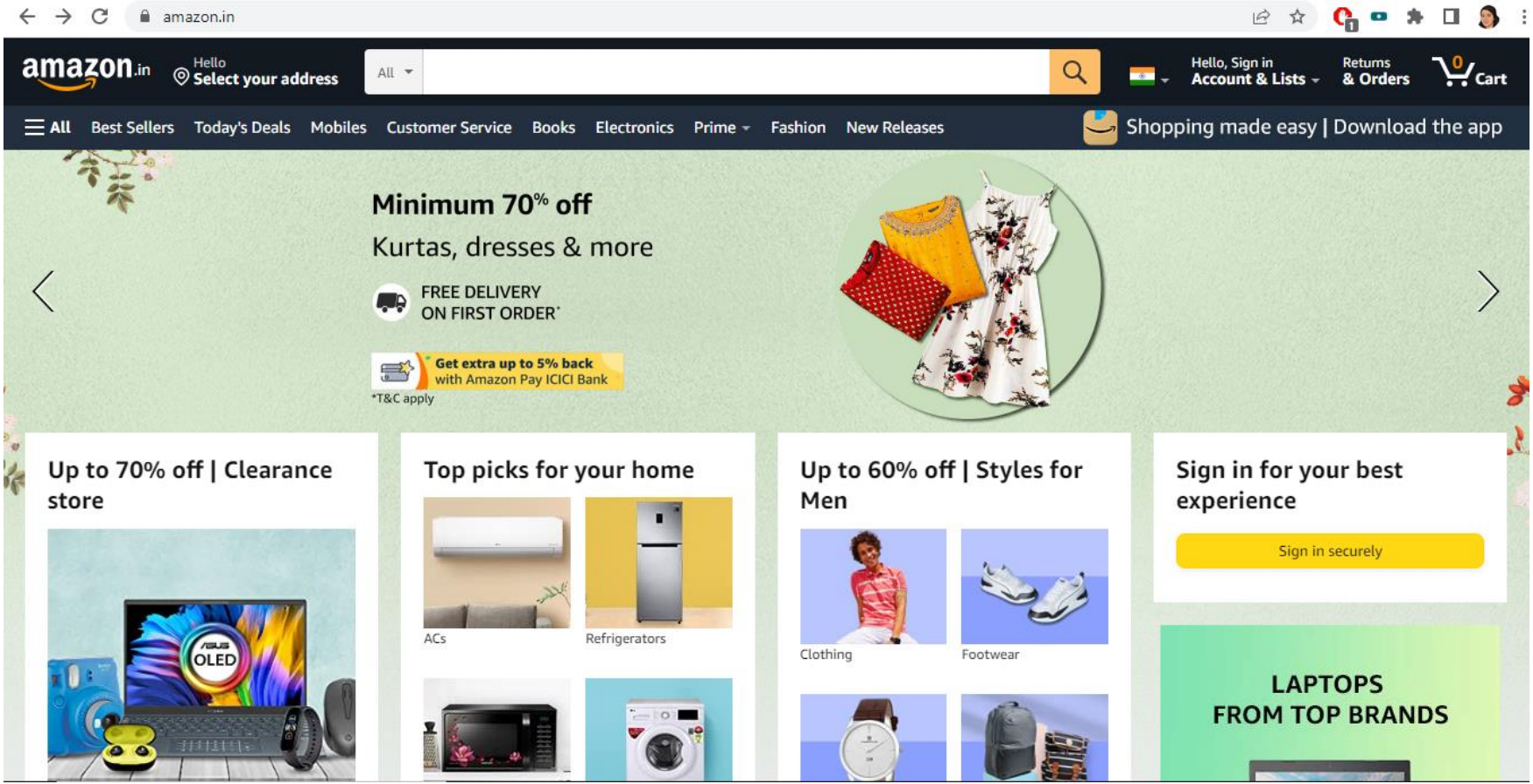
System Study, Feasibility Study, Normalization, UML Diagrams, Data Sanitization & Indexing

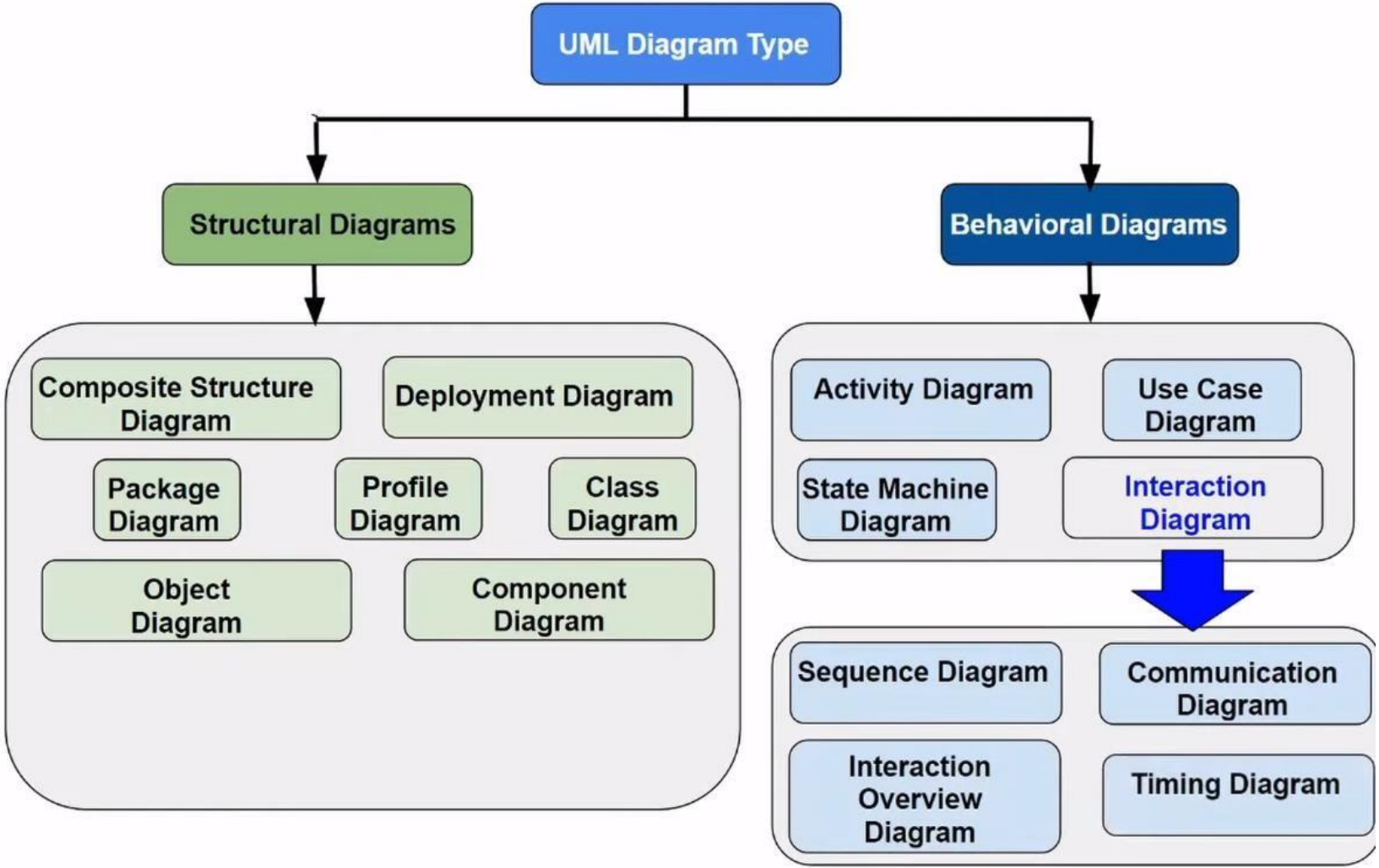
System Study

Natural System

➢ Designed System



UNIFIED
MODELING
LANGUAGE™


UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram

UML Standard Diagrams

Structural Diagrams

- **Behavioural Diagrams**

Structural Diagrams

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

Vital components of a 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.

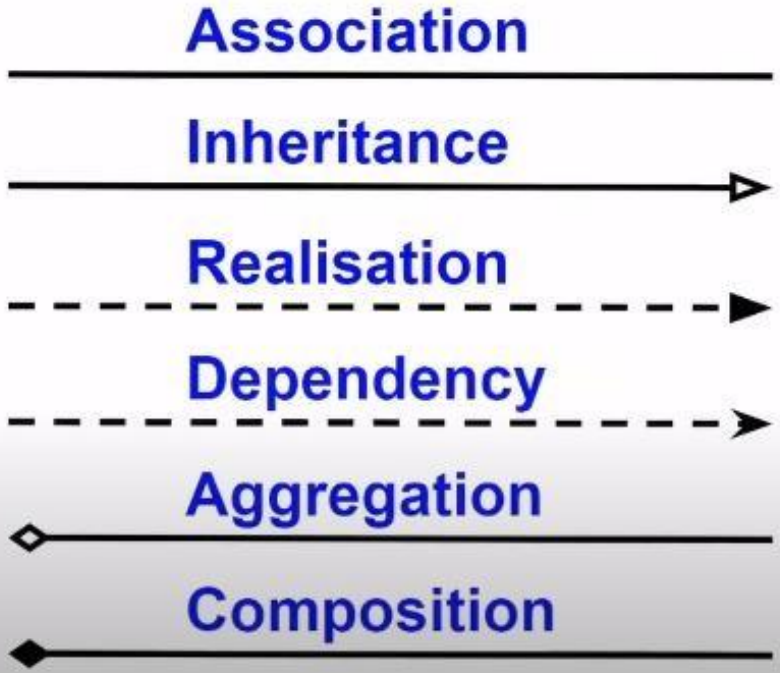
Vital components of a Class Diagram

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.

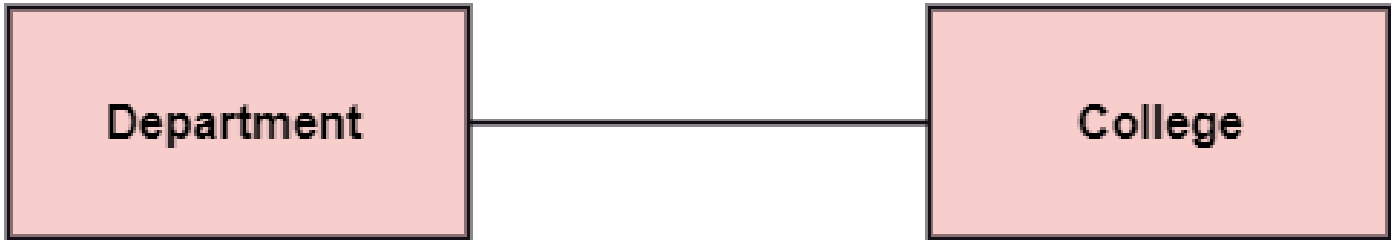
Lower Section: The lower section contain methods or operations. The methods are represented in the form of a list, where each method is written in a single line. It demonstrates how a class interacts with data.

Relationships between classes in UML

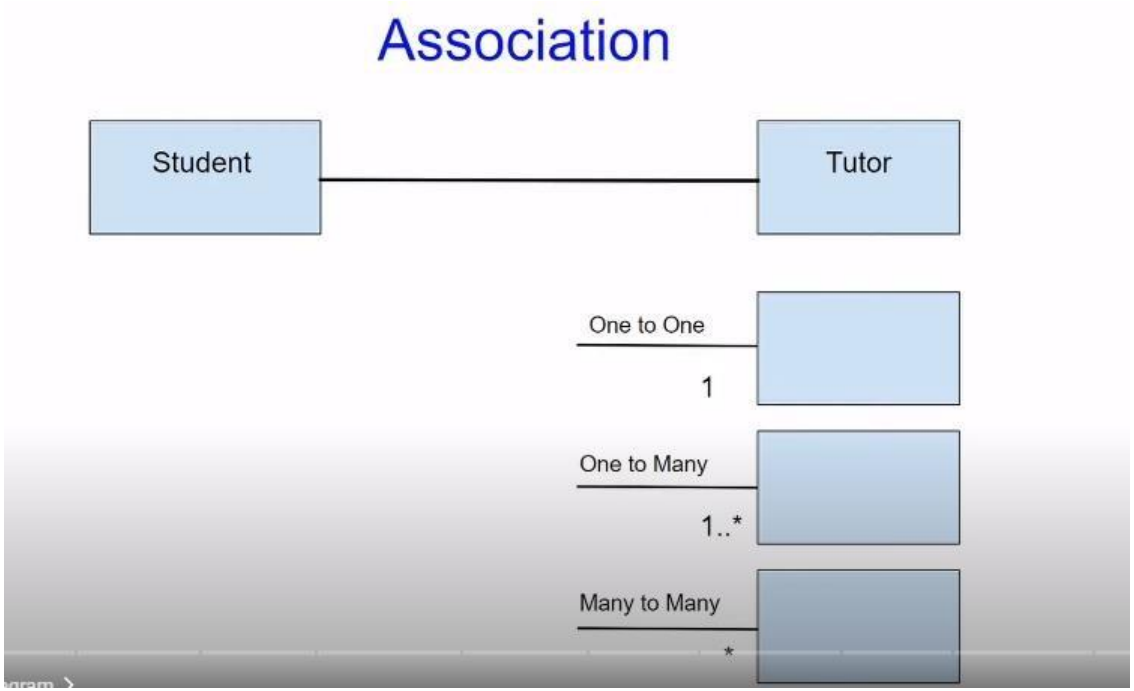


Association

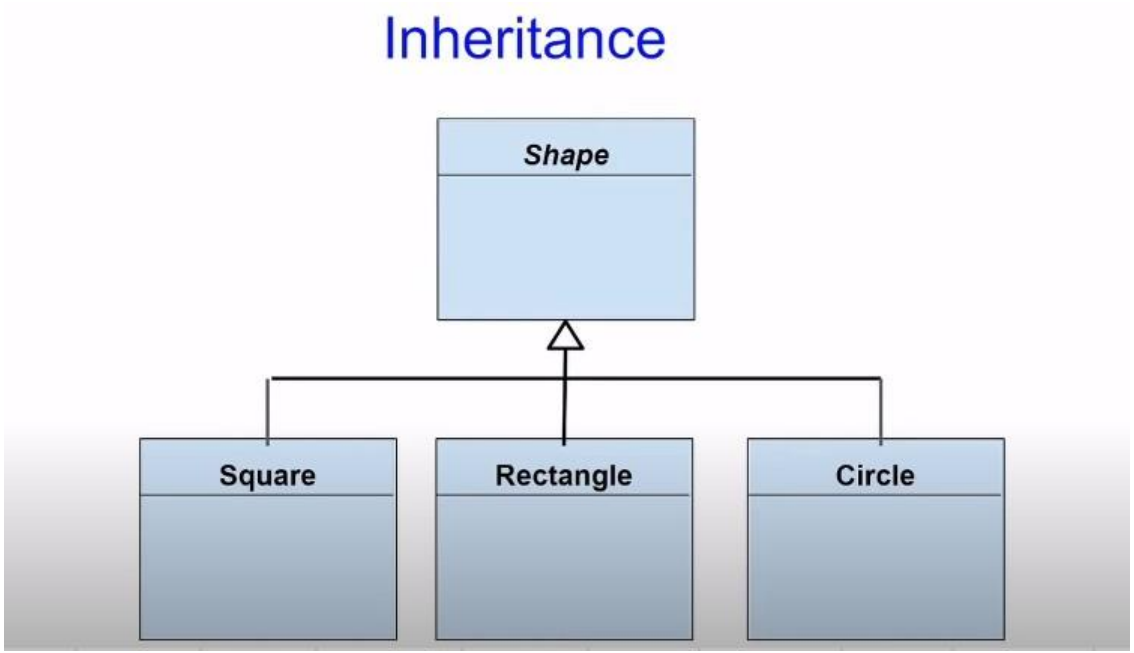
- Association: It describes a static or physical connection between two or more objects. It depicts how many objects are there in the relationship.
- For example, a department is associated with the college.



Association



Inheritance

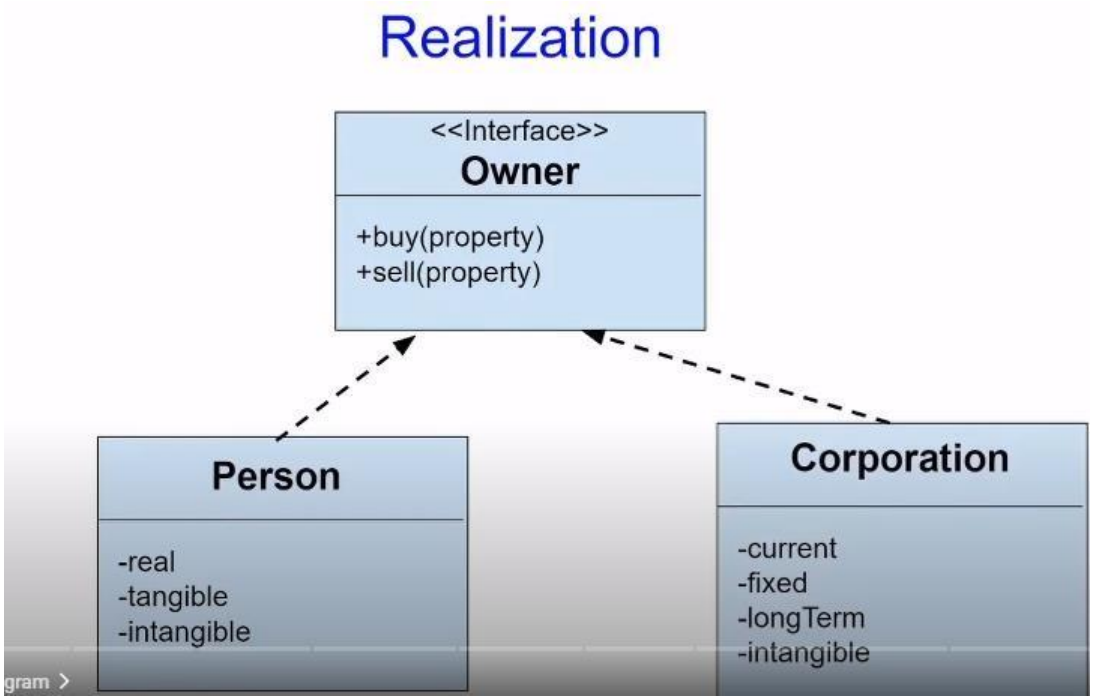


Aggregation

- 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.

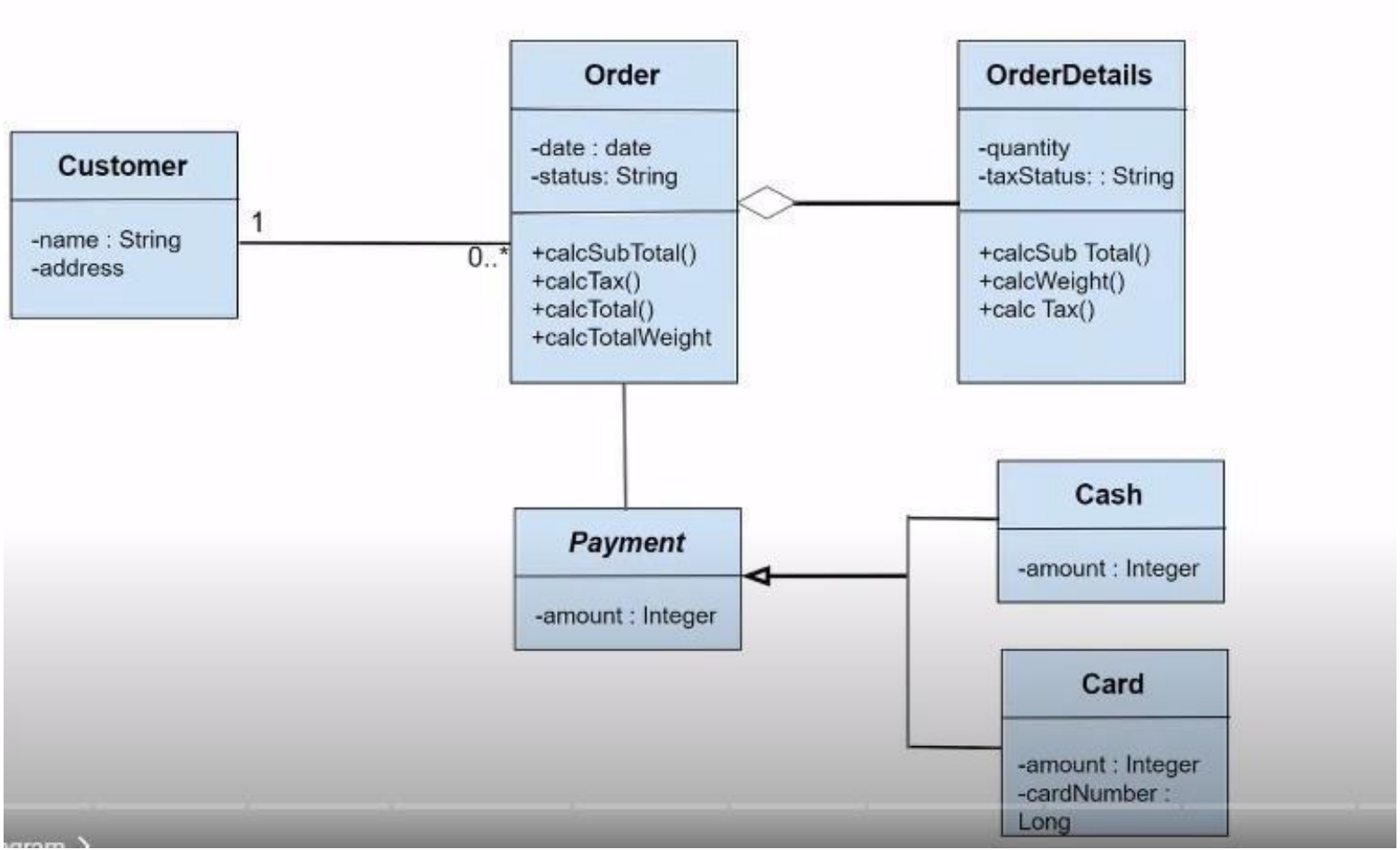
- Composition: The composition is a subset of aggregation. It portrays the dependency between the parent and its child, which means if one part is deleted, then the other part also gets discarded. It represents a whole-part relationship.

Realization

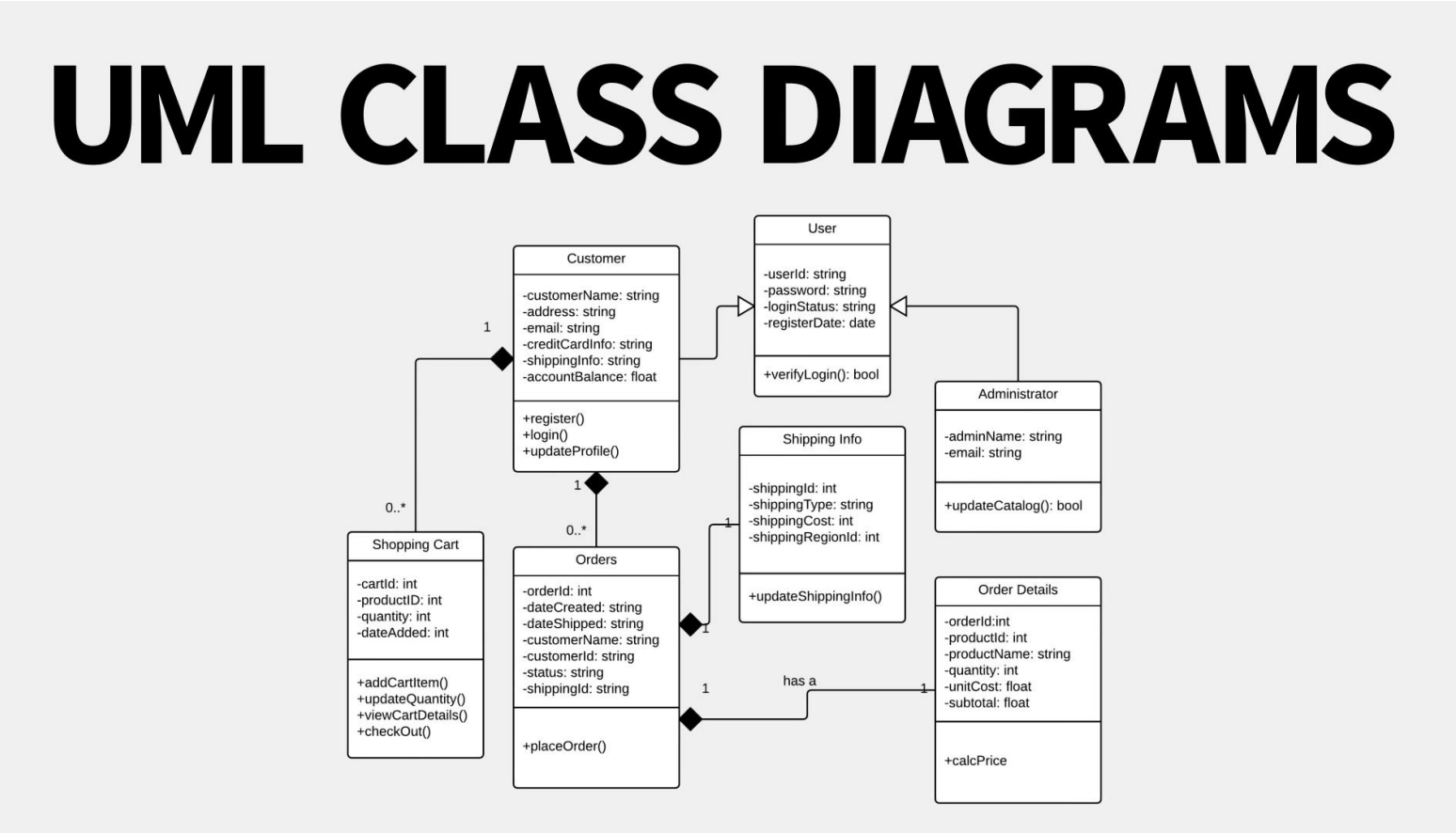


Example

- Order System of an application


Customer
-name : String
-address
1
0..*
Order
-date : date
-status: String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight
OrderDetails
-quantity
-taxStatus: : String
+calcSub Total()
+calcWeight()
+calc Tax()
Payment
-amount : Integer
Cash
-amount : Integer
Card
-amount : Integer
-cardNumber :
Long

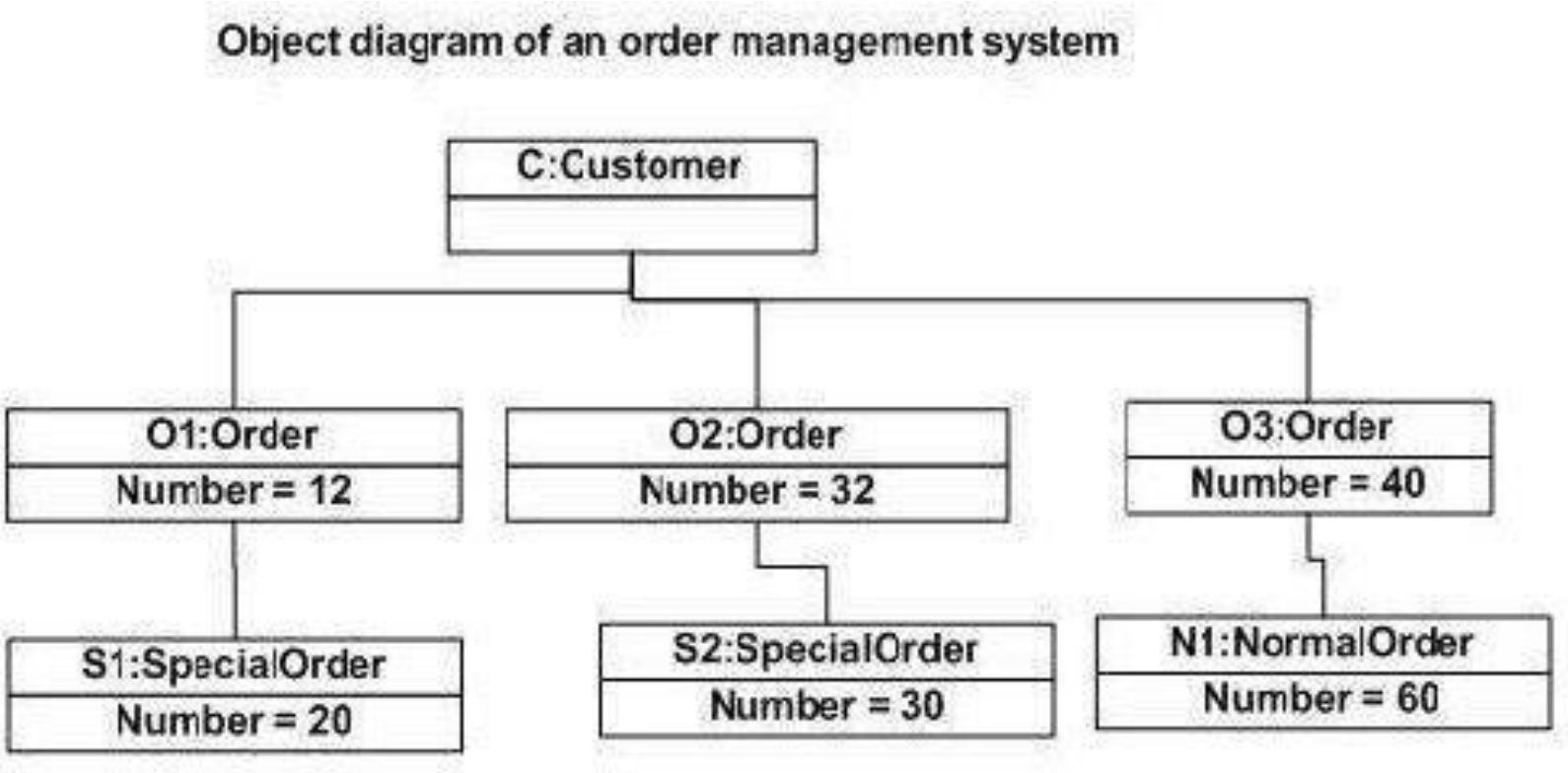
Class diagram - Static view of an application



Usage of Class diagrams

- To describe the static view of a system.
- To show the collaboration among every instance in the static view.
- To describe the functionalities performed by the system.
- To construct the software application using object-oriented languages.

Object Diagrams - instance of a class diagram



Component Diagram

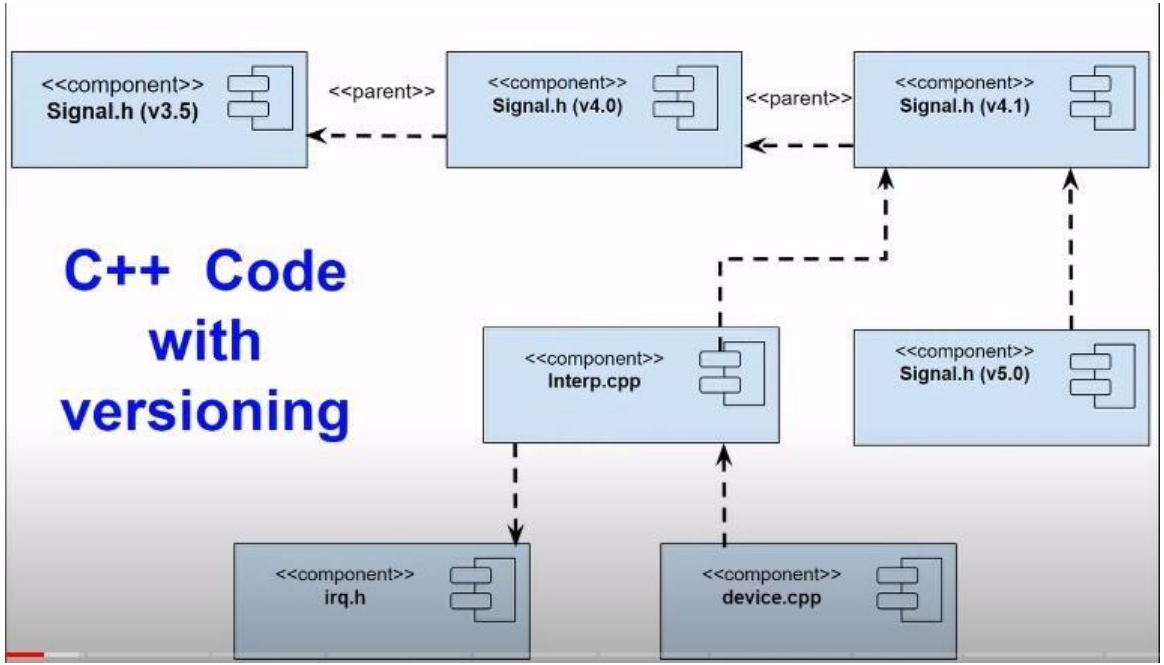
- It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

Following are some artifacts that are needed to be identified before drawing a component diagram:

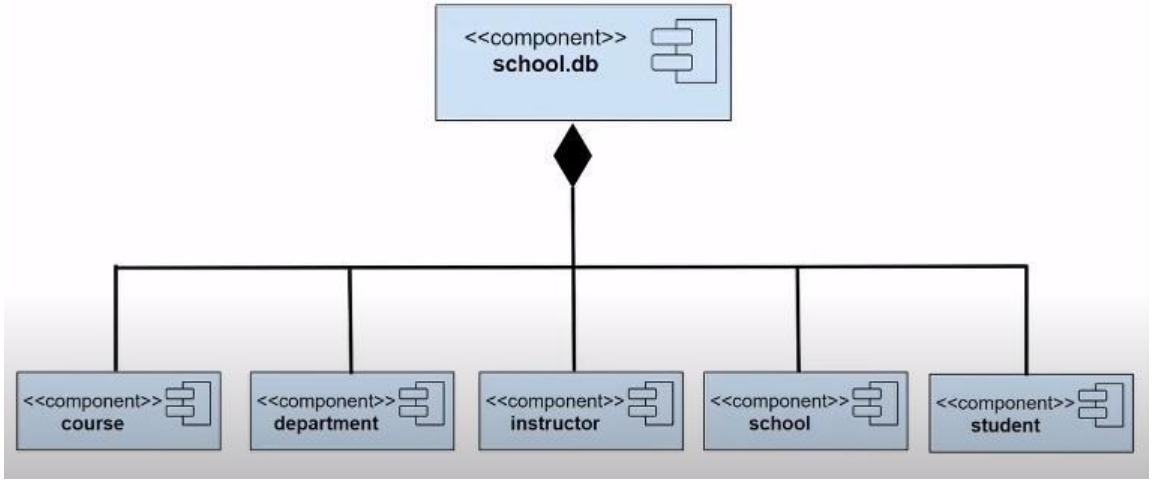
- What files are used inside the system?
- What is the application of relevant libraries and artifacts?
- What is the relationship between the artifacts?

The component diagram can be used for the followings:

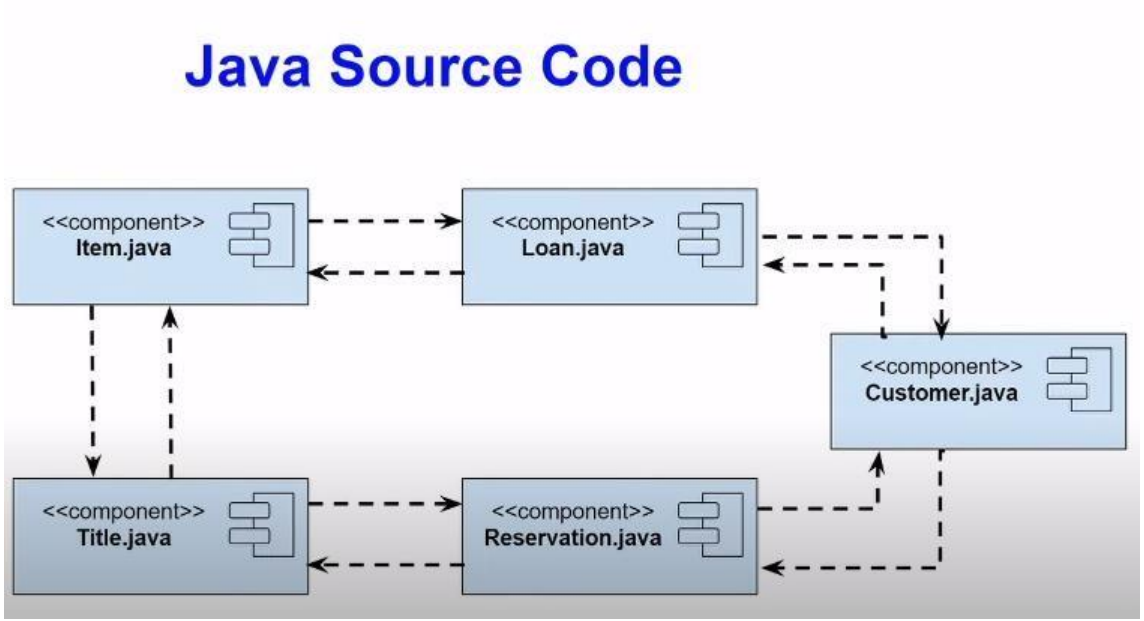
- To model the components of the system.
- To model the schemas of a database.
- To model the applications of an application.
- To model the system's source code.


<<component>>
Signal.h (v3.5)
<<parent>>
<<component>>
Signal.h (v4.0)
<<parent>>
<<component>>
Signal.h (v4.1)
C++ Code
with
versioning
<<component>>
Interp.cpp
<<component>>
Signal.h (v5.0)
<<component>>
irq.h
<<component>>
device.cpp

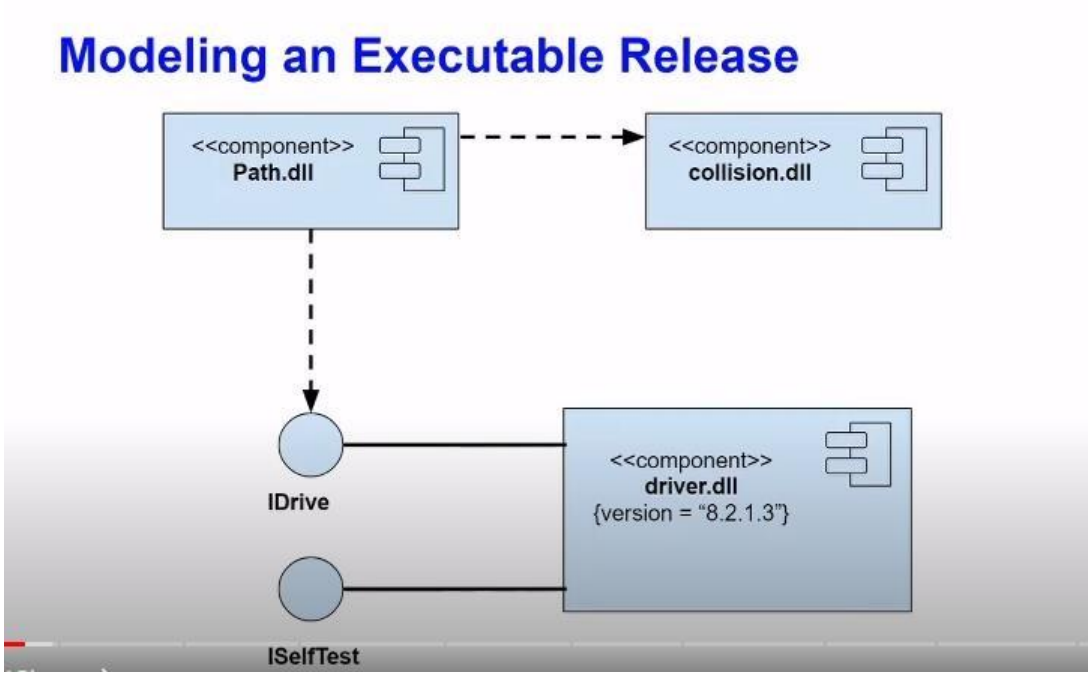
Modeling a Physical Database



Java Source Code



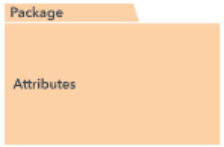
Modeling an Executable Release



Component Diagrams - Notations

Component symbol

Node symbol

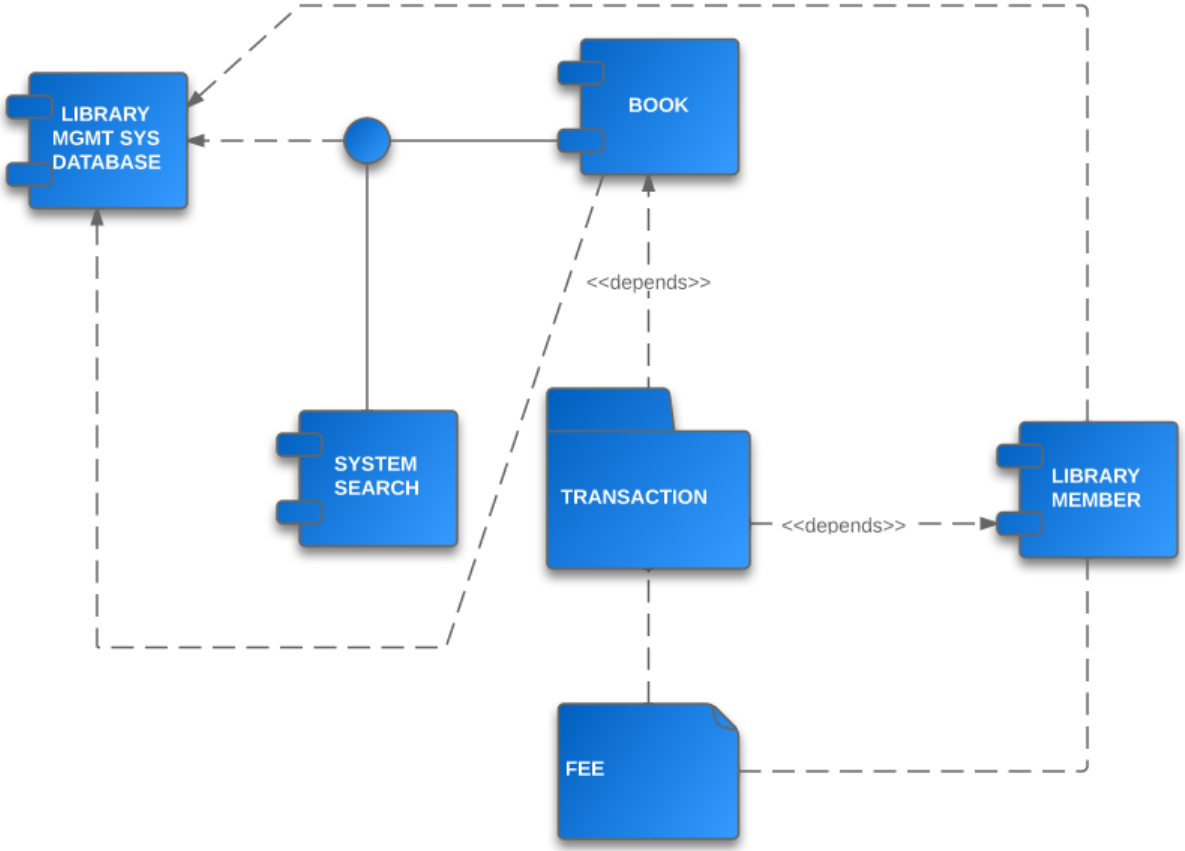


Package symbol

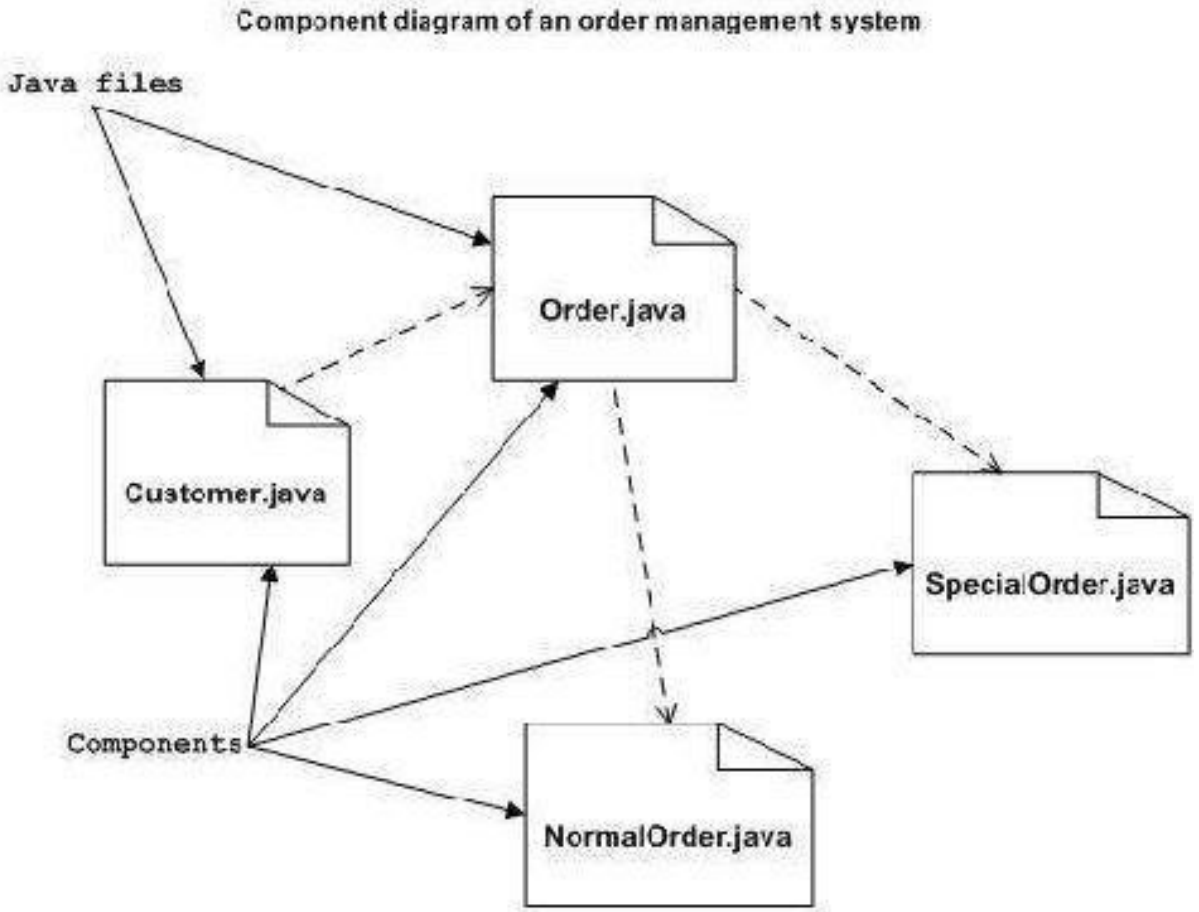
Note symbol

Dependency symbol

Component Diagrams - model the physical aspects of a system



Component Diagrams - model the physical aspects of a system

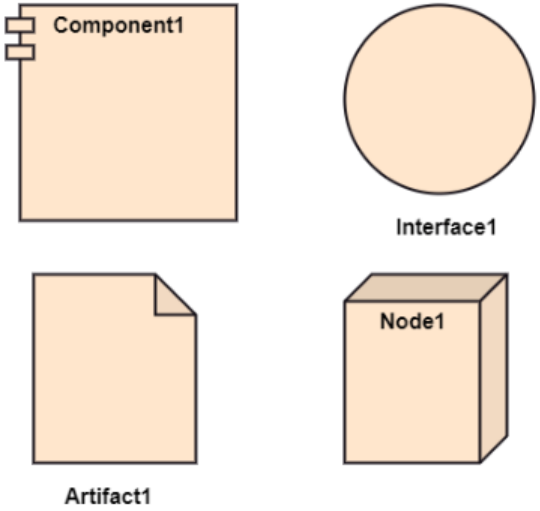


Deployment Diagrams- to visualize the topology of the physical components of a system, where the software components are deployed

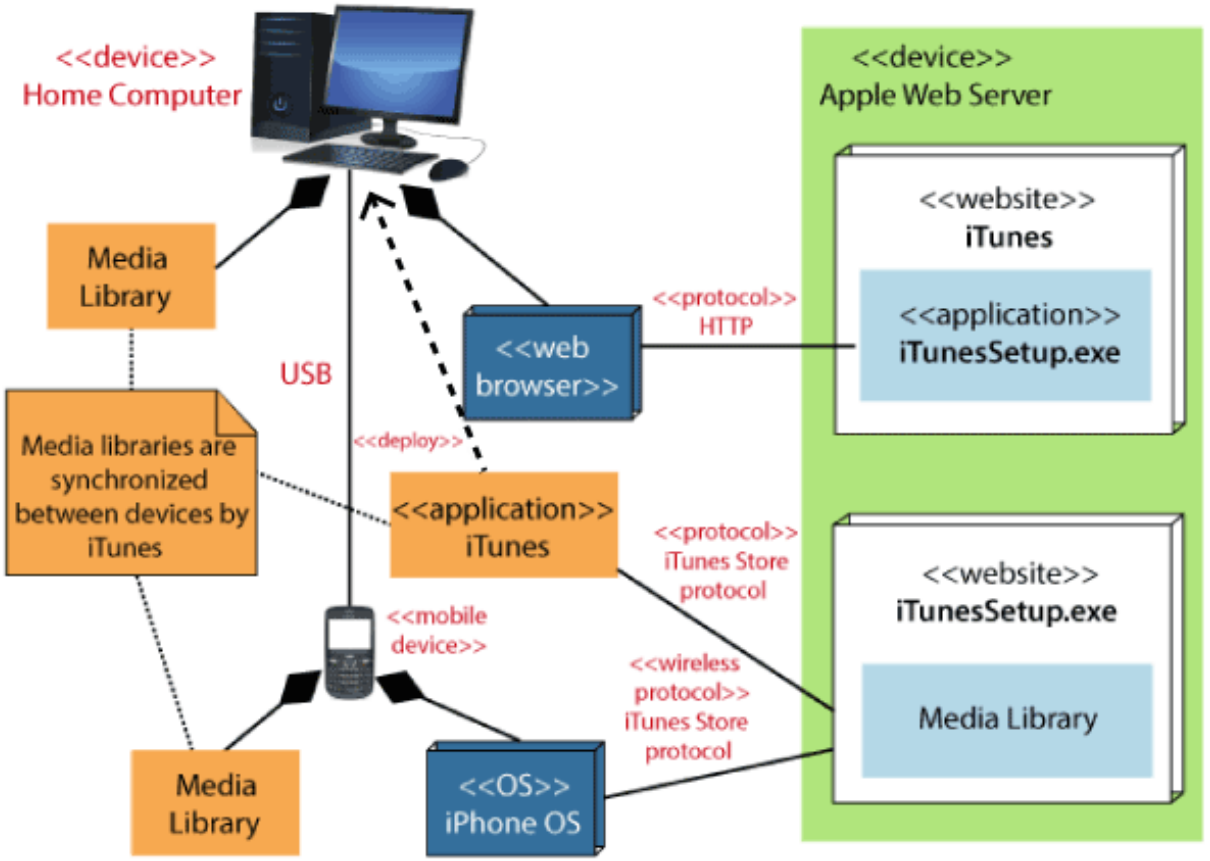
Symbol and notation of Deployment diagram

The deployment diagram consist of the following notations:

1. A component
2. An artifact
3. An interface
4. A node



Deployment Diagrams- to visualize the topology of the physical components of a system, where the software components are deployed



When to use a Deployment Diagram?

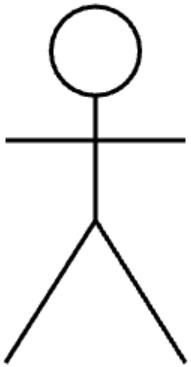
- To model the network and hardware topology of a system.
- To model the distributed networks and systems.
- Implement forwarding and reverse engineering processes.
- To model the hardware details for a client/server system.
- For modeling the embedded system.

Behavioral Diagrams - Behaviour of the system when it is running/operating

- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Activity diagram

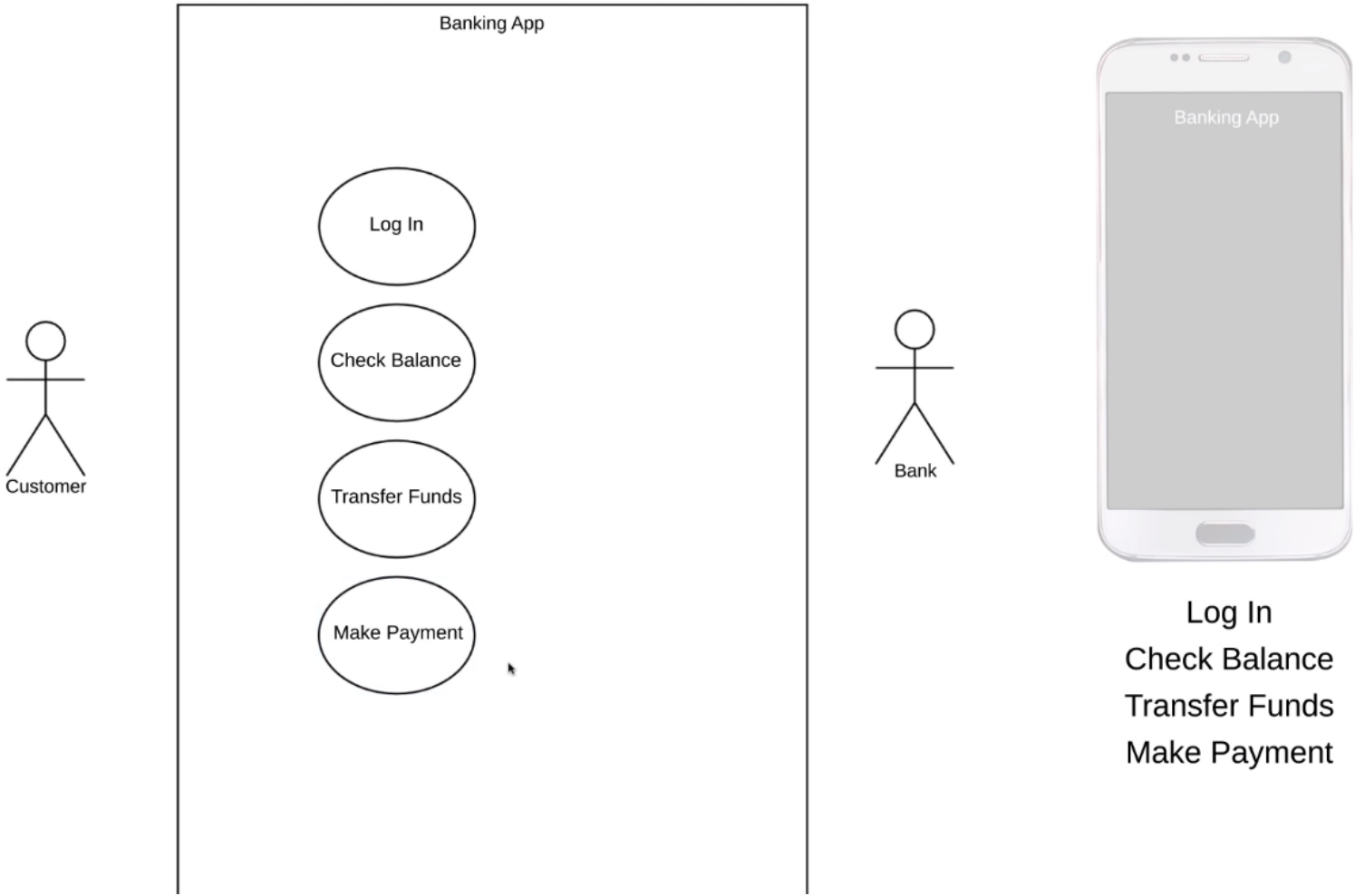
Use Case Actors

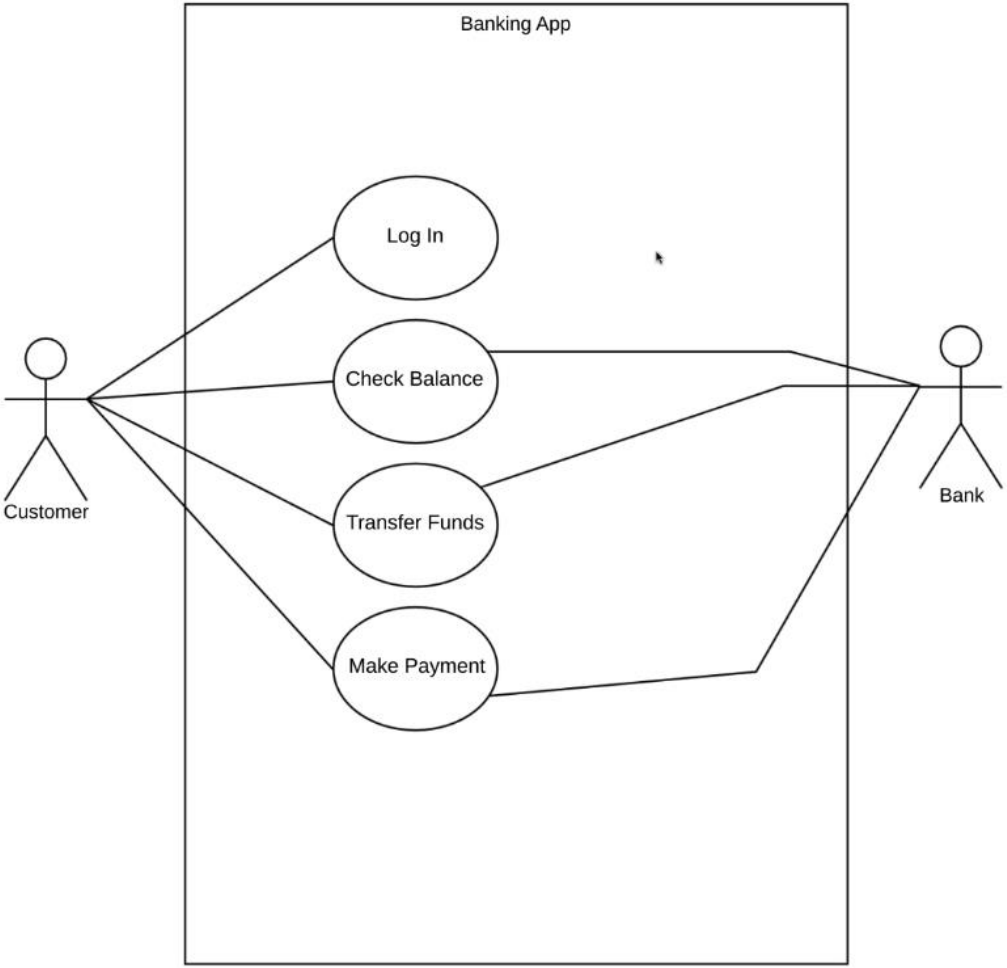
- Primary Actors
  - Initiate the use of the system



- Secondary Actors
  - Reactionary

- Actor can be
  - People
  - Organization
  - External Device


Banking App
Log In
Check Balance
Transfer Funds
Make Payment
Customer
Bank
Banking App
Log In
Check Balance
Transfer Funds
Make Payment



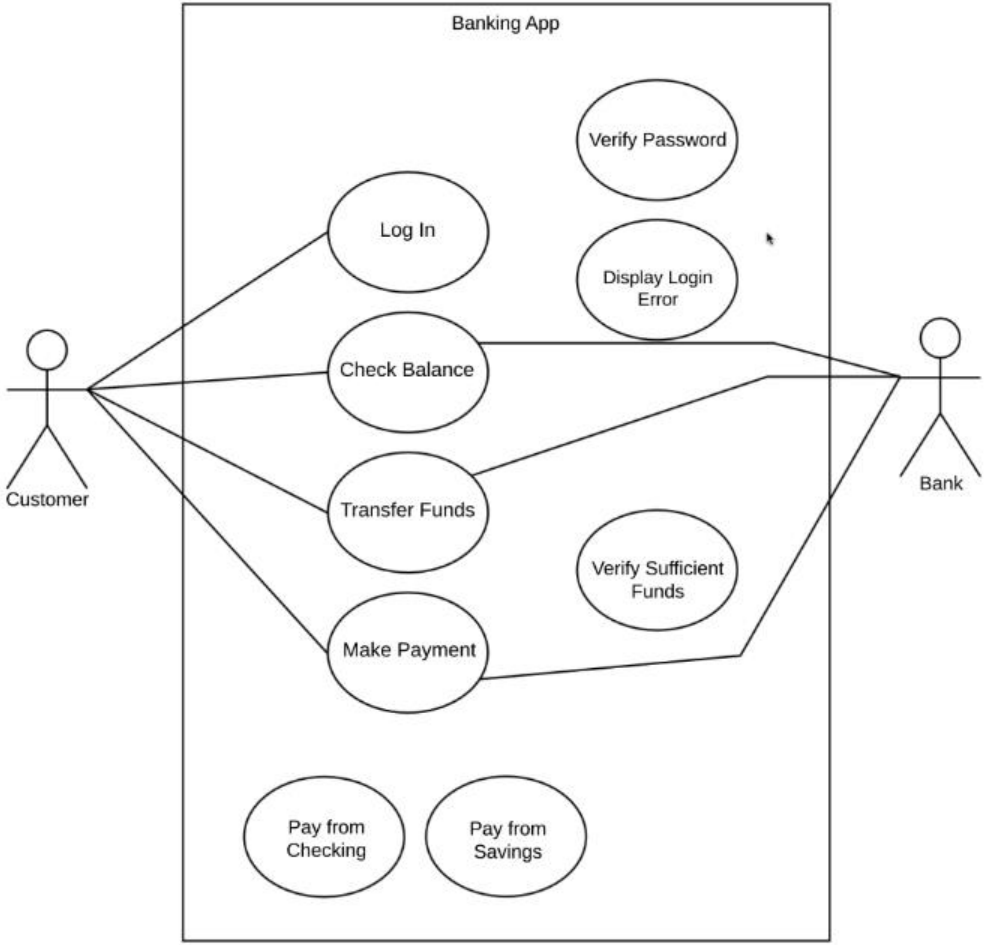
Relationships

Association

Include

Extend

Generalization



Relationships

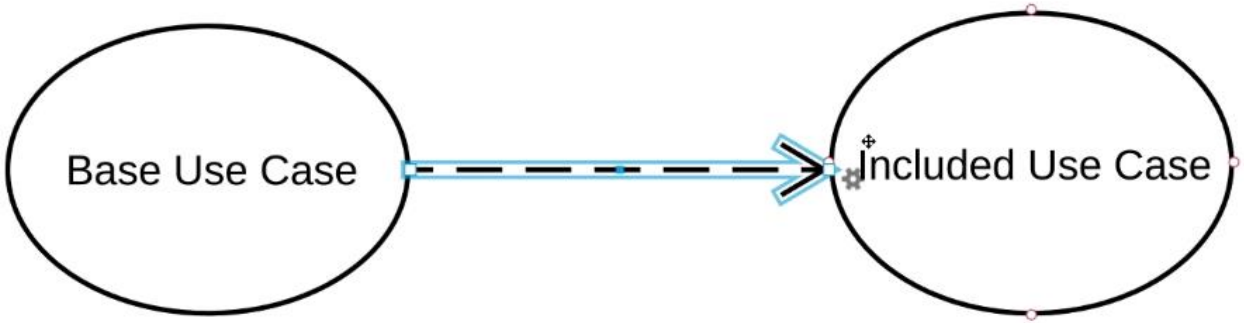
Association

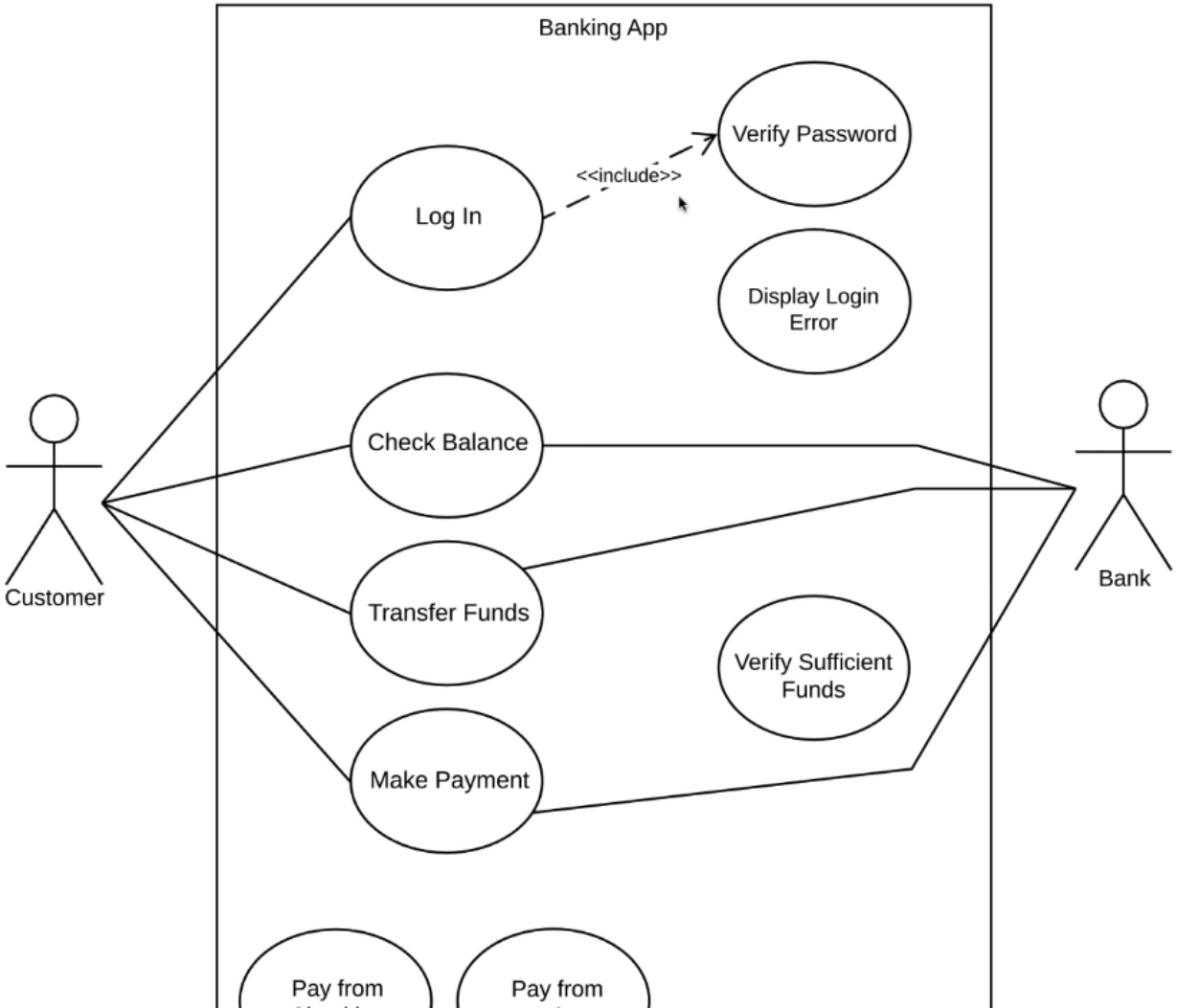
Include

Extend

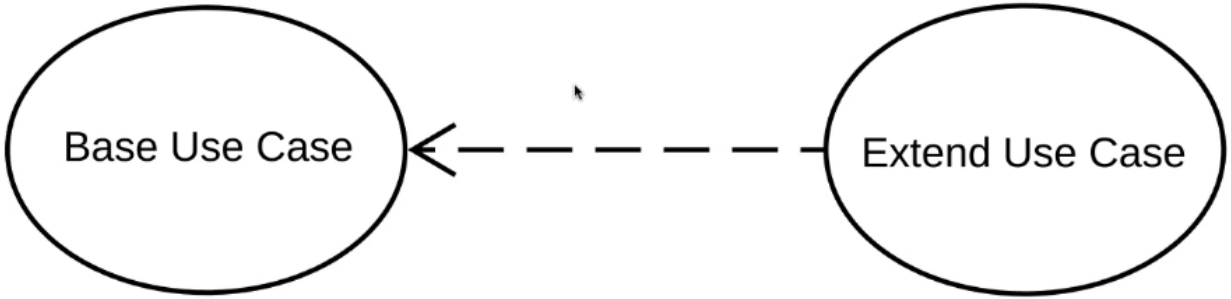
Generalization

Include

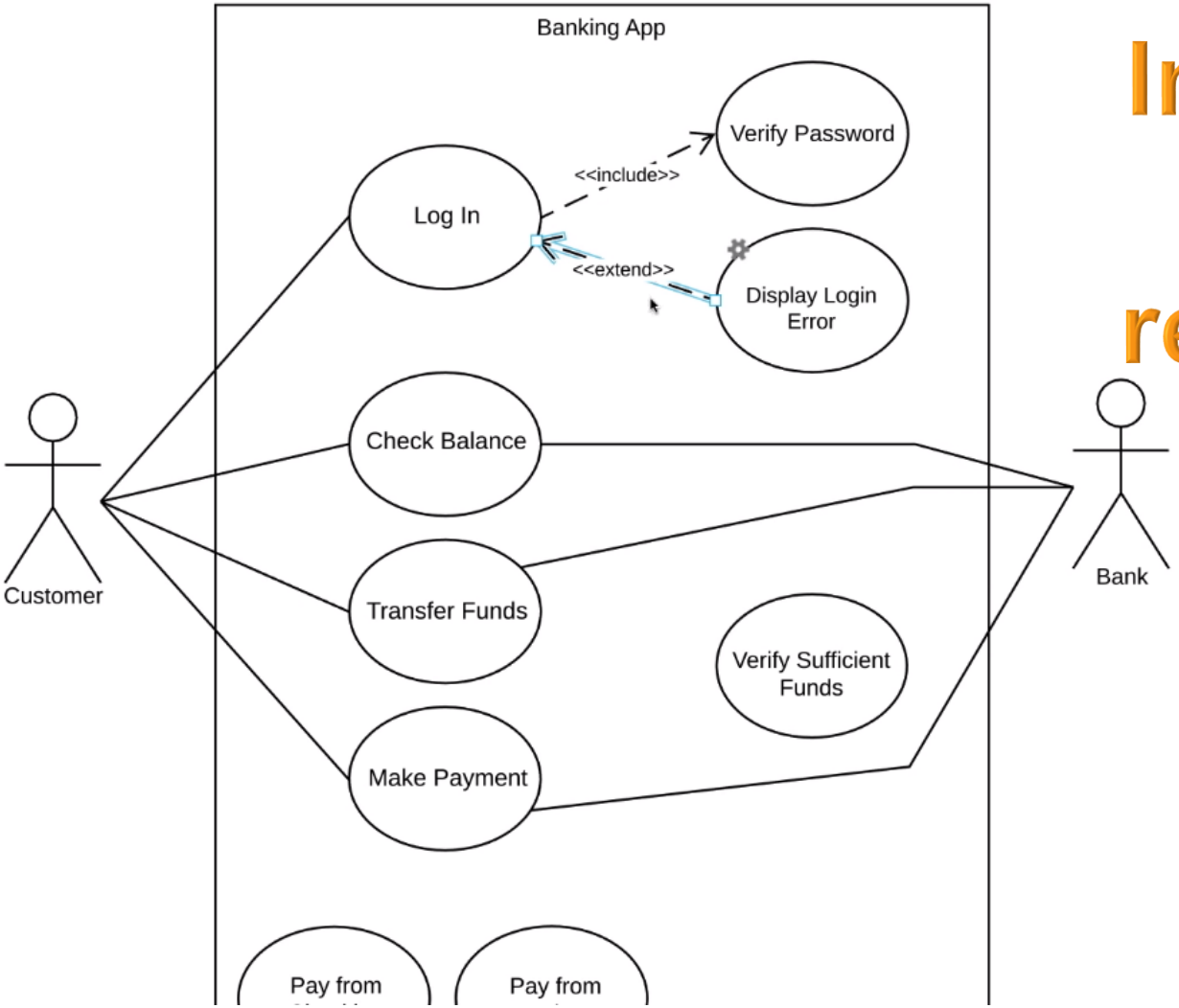


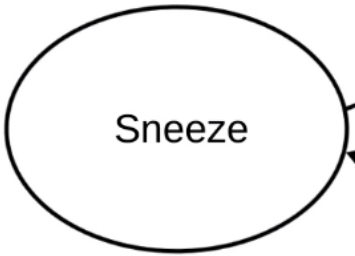

Banking App
Verify Password
<<include>>
Log In
Display Login Error
Check Balance
Customer
Bank
Transfer Funds
Verify Sufficient Funds
Make Payment
Pay from
Pay from

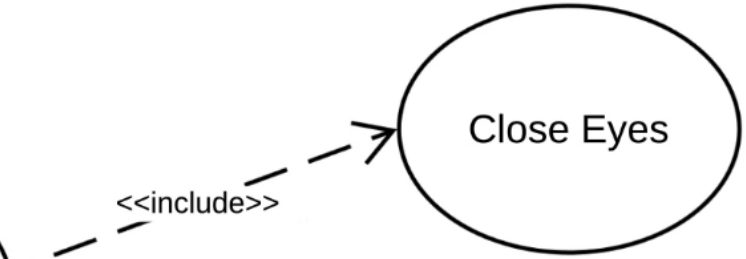
Extend

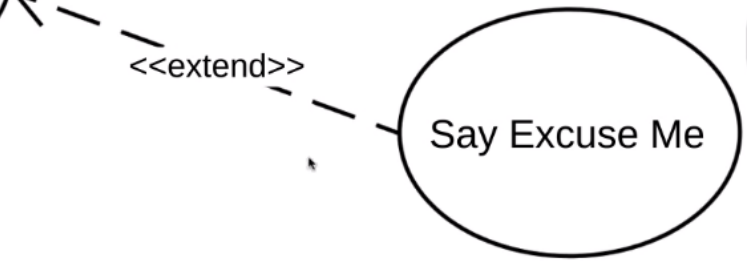


Include and extend relationship



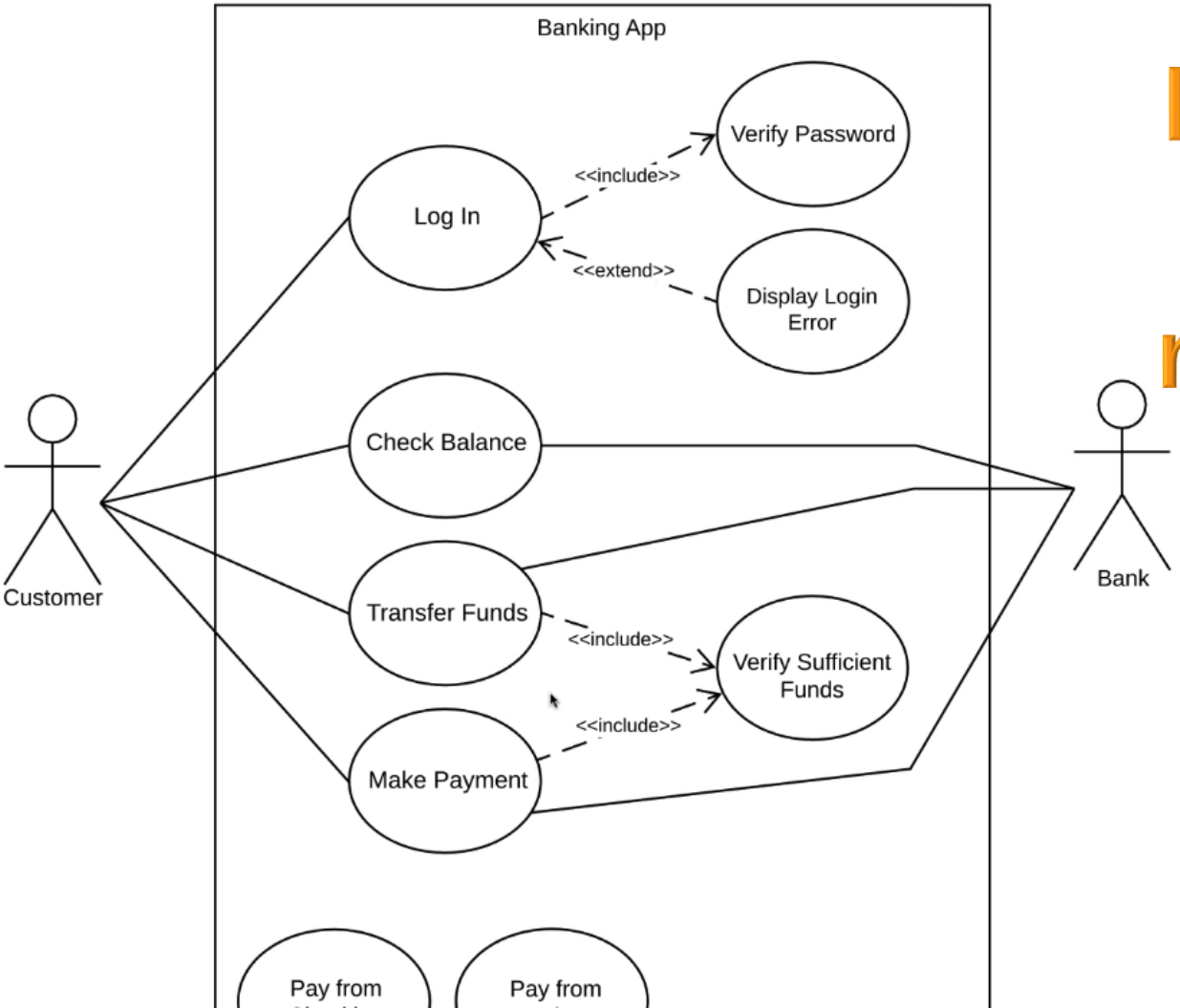




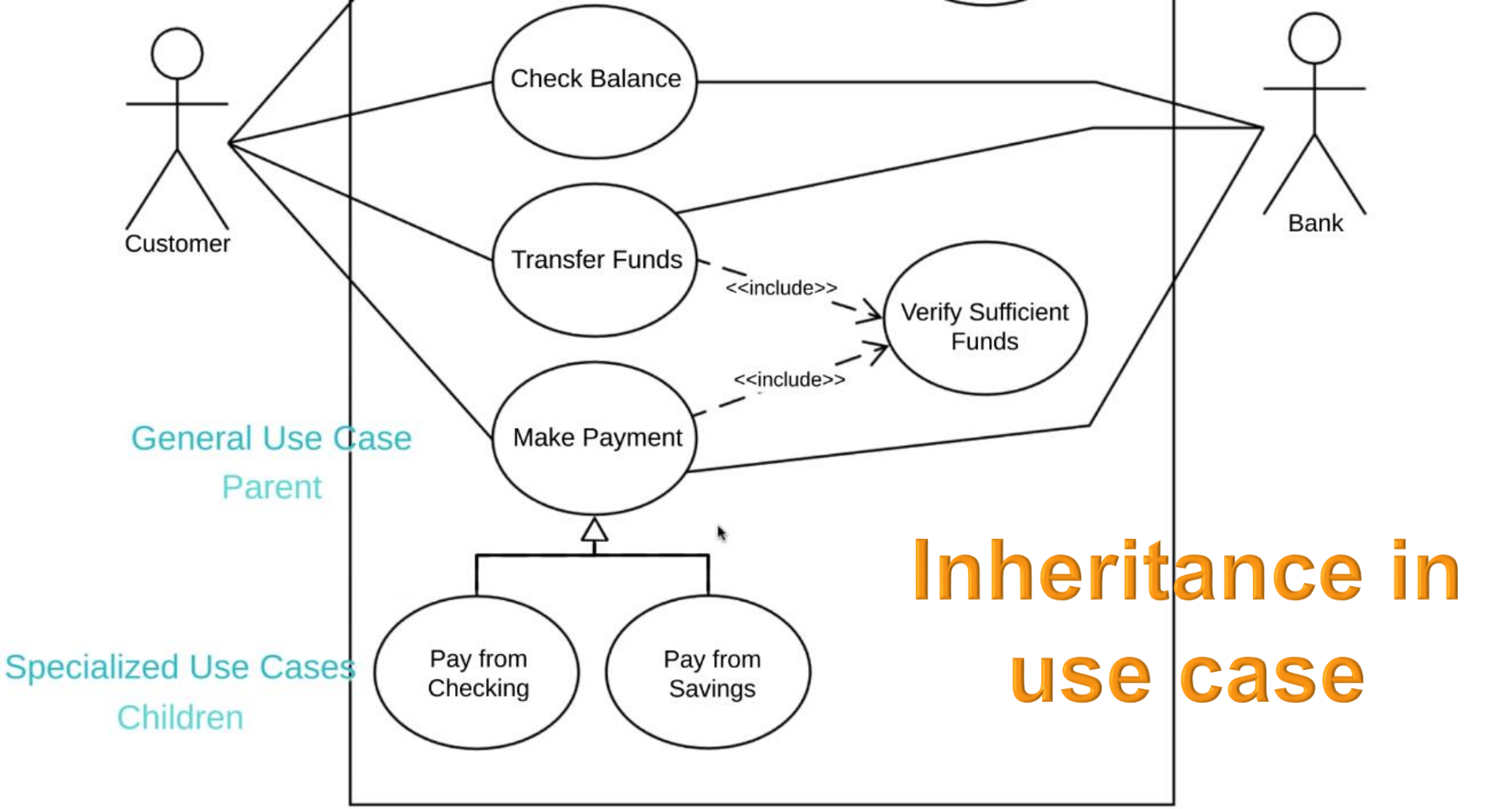


Close Eyes

Say Excuse Me

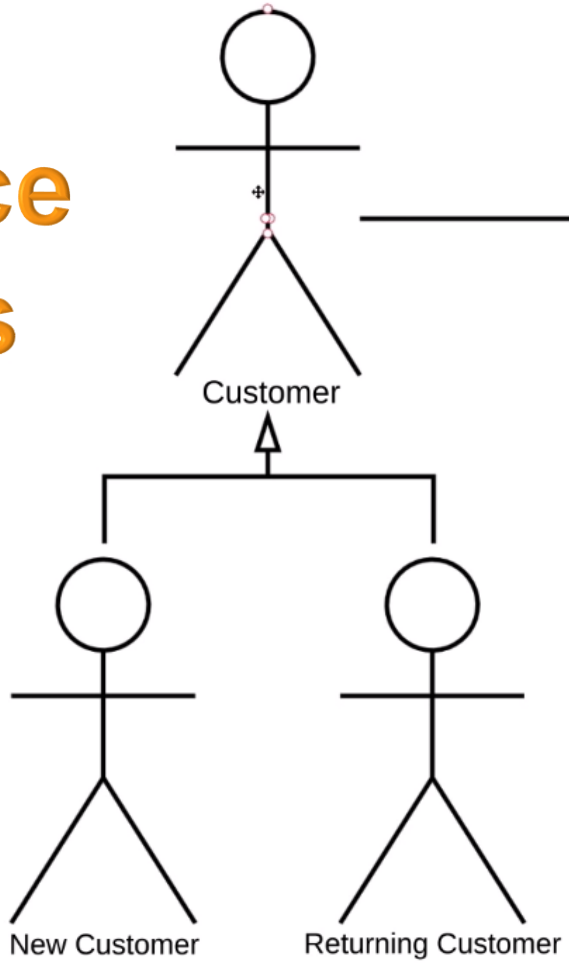


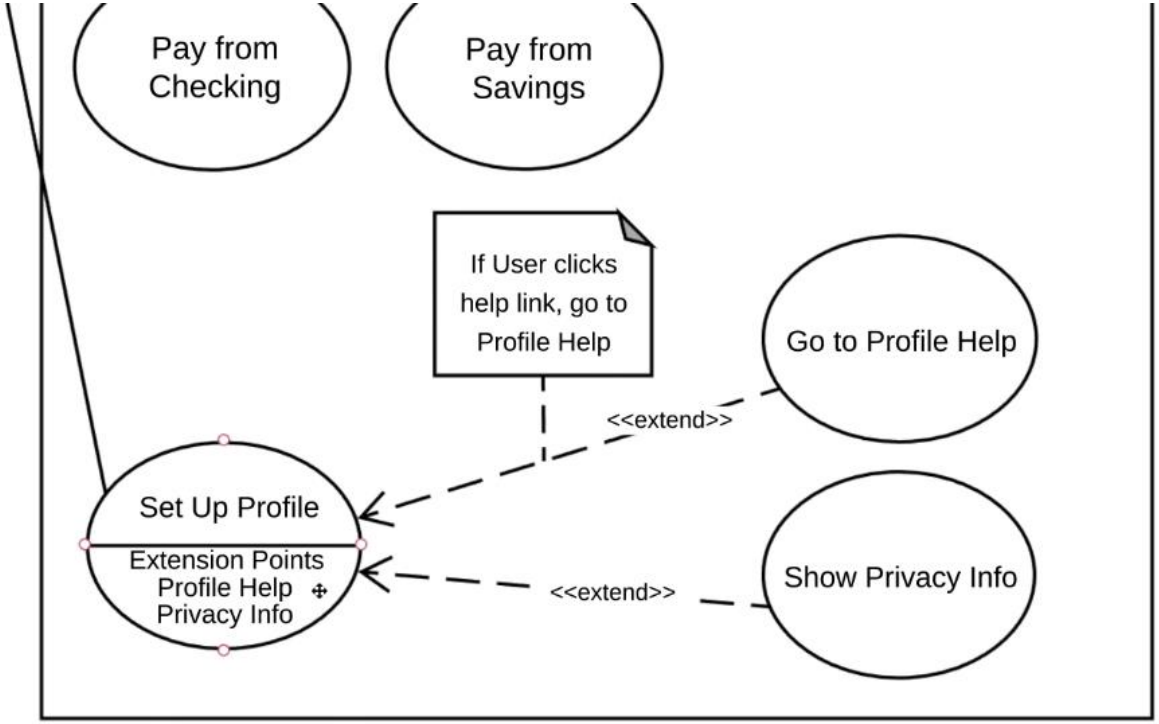
Include and extend relationship



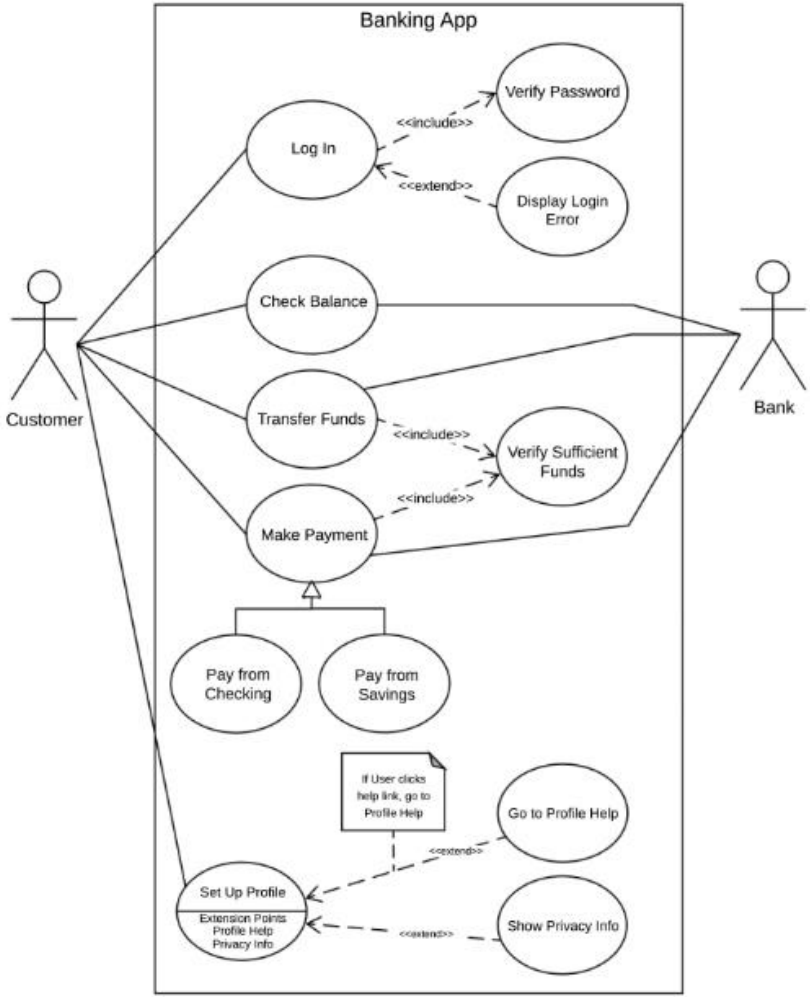
Inheritance in use case

Inheritance in Actors





Extension Point

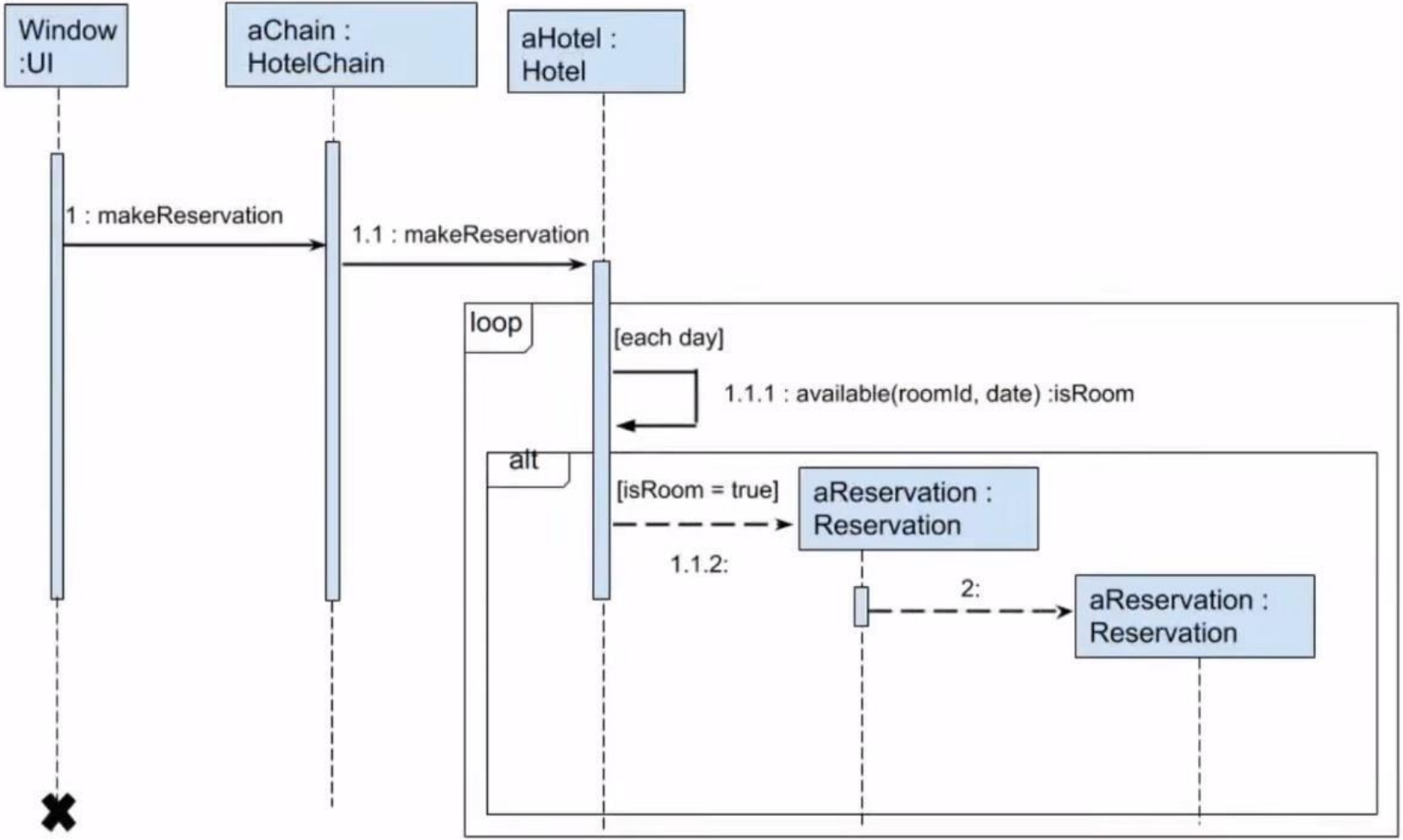

Banking App
Verify Password
Log In
<<include>>
<<extend>>
Display Login Error
Check Balance
Customer
Bank
Transfer Funds
<<include>>
Verify Sufficient Funds
<<include>>
Make Payment
Pay from Checking
Pay from Savings
If User clicks help link, go to Profile Help
Go to Profile Help
<<extend>>
Set Up Profile
Extension Points
Profile Help
Privacy Info
<<extend>>
Show Privacy Info

Interaction Diagrams - Interactions among the different elements in the model

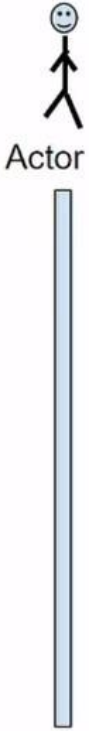
- **Sequence diagram**
- **Collaboration diagram**

Purpose of a Sequence Diagram

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either models generic interactions or some certain instances of interaction.

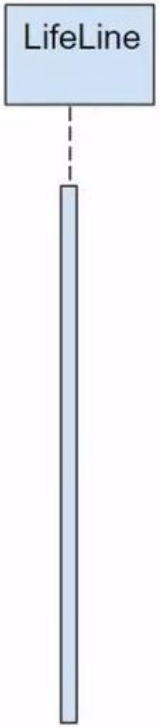

Window
:UI
aChain :
HotelChain
aHotel :
Hotel
1 : makeReservation
1.1 : makeReservation
loop
[each day]
1.1.1 : available(roomId, date) :isRoom
alt
[isRoom = true]
aReservation :
Reservation
1.1.2:
2:
aReservation :
Reservation

Actor



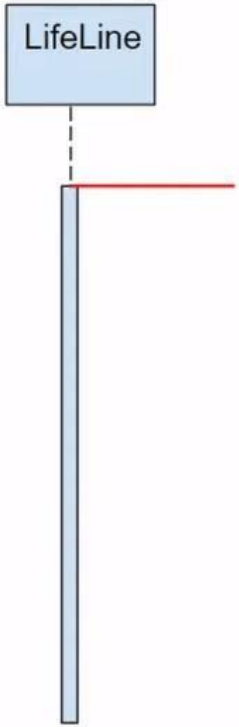
- A type of role played by the instance that interacts with the subject (for example, by exchanging signals and data).
- Actor represents the roles played by users, external equipment, or other actors.
- Actor does not necessarily represent a specific physical entity, but simply a specific role of some entity.
- A person can play the role of several different actors, and, conversely, an actor may be played by multiple different people.

LifeLine



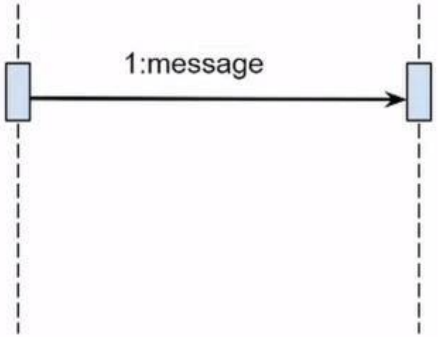
Represents an individual participant in the interaction and displays the passage of time.

Activation



- The thin rectangle on the lifeline indicates the period during which the element performs an operation.

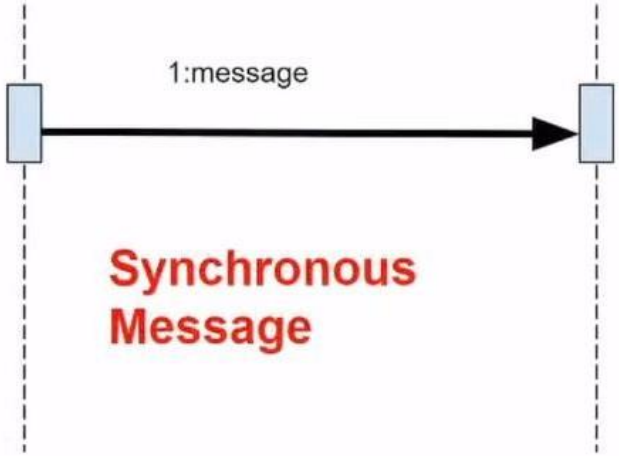
- Top and bottom of the rectangle are aligned with the start and end of a specific function.



Call Message

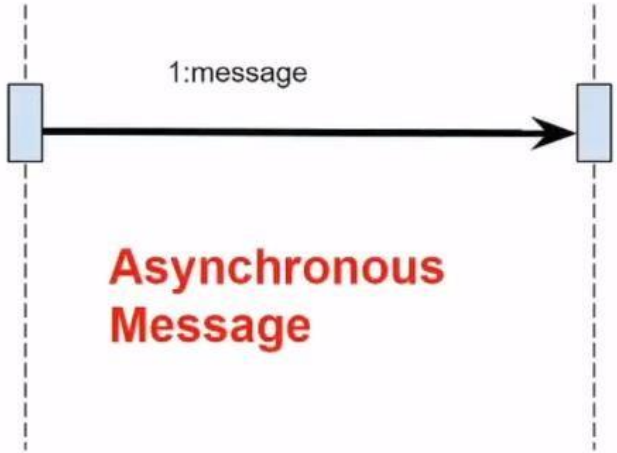
- Used to call procedures, execute operations, or designate individual nested control flows.
- One end of the arrow always touches the control focus or lifeline of the client object that triggers the message.
- The tip of the arrow points at the lifeline of the object that receives the message and takes action as a result. At the same time, this object also often receives a control focus, becoming active.

Call Message



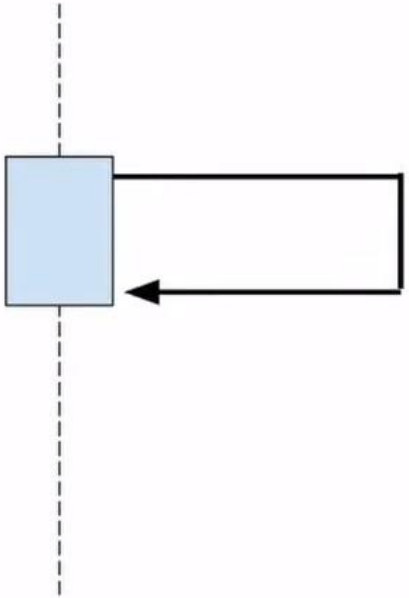
The message is synchronous (depicted with a filled arrow), when the client sends a message, for example, to the server and waits for a response before doing anything else.

Call Message



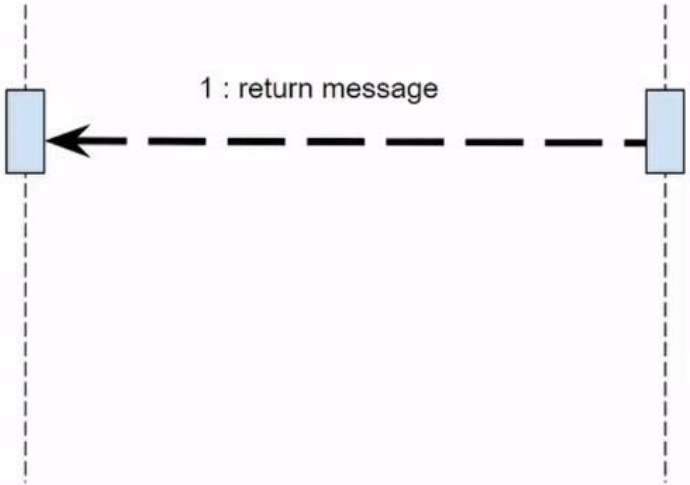
Message is asynchronous (depicted with a regular arrow), when the client continues to perform operations without waiting for a response.

Self Message



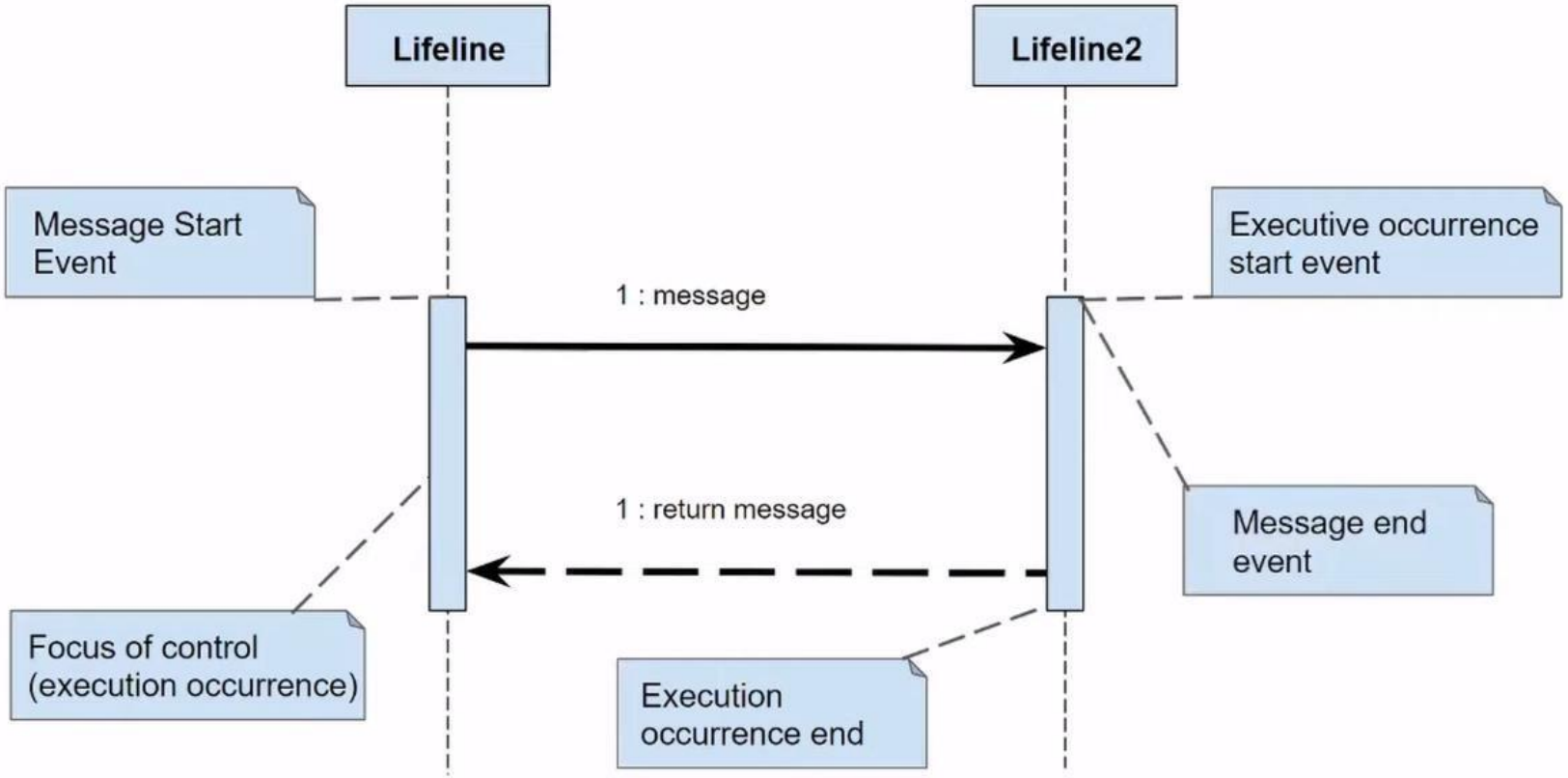
A participant sends a message (command) to itself.

Return Message

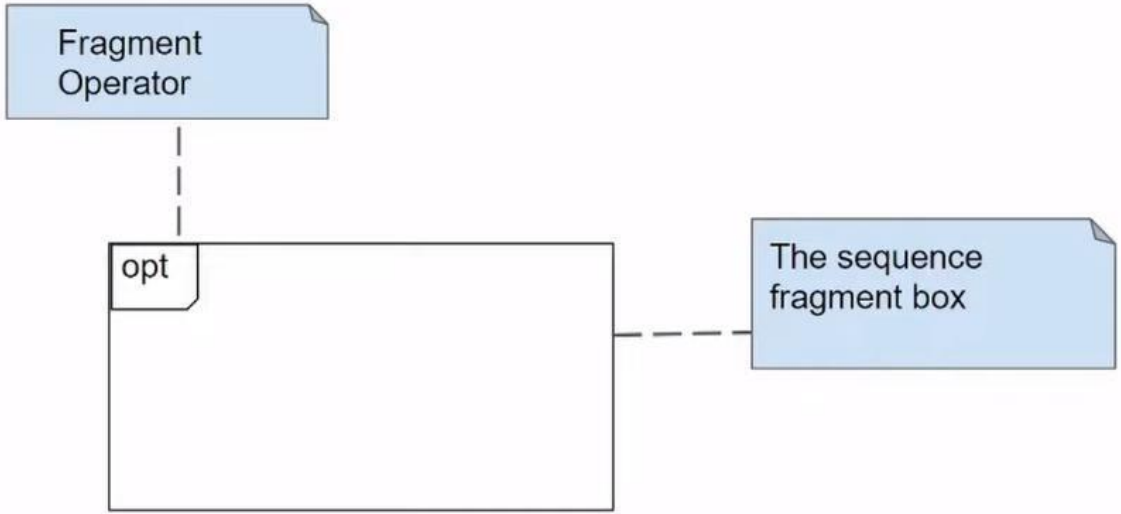


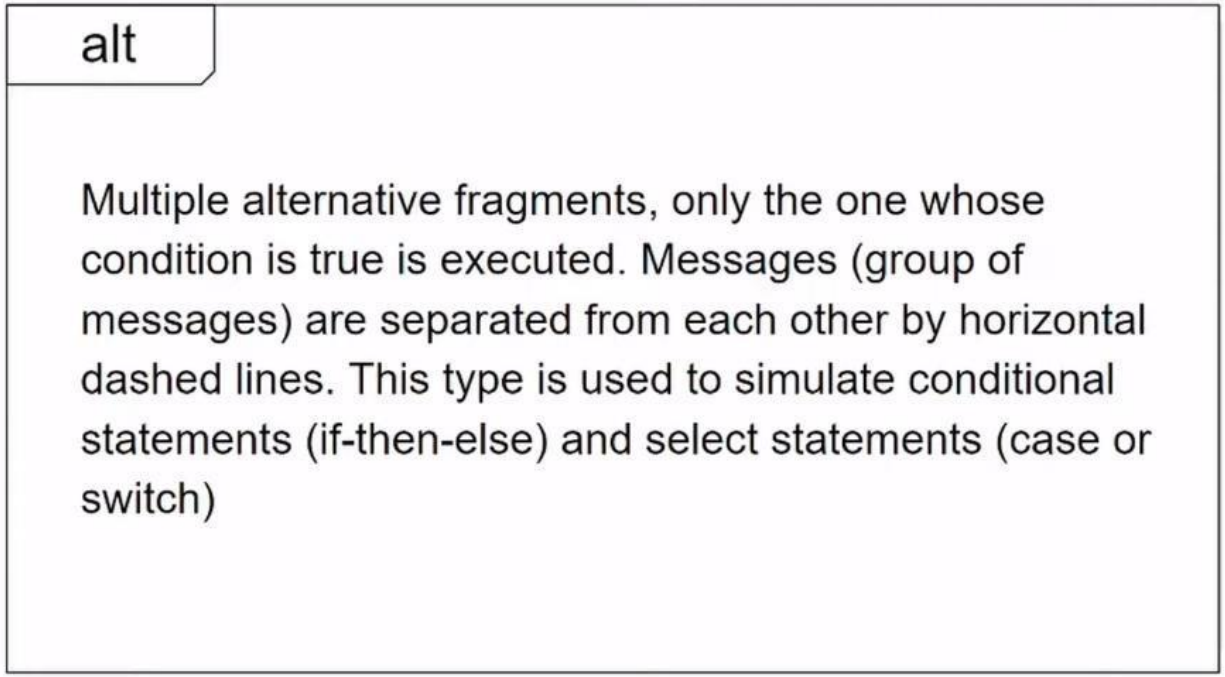
Return message identifies the specific communication between the lifelines of an interaction and represents the transfer of information to the caller of the corresponding message.

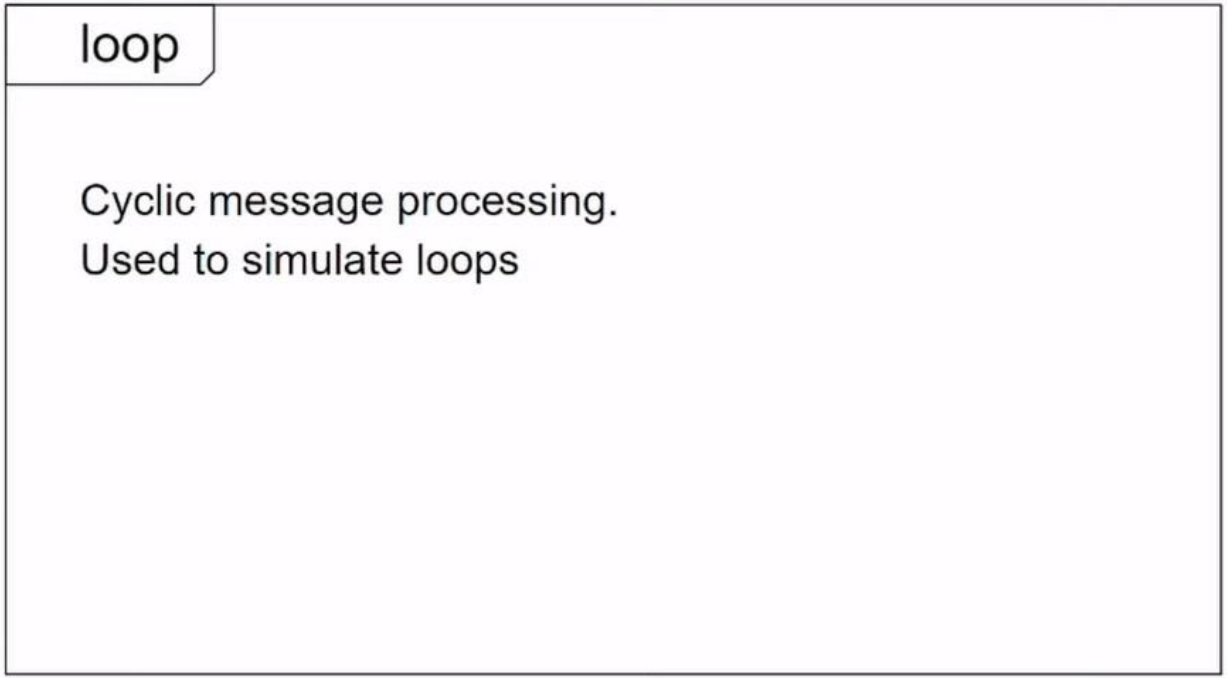
Focus of Control

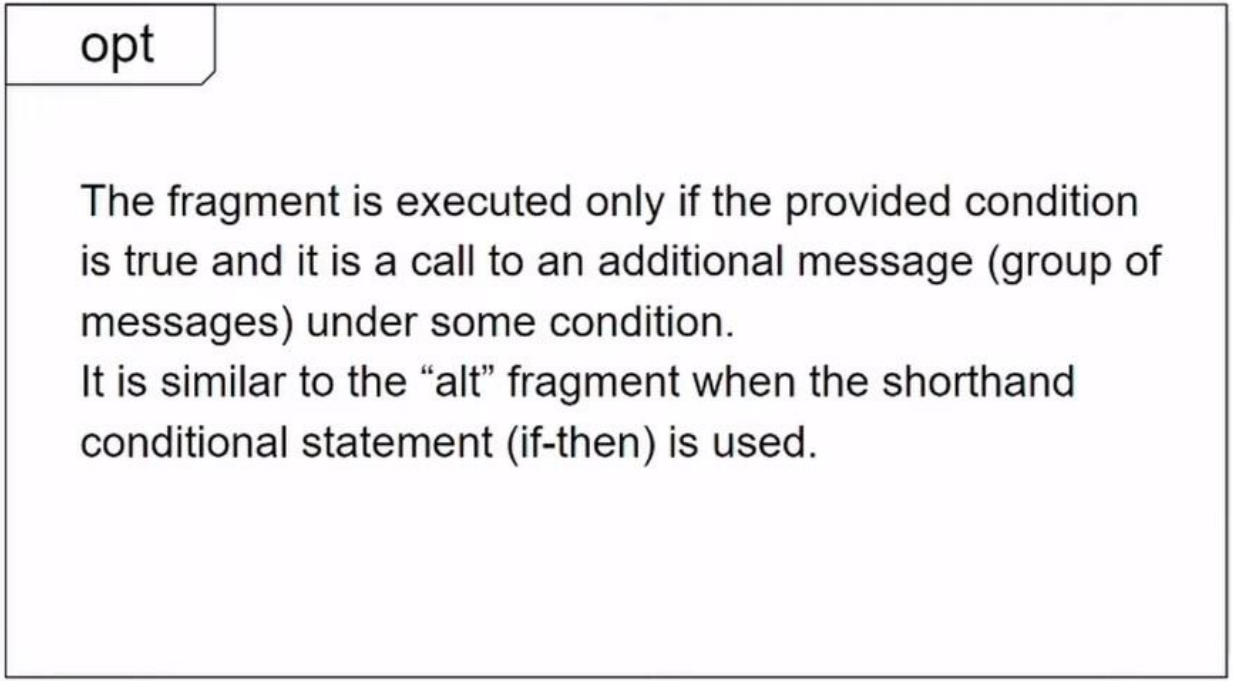


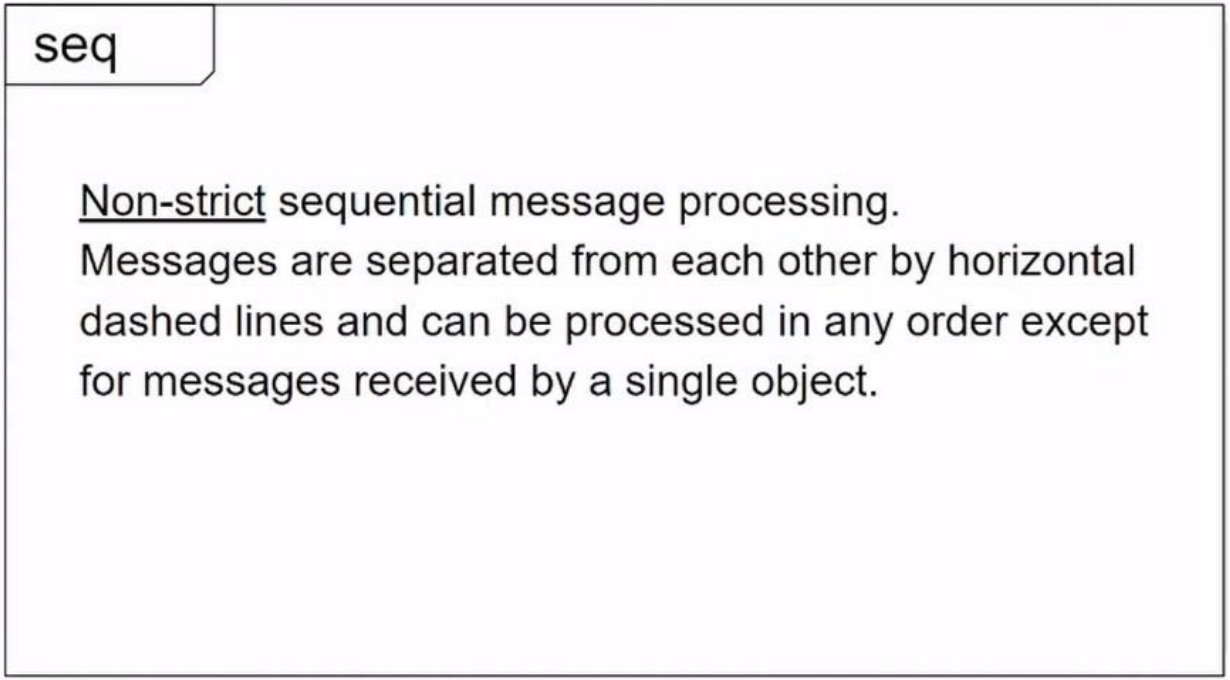
Sequence Fragments

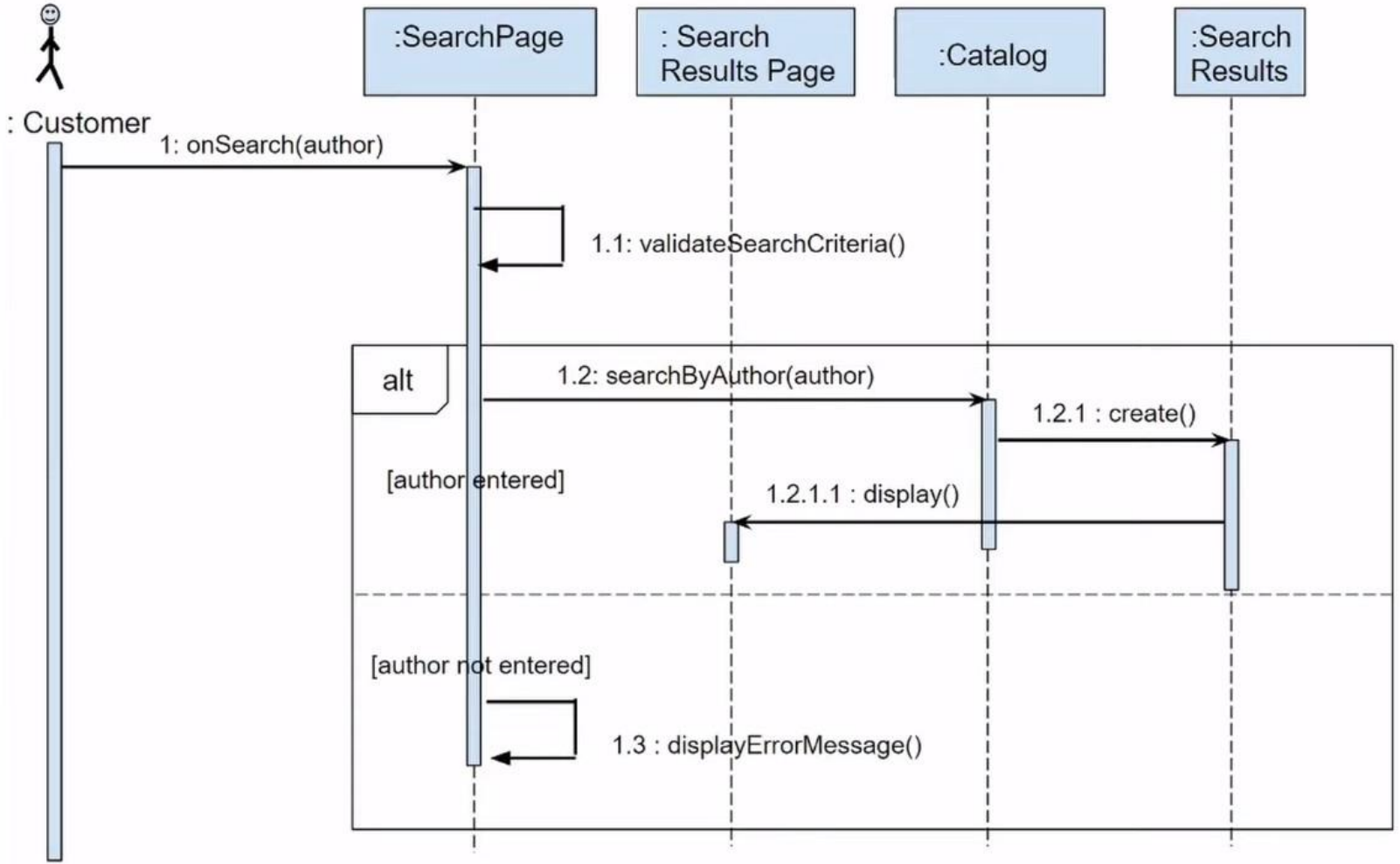



alt
Multiple alternative fragments, only the one whose condition is true is executed. Messages (group of messages) are separated from each other by horizontal dashed lines. This type is used to simulate conditional statements (if-then-else) and select statements (case or switch)

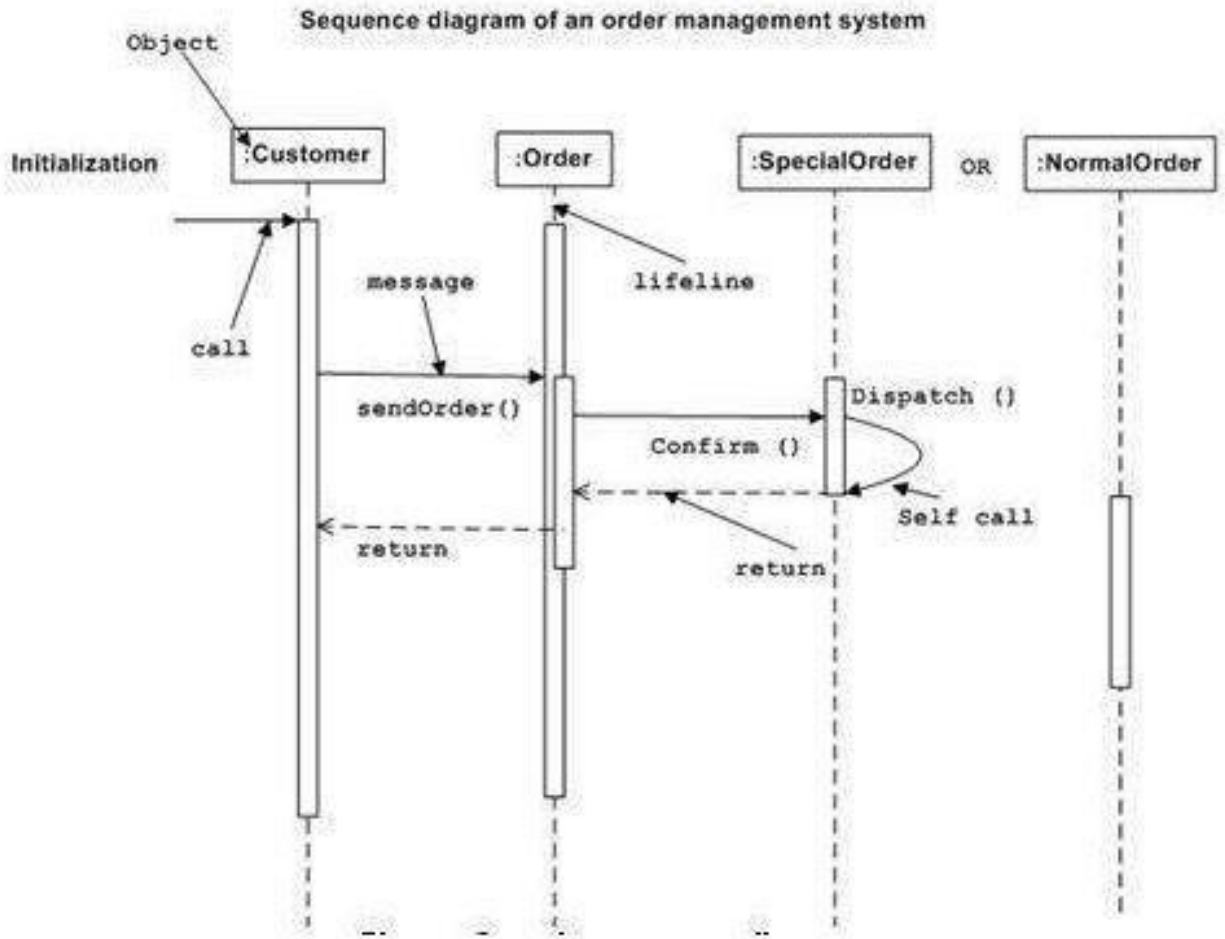

loop
Cyclic message processing.
Used to simulate loops


opt
The fragment is executed only if the provided condition is true and it is a call to an additional message (group of messages) under some condition.
It is similar to the “alt” fragment when the shorthand conditional statement (if-then) is used.

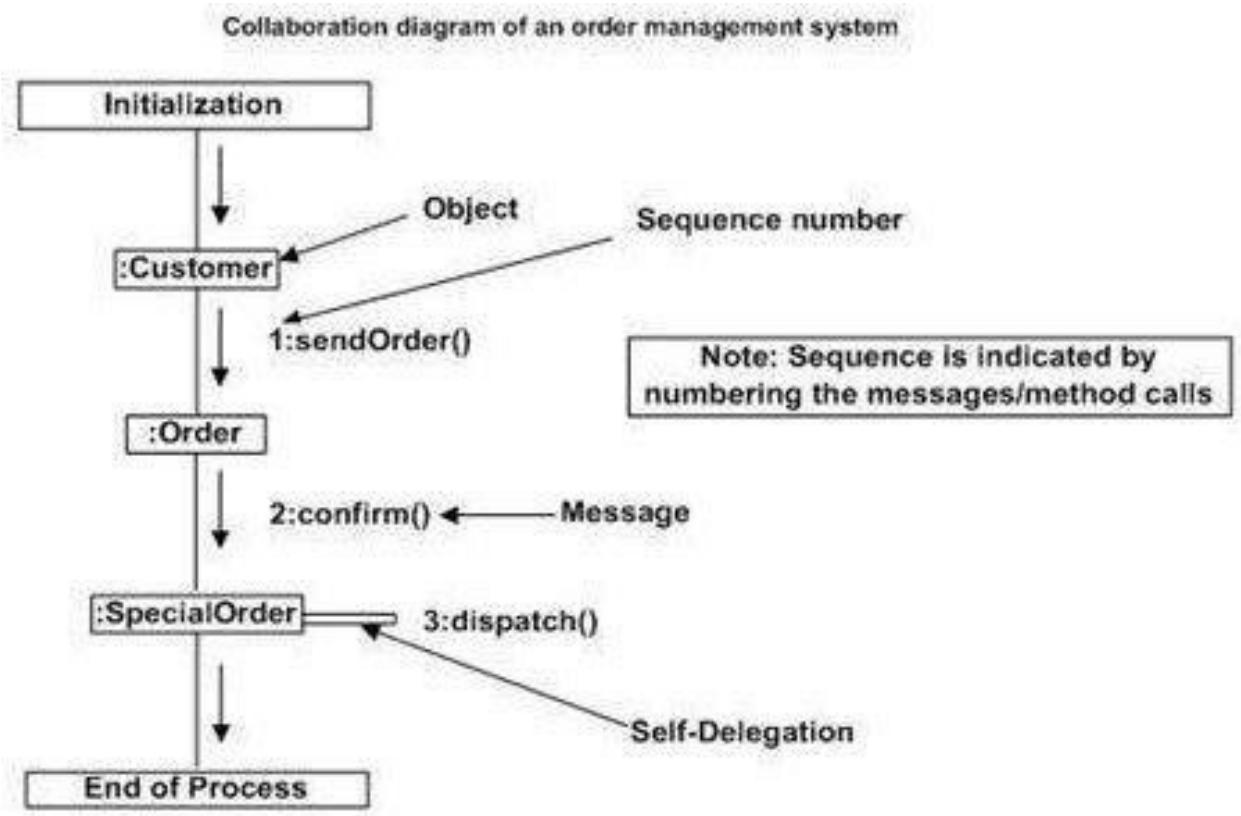

seq
Non-strict sequential message processing.
Messages are separated from each other by horizontal dashed lines and can be processed in any order except for messages received by a single object.


: Customer
:SearchPage
: Search Results Page
:Catalog
:Search Results
1: onSearch(author)
1.1: validateSearchCriteria()
alt
1.2: searchByAuthor(author)
1.2.1 : create()
[author entered]
1.2.1.1 : display()
[author not entered]
1.3 : displayErrorMessage()

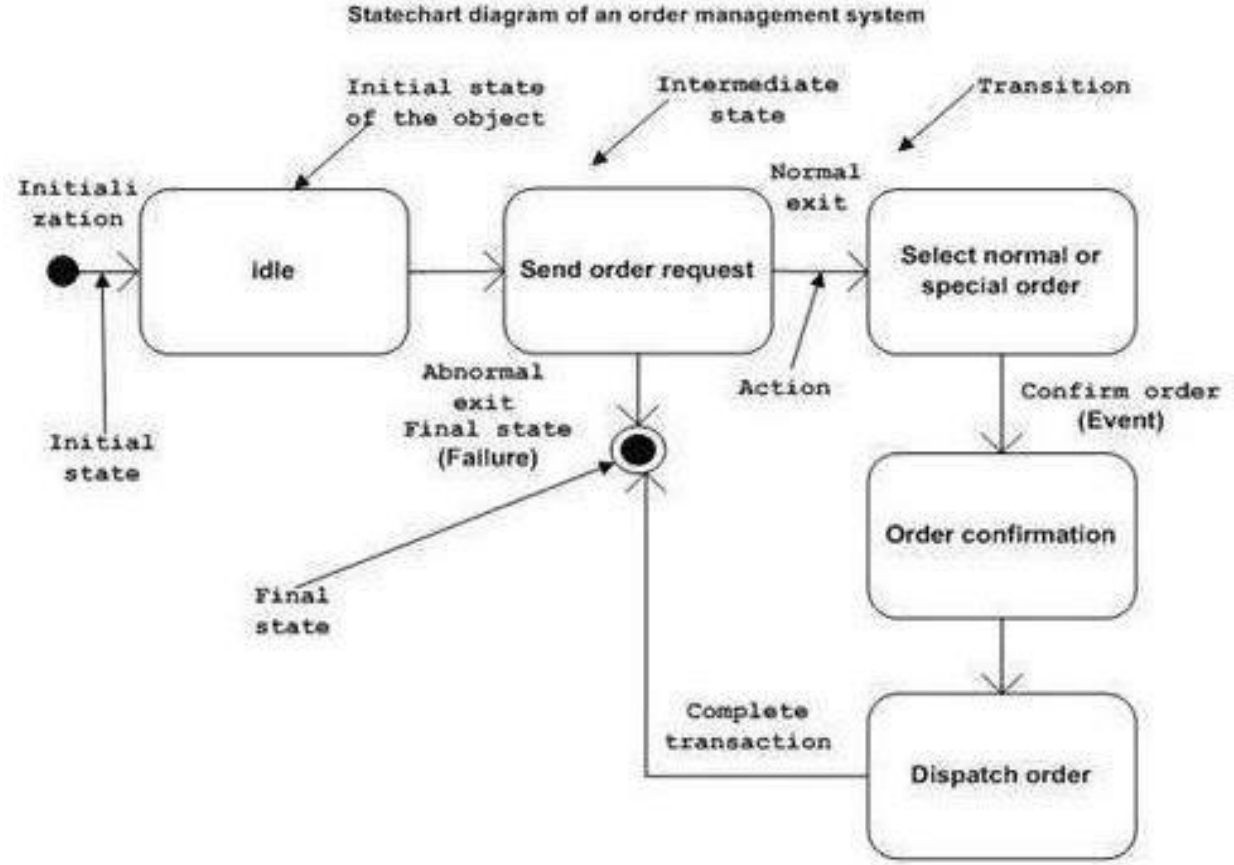
Sequence Diagram- Emphasizes on time sequence of messages



Collaboration diagram - Emphasizes on the structural organization of the objects that send and receive messages.

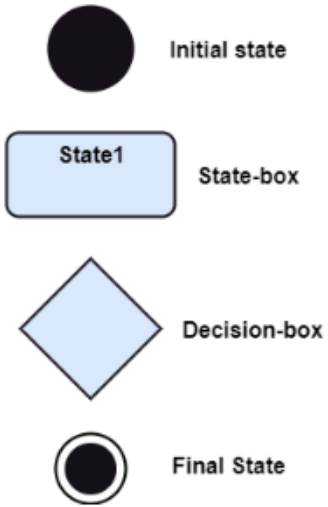


State chart Diagrams - State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.



State chart Diagrams

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



a. **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.

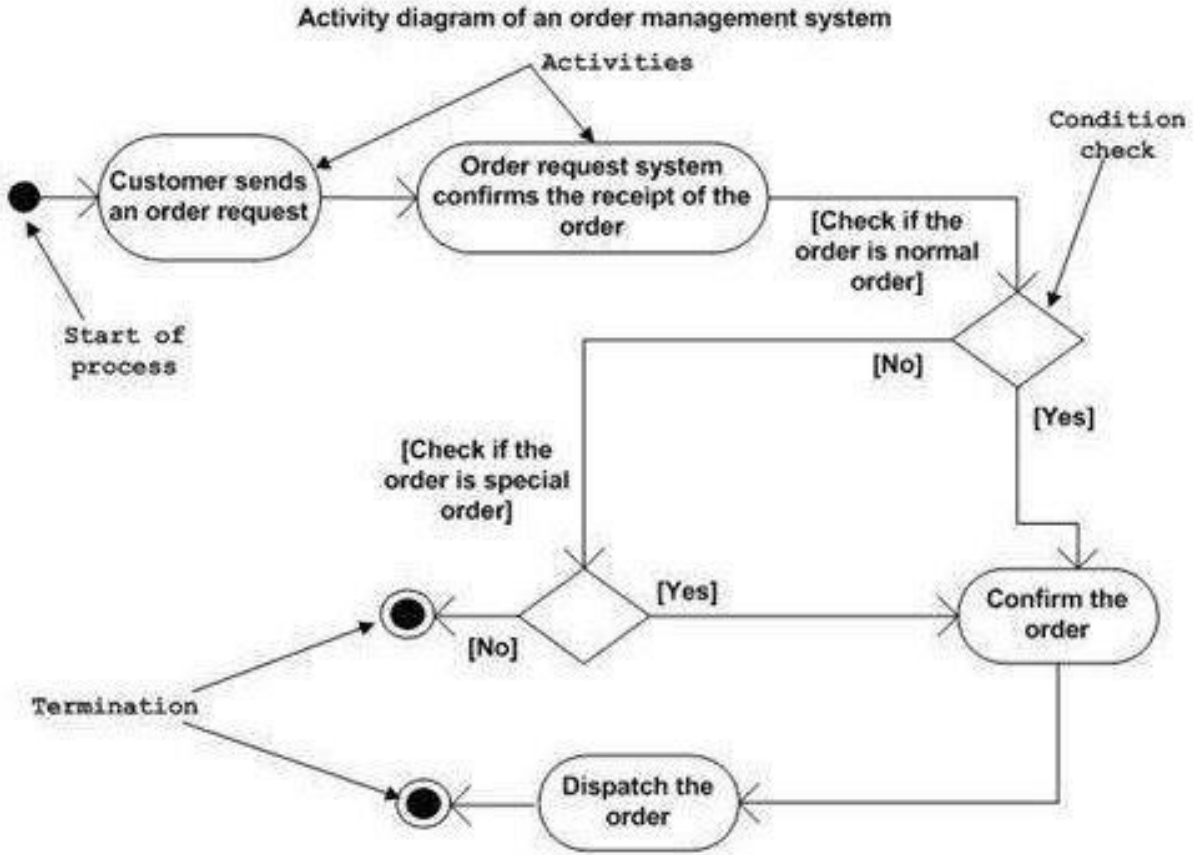
b. **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.

c. **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.

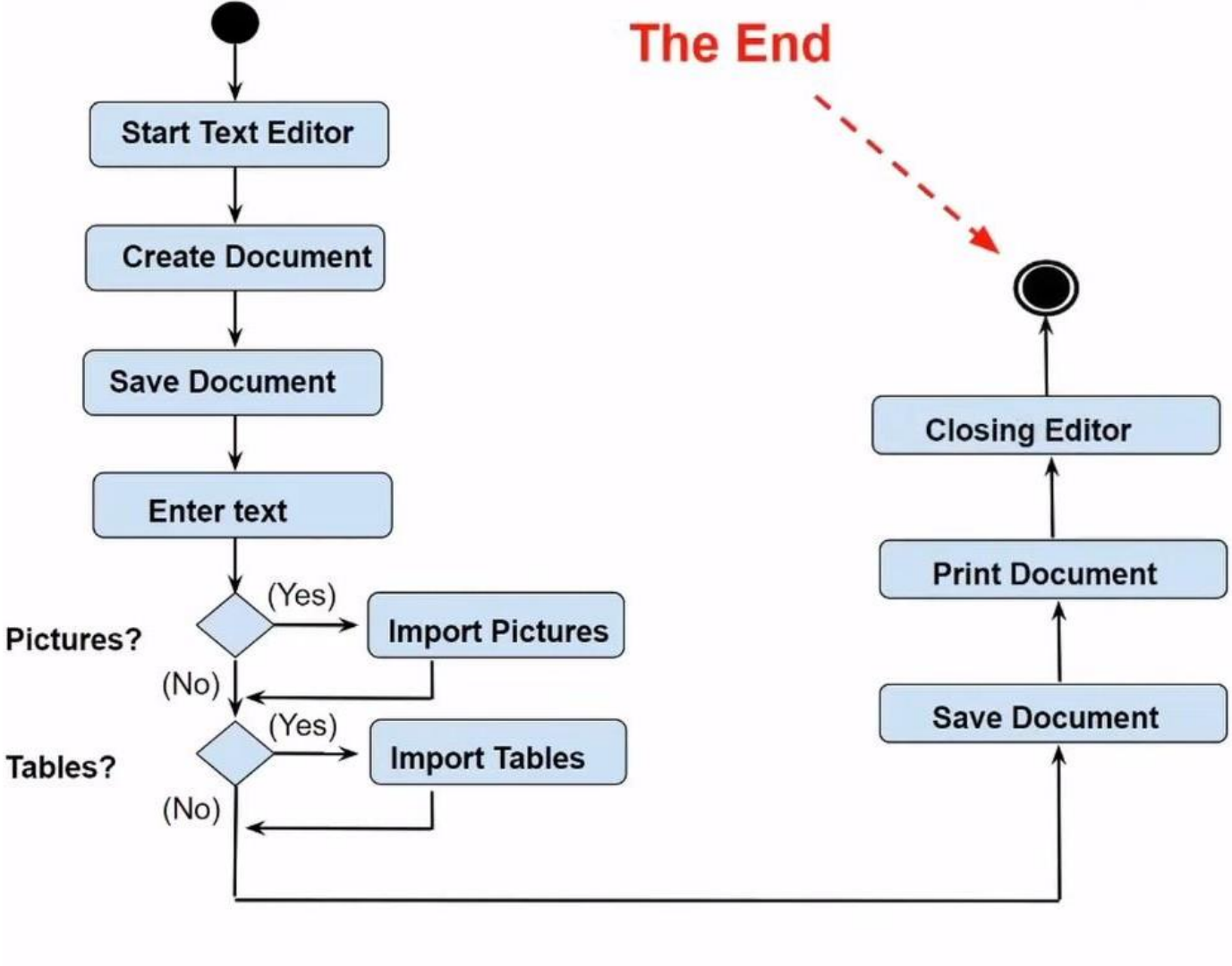
d. **Transition:** A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.

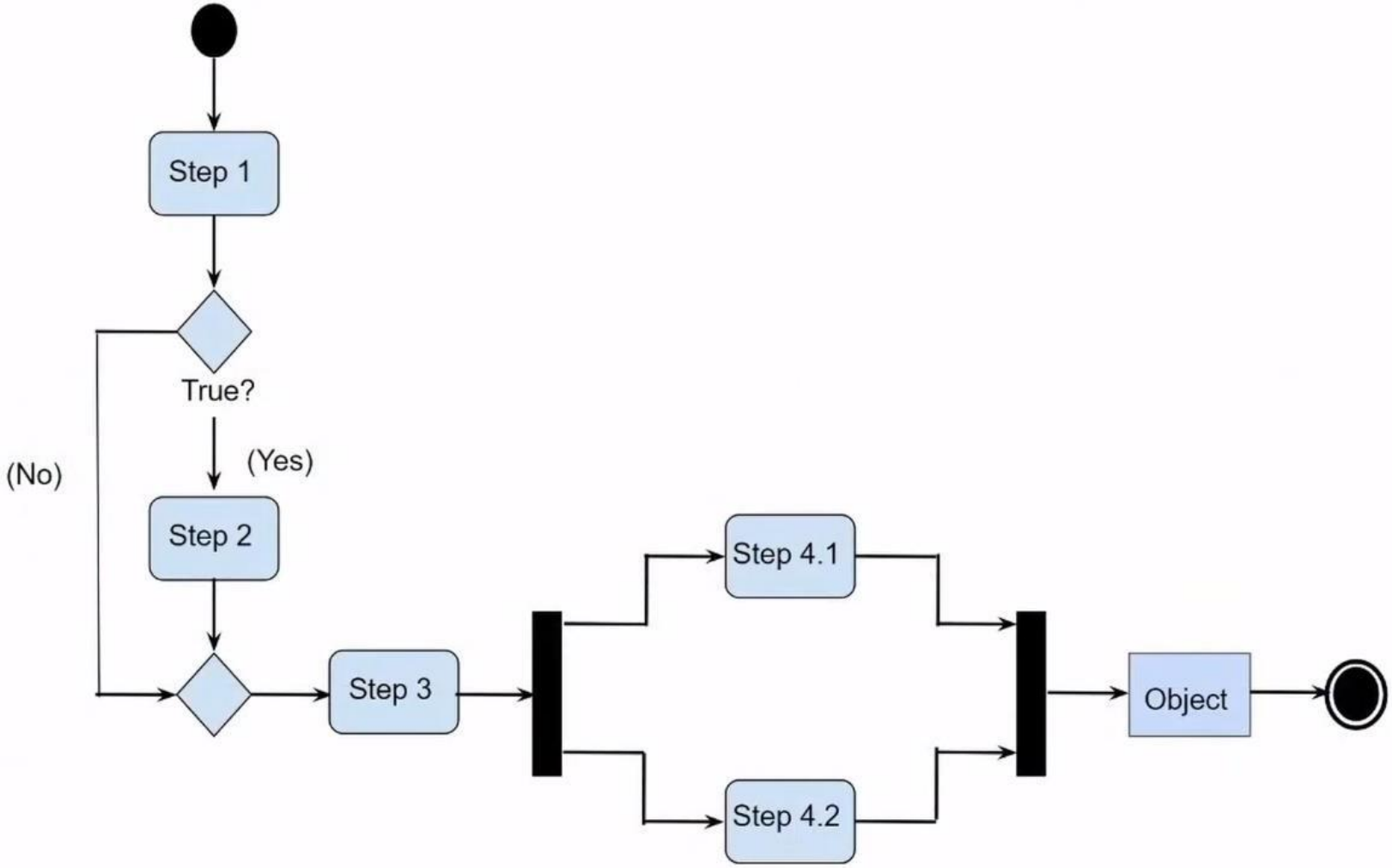
e. **State box:** It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.

Activity Diagrams – Basically a flowchart to represent the flow from one activity to another activity

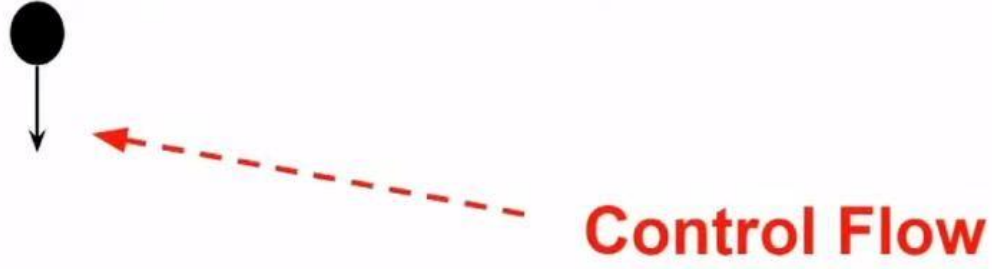


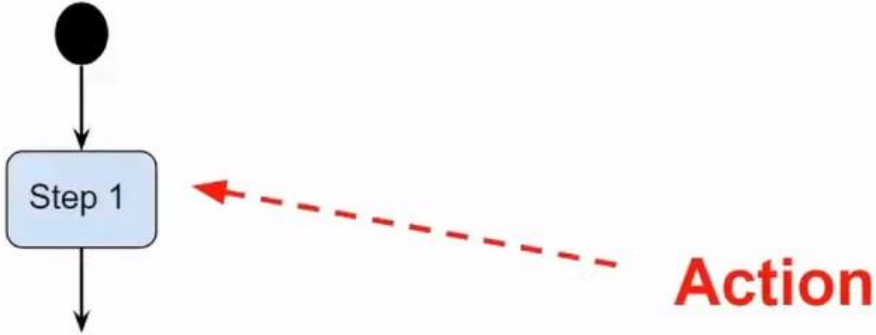
Creating a document in a text editor

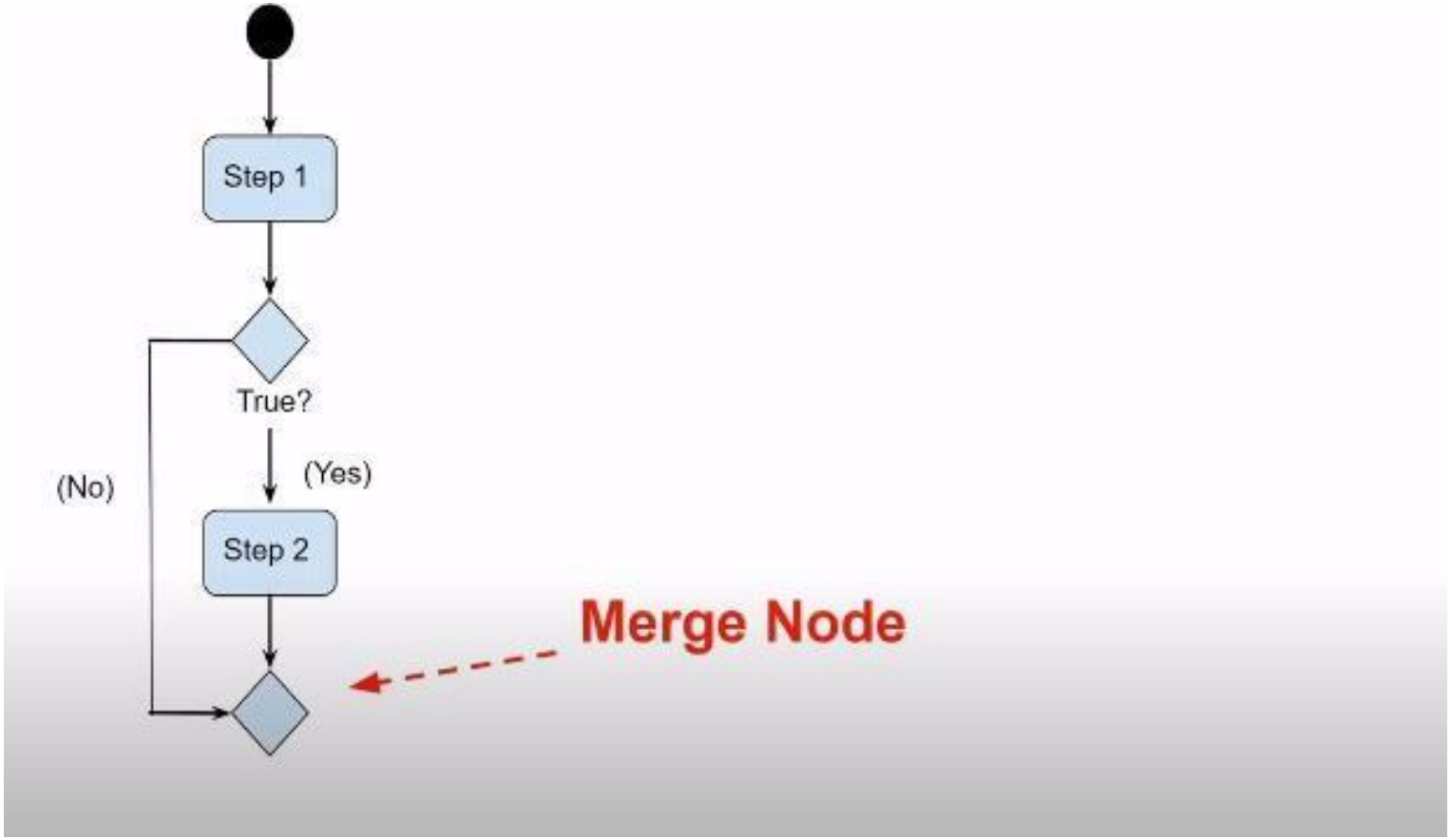

The End
Start Text Editor
Create Document
Save Document
Enter text
Pictures?
(Yes)
Import Pictures
(No)
Tables?
(Yes)
Import Tables
(No)
Closing Editor
Print Document
Save Document

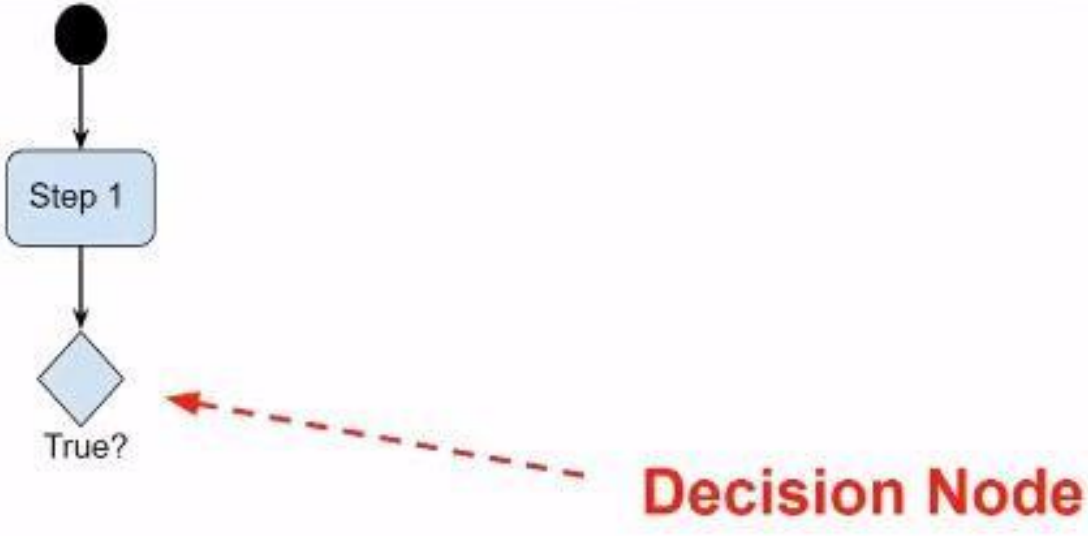

Step 1
True?
(No)
(Yes)
Step 2
Step 3
Step 4.1
Step 4.2
Object

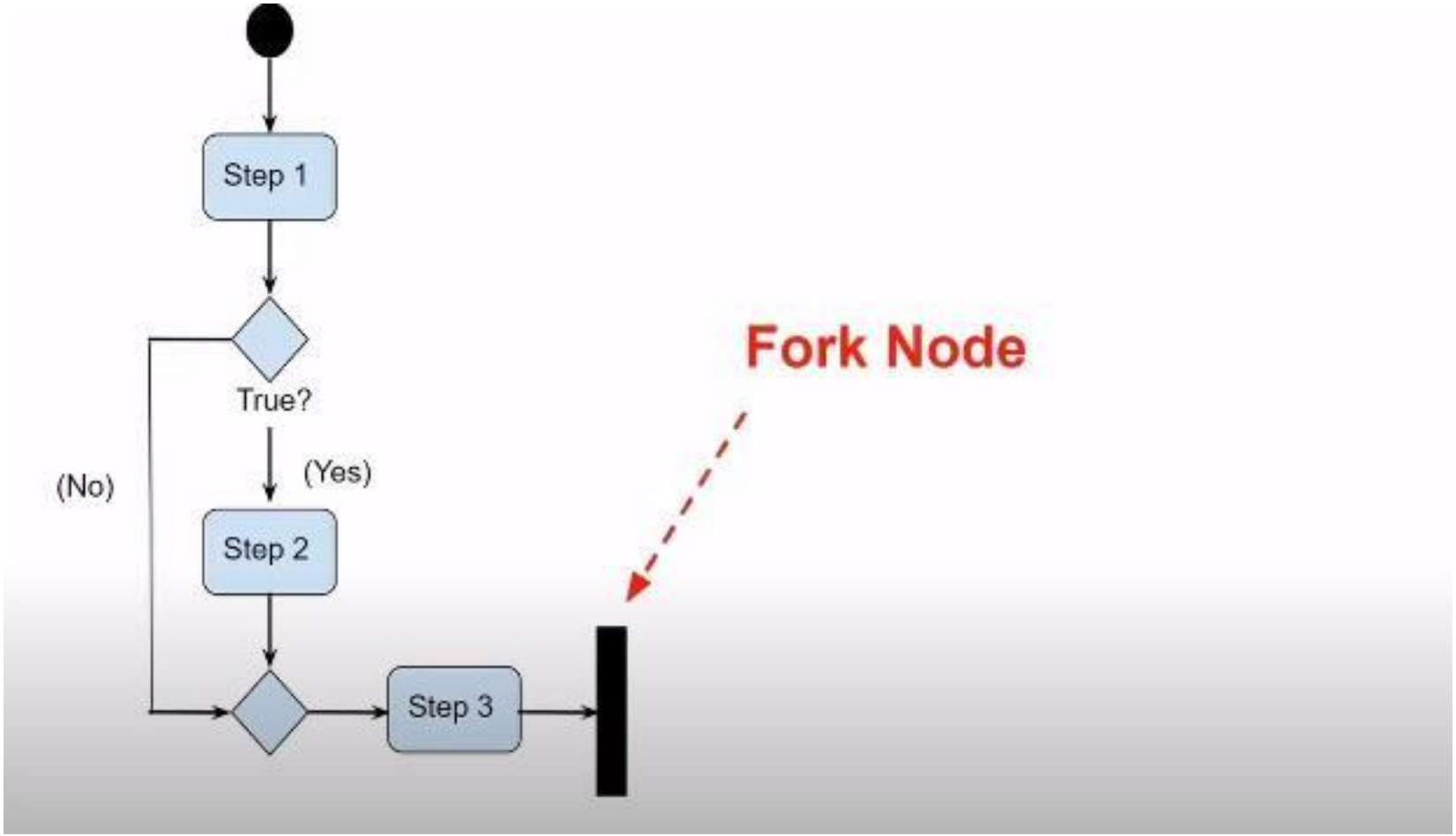

Initial Node

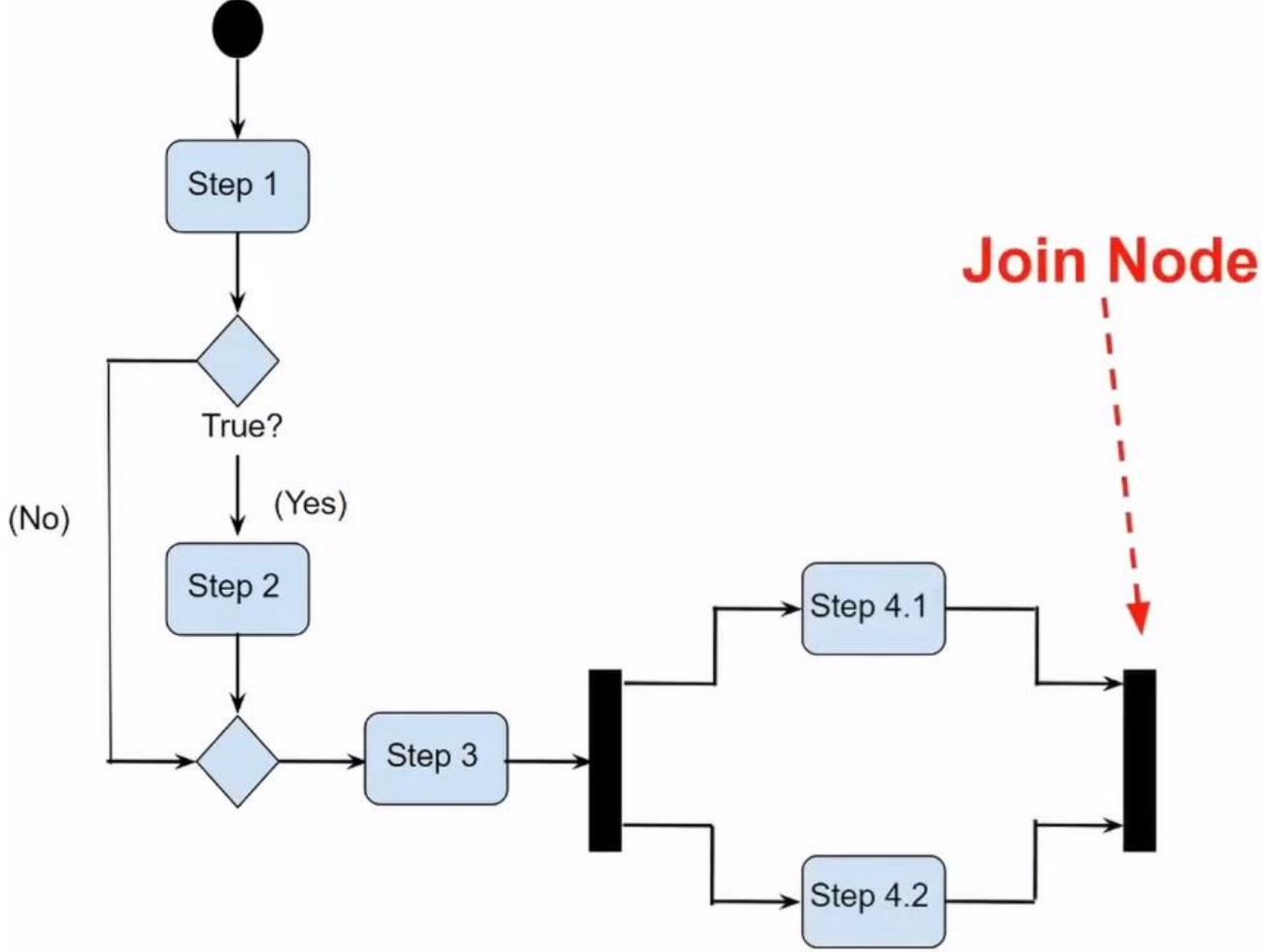

Control Flow

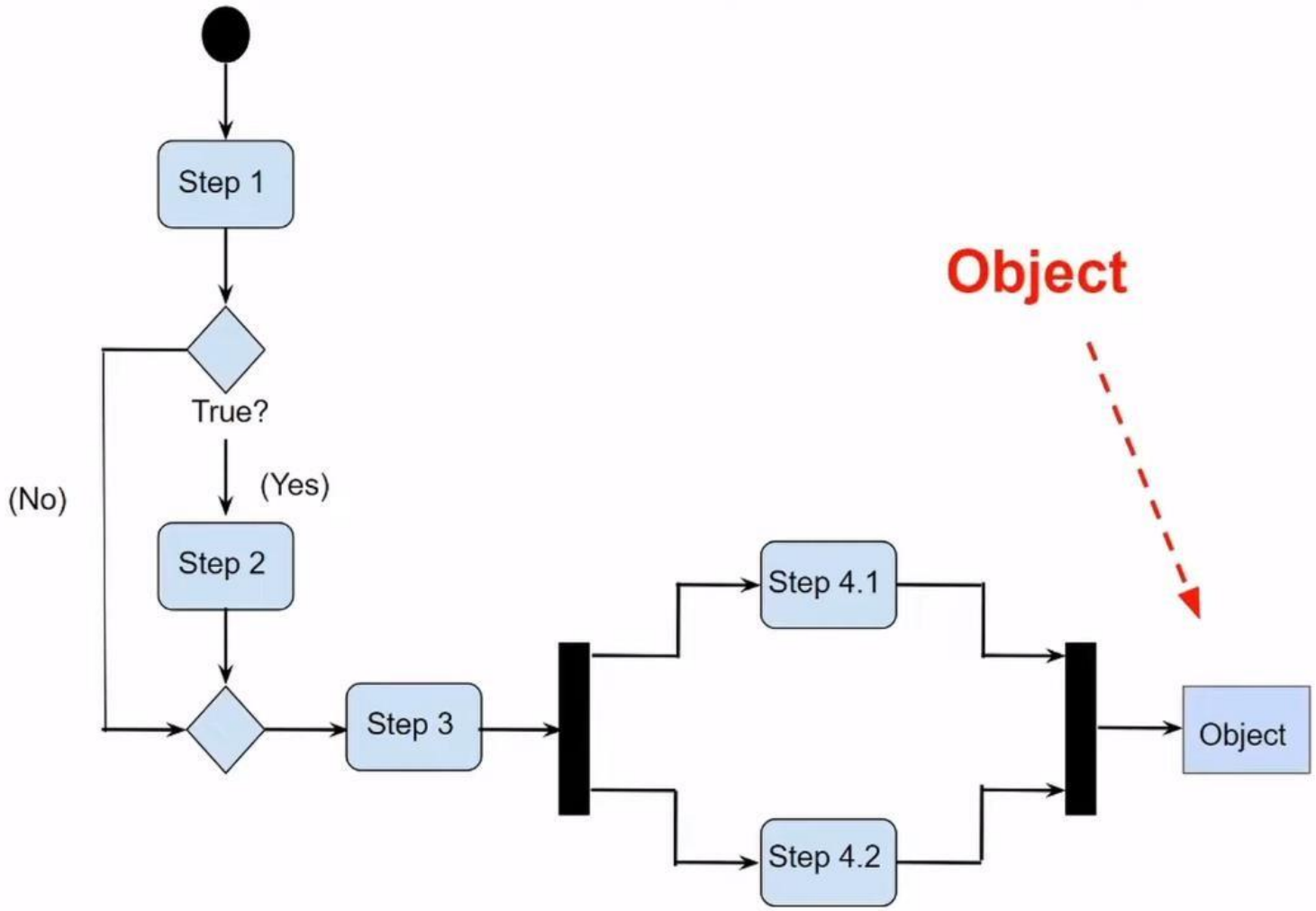

Step 1
Action


Step 1
True?
(No)
(Yes)
Step 2
Merge Node

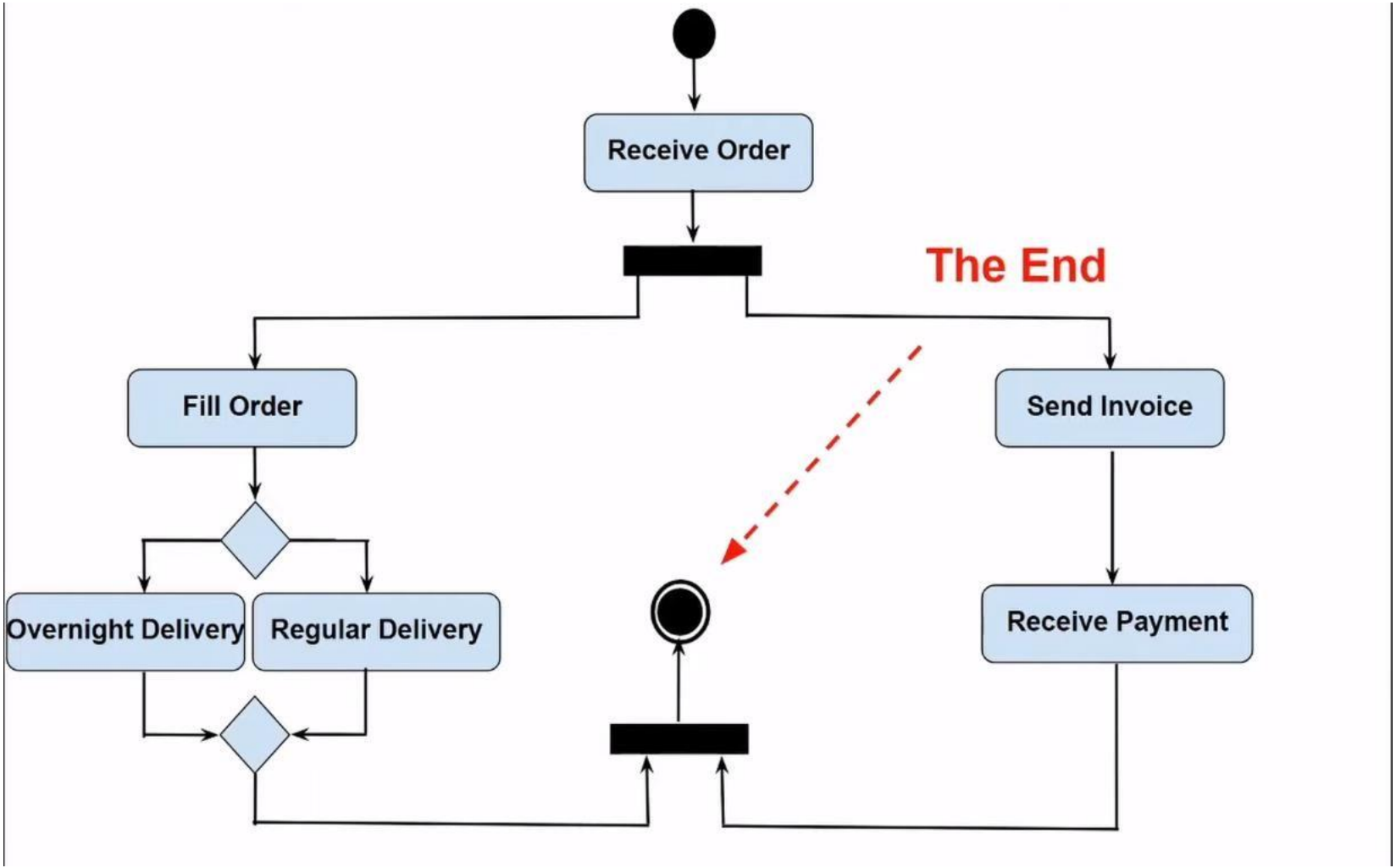

Step 1
True?
Decision Node

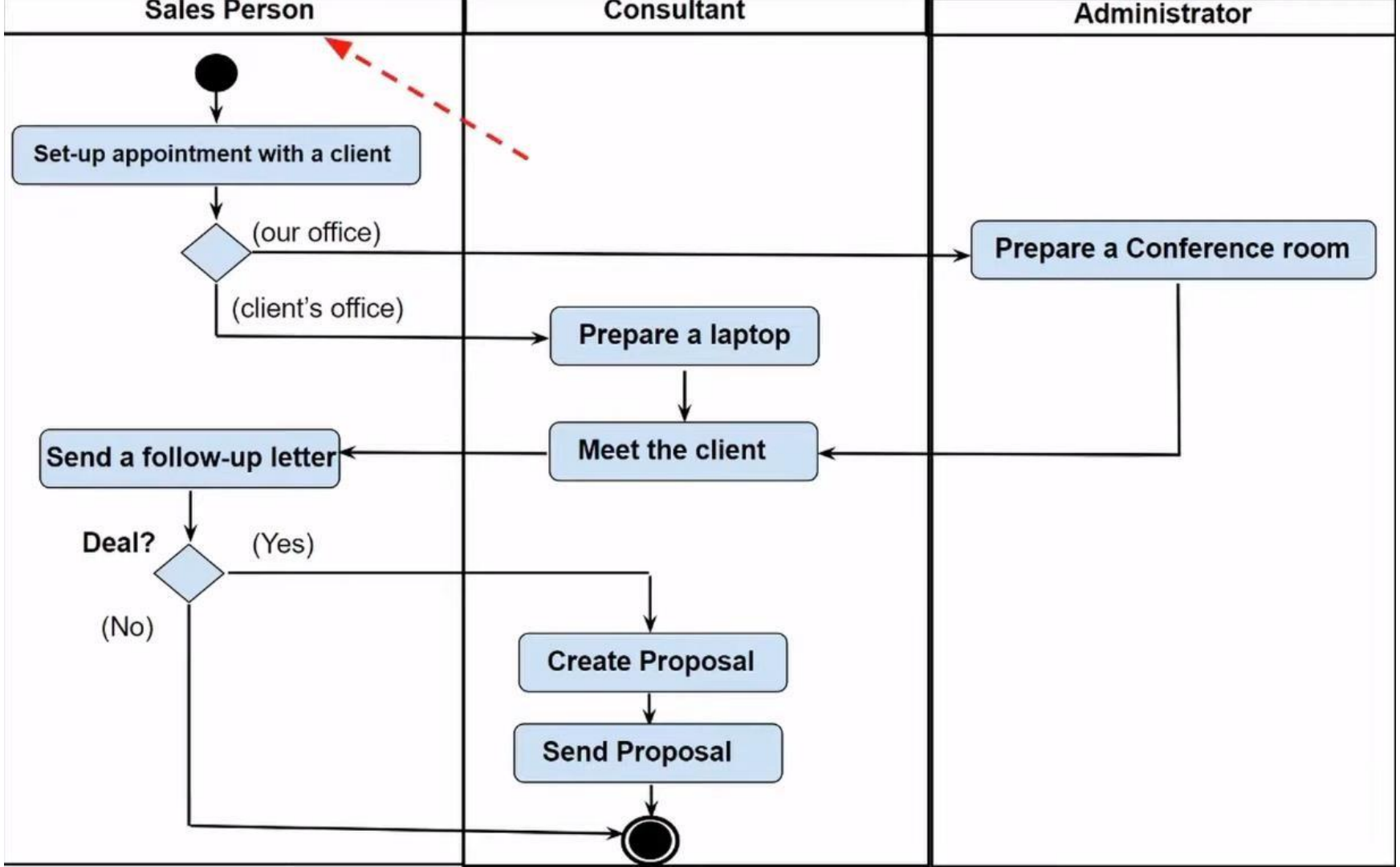

Step 1
True?
(No)
(Yes)
Step 2
Step 3
Fork Node


Step 1
True?
(No)
(Yes)
Step 2
Step 3
Step 4.1
Step 4.2
Join Node


Step 1
True?
(No)
(Yes)
Step 2
Step 3
Step 4.1
Step 4.2
Object
Object

Order Processing


Receive Order
The End
Fill Order
Send Invoice
Overnight Delivery
Regular Delivery
Receive Payment


Sales Person
Consultant
Administrator
Set-up appointment with a client
(our office)
(client's office)
Prepare a Conference room
Prepare a laptop
Meet the client
Send a follow-up letter
Deal?
(Yes)
(No)
Create Proposal
Send Proposal

NORMALIZATION

What is Normalization?

- Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divides larger tables into smaller tables and links them using relationships.
- The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

Example-1:

Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. Its decomposition into 1NF has been shown in table 2.

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Example-1:

Consider table as following below.

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

Table 1: STUD_NO, COURSE_NO

Table 2: COURSE_NO, COURSE_FEE

Table 1

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

Table 2

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

Unnormalized

<StudentProject>

StudentID	ProjectID	StudentName	ProjectName
S89	P09	Olivia	Geo Location
S76	P07	Jacob	Cluster Exploration
S56	P03	Ava	IoT Devices
S92	P05	Alexandra	Cloud Deployment

Normalization (Second Normal Form)

<StudentInfo>

StudentID	ProjectID	StudentName
S89	P09	Olivia
S76	P07	Jacob
S56	P03	Ava
S92	P05	Alexandra

<ProjectInfo>

ProjectID	ProjectName
P09	Geo Location
P07	Cluster Exploration
P03	IoT Devices
P05	Cloud Deployment

Example

Consider a relation student (rollno, game, feestructure)

Rollno	Game	Feestructure
1	Basketball	500
2	Basketball	500
3	Basketball	500
4	Cricket	600
5	Cricket	600
6	Cricket	600
7	Tennis	400

Example

Consider a relation student (rollno, game, feestructure)

Rollno	Game	Feestructure
1	Basketball	500
2	Basketball	500
3	Basketball	500
4	Cricket	600
5	Cricket	600
6	Cricket	600
7	Tennis	400

Anomalies

- The above student table is also suffering from all three anomalies –
- Insertion anomaly – A new game can't be inserted into the table unless we get a student to play that game.
- Deletion anomaly – If rollno 7 is deleted from the table we also lost the complete information regarding tennis.
- Updation anomaly –To change the fee structure for basketball we need to make changes in more than one place.

Decomposition for 3NF

To overcome these anomalies, the student table should be divided into smaller tables.

If X->Y is transitive dependency then divide R into R1(X^+) and R2(R-Y^+).

Game->feestructure is a transitive dependency [since neither game is a key nor fee is a key attribute]

R1=game+=(game, feestructure)

R2=(student-feestructure$^+$) = (rollno,game)

So divide the student table into R1(game, feestructure) and R2 (rollno, game).

R1

Rollno	Game
1	Basketball
2	Basketball
3	Basketball
4	Cricket
5	Cricket
6	Cricket
7	tennis

R2

Game	Feestructure
Basketball	500
Cricket	600
Tennis	400

INDEXING

What is Indexing?

- Primary Index
- When you create a table with a primary key or unique key, MySQL automatically creates a special index named PRIMARY . This index is called the **clustered index**. The PRIMARY index is special because the index itself is stored together with the data in the same table.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	NULL	ALL	NULL	NULL	NULL	NULL	23	10.00	Using where

- Secondary Index
- **Indexes other than the clustered index** are known as secondary indexes.
- A secondary index is **a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations**. You can retrieve data from the index using a Query , in much the same way as you use Query with a table.

DATA SANITIZATION

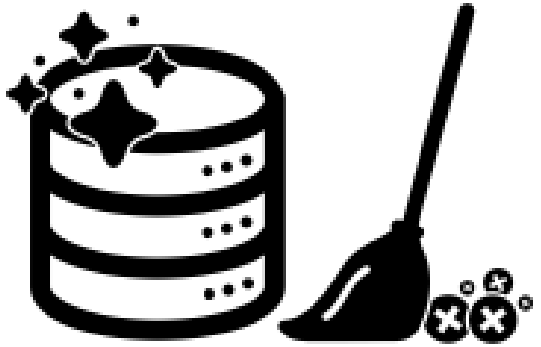
Data Sanitization

- Sanitizing data means removing any illegal character from the data.
- Code injection
- Sanitizing user input is one of the most common tasks in a web application.
- To make this task easier PHP provides native filter extension that you can use to sanitize the data such as e-mail addresses, URLs, IP addresses, etc.
- In python you can use

```
import html
input = '<>&'
output = html.escape(input)
print(output)
```

Data Sanitization - Advantages

- Many web applications receive external input. External input/data can be:
- User input from a form
- Cookies
- Web services data
- Server Variables
- Database query results



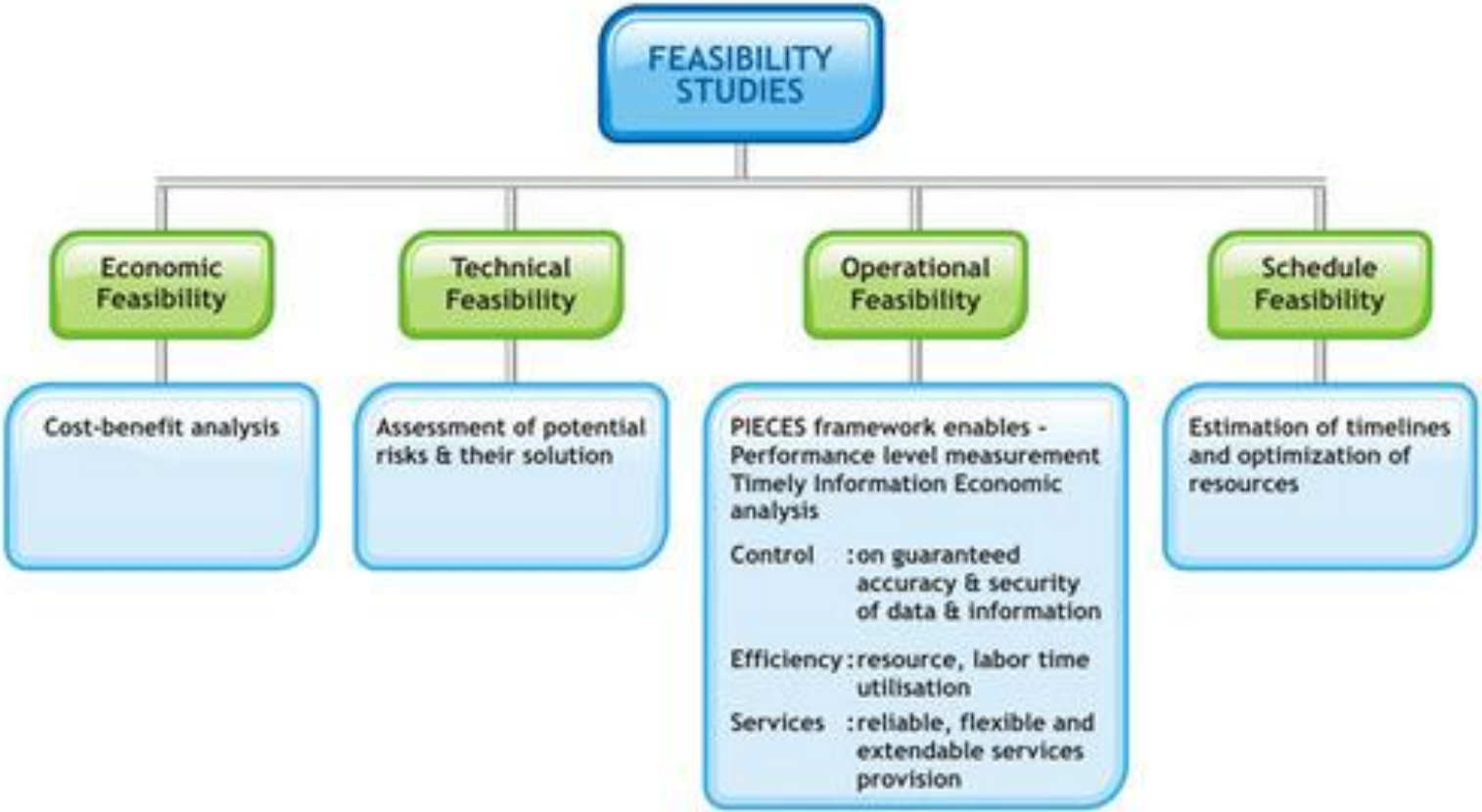
Data Sanitization

- **Sanitizing a String:** The following example uses the **filter_var()** function to remove all HTML tags from a string in php.


FEASIBILITY
STUDY
IN PROJECT MANAGEMENT

Feasibility Study

- A **feasibility study** looks at a set of criteria in order to provide decision makers with a recommendation, whether a specific solution is feasible and viable in a certain context or not.

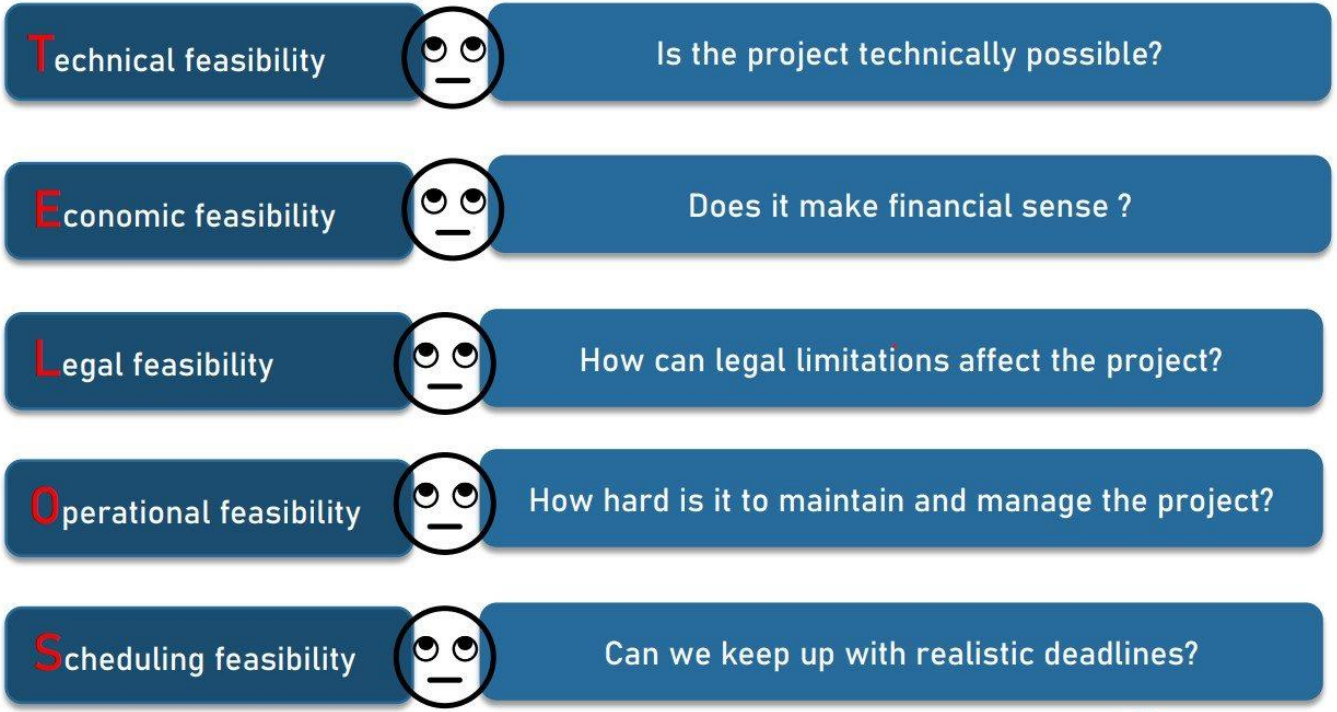

FEASIBILITY STUDIES
Economic Feasibility
Technical Feasibility
Operational Feasibility
Schedule Feasibility
Cost-benefit analysis
Assessment of potential risks & their solution
PIECES framework enables - Performance level measurement Timely Information Economic analysis
Control : on guaranteed accuracy & security of data & information
Efficiency : resource, labor time utilisation
Services : reliable, flexible and extendable services provision
Estimation of timelines and optimization of resources

Types of Feasibility

- Technical Feasibility
- Economic Feasibility
- It refers to the analysis of the cost-effectiveness of a project in order to determine whether the company should undertake the project on the basis of profitability or not.
- Example
- Operational Feasibility

▷ Technical Feasibility

TELOS framework



altexsoft

T

Technology and System Feasibility

- Do stakeholders have the expertise needed?
- Are additional resources needed in the health system including infrastructure, skills-sets or job aids?
- Is the health system ready in terms of the technology required?

E

Economic Feasibility

- Do the resources needed exist?
- Will the proposed health service or initiative lead to better use of resources to improve health outcomes, when compared with other options?

L

Legal Feasibility

- Are rules and regulations in place to enable stakeholders to support the new service or initiative?
- Does the essential political will exist?
- Is there a legal framework to engage with the private sector or other key service providers?

O

Operational Feasibility

- Do existing health system procedures and protocols support the new service or intiative?
- How will key collaborators be involved?

S

Schedule Feasibility

- What are the prerequisites before the new service or initiative can begin?
- Is the service or initiative likely to be developed in time to be useful to the health system?

Thanks!

Any questions?

You can find me at:

@NAVYA

navyamolkt@amaljyothi.ac.in