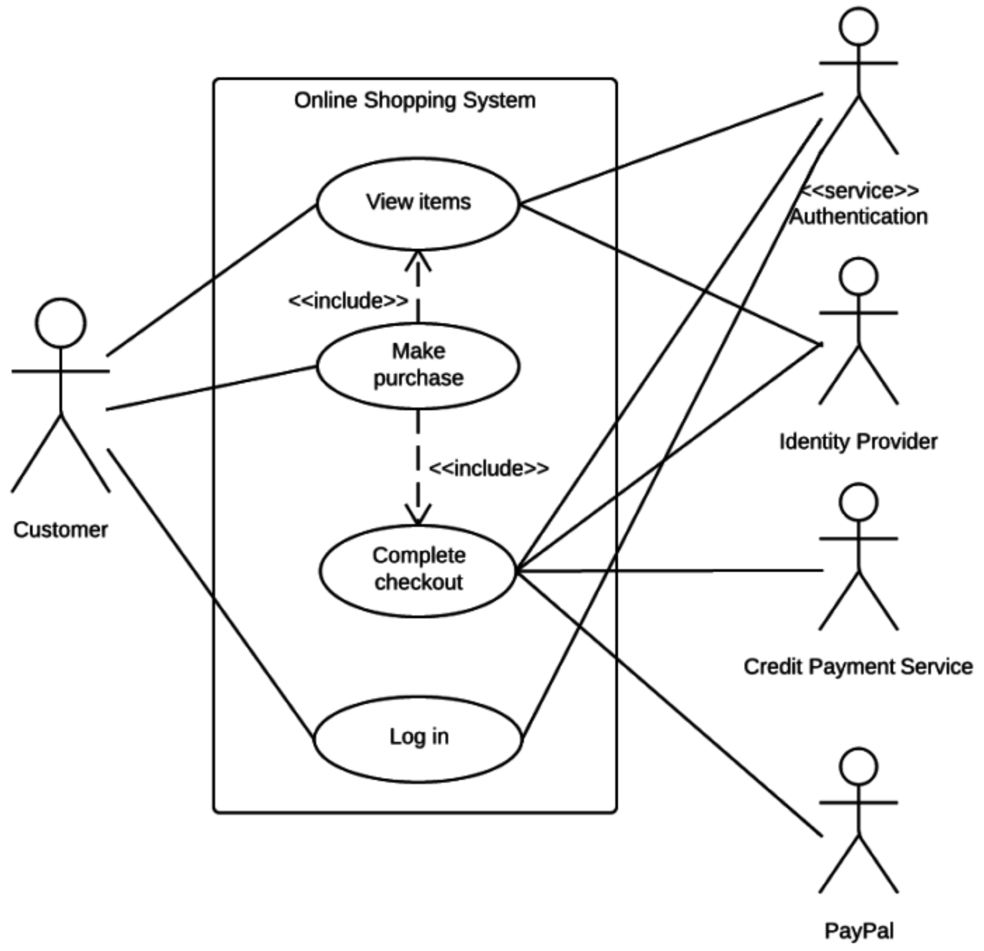


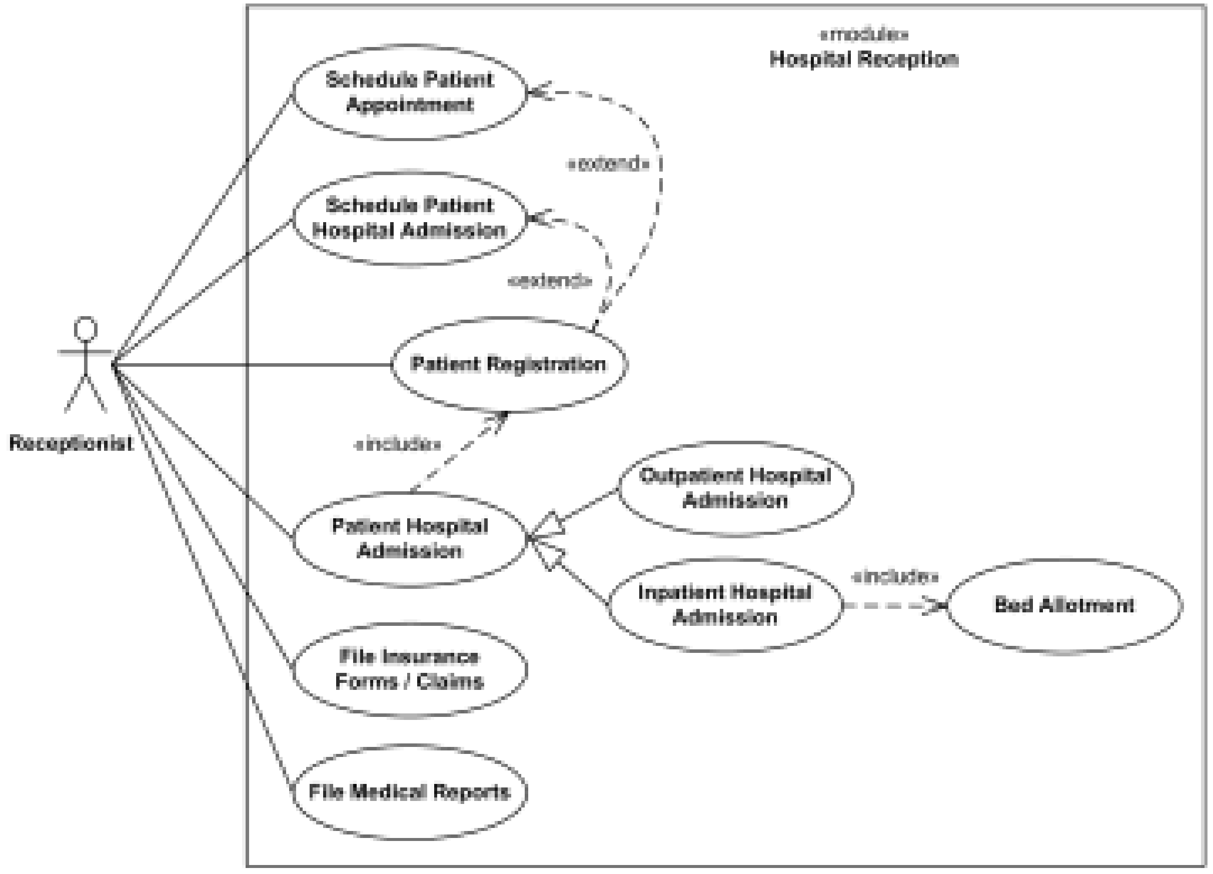
Figure: Sample Use Case diagram

ATM: Use-case diagrams

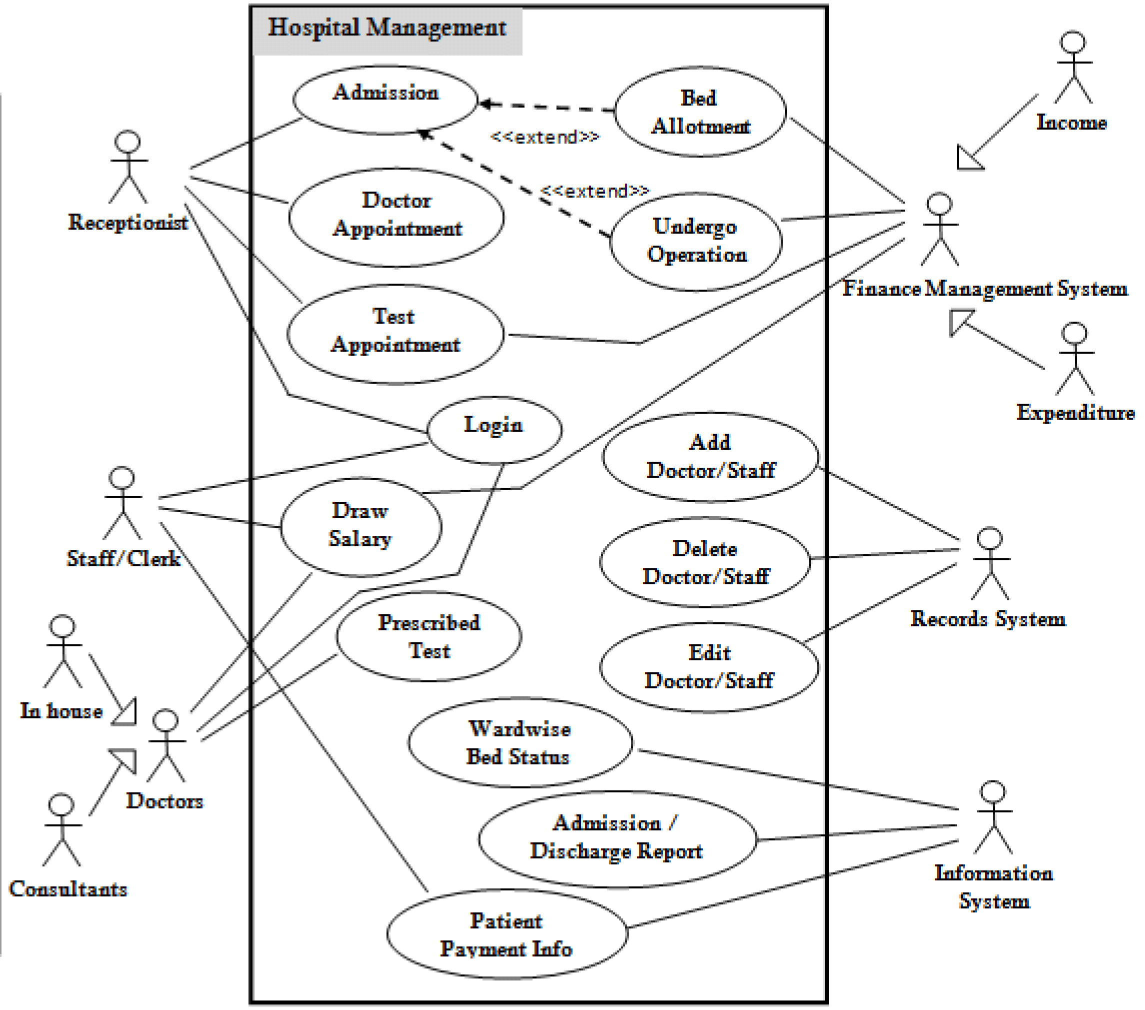


Order Management System: Use-case diagrams





«module»
Hospital Reception
Receptionist
Schedule Patient Appointment
Schedule Patient Hospital Admission
«extend»
«extend»
Patient Registration
«include»
Patient Hospital Admission
Outpatient Hospital Admission
Inpatient Hospital Admission
«include»
Bed Allotment
File Insurance Forms / Claims
File Medical Reports



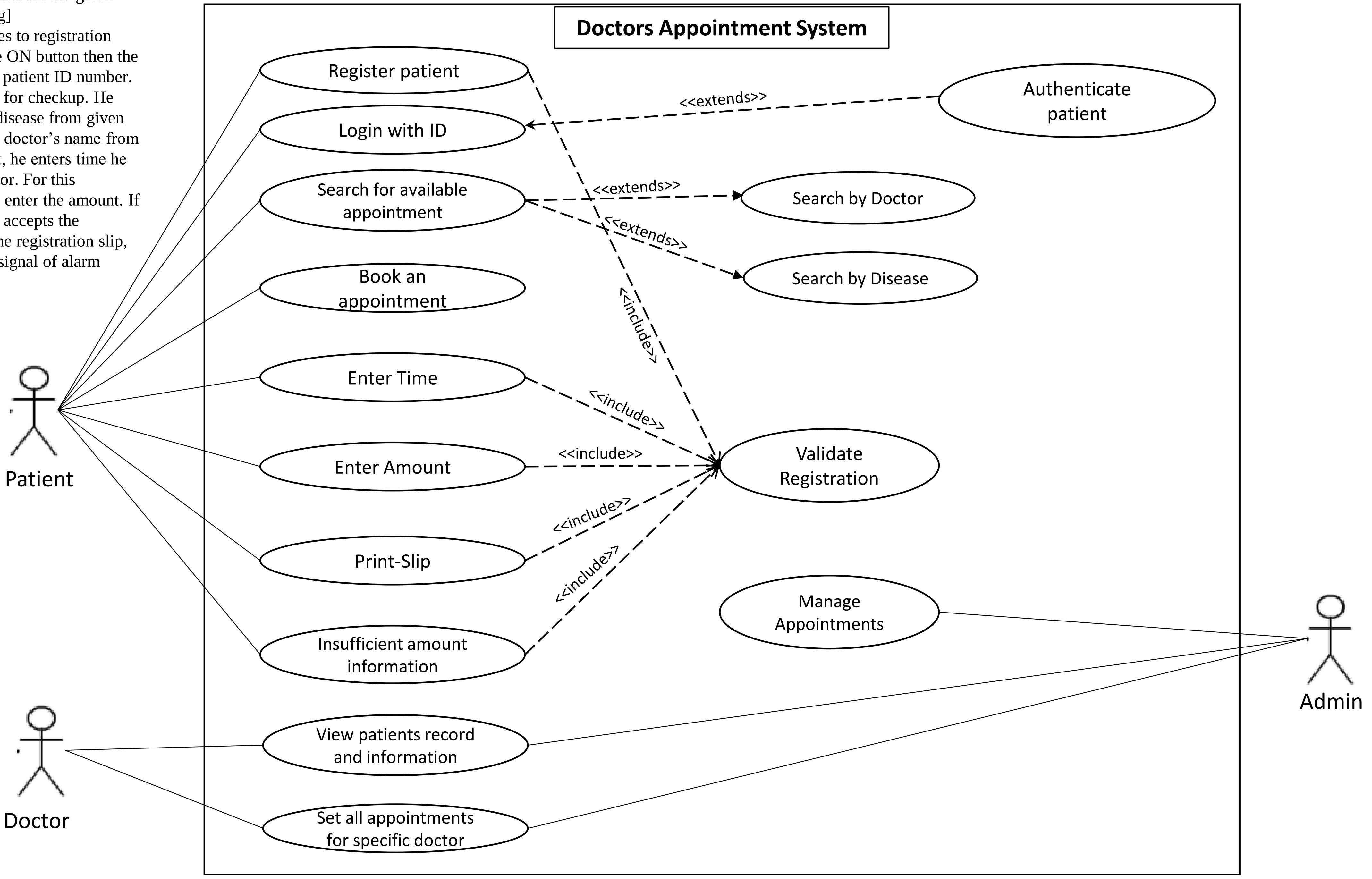
Hospital Management
Admission
Bed Allotment
<<extend>>
<<extend>>
Doctor Appointment
Undergo Operation
Test Appointment
Login
Draw Salary
Prescribed Test
Add Doctor/Staff
Delete Doctor/Staff
Edit Doctor/Staff
Wardwise Bed Status
Admission / Discharge Report
Patient Payment Info
Receptionist
Staff/Clerk
In house
Doctors
Consultants
Income
Finance Management System
Expenditure
Records System
Information System

Assignment:

- Draw a use case diagram from the given case study.

  In hospital, a patient goes to registration machine. He presses the ON button then the screen opens. He enters patient ID number. He books for the doctor for checkup. He checks the category of disease from given list, then he chooses the doctor's name from given doctor's name list, he enters time he wants to meet with doctor. For this registration, he needs to enter the amount. If the amount digit is ok it accepts the registration and prints the registration slip, otherwise it will give a signal of alarm "insufficient amount".

Draw a use case diagram from the given case study. [2019 Spring]
In hospital, a patient goes to registration machine. He presses the ON button then the screen opens. He enters patient ID number. He books for the doctor for checkup. He checks the category of disease from given list, then he chooses the doctor's name from given doctor's name list, he enters time he wants to meet with doctor. For this registration, he needs to enter the amount. If the amount digit is ok it accepts the registration and prints the registration slip, otherwise it will give a signal of alarm "insufficient amount".



Sequence diagrams

- Sequence diagrams are used to **model the interactions between the actors and the objects within a system.**
- A sequence diagram **shows the sequence of interactions** that take place during a particular use case or use case instance.
- The **objects and actors** involved are **listed along the top of the diagram**, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

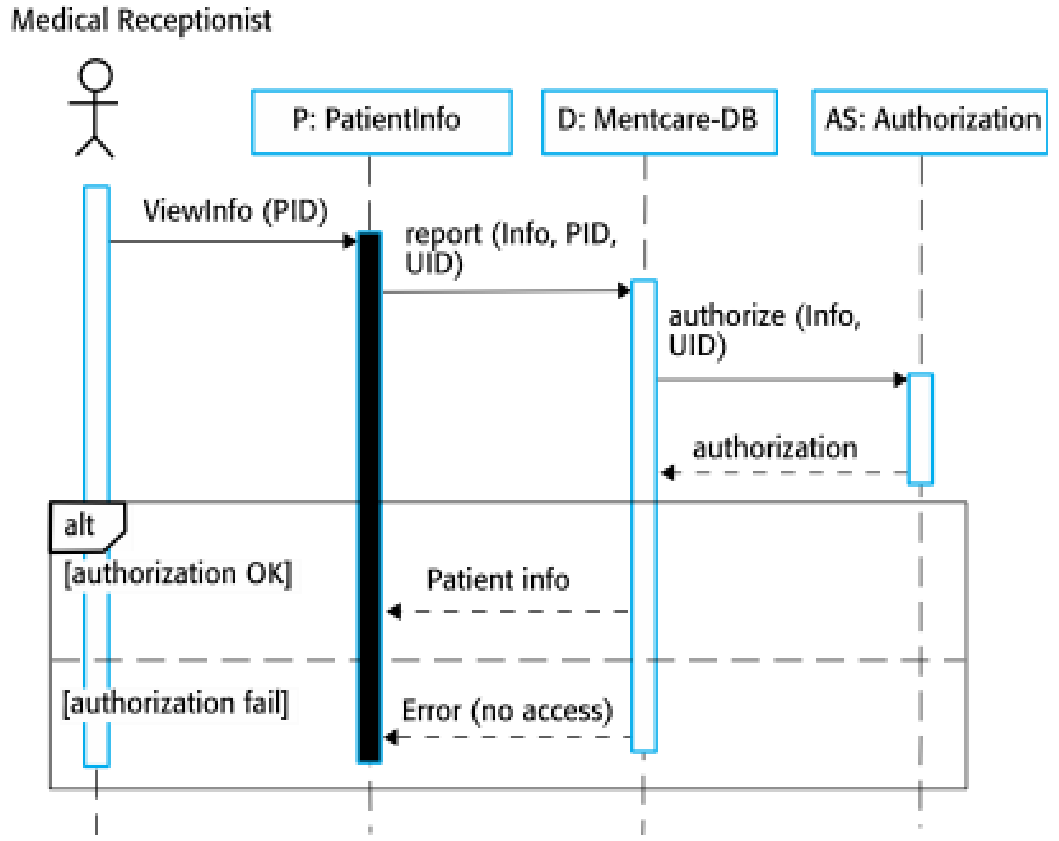


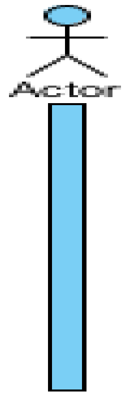
Figure: Sequence diagram for View patient information

Sequence Diagram

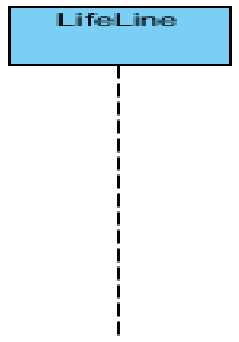
- A *sequence diagram* is an interaction diagram that emphasizes the time ordering of messages.
- It shows a set of objects and the messages sent and received by those objects.
- Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis.

Notations in Sequence Diagram :

1. **Actor:**
   - that interact with the system.
   - represent roles played by human users, external hardware, or other subjects.



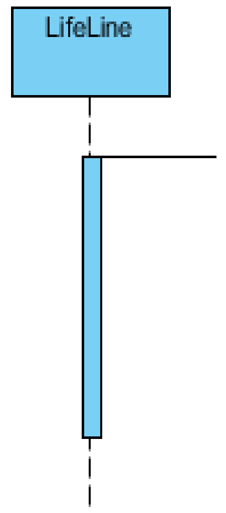
2. **Lifeline**
   - A lifeline represents an individual participant in the Interaction.



Notations in Sequence Diagram :

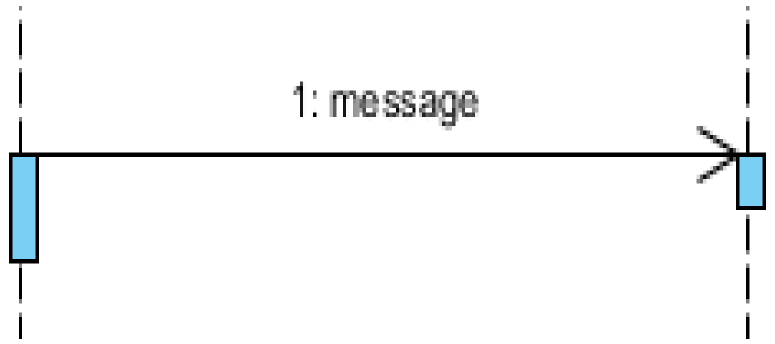
4. Activations

- A thin rectangle on a lifeline) represents the period during which an element is performing an operation.



5 Call Message

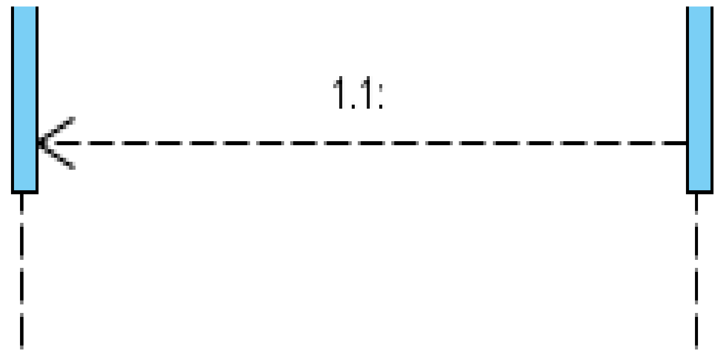
- A message defines a particular communication between Lifelines of an Interaction.
- Call message is a kind of message that represents an invocation of operation of target lifeline.



Notations in Sequence Diagram:

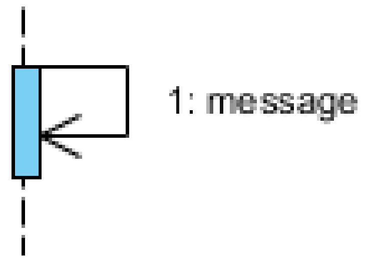
6. **Return Message**

   Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message



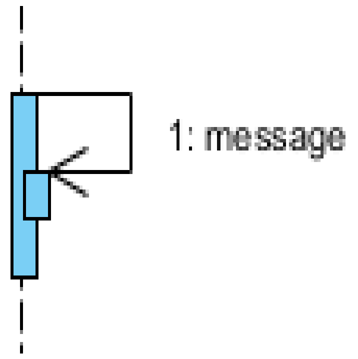
7. **Self Message**

   Self message is a kind of message that represents the invocation of message of the same lifeline..



8. **Recursive Message**

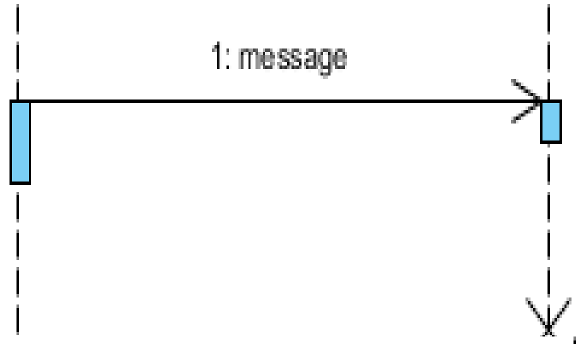
   Recursive message is a kind of message that represents the invocation of message of the same lifeline. It's target points to an activation on top of the activation where the message was invoked from.



Notations in Sequence Diagram :

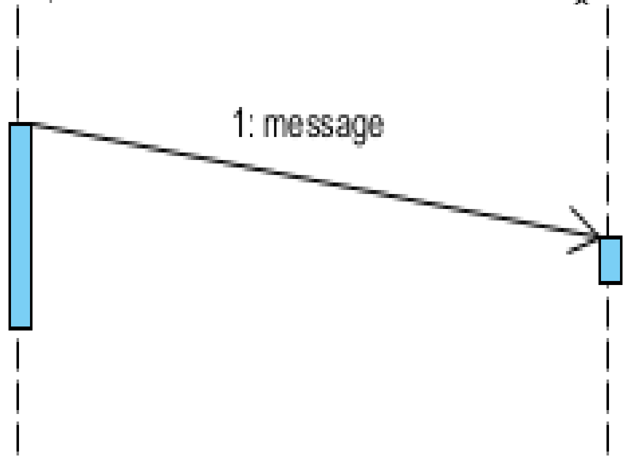
9. Destroy Message

Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.



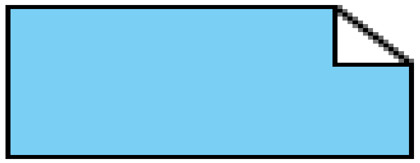
10.Duration Message

Duration message shows the distance between two time instants for a message invocation.



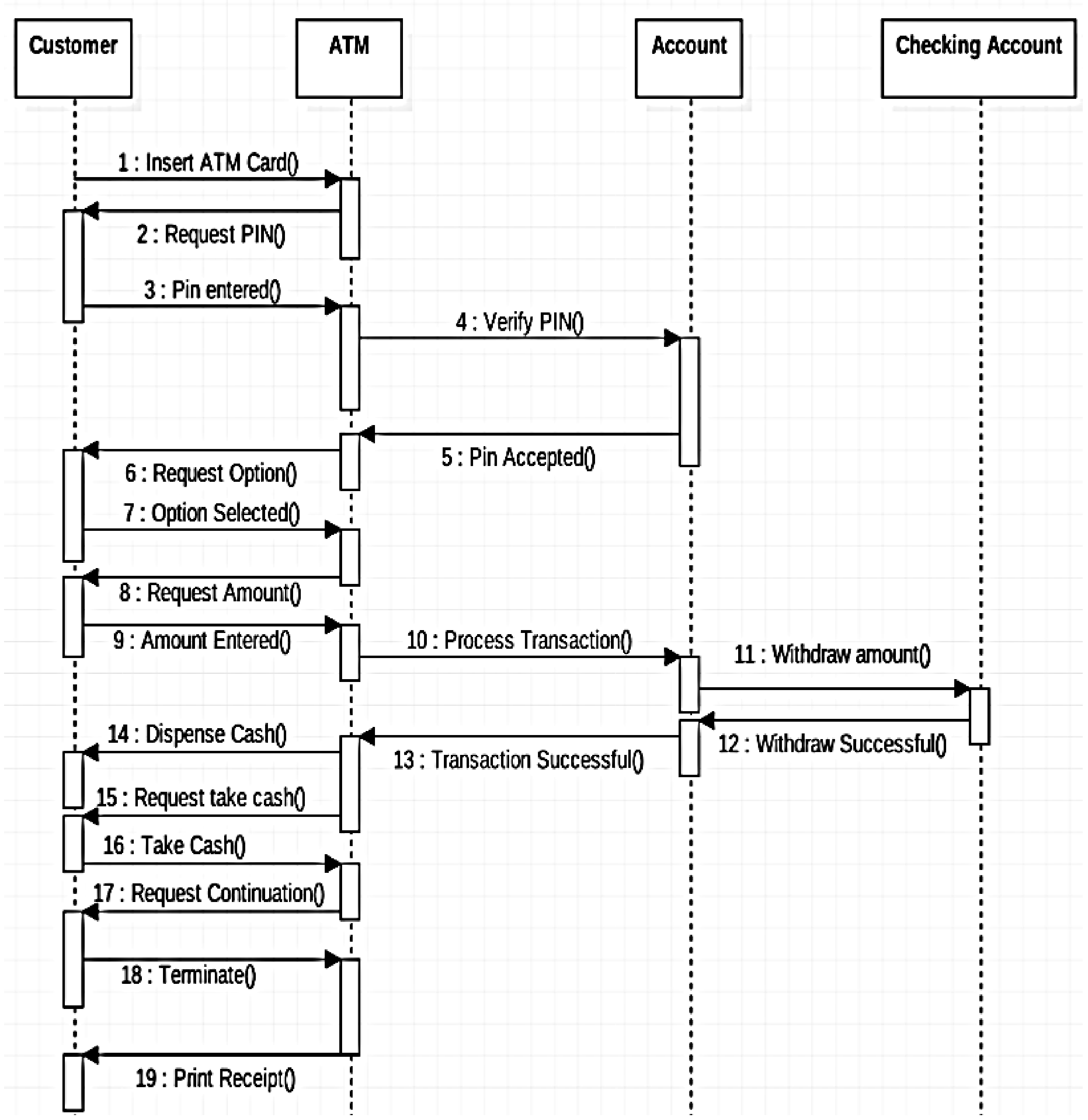
11.Note

A note (comment) gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.sage invocation.

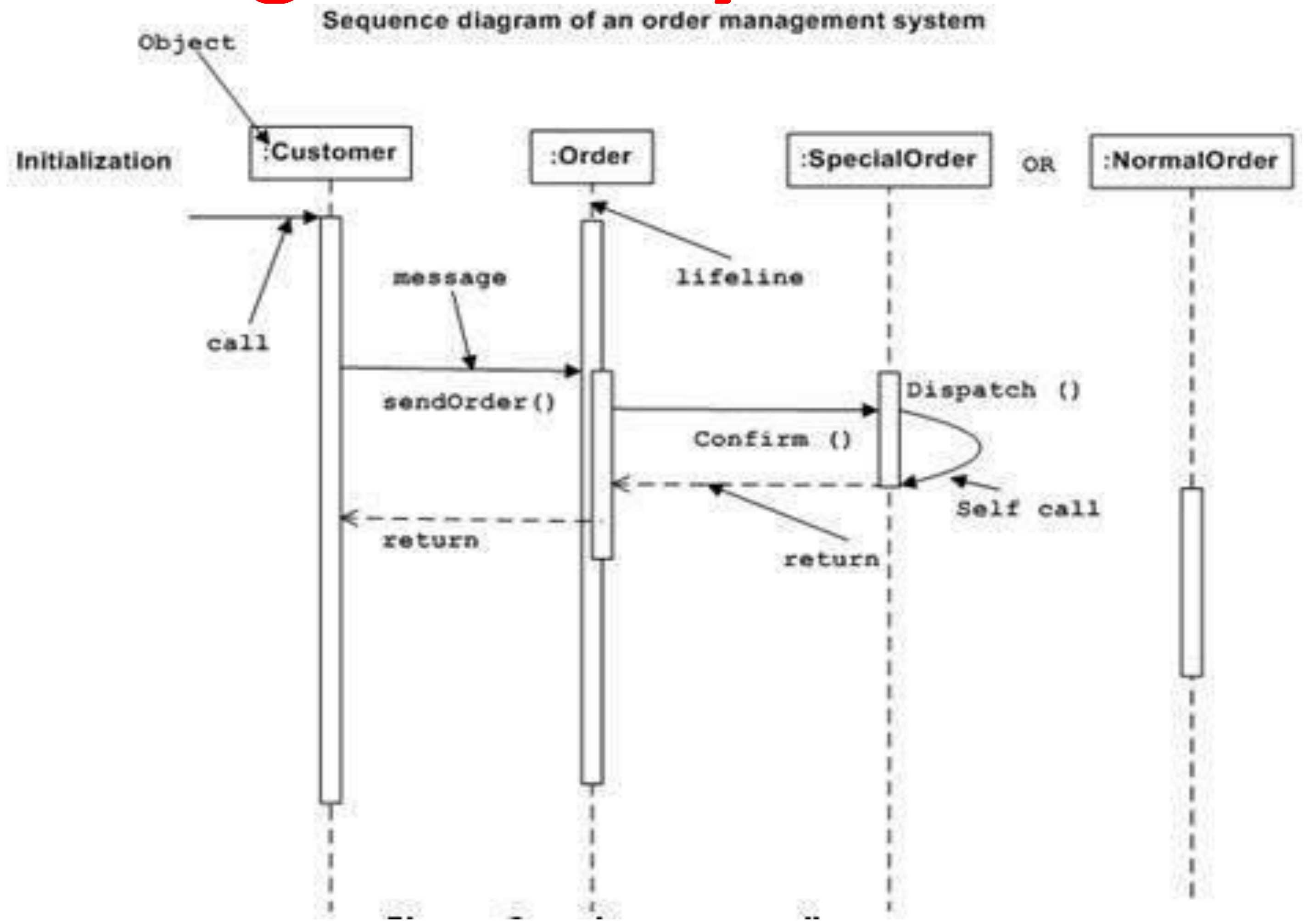


ATM System

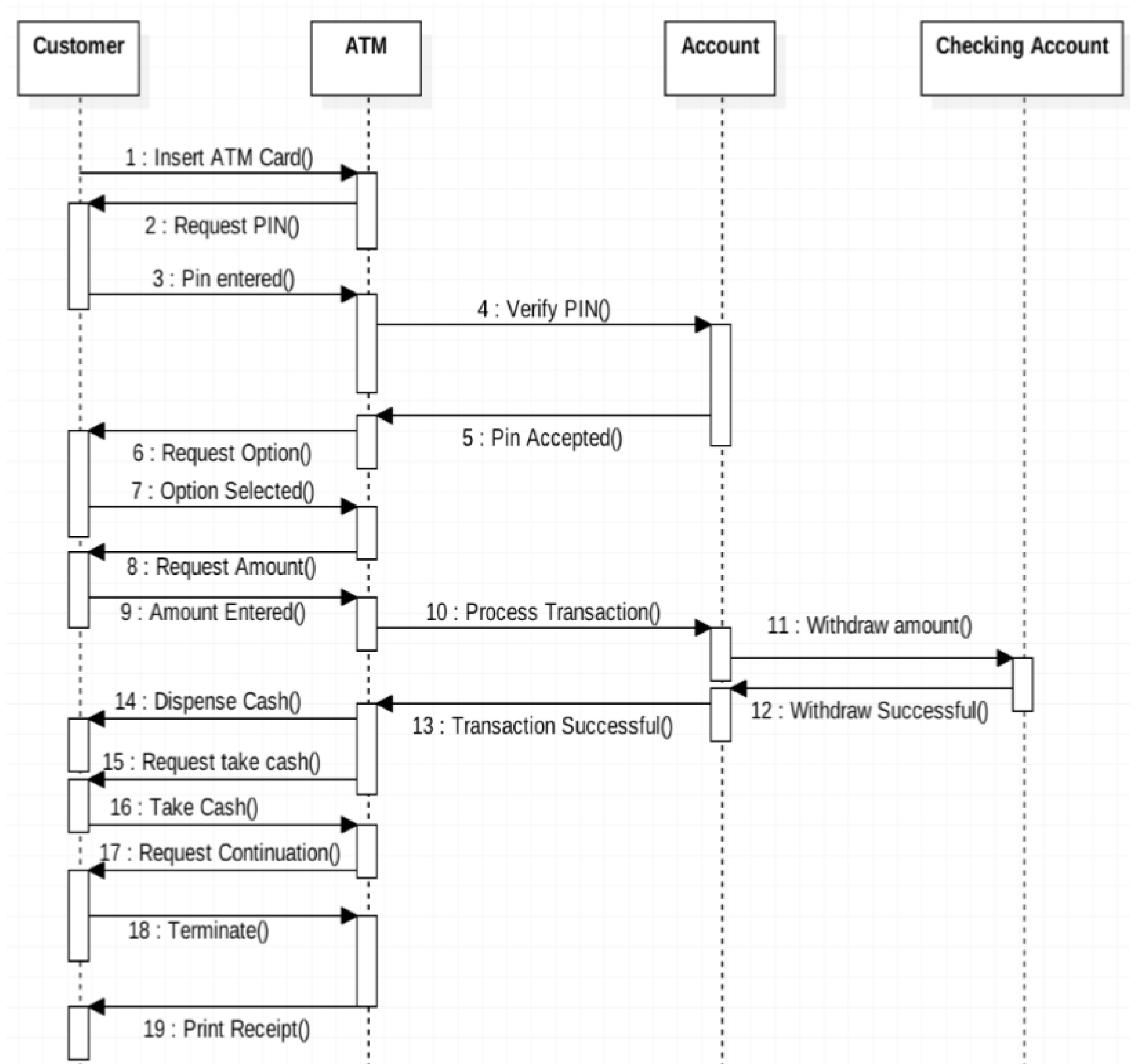
Sequence Diagram



Order Management System : Sequence Diagram



Sequence diagram for ATM system



4.3 Structural models

Structural models

- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.

Class diagrams

- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An object class can be thought of as a general definition of one kind of system object.
- An association is a link between classes that indicates that there is some relationship between these classes.
- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

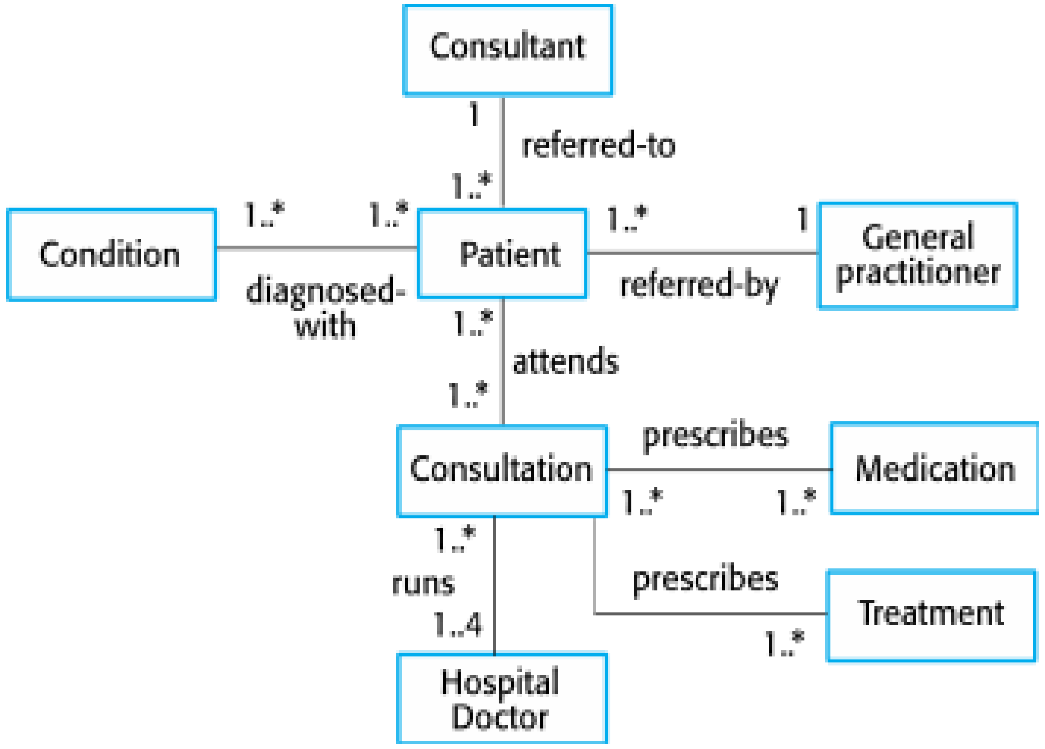
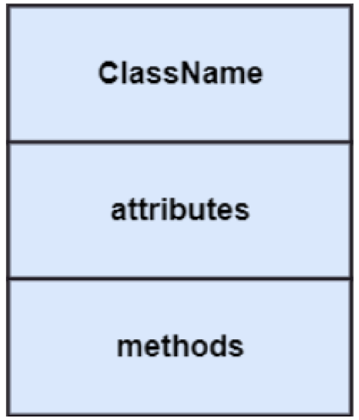


Figure: Classes and associations in the MHC-PMS

Class Diagram

- A type of Static structure diagram
- Describes the structure of a system by showing the system’s classes, their attributes, operations (or methods) and their relationships between the classes.

Class Diagram Notations:

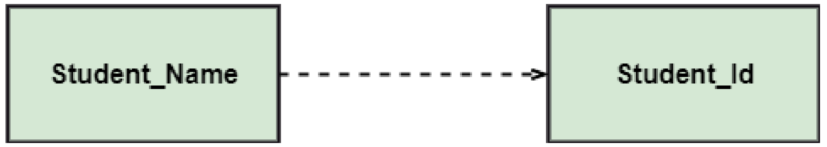


Class

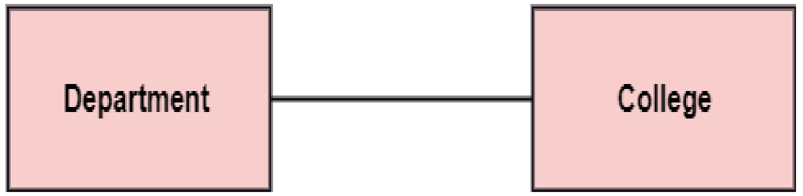
- The upper section encompasses class name
- Middle section constitutes the attributes.
- Lower section contains methods or operations.

Relationships:

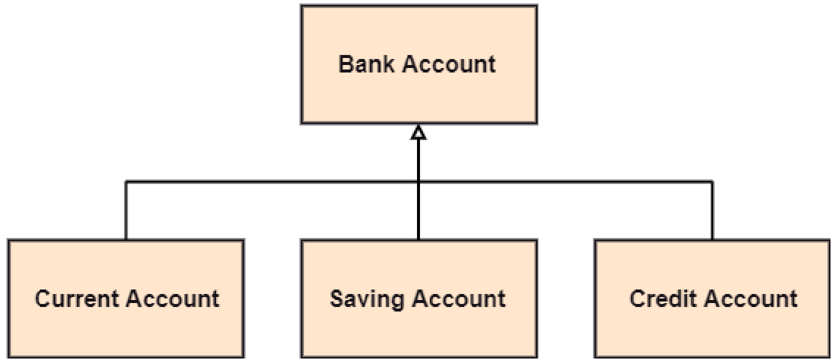
Dependency:



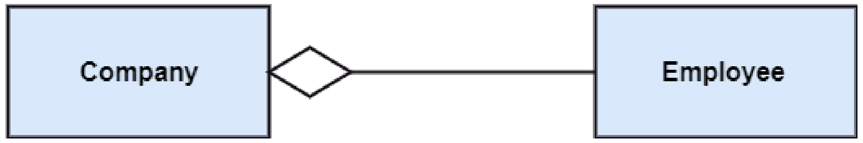
Association:



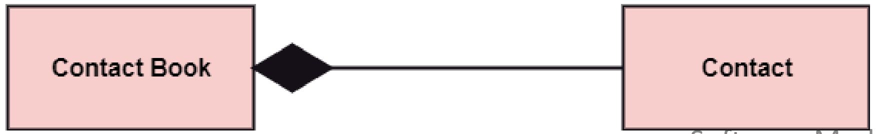
Generalization:



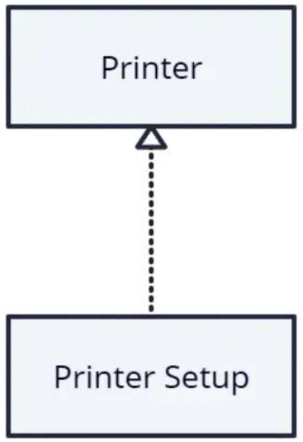
Aggregation:



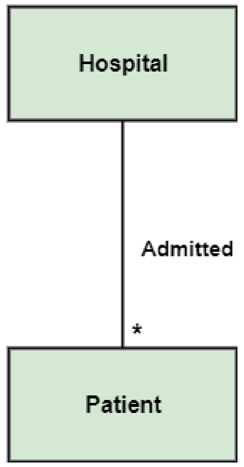
Composition:



Realization:



Multiplicity:



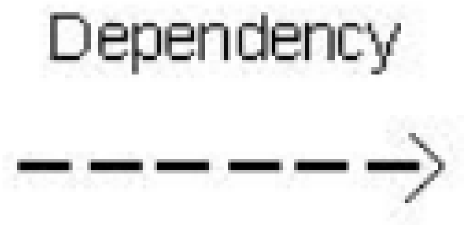
Relationships:

- **Dependency :** Eg: Student_Name is dependent on Student_Id
- **Association :** describes connection between two or more objects
- **Generalization :** Relationship between parent class and child class
- **Aggregation :** subset of association. Eg. Company encompasses a number of employees
- **Composition:** It represents a whole-part relationship.
- **Realization:** denotes the implementation of the functionality defined in one class by another class
- **Multiplicity:** Multiple patients are admitted to one hospital

Reference: https://www.javatpoint.com/uml-class-diagram

i) Dependency

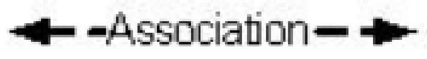
- Dependency is a semantic relationship where a change in one thing (the independent thing) causes a change in the semantics of the other thing (the dependent thing).
- The notation used for dependency is a dashed line with an arrowhead.



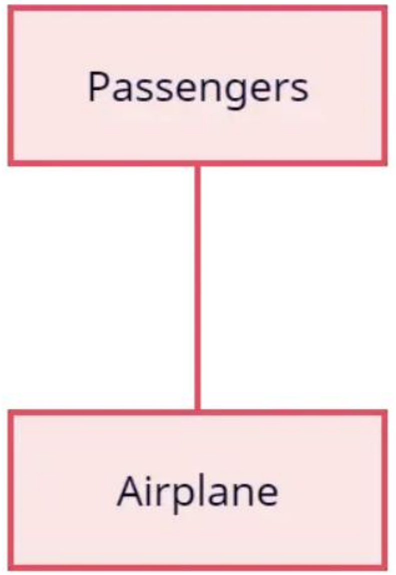
- For example, When the date of birth (the independent thing) is changed, the age (the dependent thing) changes with it.

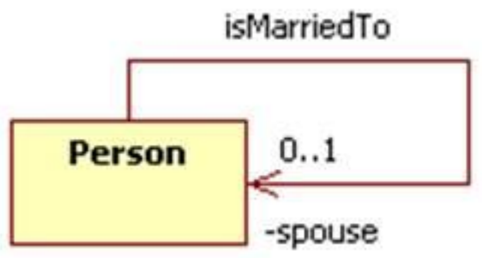
ii) Association

- Association is a structural relationship that describes the connection between two things.
- It also describes how many objects are taking part in that relationship.
- The notation used for association is a simple line segment or a line segment with arrowhead.



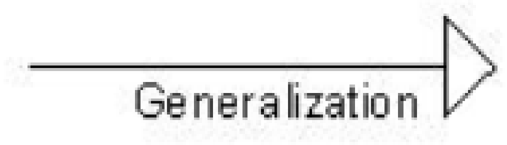
- For example, passenger and airline are associated.
- Example of Self association

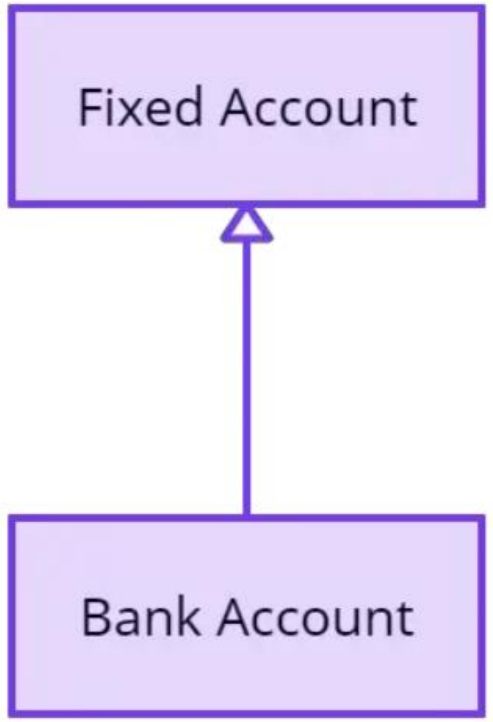




iii) Generalization

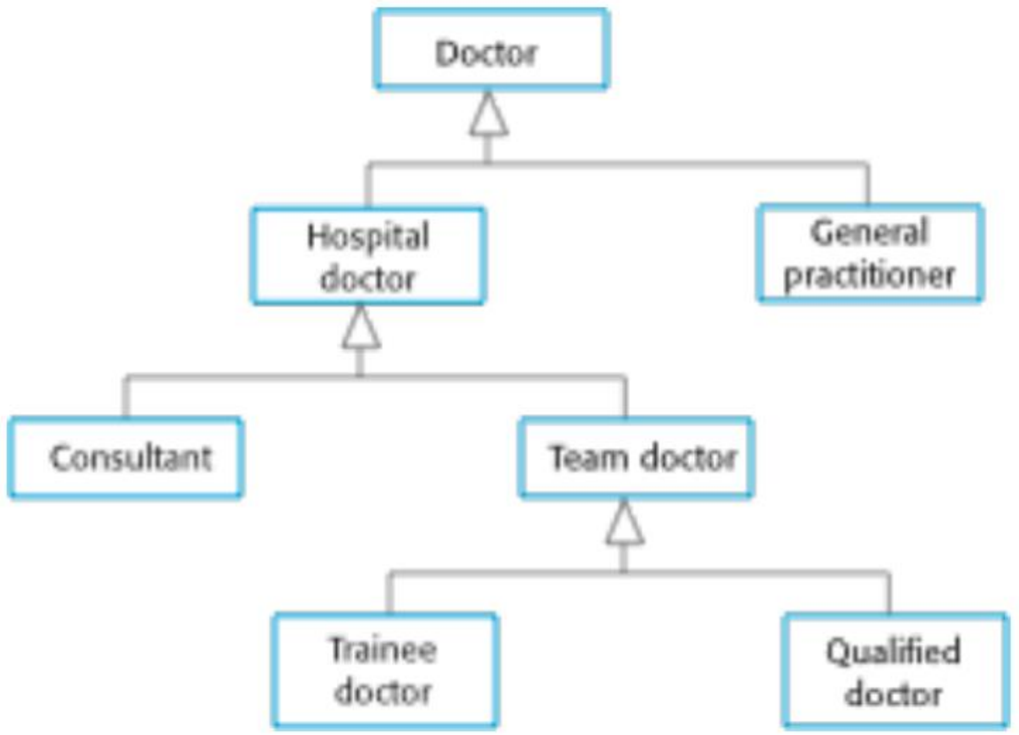
- Generalization is a relationship between a general thing (a parent or superclass) and a more specific kind of that thing (a child or subclass).
- The notation used for generalization is a line segment with an empty block triangular arrowhead.
- The arrowhead points toward the generalized class or use case or package.

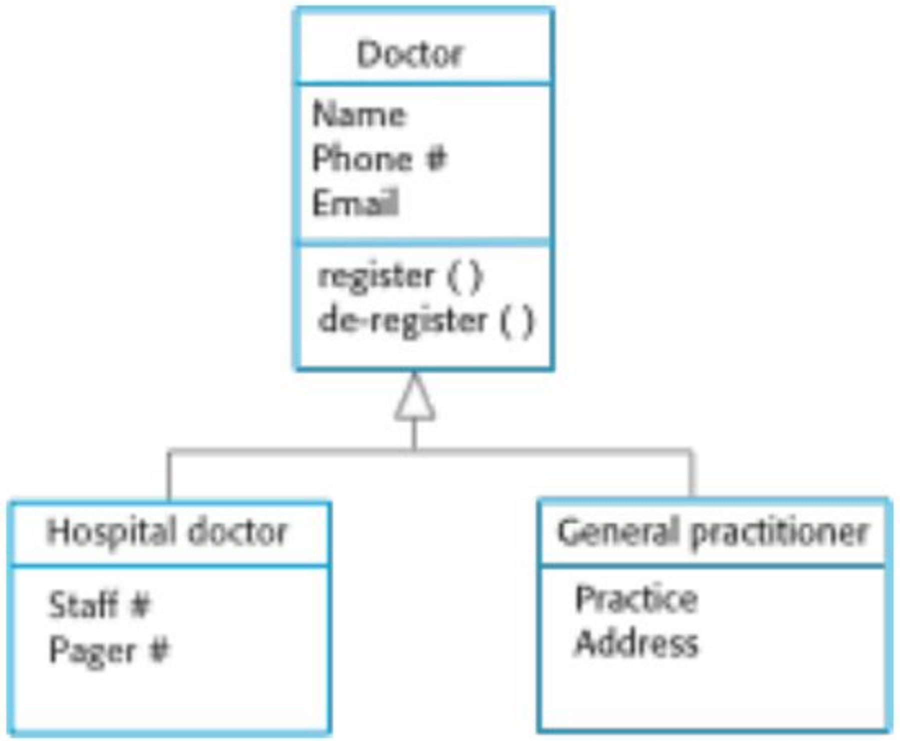




- For example, an animal (generalization) and a cat (specialization) are related by the generalization-specialization relationship.

A generalization hierarchy

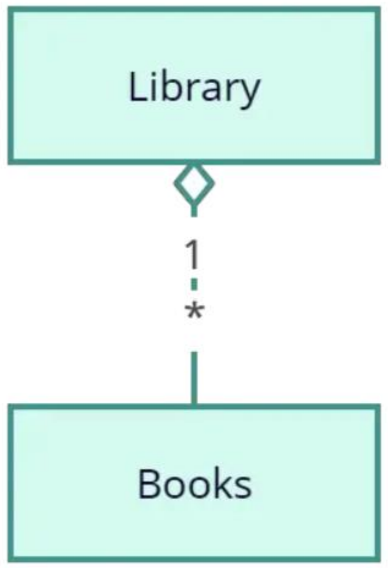


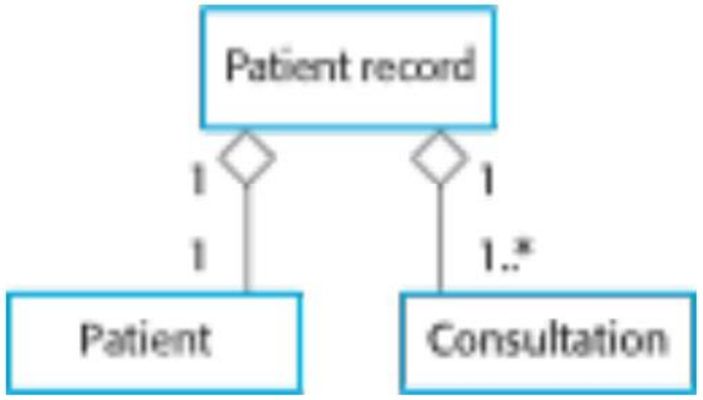


A generalization hierarchy with added detail

iv) Aggregation

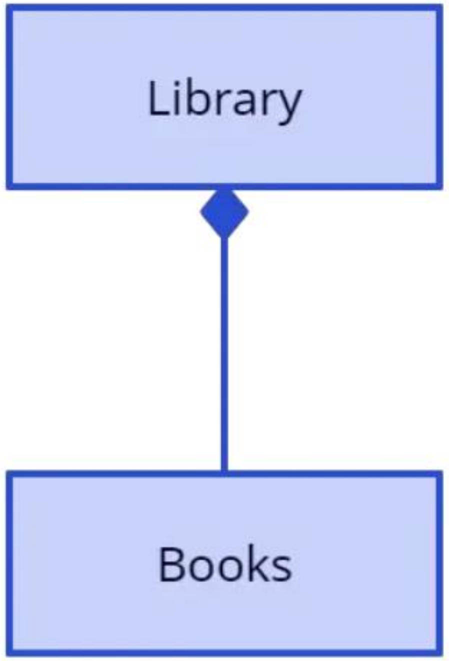
- refers to the formation of a particular class as a result of one class being aggregated or built as a collection.
- To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.
- For example, the class “library” is made up of one or more books, among other materials.





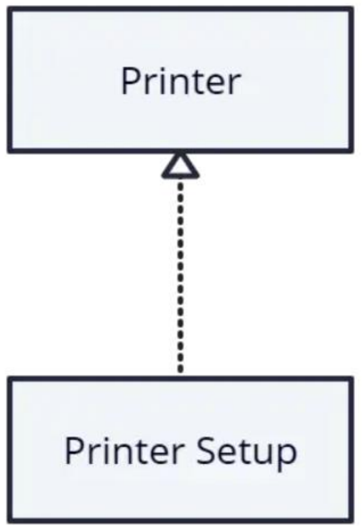
v) Composition

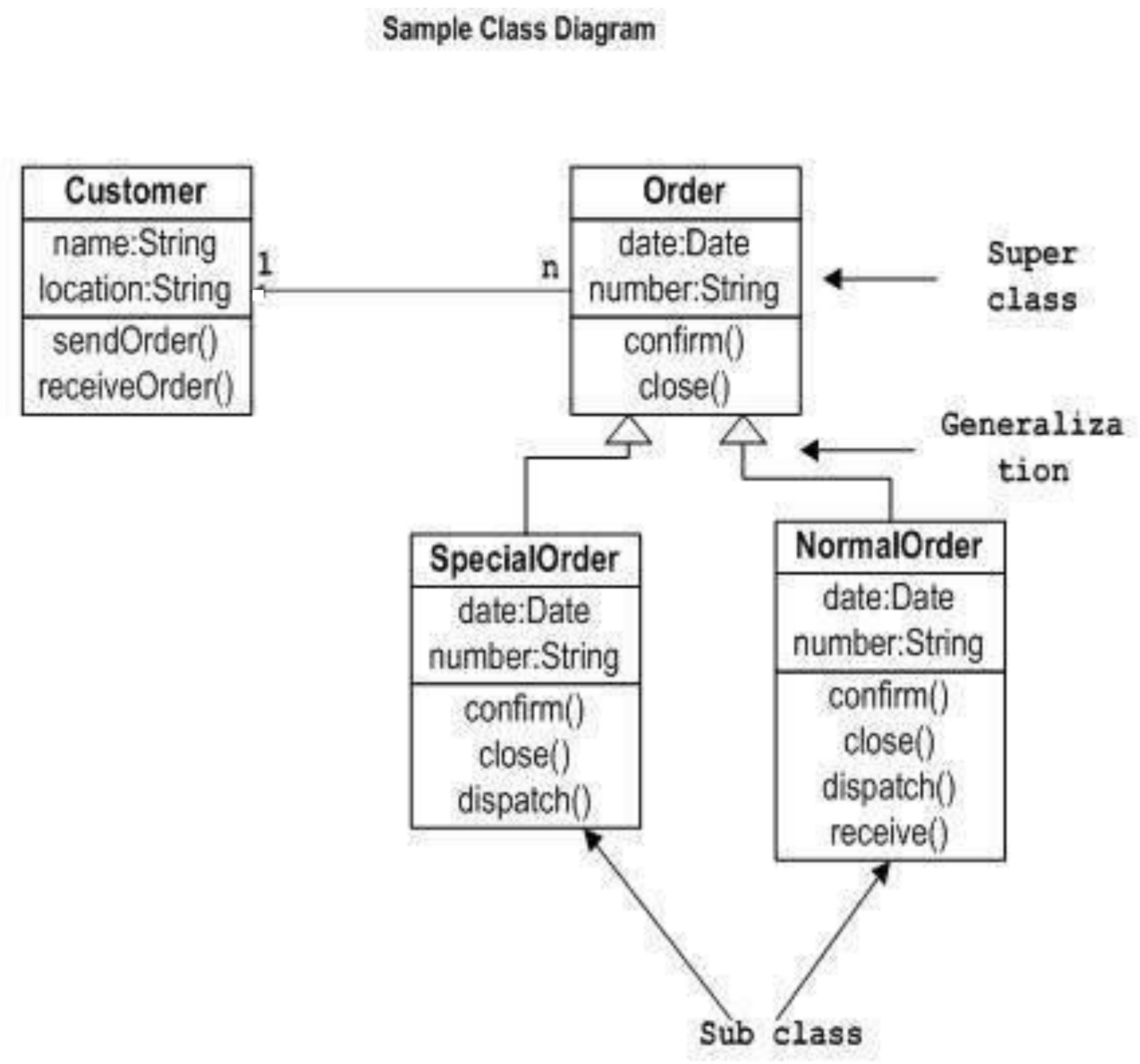
- The composition relationship is very similar to the aggregation relationship.
- the only difference being its key purpose of emphasizing the dependence of the contained class to the life cycle of the container class.
- That is, the contained class will be obliterated (cancelled) when the container class is destroyed.
- For example, a shoulder bag's side pocket will also cease to exist once the shoulder bag is destroyed.



vi) Realization

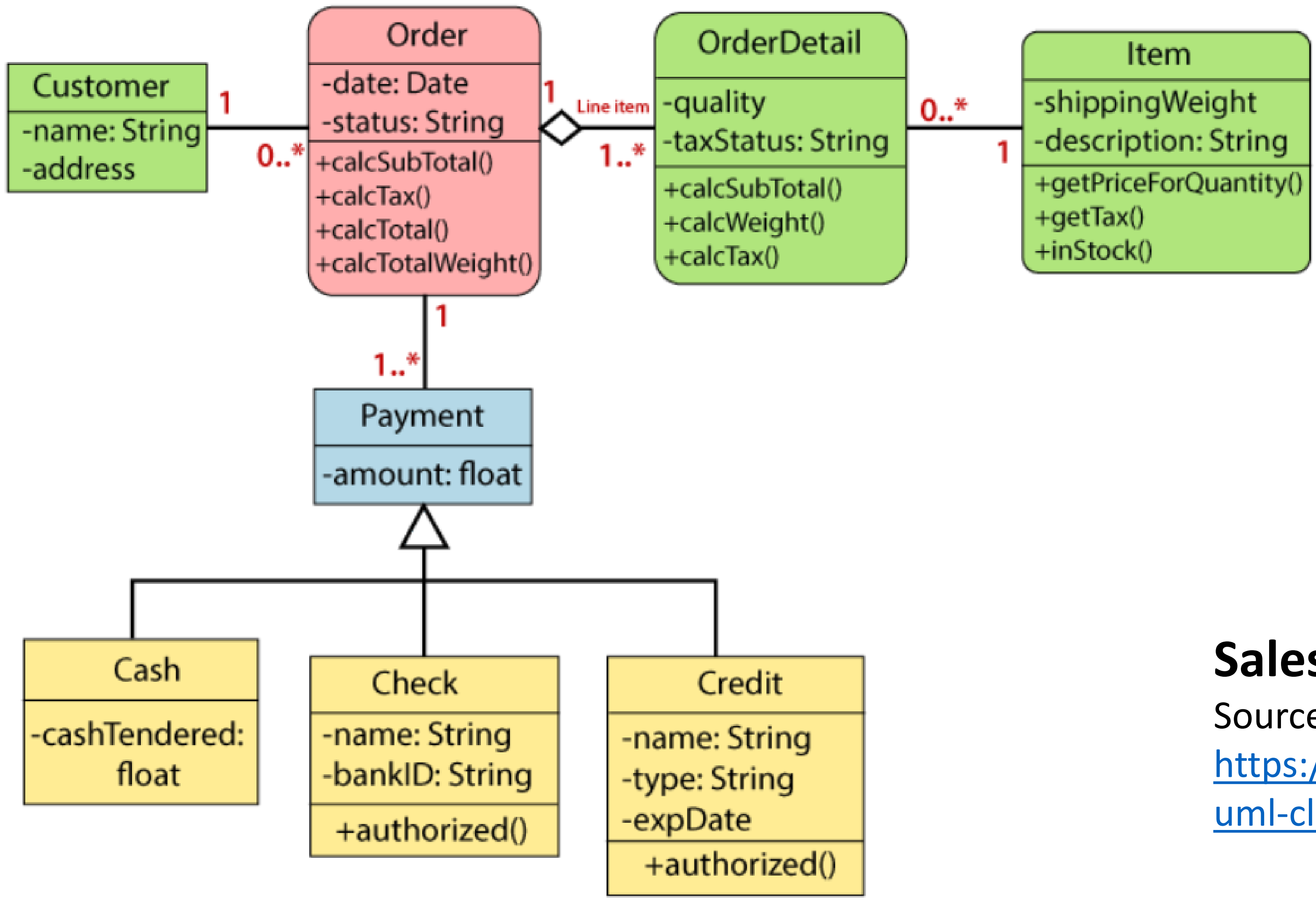
- denotes the implementation of the functionality defined in one class by another class.
- To show the relationship in UML, a broken line with an unfilled solid arrowhead is drawn from the class that defines the functionality of the class that implements the function.
- In the example, the printing preferences that are set using the printer setup interface are being implemented by the printer.





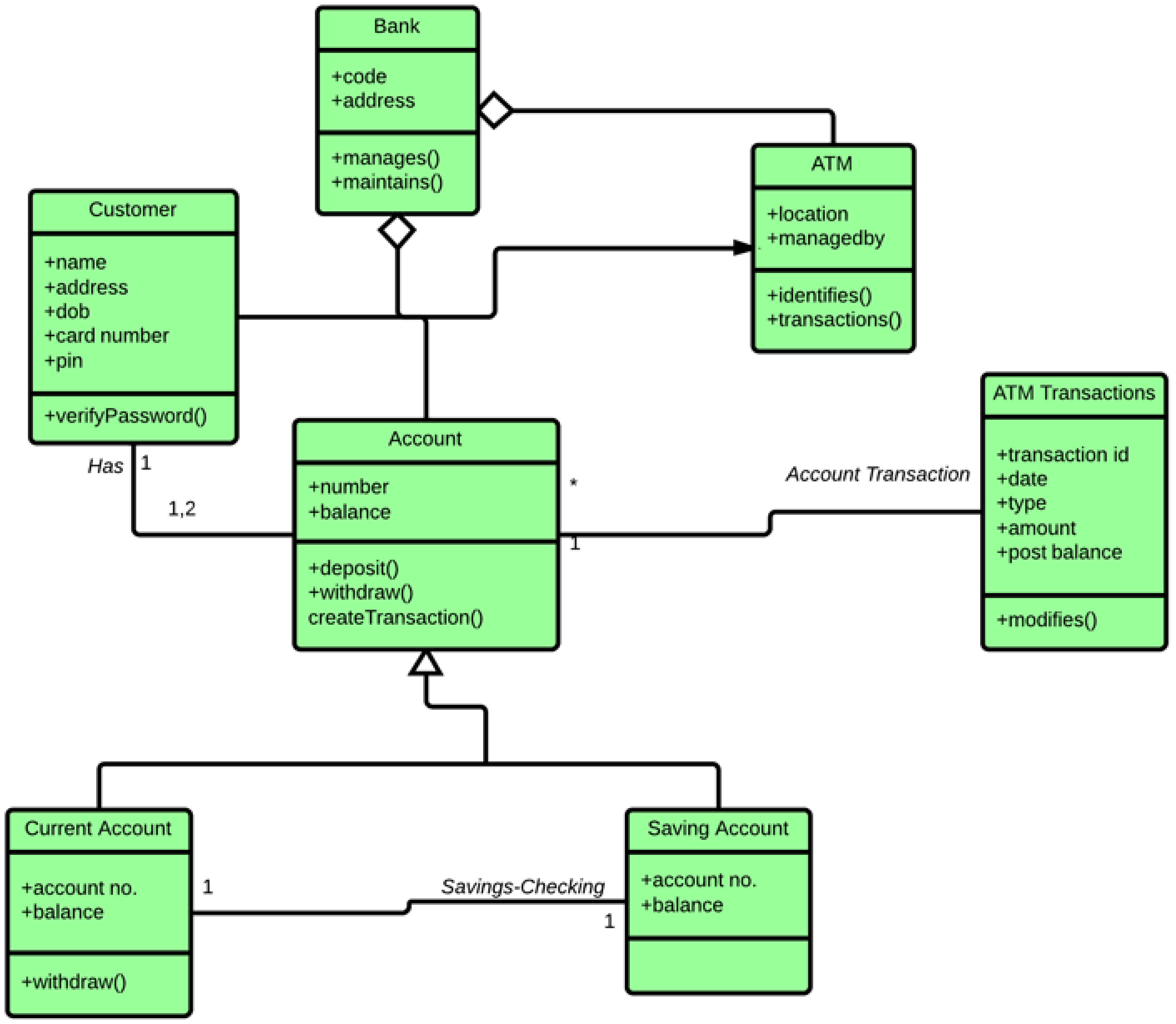
Order Management System

Source: https://www.tutorialspoint.com/uml/uml_class_diagram.htm



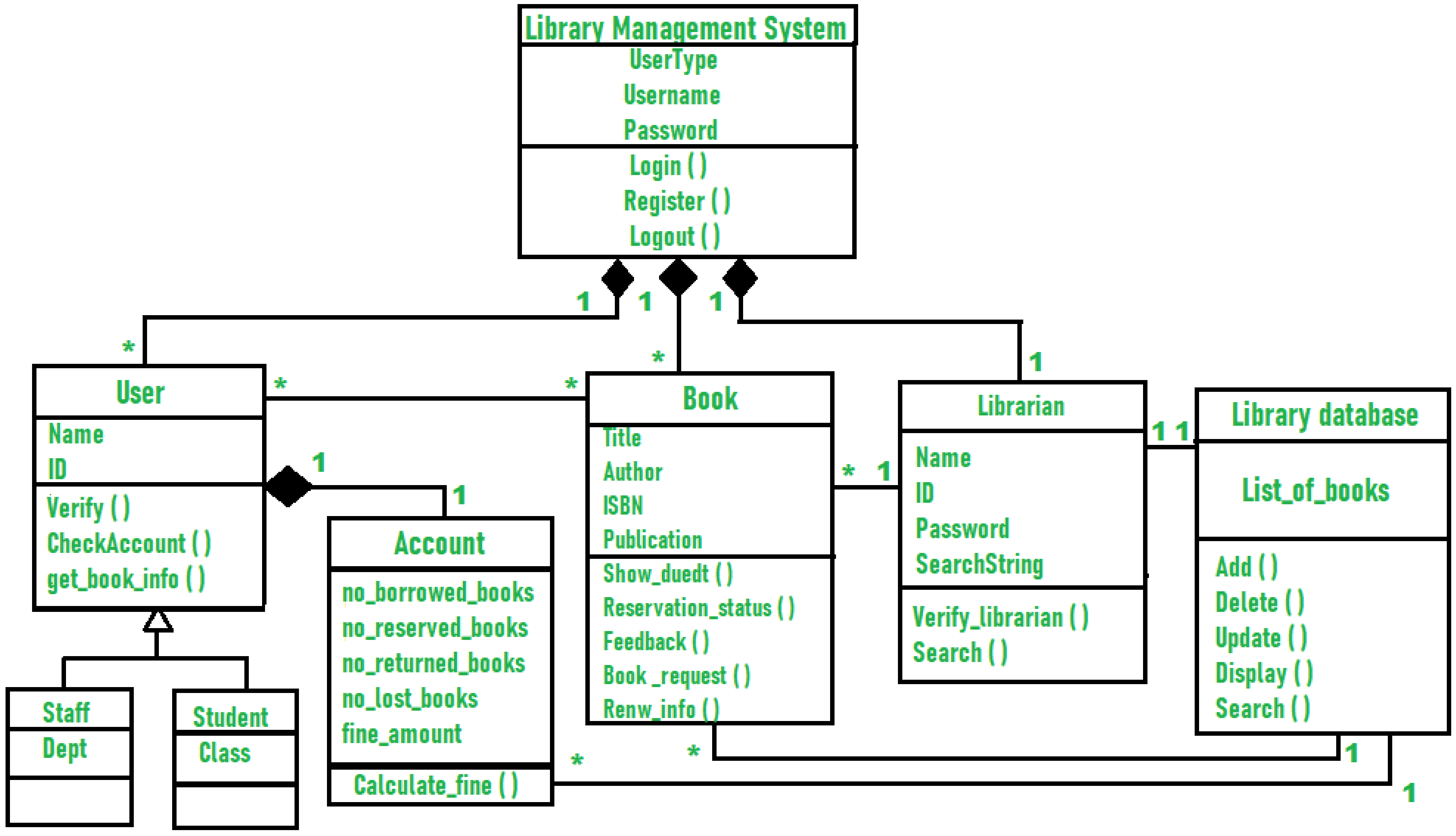
Sales Order System

Source:
https://www.javatpoint.com/uml-class-diagram



ATM system

Source: https://www.guru99.com/uml-class-diagram.html



CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

Source:https://www.geeksforgeeks.org/class-diagram-for-library-management-system/

4.4 Behavioral models

Behavioral models

- Behavioral model describe the overall behavior of the system.
- Behavioral models are models of the dynamic behavior of a system as it is executing.
- They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - Data Some data arrives that has to be processed by the system.
  - Events Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

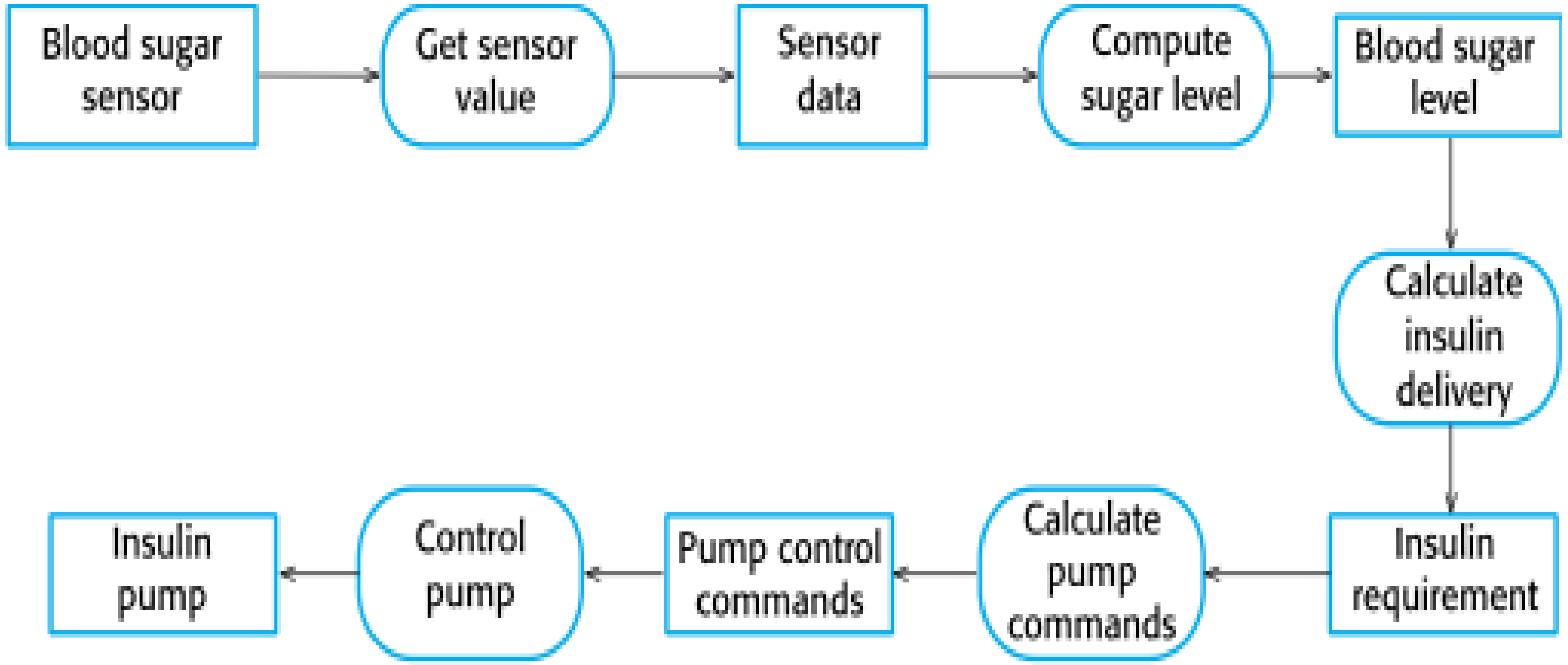
Data-driven modeling

- Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.
- Data-driven models show the sequence of actions involved in processing input data and generating an associated output.
- They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.

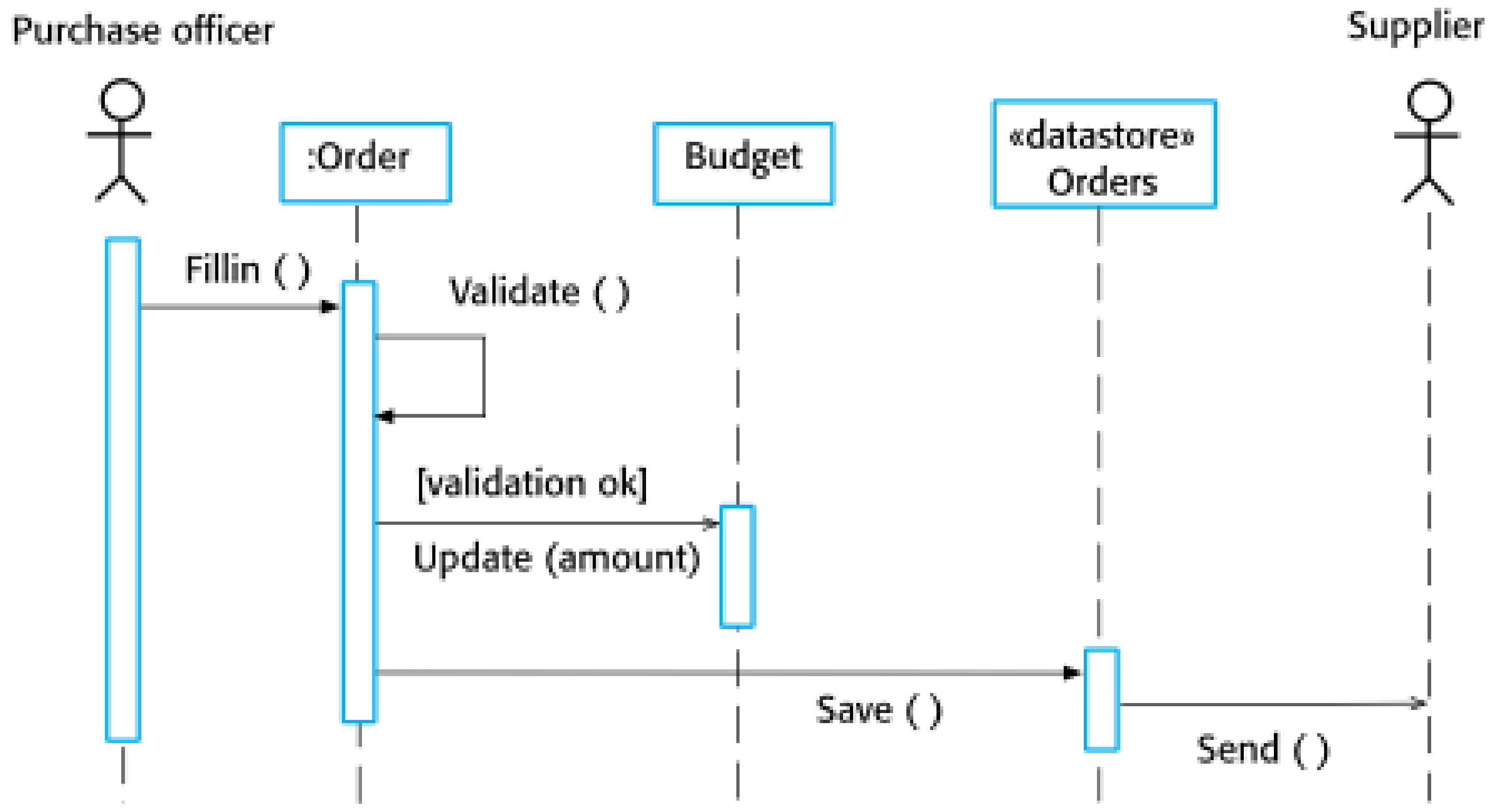
Data-driven modeling

- Data-driven models show the sequence of actions involved in processing input data and generating the associated output.
- This is very useful during the analysis stage since they show end-to-end processing in a system which means that they show the entire action sequence of how input data become output data.
- In other words, it shows the response of the system to particular input.
- In UML, activity and sequence diagrams can be used to describe such data flows.

An activity model of the insulin pump's operation



Order processing

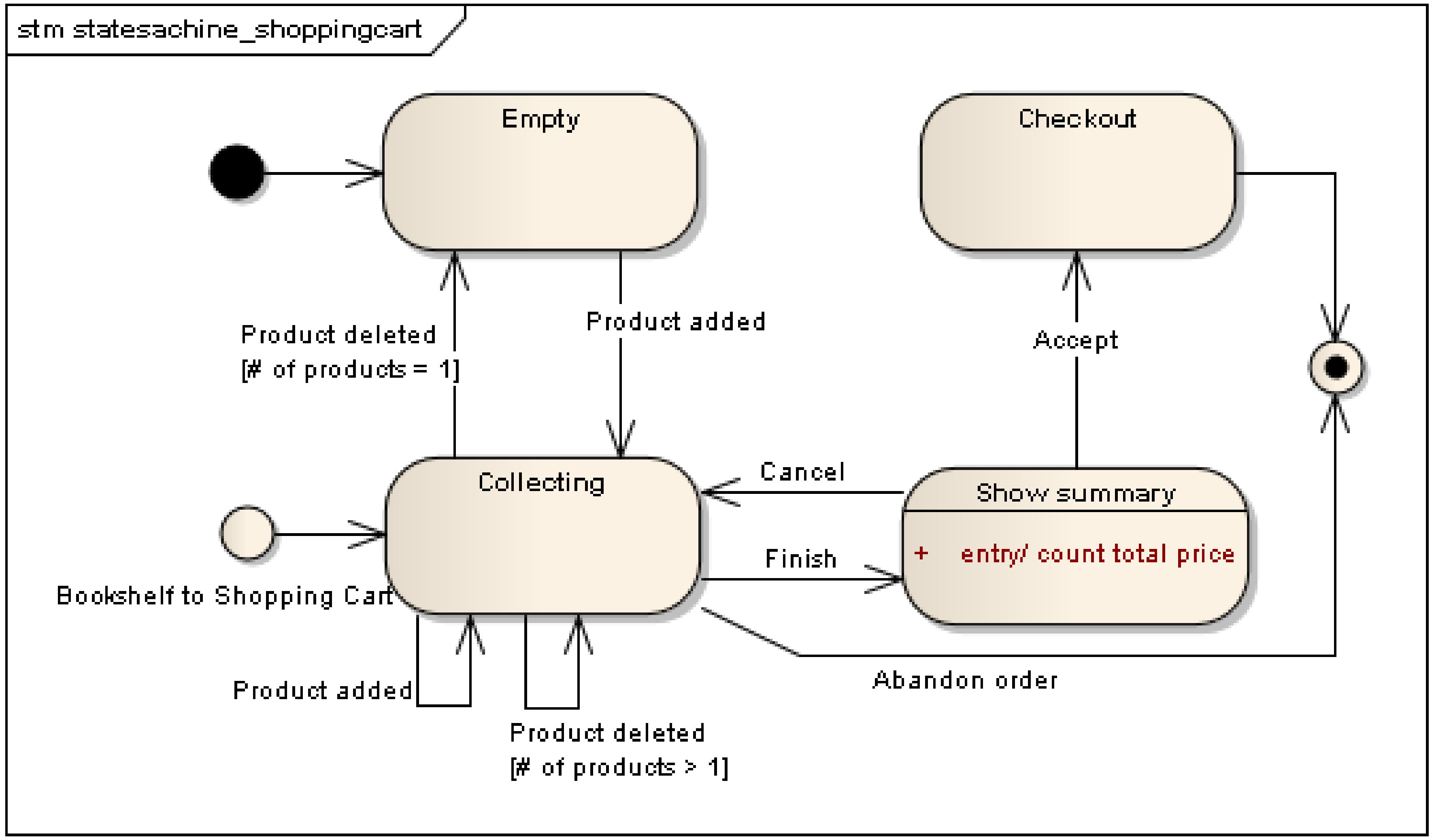


Event-driven modeling

- Event-driven modeling shows how a system responds to external and internal events (stimuli).
- It is based on the assumption that a system has a finite number of states and that an event (stimulus) may cause a transition from one state to another.
- The UML supports event-based modeling using ***state machine diagrams***

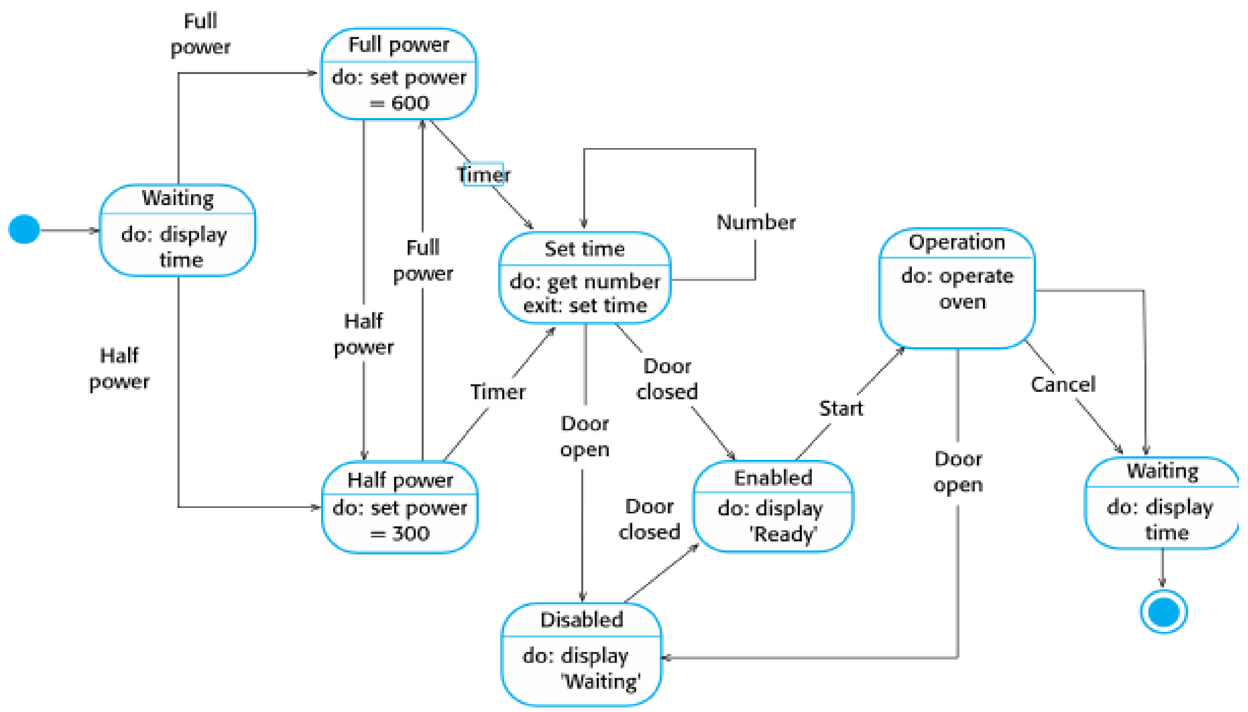
State machine diagrams

- A state machine diagram models the behaviour of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events. It contains the following elements:
- **State.** A state is denoted by a round-cornered rectangle with the name of the state written inside it. There are two special states:
  - **Initial state.** The initial state is denoted by a filled black circle and may be labeled with a name.
  - **Final state.** The final state is denoted by a circle with a dot inside and may also be labeled with a name.
- **Transitions.** Transitions from one state to the next are denoted by lines with arrowheads.
- **State action.** State actions describe effects associated with a state. A state action is an activity label/behavior expression pair. The activity label identifies the circumstances under which the behavior will be invoked. There are three reserved activity labels:
  - entry: the behavior is performed upon entry to the state,
  - do: ongoing behavior, performed as long as the element is in the state,
  - exit: a behavior that is performed upon exit from the state.


stm statesachine_shoppingcart
Empty
Checkout
Product deleted
[# of products = 1]
Product added
Accept
Collecting
Cancel
Show summary
+ entry/ count total price
Finish
Bookshelf to Shopping Cart
Product added
Product deleted
[# of products > 1]
Abandon order

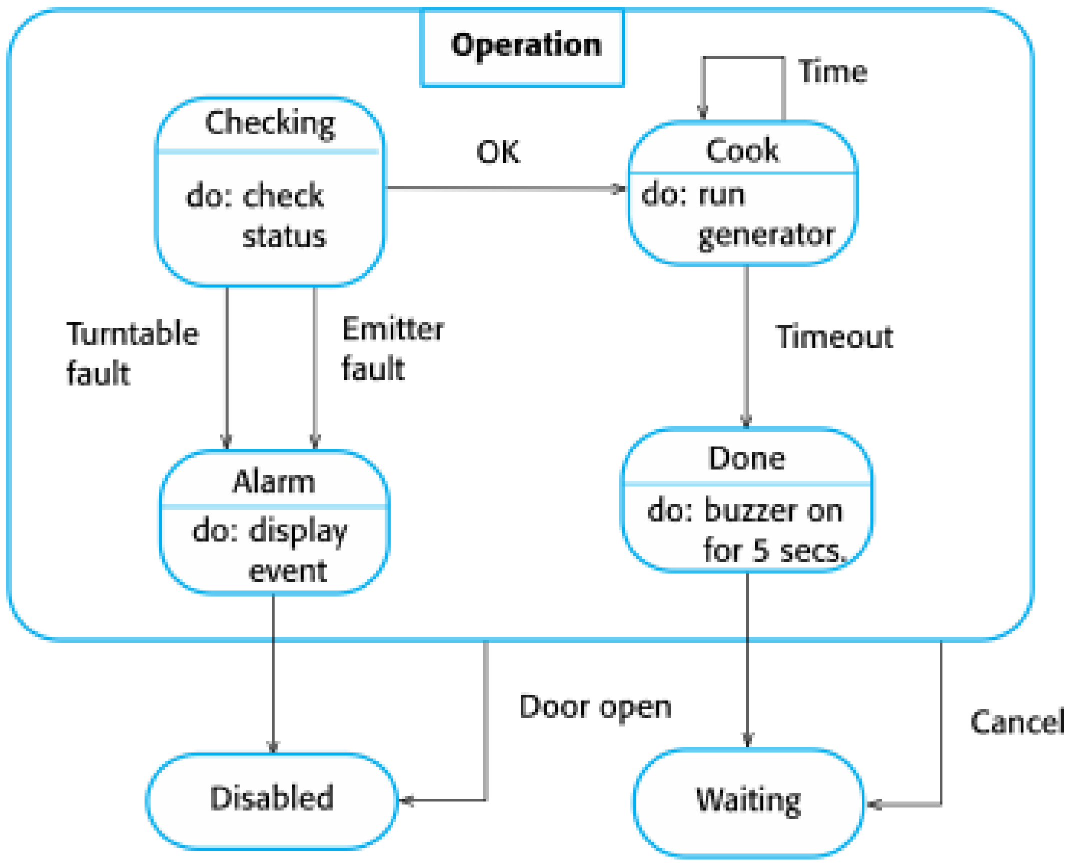
State machine models

- These model the behavior of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- State charts are an integral part of the UML and are used to represent state machine models.



State diagram of a microwave oven

Microwave oven operation



States and stimuli for the microwave oven (a)

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ‘Half power’.
Full power	The oven power is set to 600 watts. The display shows ‘Full power’.
Set time	The cooking time is set to the user’s input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ‘Not ready’.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ‘Ready to cook’.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows ‘Cooking complete’ while buzzer is sounding.

States and stimuli for the microwave oven (b)

Stimulus	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

4.5 Package Structure

To be continued…

Keypoints

- A model is an abstract view of a system that ignores system details. Complementary system models can be developed to show the system’s context, interactions, structure and behavior.
- Context models show how a system that is being modeled is positioned in an environment with other systems and processes.
- Use case diagrams and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; sequence diagrams add more information to these by showing interactions between system objects.
- Structural models show the organization and architecture of a system. Class diagrams are used to define the static structure of classes in a system and their associations.
- Behavioral models are used to describe the dynamic behavior of an executing system. This behavior can be modeled from the perspective of the data processed by the system, or by the events that stimulate responses from a system.
- Activity diagrams may be used to model the processing of data, where each activity represents one process step.
- State diagrams are used to model a system’s behavior in response to internal or external events.