



CSE- 321

Software Engineering

Lecture: 11

System modeling and UML

Fahad Ahmed

Lecturer, Dept. of CSE

E-mail: fahadahmed@uap-bd.edu

- ✧ **Model**
- ✧ **System Modeling**
- ✧ **UML-Building Blocks**
- ✧ **UML : Diagrams**
- ✧ **Class Diagram**
- ✧ **Object Diagram**
- ✧ **Use case Diagram**
- ✧ **UML Tools**

What is model?

- ❖ Model is a **simplification of reality**.
- ❖ A **model** is an abstraction of some aspect of an existing or planned system.
- ❖ Blueprint of the actual system.
- ❖ Specify the structural and behavior of the system.
- ❖ Templates for designing the system.

System Modeling

- Modeling is a **central part** of all the activities that lead to the development of good software.

- We build models to
 - 1) To **Communicate** the desired structure and behavior of our system.
 - 2) To **visualize and control** the systems architecture.
 - 3) To **better understanding** the system we are building often exposing opportunities for simplification and reuse.
 - 4) To **manage risk**.

- Modeling is a **proven and well accepted** engineering tech.

Why Model ?

- Analyse the problem-domain
 - simplify reality
 - capture requirements
 - visualize the system in its entirety
 - specify the structure and/or behaviour of the system
- Design the solution
 - document the solution - in terms of its structure, behaviour, etc.

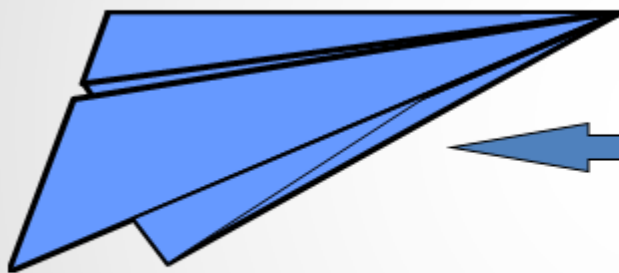
Why Model ?

- Modeling achieves four aims:
 - Helps you to **visualize a system** as you want it to be.
 - Permits you to **specify the structure or behavior of a system**.
 - Gives you a **template that guides you in constructing** a system.
 - **Documents the decisions** you have made.
- You build models of complex systems because you cannot comprehend such a system in its entirety.
- You build models to better understand the system you are developing.

What are the advantages of creating a model?

Advantages:

- Provides standard for software development.
- Reducing of costs to develop diagrams of UML using supporting tools.
- Development time is reduced.
- The past faced issues by the developers are no longer exists.
- Has large visual elements to construct and easy to follow.



Paper Airplane



Fighter Jet

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.

Unified Modeling Language (UML)

✧ The **UML** stands for **Unified modeling language**.

✧ The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

the artifacts of a software-intensive system.



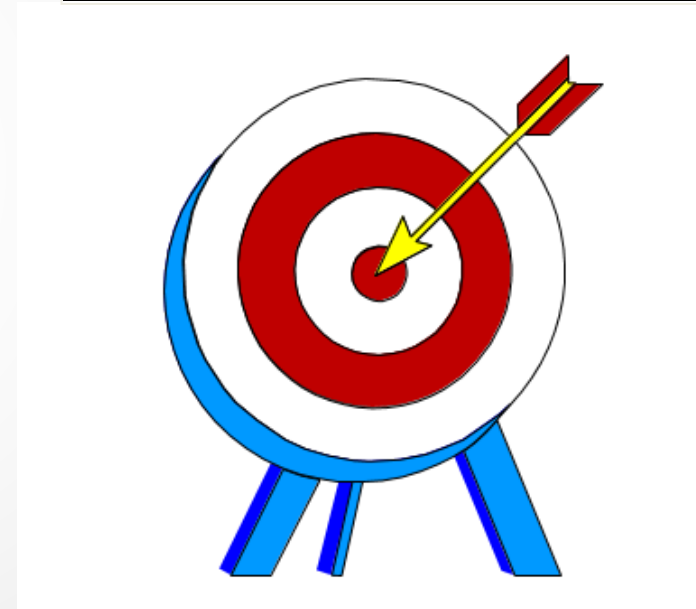
✧ The Unified Modelling Language (UML) is an industry standard for object oriented design notation.

The Unified Modeling Language User Guide SECOND (2.0) EDITION

<http://umlguide2.uw.hu/index.html>

Unified Modeling Language (UML)

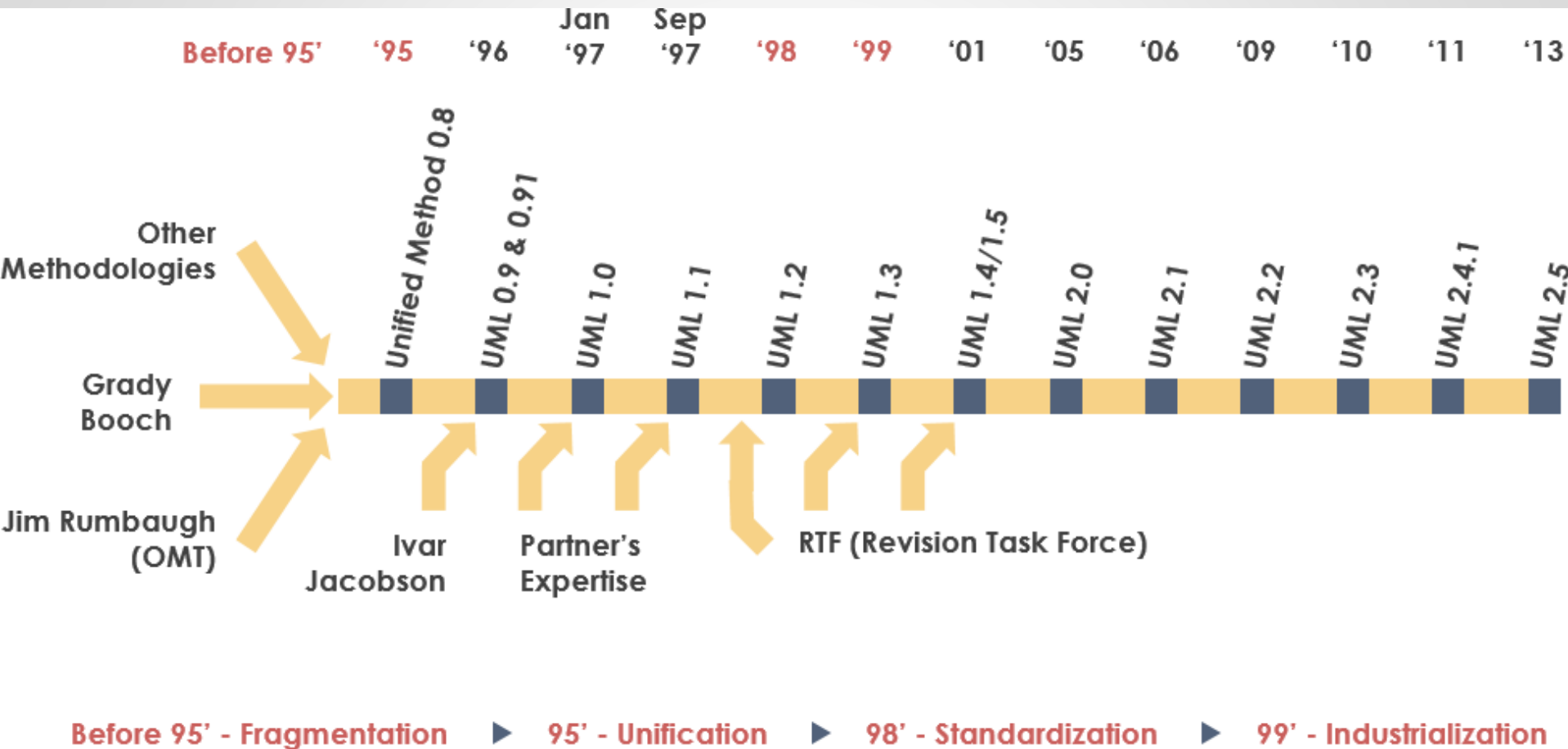
- ✧ UML is **not a programming language**, it is rather a visual language.
- ✧ We use UML diagrams to portray the **behavior and structure** of a system.
- ✧ The UML builds models that are precise, unambiguous, and complete.
- ✧ UML helps software engineers, businessmen and system architects with modelling, design and analysis.



Unified Modeling Language (UML)

- ✧ The UML was developed in 1994-95 by Grady Booch, Ivar Jacobson, and James Rumbaugh at the Rational Software.
- ✧ In 1997, it got adopted as a standard by the **Object Management Group (OMG)**. This was the first version (V1.1) of UML. The OMG is best recognized for the Common Object Request Broker Architecture (**CORBA**) standards.
- ✧ **International Organization for Standardization (ISO)** published UML as an approved standard in 2005.

Unified Modeling Language (UML)



Characteristics of UML

The UML has the following features:

- ✧ It is a generalized modeling language.
- ✧ It is distinct from other programming languages like C++, Python, etc.
- ✧ It is interrelated to object-oriented analysis and design.
- ✧ It is used to visualize the workflow of the system.
- ✧ It is a pictorial language, used to generate powerful modeling artifacts.

Where Can the UML Be Used?

The UML is intended primarily for software-intensive systems. It has been used effectively for such domains as

- Enterprise information systems
- Banking and financial services
- Telecommunications
- Transportation
- Defence/aerospace
- Retail
- Medical electronics
- Scientific
- Distributed Web-based services

The UML is not limited to modeling software. In fact, it is expressive enough to model non-software systems, such as workflow in the legal system, the structure and behavior of a patient healthcare system, software engineering in aircraft combat systems, and the design of hardware.

Basics of the conceptual model

Some object-oriented concepts that are needed to begin with UML:

Object: An object is a real world entity. There are many objects present within a single system. It is a fundamental building block of UML.

Class: A class is a software blueprint for objects, which means that it defines the variables and methods common to all the objects of a particular type.

Abstraction: Abstraction is the process of portraying the essential characteristics of an object to the users while hiding the irrelevant information. Basically, it is used to envision the functioning of an object.

Basics of the conceptual model

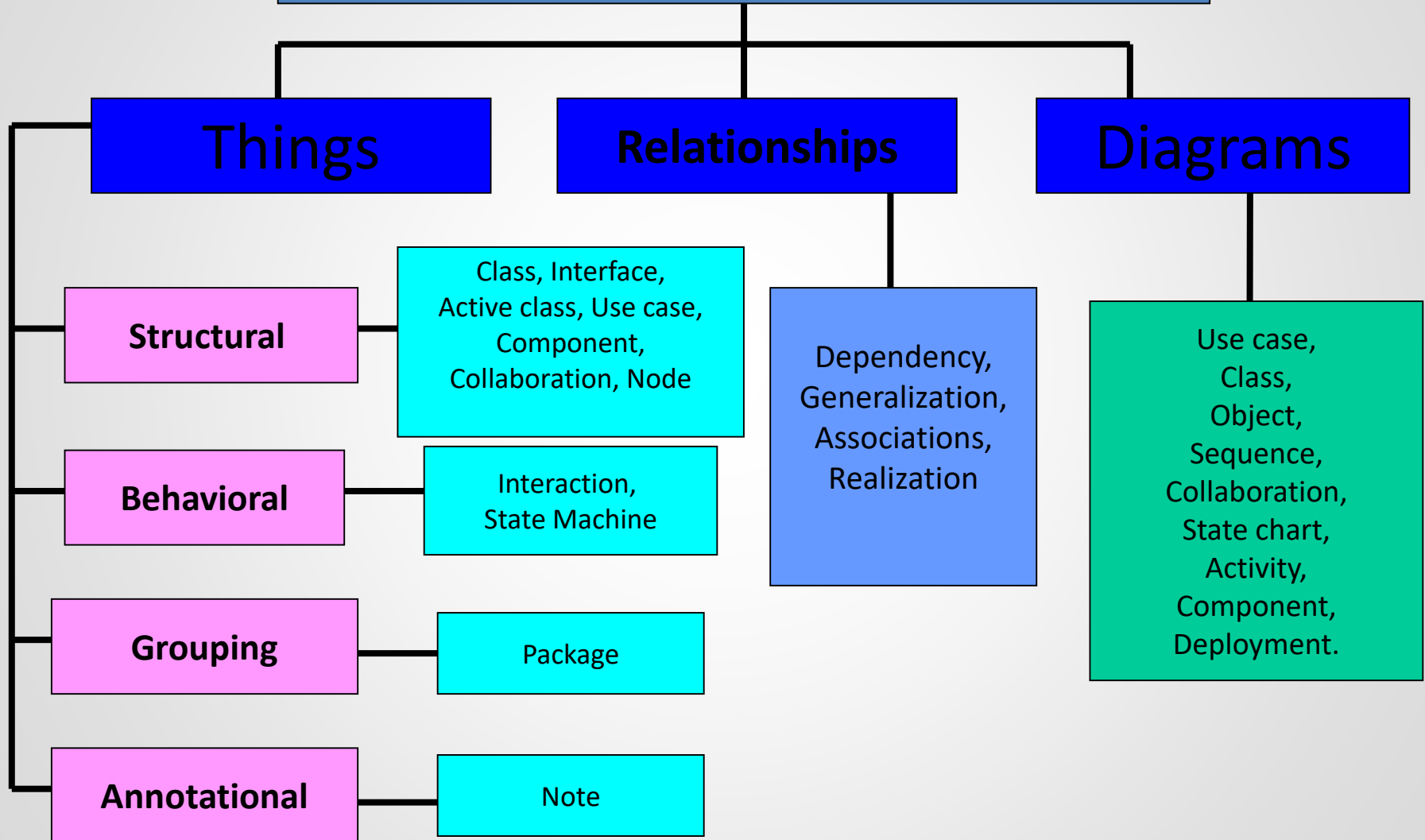
Inheritance: Inheritance is the process of deriving a new class from the existing ones.

Polymorphism: It is a mechanism of representing objects having multiple forms used for different purposes.

Encapsulation: It binds the data and the object together as a single unit, enabling tight coupling between them.

UML-Building Blocks

Building Blocks of UML



UML-Building Blocks: Structural Things

There are 4 kinds of things in the UML

1. Structural Things
2. Behavioral Things
3. Grouping Things
4. Annotational Things

Structural Things

➤ Structural things are the **nouns of UML models**. These are mostly **static** parts of a model, representing elements that are either conceptual or physical.

There are 7 kinds of Structural things:

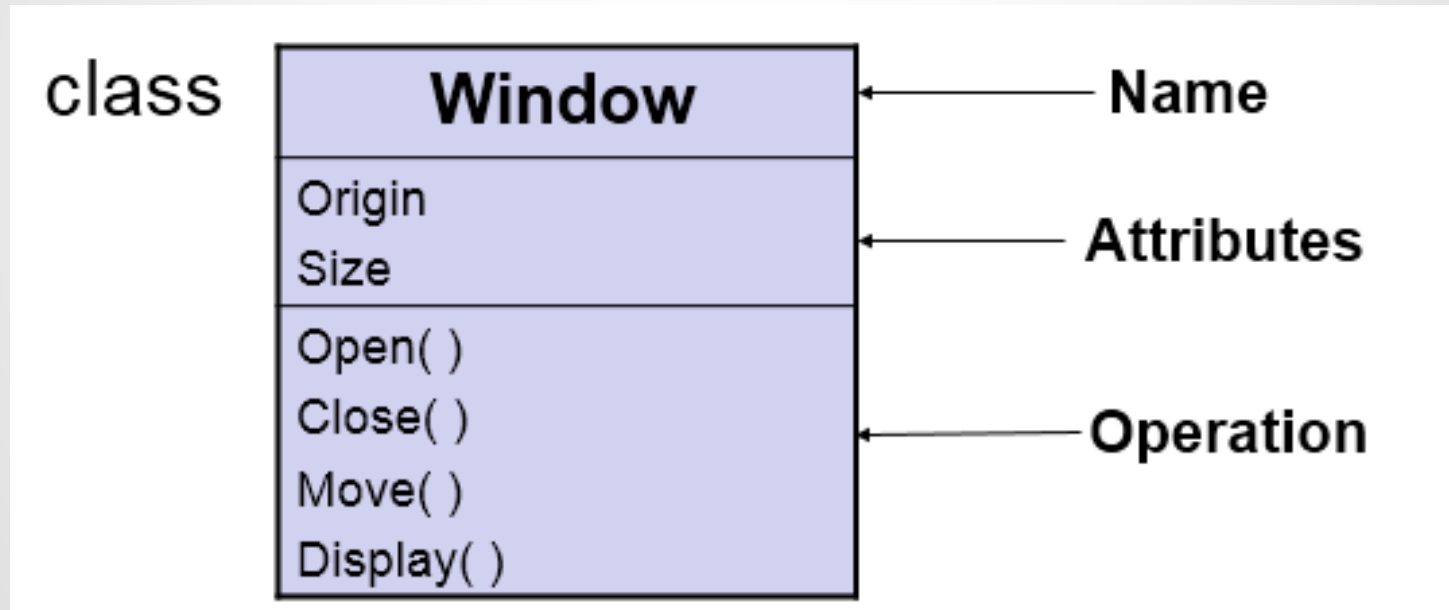
- | | |
|-----------------|-------------|
| ▪ Class | ▪ Actor |
| ▪ Interface | ▪ Component |
| ▪ Collaboration | ▪ Node |
| ▪ Use case | |

UML-Building Blocks: Structural Things

(1) Class:-

A class is a **description of a set of objects** that share the same attributes, operations, relationships and semantics.

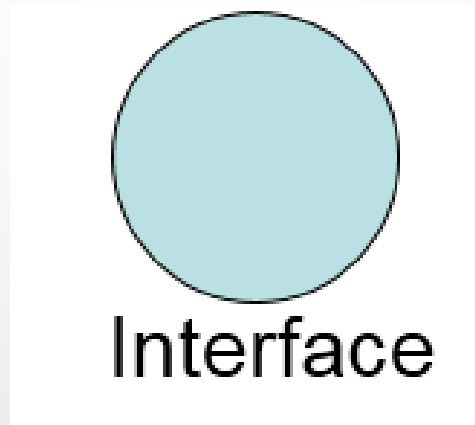
A class implements one or more interfaces.



UML-Building Blocks: Structural Things

(2) Interface:-

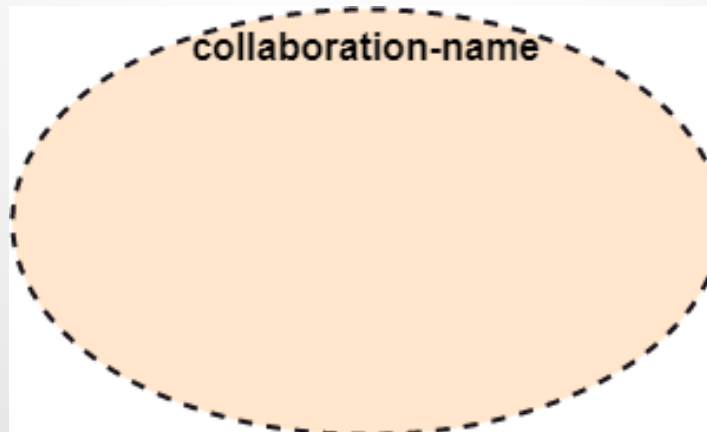
- Interface is a collection of operations that specify a service of a class or component.
- An interface describes the externally visible behavior of that element.
- An interface defines a set of operation specifications but never a set of operation implementations.
- Graphically, an interface is represented as a circle with its name.



UML-Building Blocks: Structural Things

(3) Collaboration:-

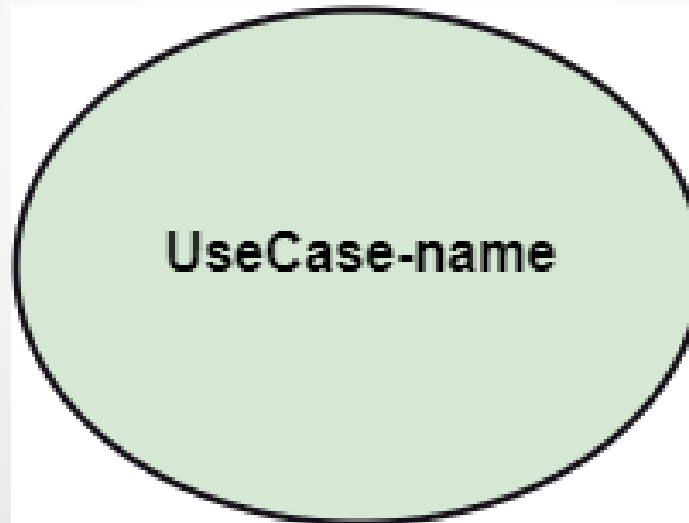
- A collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior.
- Therefore, collaboration have structural as well as behavioral dimensions.
- Graphically a collaboration is represented as an ellipse with dashed lines, usually including only its name.



UML-Building Blocks: Structural Things

(4) Use case:-

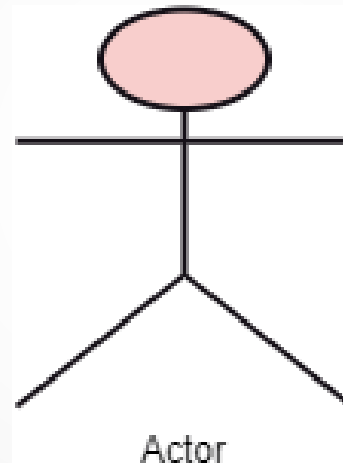
- A Use case is a **description of set of sequence of actions** that a system performs, which results a value to a particular actor.
- A use case is **used to structure the behavioral things** in a model.
- A use case is realized by a collaboration.



UML-Building Blocks: Structural Things

(5) Actor:-

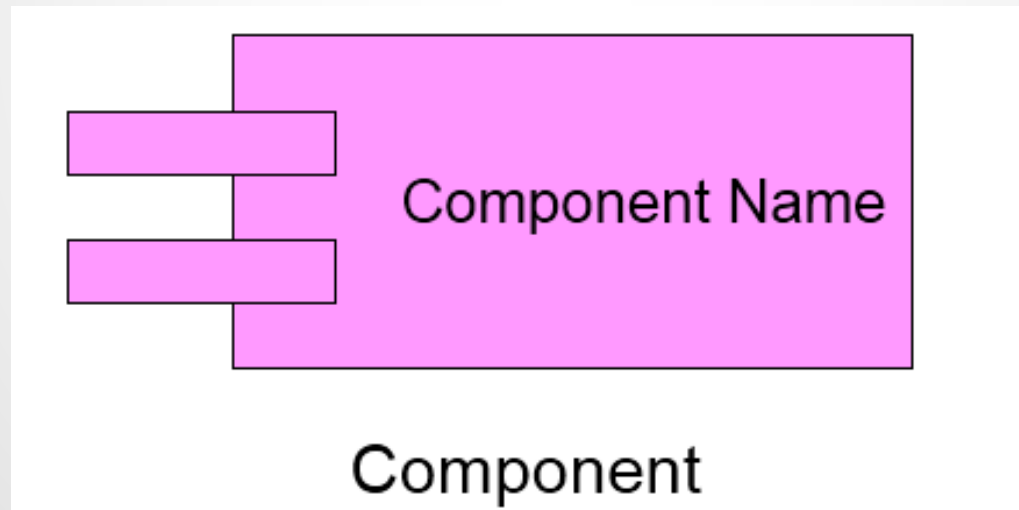
- Actor: It comes under the use case diagrams. It is an object that interacts with the system, for example, a user.



UML-Building Blocks: Structural Things

(6) Component:-

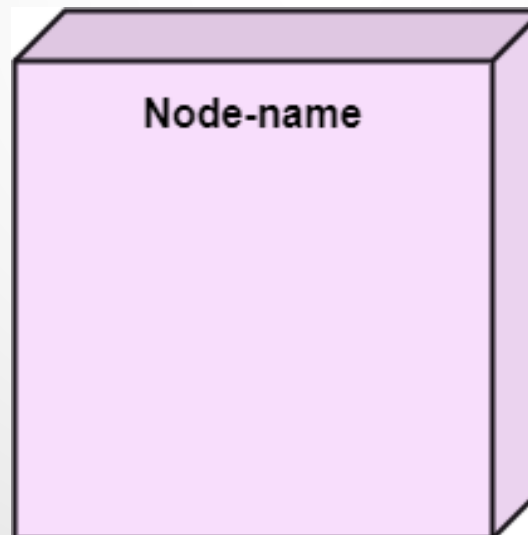
- A component is a **physical and replaceable part** of a system that conforms to and provides the **realization of a set of interfaces**.
- A **component represents** the **physical packaging of logical elements** such as classes, interfaces and collaborations.



UML-Building Blocks: Structural Things

(7) Node:-

- A node is **physical element** that **exists at run-time** and represents a computational resource, generally **having some memory and processing capability**.
- A set of components may reside on a node and may also migrate from node to node.



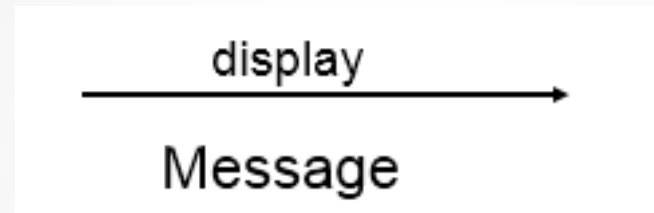
UML: Behavioral Things

- These are the **dynamic parts of UML** models.
- These are the **verbs of a model** representing the behavior over time and space.
- There are two types of behavioral things
 1. Interaction
 2. State machine.

UML: Behavioral Things

(1) Interaction:-

- An Interaction is a **behavior** that comprises a set of messages exchanged among a set of objects within a particular context to perform a specific purpose.

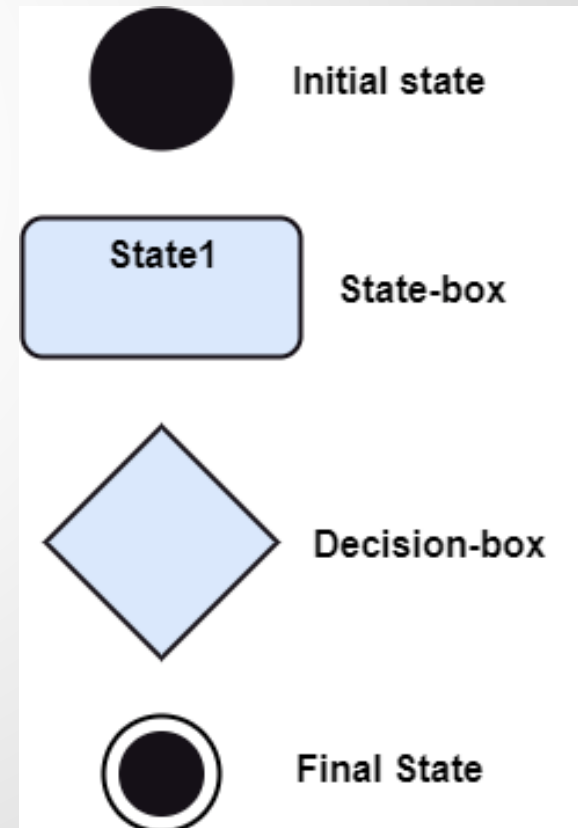


- An interaction involves a number of other elements, including messages, action sequences and links. (connection between objects.)

UML: Behavioral Things

(2) State machine:-

- A State machine is a behavior that specifies the **sequences of states** an object or an interaction goes through during its **lifetime** in response to events, together with its responses to those events.
- A State machine involves a number of other elements, including **states**, **transitions** (the flow from state to state), **events** (things that trigger a transition), and **activities** (the response to a transition).

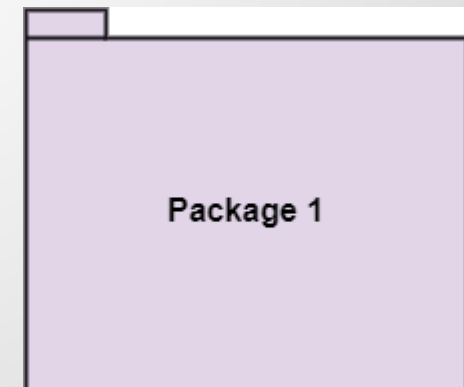


UML: Grouping Things

- Grouping things are the **organizational** parts of UML models.
- In all, there is one primary kind of grouping thing, namely, **Package**.

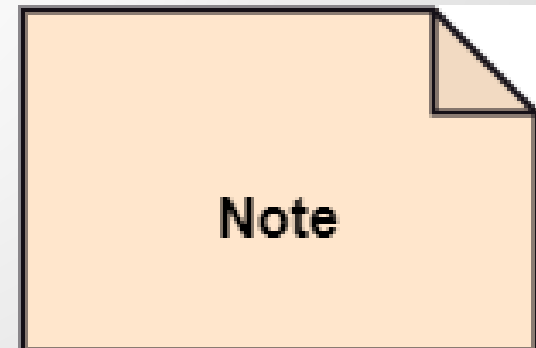
Package:-

- A Package is a general-purpose mechanism for **organizing elements into groups**.
- Structural things, behavioral things and even other grouping things that may be placed in a package.
- Unlike **component** (which exist at run time), **package is purely conceptual** (meaning that it exist only at development time).



UML: Annotational Things

- Annotational Things are the **explanatory parts of the UML models**.
- These are the **comments** you may apply to describe, illuminate and remark about any element in a model.
- There is one primary kind of Annotational thing, called a **Note**.
- **Note:-** A Note is simply a **symbol for representing constraints and comments** attached to an element or a collection of elements.
- A note is rendered as a rectangle with a **dog-eared corner**.



Relationships in the UML

There are **four kinds** of relationships in the UML.

1. Dependency
2. Association
3. Generalization
4. Realization

These relationships are the **basic relational building blocks** of the UML. **These are used to write well-formed models.**

Relationships in the UML

DEPENDENCY:-

- A Dependency is a semantic relationship between two things in which a change to one thing (the independent thing) may affect the semantics of the other thing (the dependent thing).

- - - - Dependency - - ->

ASSOCIATION:-

- A set of links that associates the entities to the UML model. It tells how many elements are actually taking part in forming that relationship.
- It is denoted by a dotted line with arrowheads on both sides to describe the relationship with the element on both sides.

← - - Association - - - →

GENERALIZATION:-

- A generalization is a specialization/generalization relationship in which objects of the specialized elements (the child) are substitutable for objects of the generalized elements (the parent).
- In this way, the child shares the structure and the behavior of the parent.
- Represented as a solid line with a hollow arrow head pointing to the parent.



REALIZATION:-

- A realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carryout.
- We will encounter realization relationships in two places: between interfaces and classes or components that realize them, and between use cases and the collaborations that realize them.

- - - - - Realization - - - 

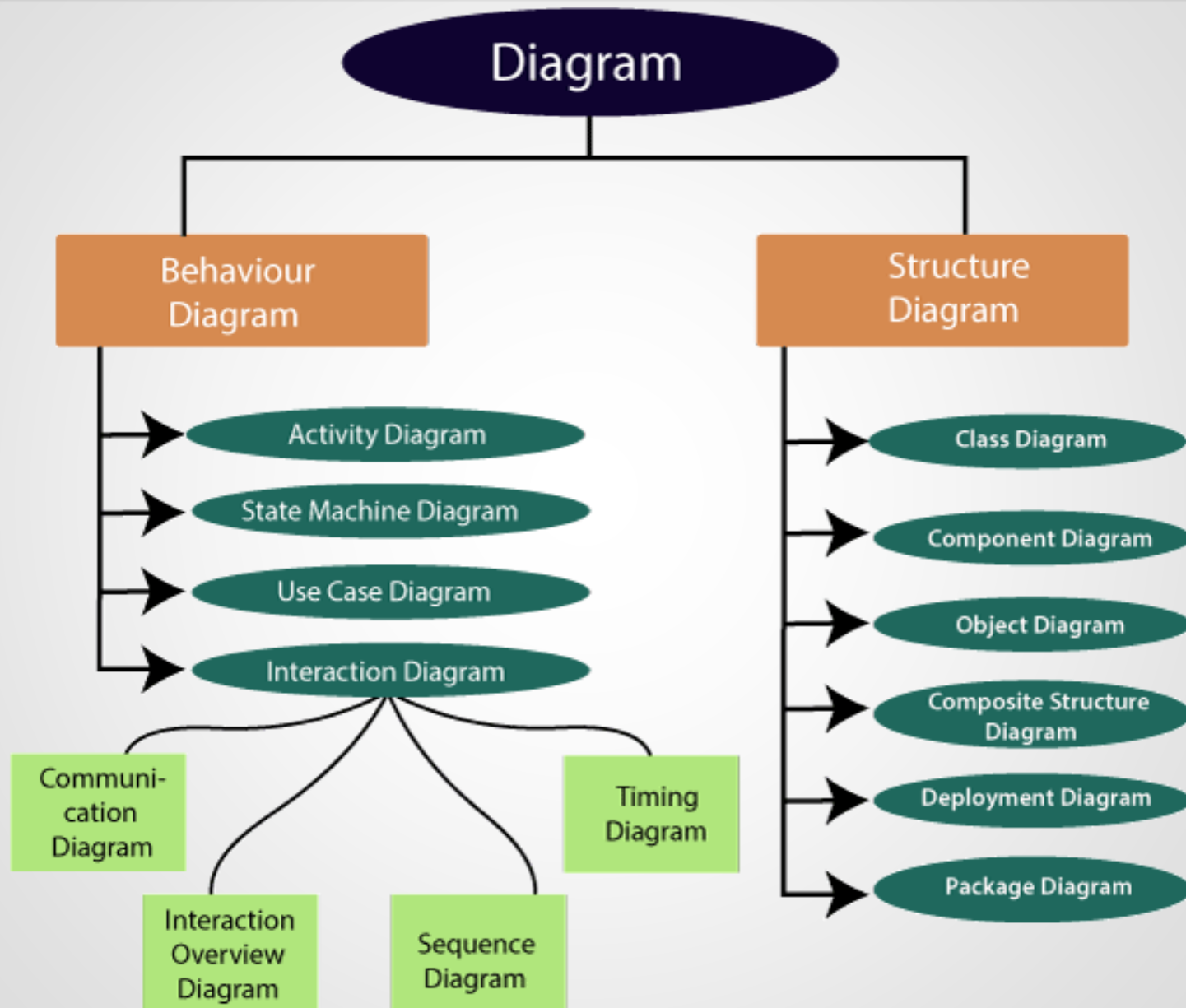
UML : Diagrams

- A diagram is the **graphically presentation** of a set of elements, most often represented as a connected **graph of vertices (things) and arcs (relationships)**.
- Diagrams **visualize a system** from different perspectives, so a diagram is a projection into system.
- **UML** is linked with **object oriented** design and analysis.

Diagrams in UML can be broadly classified as:

- ❖ **Structural Diagrams**
- ❖ **Behavior Diagrams**

UML : Diagrams



UML : Class Diagram

UML Class Diagram

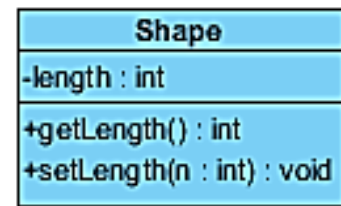
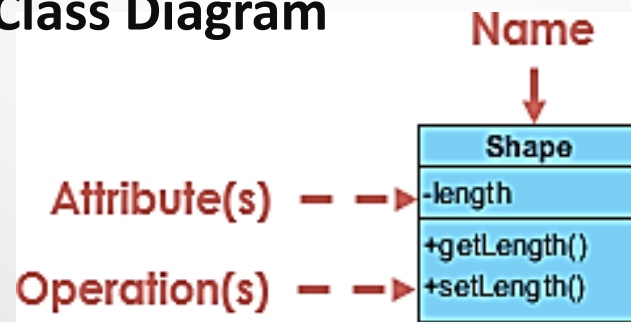
The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them.

Purpose of Class Diagrams

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

Vital components of a Class Diagram

- ❖ Upper Section
- ❖ Middle Section
- ❖ Lower Section



Class without signature

Class with signature

Cardinality

Cardinality is expressed in terms of:

- one to one
- one to many
- many to many

How to draw a Class Diagram?

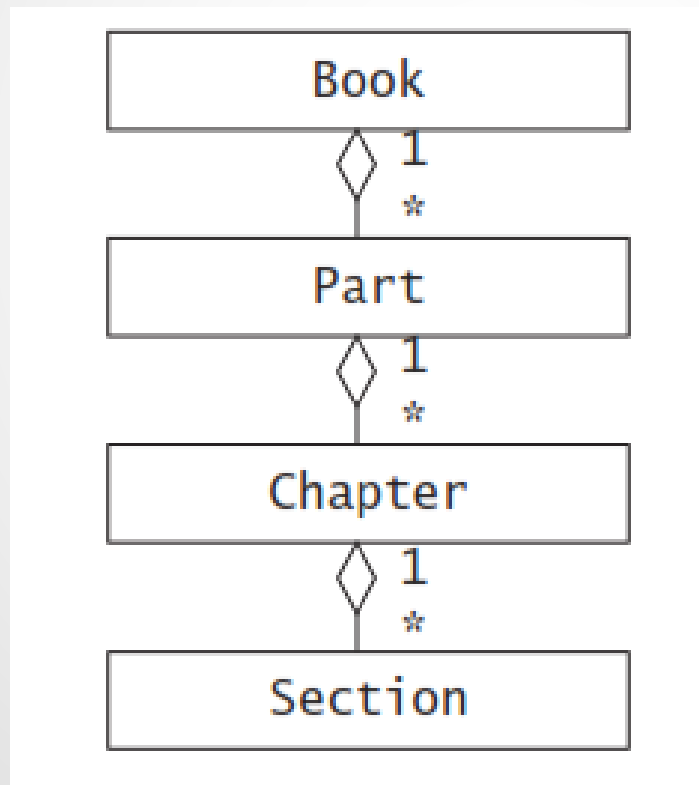
1. To describe a complete aspect of the system, it is suggested to give a meaningful name to the class diagram.
2. The objects and their relationships should be acknowledged in **advance**.
3. The attributes and methods (responsibilities) of each class must be **known**.
4. A minimum number of desired 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.

UML : Class Diagram

Draw a class diagram representing a book defined by the following statement:

“A book is composed of a number of parts, which in turn are composed of a number of chapters. Chapters are composed of sections.”

Focus only on classes and relationships.

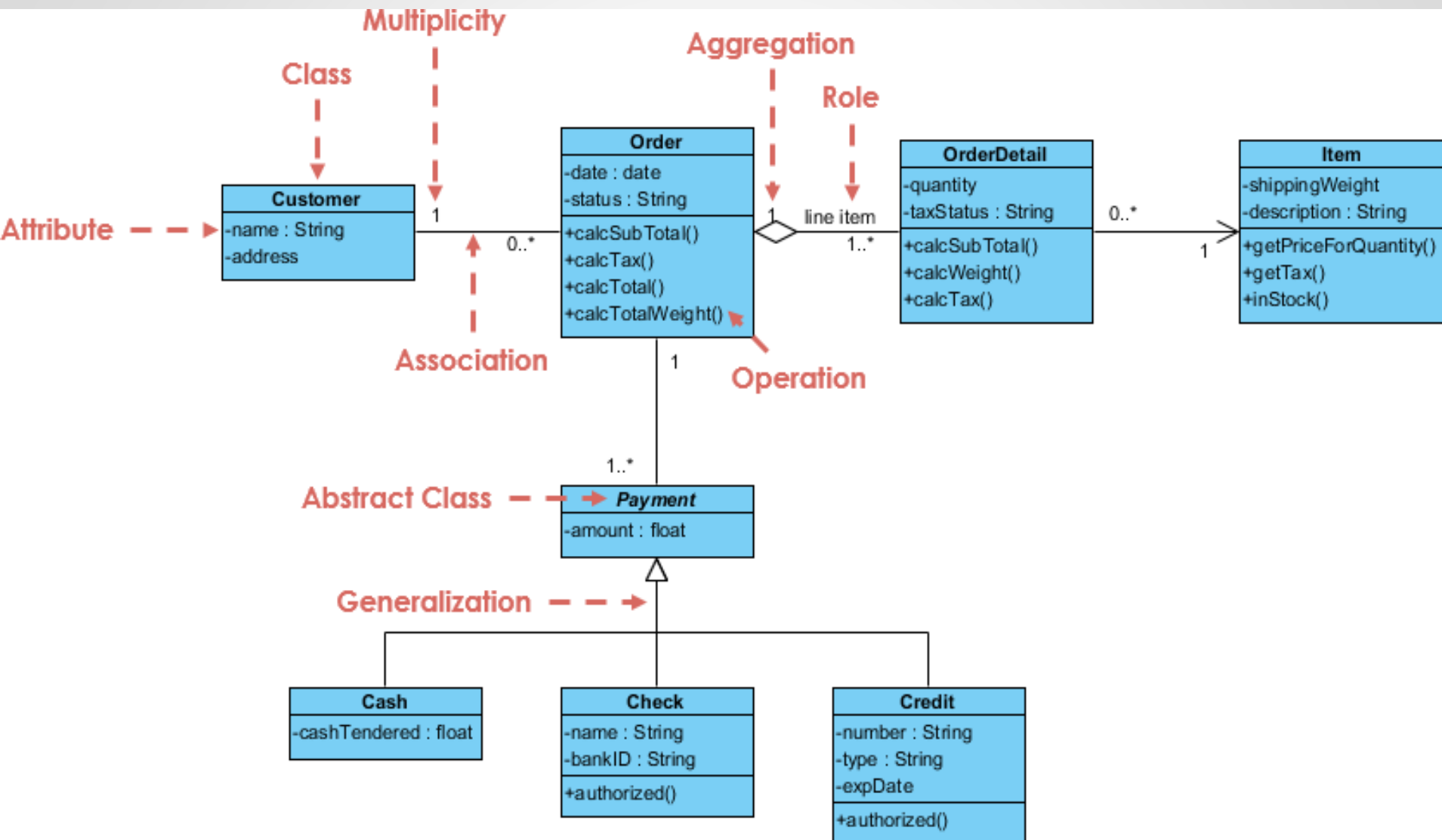


- Classes are represented with rectangles.
- The attribute and operations compartment can be omitted.
- **Aggregation** relationships are represented with diamonds.
- Class names start with a capital letter and are singular.

UML : Class Diagram

Class Diagram Example: Order System

<https://www.uml-diagrams.org/class-diagrams-examples.html>



UML Object Diagram

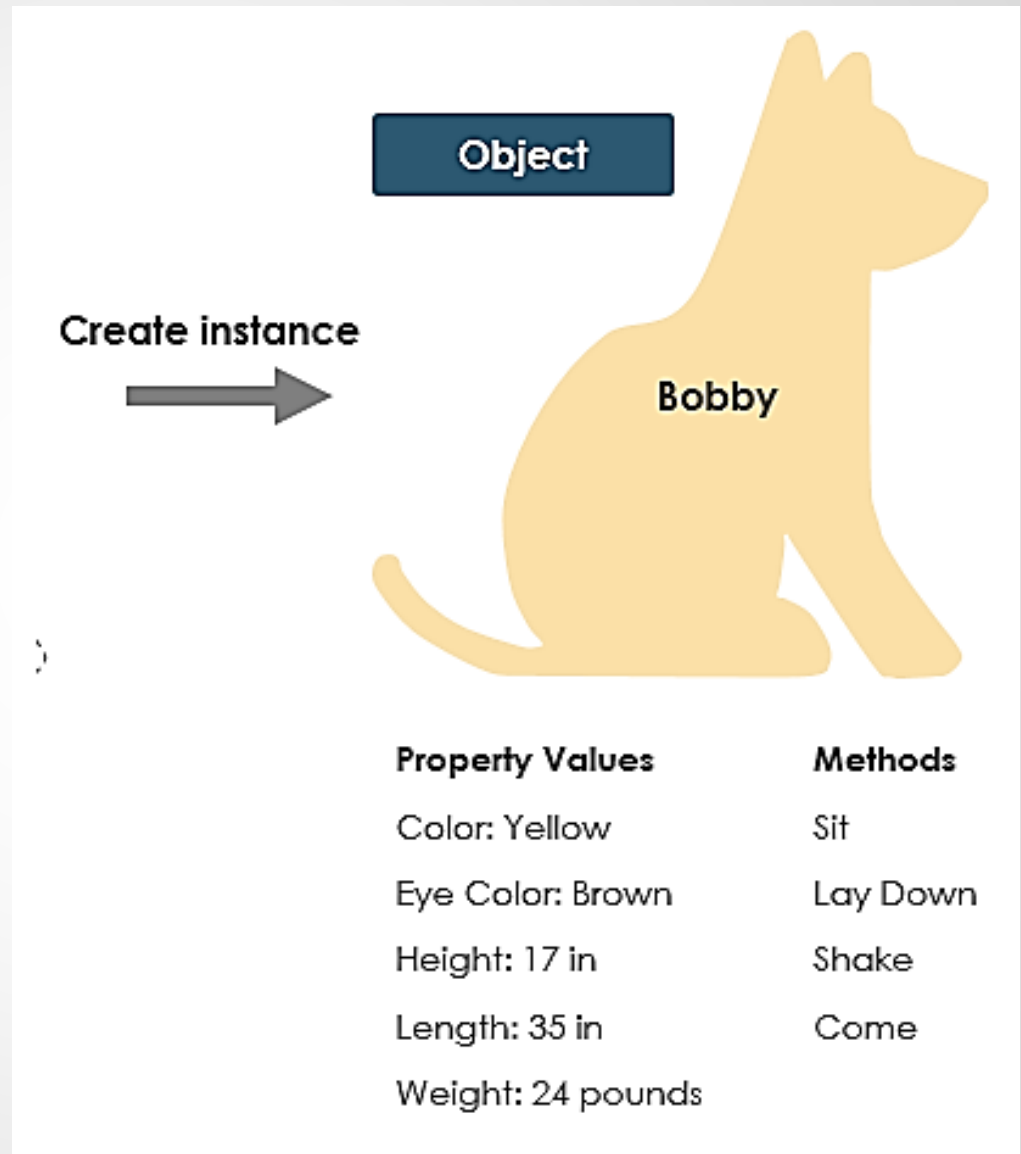
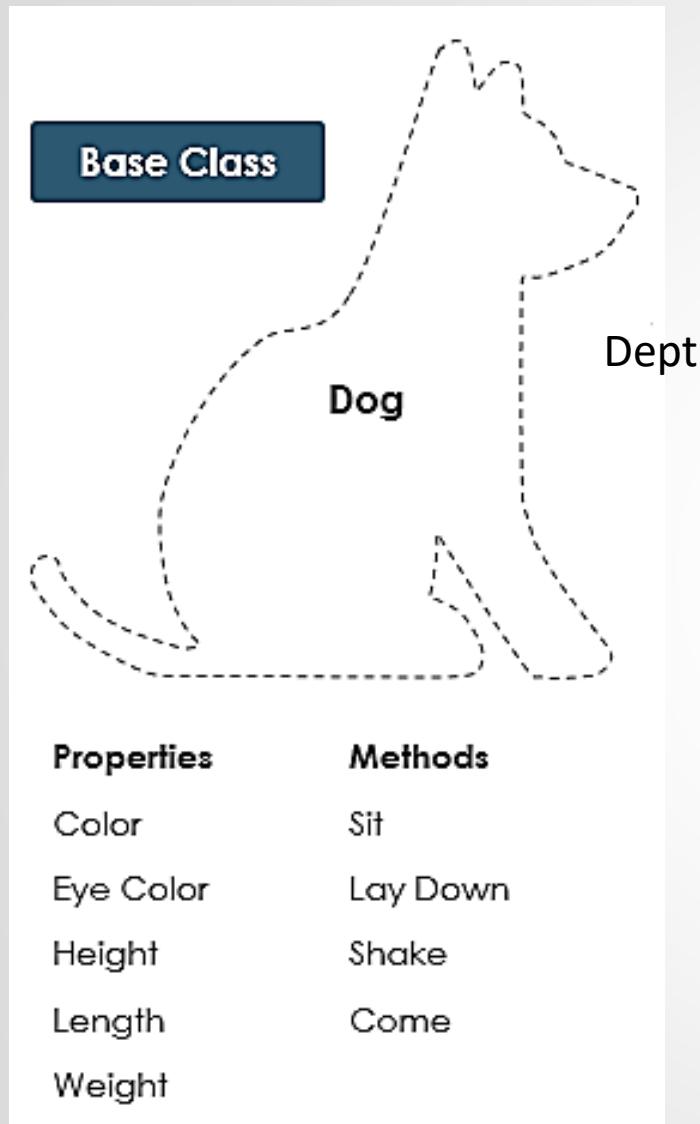
Object is an **instance of a class** in a particular moment in runtime that can have its own state and data values.

Likewise a static UML object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time, thus an object diagram encompasses objects and their relationships which may be considered a special case of a class diagram or a communication diagram.

Purpose of Object Diagram

- ❖ The use of object diagrams is fairly limited, mainly to **show examples of data structures**.
- ❖ During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.
- ❖ Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.

UML : Object Diagram



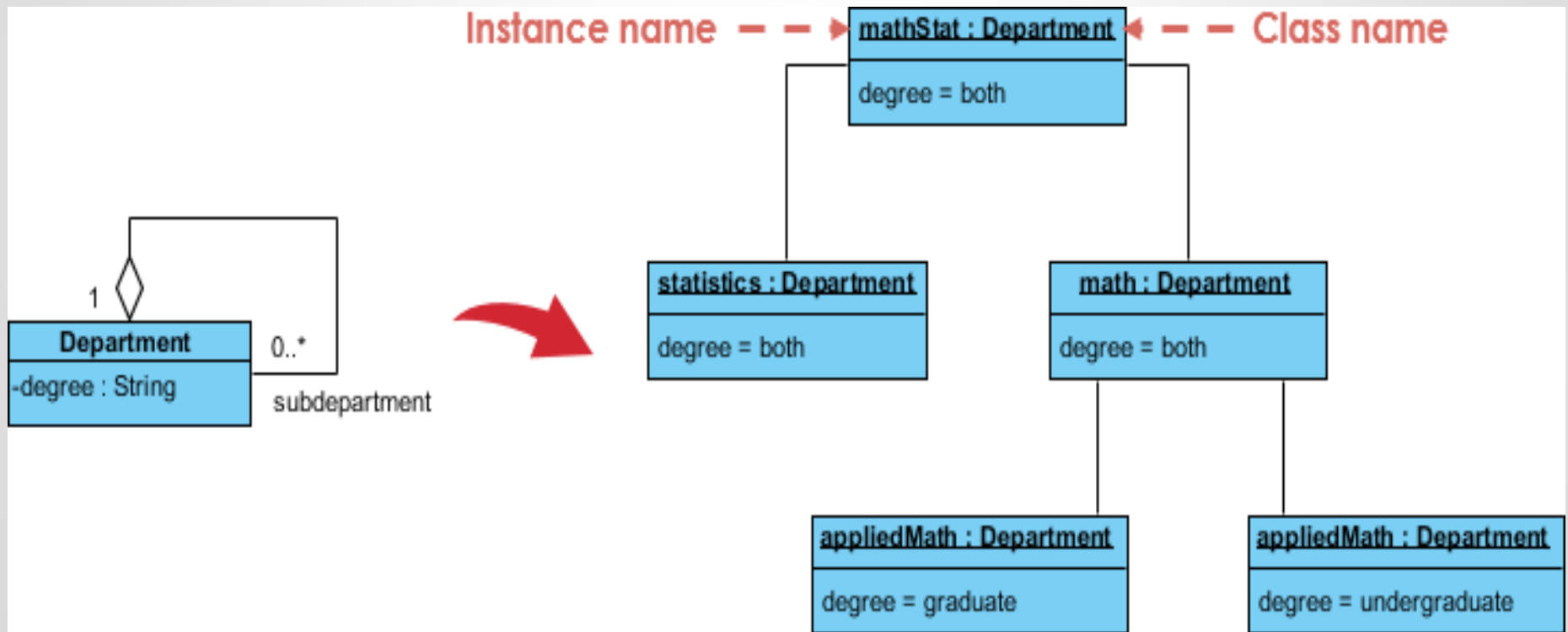
Name

Degree

UML : Object Diagram

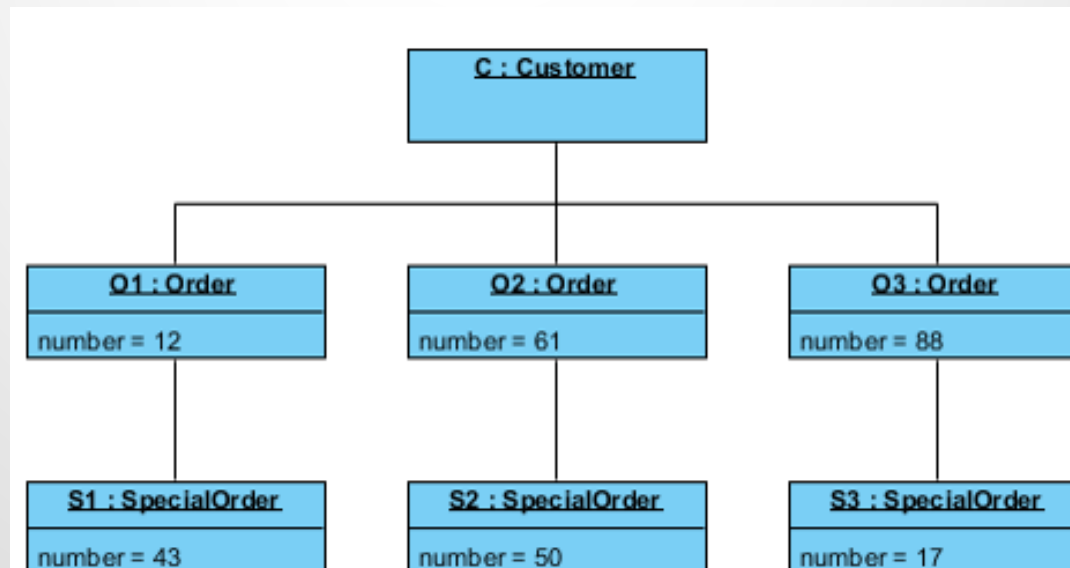
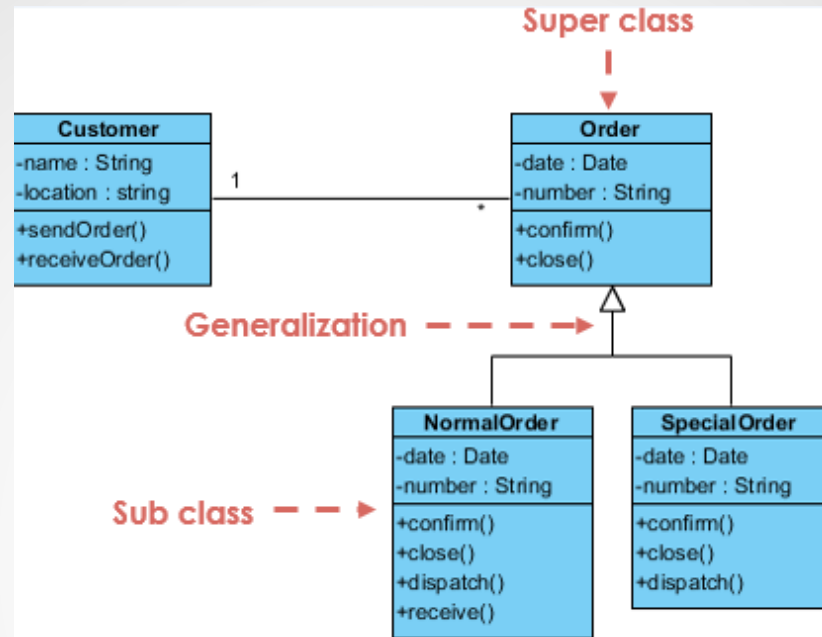
Class to Object Diagram

This small class diagram shows that a university Department can contain lots of other Departments and the object diagram below instantiates the class diagram, replacing it by a concrete example.



UML : Object Diagram

Class to Object Diagram



UML : Object Diagram

Class vs. Object diagram

SN	Class Diagram	Object Diagram
1.	It depicts the static view of a system.	It portrays the real-time behavior of a system.
2.	Dynamic changes are not included in the class diagram.	Dynamic changes are captured in the object diagram.
3.	The data values and attributes of an instance are not involved here.	It incorporates data values and attributes of an entity.
4.	The object behavior is manipulated in the class diagram.	Objects are the instances of a class.

Use Case Diagram

- ❖ **Use Case Diagram** captures the system's functionality and requirements by using actors and use cases.
- ❖ It only summarizes **some of the relationships** between use cases, actors, and systems.
- ❖ It does **not show the order** in which steps are performed to achieve the goals of each use case.

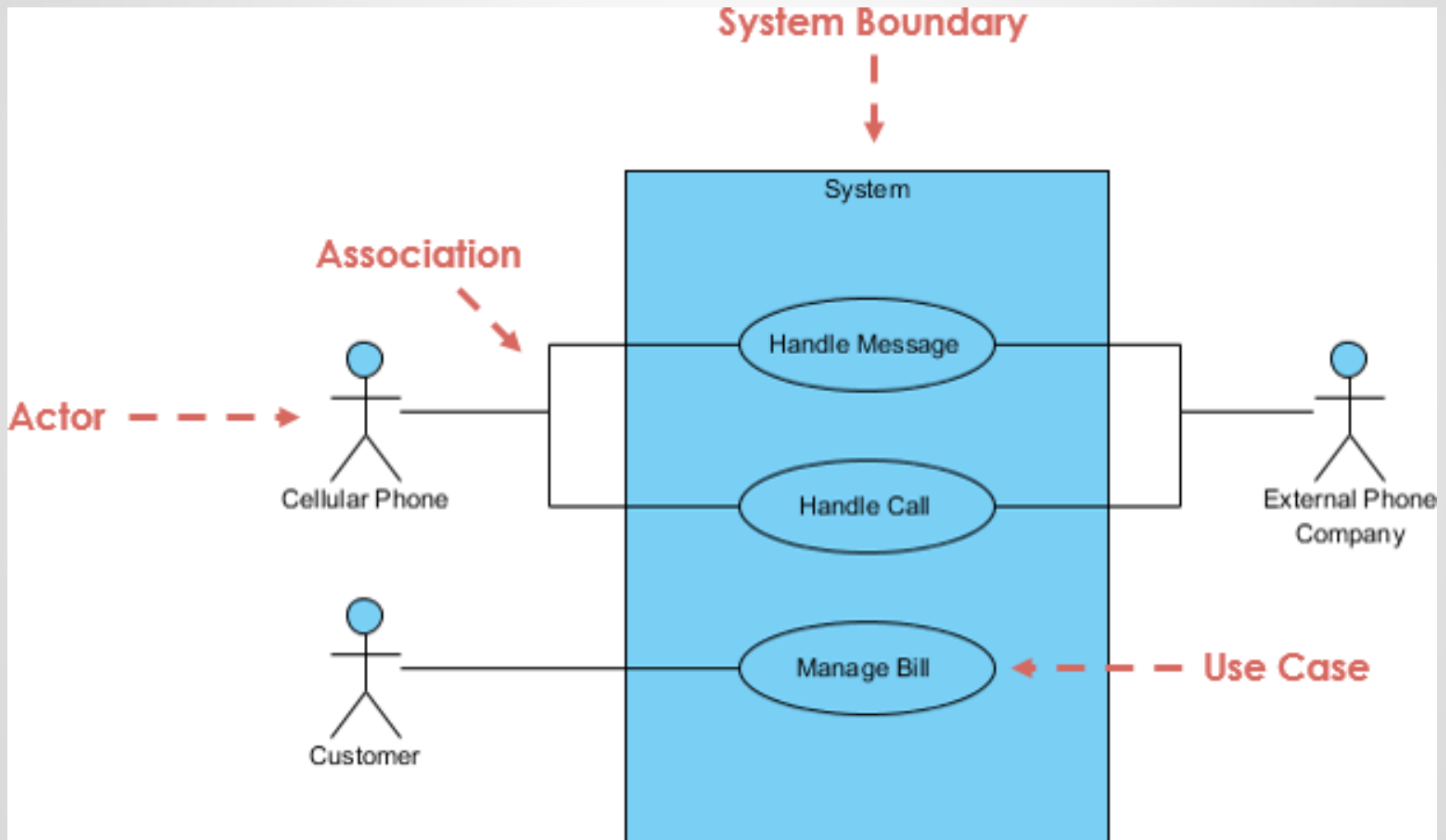
Purpose of Use Case Diagram

- Analyzing the requirements of a system
- High-level visual software designing
- Capturing the functionalities of a system
- Modeling the basic idea behind the system
- Forward and reverse engineering of a system using various test cases.

UML : Use Case Diagram

Example of Use Case Diagram

- <https://www.uml-diagrams.org/use-case-diagrams-examples.html>



Example of Use Case Diagram : Bank ATM

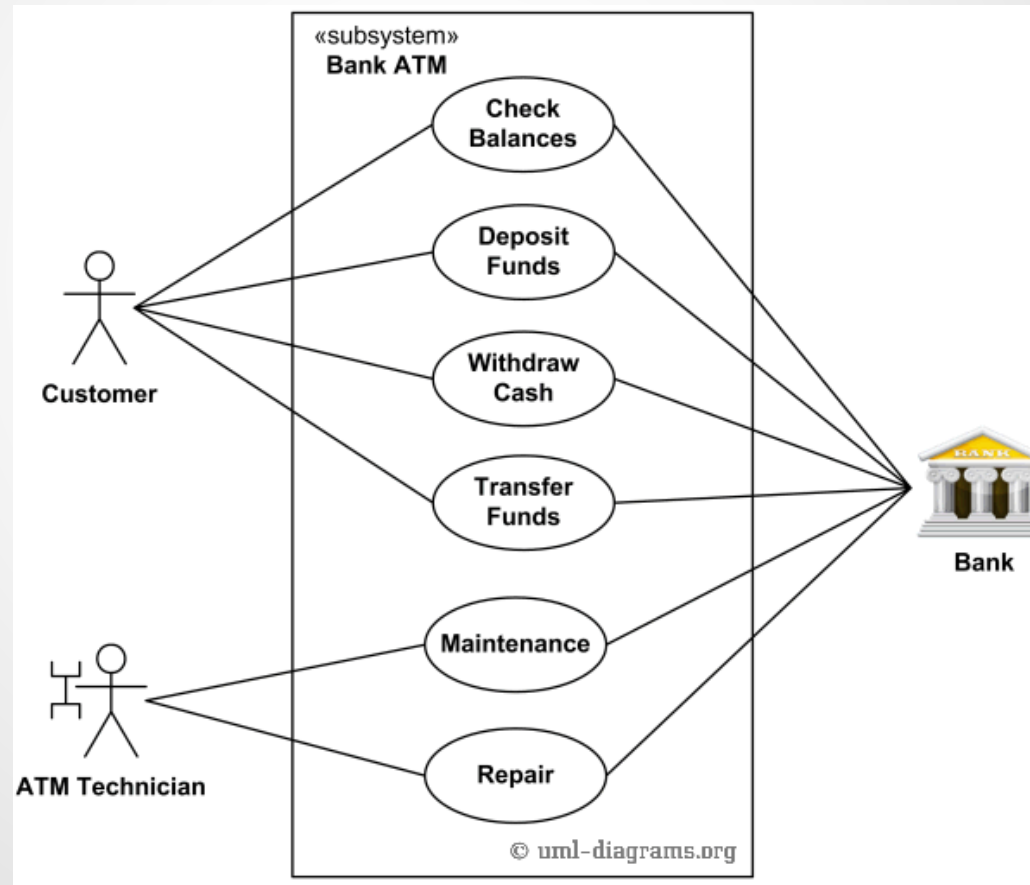
An automated teller machine (ATM) or the automatic banking machine (ABM) is a banking subsystem (subject) that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller.

Customer (actor) uses bank ATM to Check Balances of his/her bank accounts, Deposit Funds, Withdraw Cash and/or Transfer Funds . ATM Technician provides Maintenance and Repairs. All these use cases also involve Bank actor whether it is related to customer transactions or to the ATM servicing.

UML : Use Case Diagram

Example of Use Case Diagram : Bank ATM

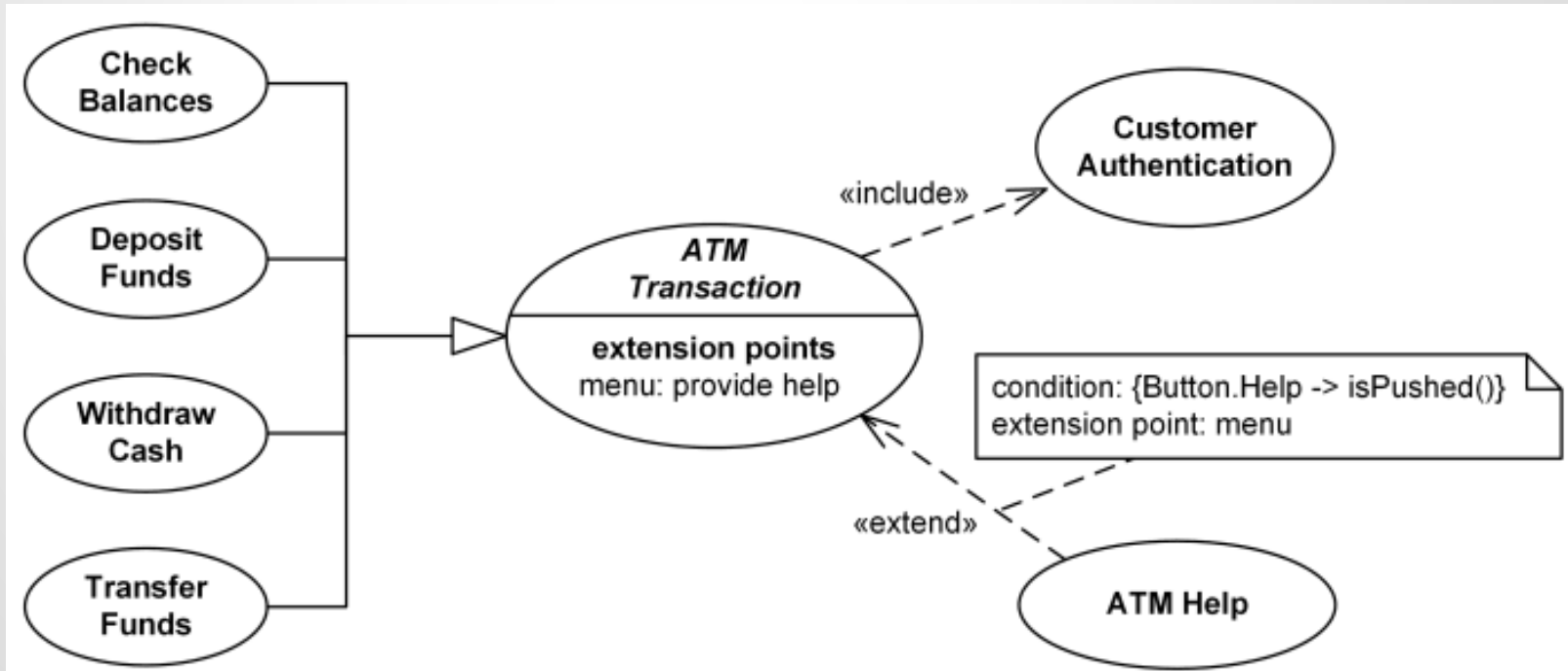
Customer (actor) uses bank ATM to **Check Balances** of his/her bank accounts, **Deposit Funds**, **Withdraw Cash** and/or **Transfer Funds**. ATM Technician provides **Maintenance** and **Repairs**. All these use cases also involve Bank actor whether it is related to customer transactions or to the ATM servicing.



UML : Use Case Diagram

Example of Use Case Diagram : Bank ATM

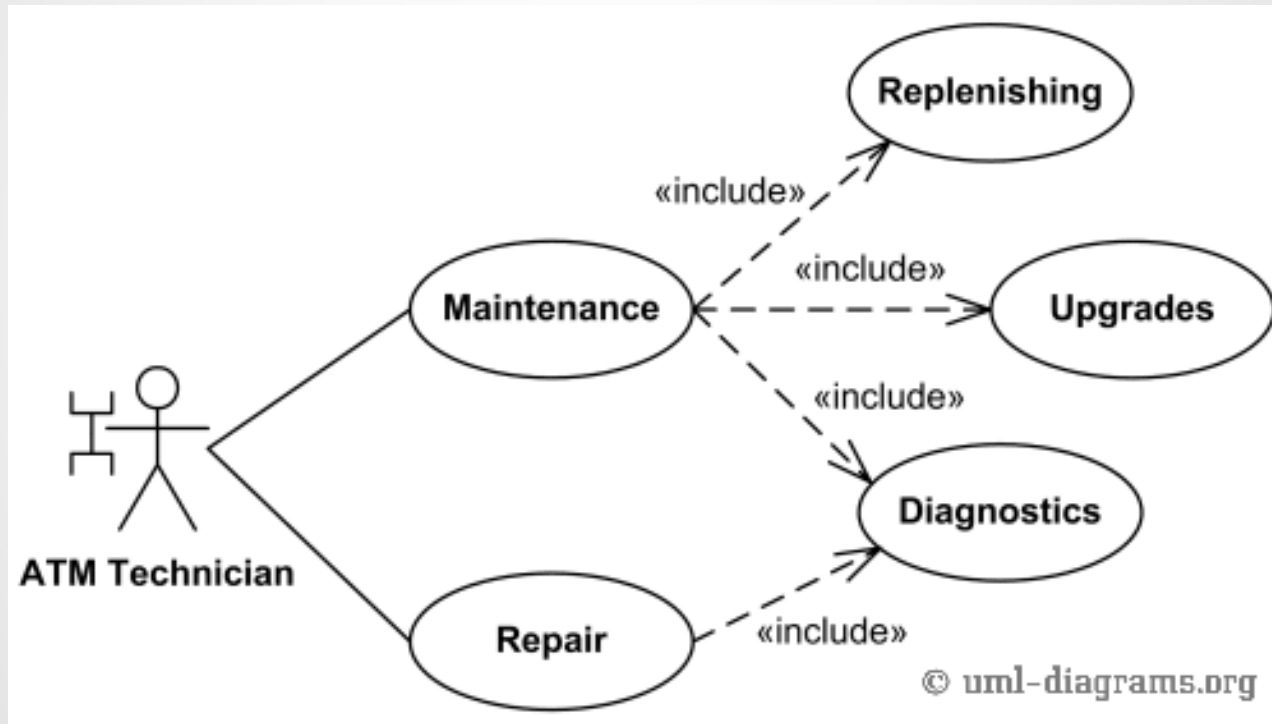
On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN). **Customer Authentication** use case is required for every ATM transaction so we show it as include relationship. Including this use case as well as transaction generalizations make the ATM Transaction an abstract use case. Customer may need some help from the ATM. ATM Transaction use case is extended via extension point called menu by the ATM Help use case whenever ATM Transaction is at the location specified by the menu and the bank customer requests help, e.g. by selecting Help menu item.



UML : Use Case Diagram

Example of Use Case Diagram : Bank ATM

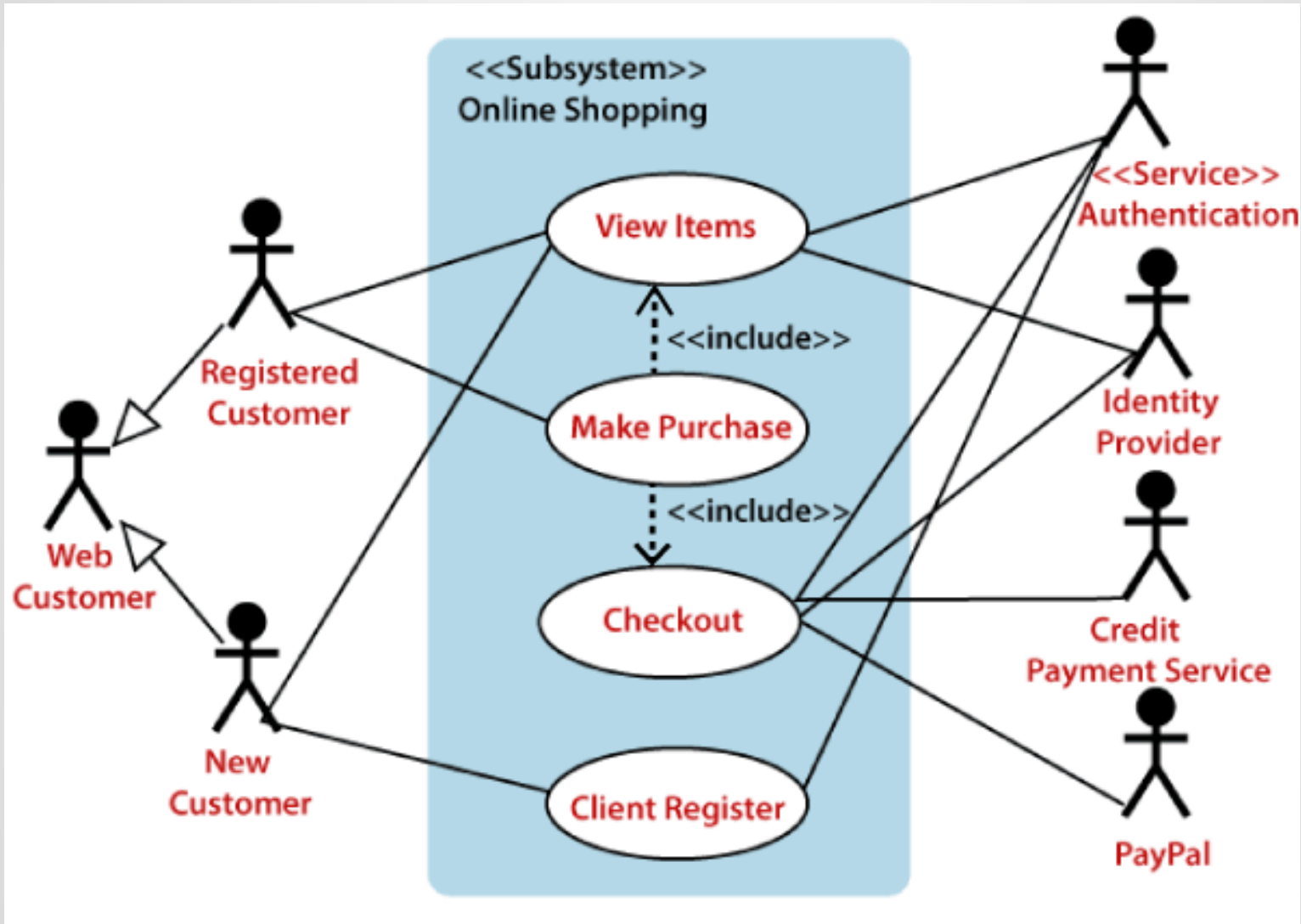
ATM Technician maintains or repairs Bank ATM. Maintenance use case includes **Replenishing** ATM with cash, ink or printer paper, **Upgrades** of hardware, **firmware or software**, and **remote or on-site Diagnostics**. Diagnostics is also included in (shared with) Repair.



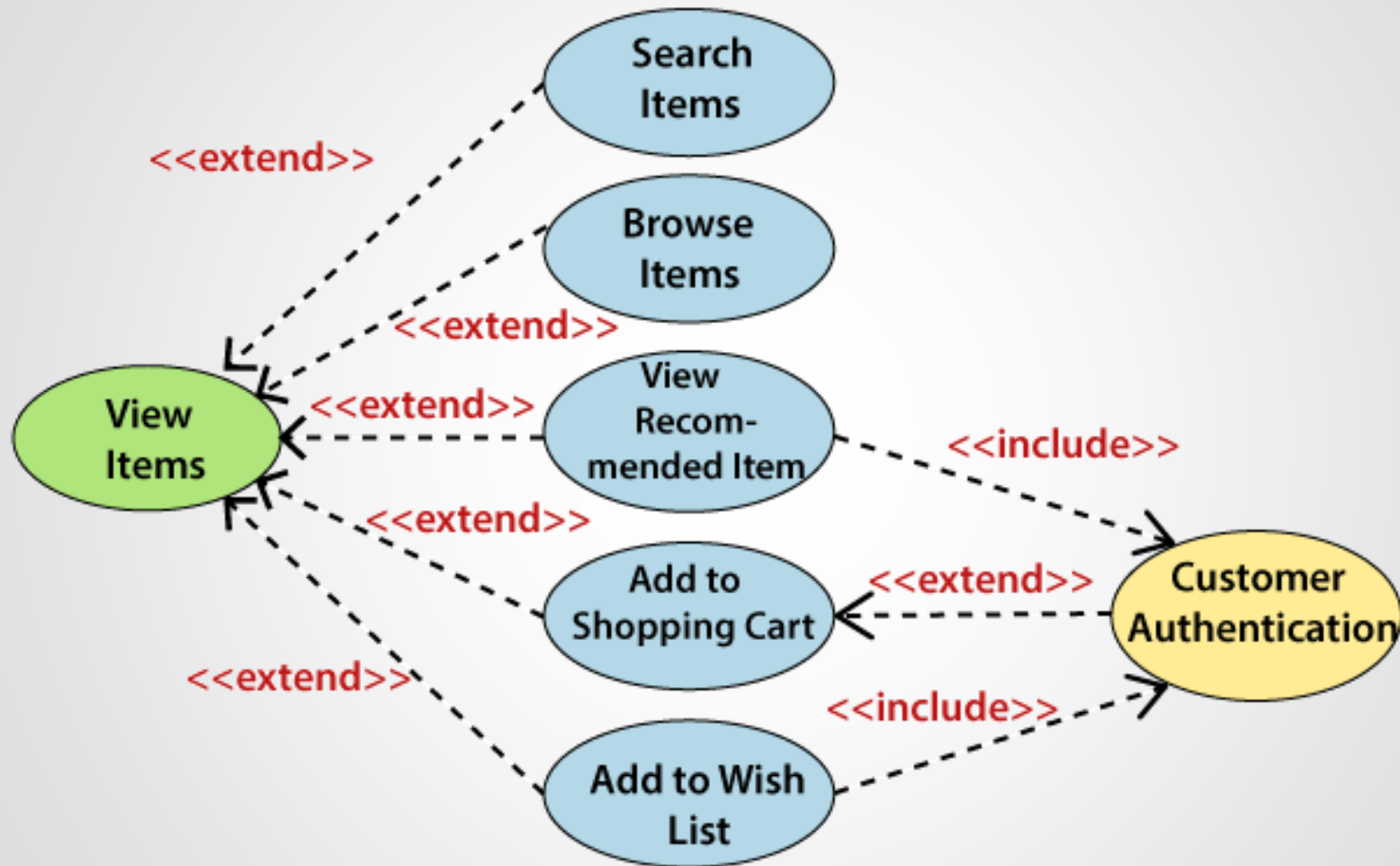
Bank ATM Maintenance, Repair, Diagnostics Use Cases Example

UML : Use Case Diagram

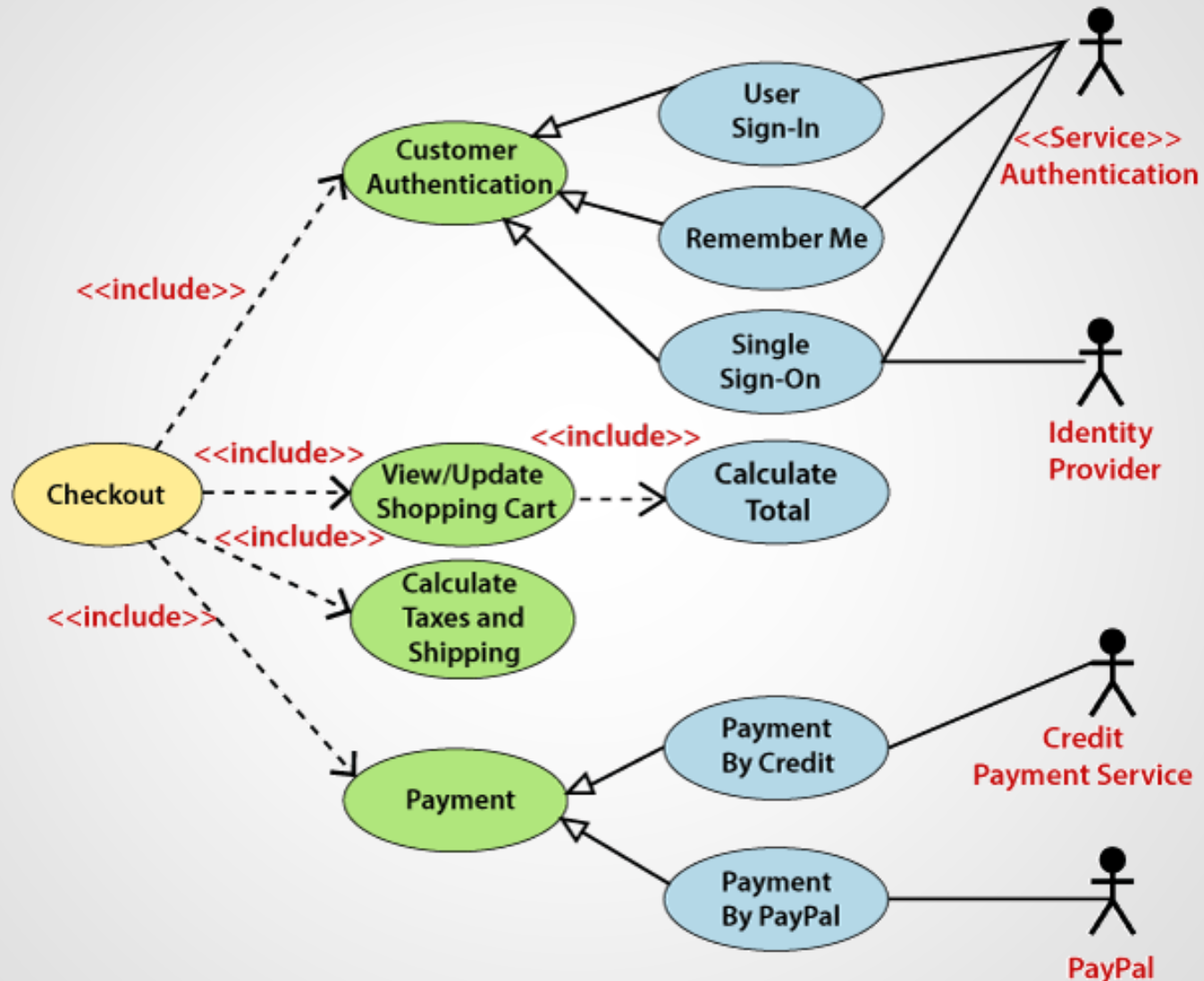
Example of Use Case Diagram



UML : Use Case Diagram



UML : Use Case Diagram



How to Identify Actor?

The following questions can help you identify the actors of a system.

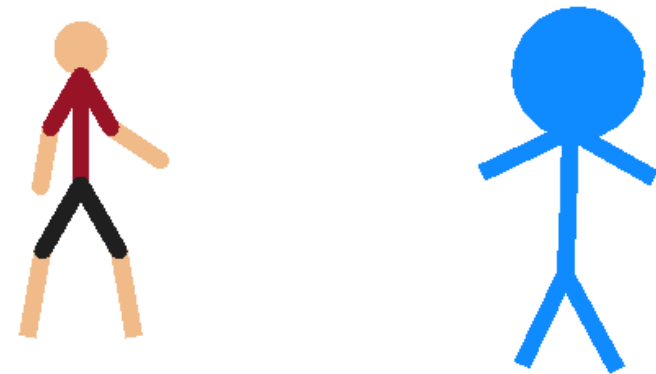
- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintains the system?
- Who shuts down the system?
- What other systems use this system?
- Who gets information from this system?
- Who provides information to the system?
- Does anything happen automatically at a present time?



How to Identify Use Cases?

The following questions can be asked to identify use cases, once your actors have been identified .


- What functions will the actor want from the system?
- Does the system store information?
- What actors will create, read, update or delete this information?
- Does the system need to notify an actor about changes in the internal state?
- Are there any external events the system must know about?
- What actor informs the system of those events?




StickNodes.com

Ticket management System

NOW PLAYING:
Sky Captain & The World of Tomorrow



TIME: 9:00PM
TICKET PRICE: \$6.00

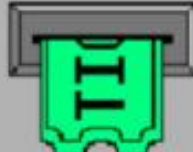


of Tickets: 2

+
-

Total Due: \$12.00

BUY






BALANCE \$15.00

—

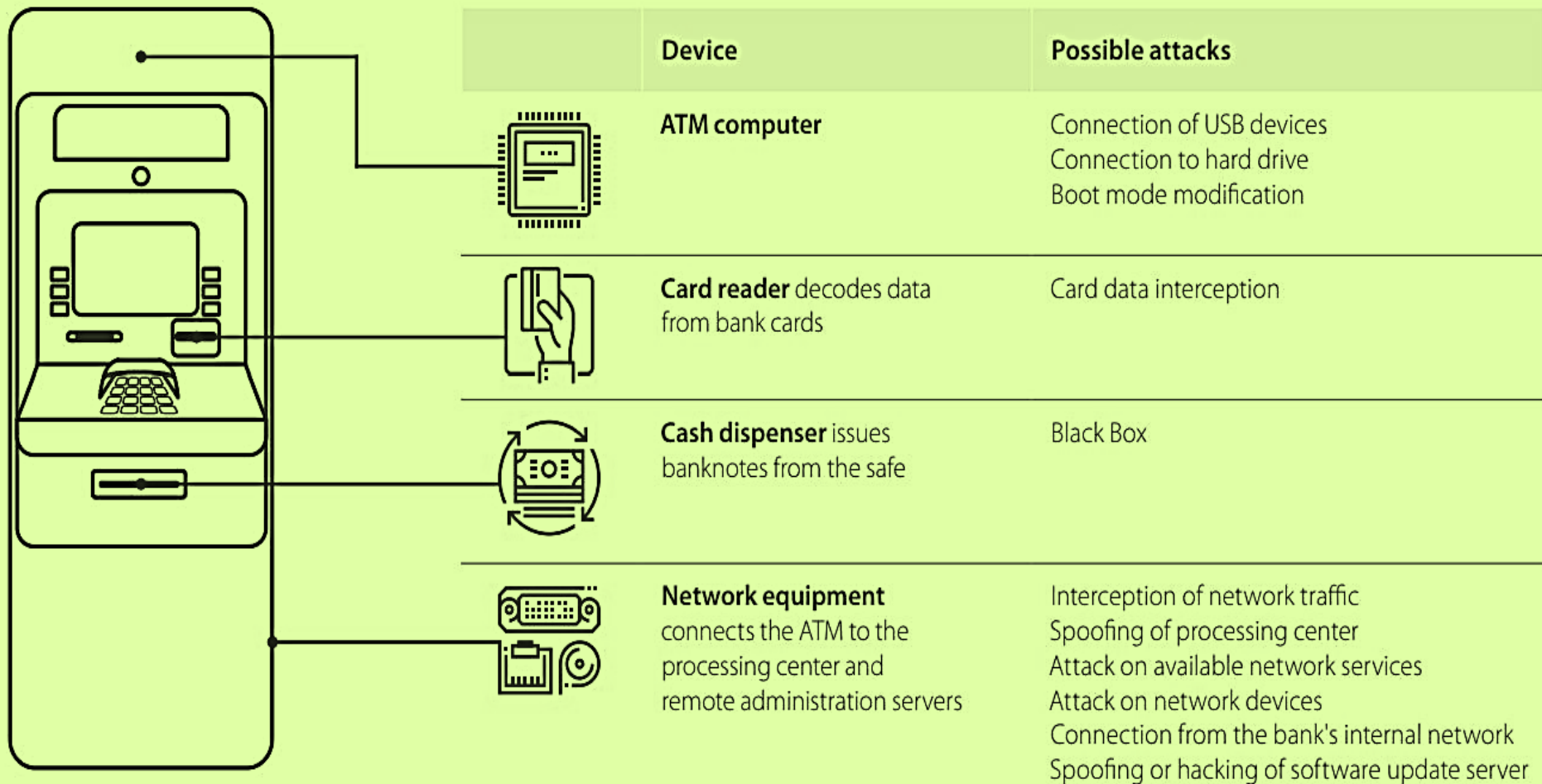
CHANGE

—

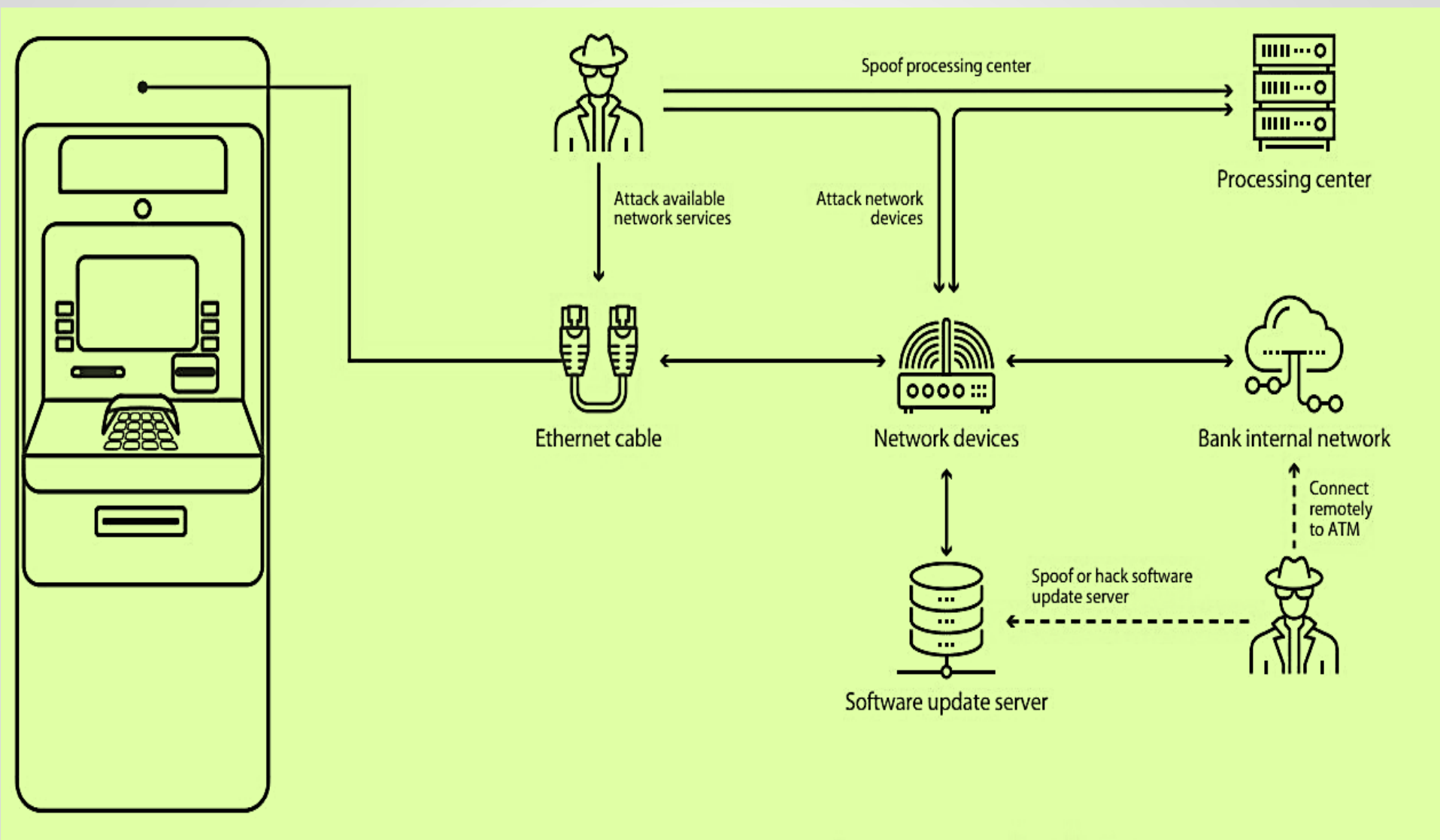




ATM network attacks



ATM network attacks



UML Tools

There are numerous tools, both commercial and open-source, which are available for designing UML diagrams.

- StarUML
- Umbrello
- Visual Paradigm
- Gliffy
- Altova
- draw.io





Thanks to All