# Object Oriented Analysis and Design

## Class Diagrams

Miss. Subodha Rathnayake
rmslr231@kln.ac.lk
Dept. of Software Engineering, Faculty of Computing and Technology , University of Kelaniya

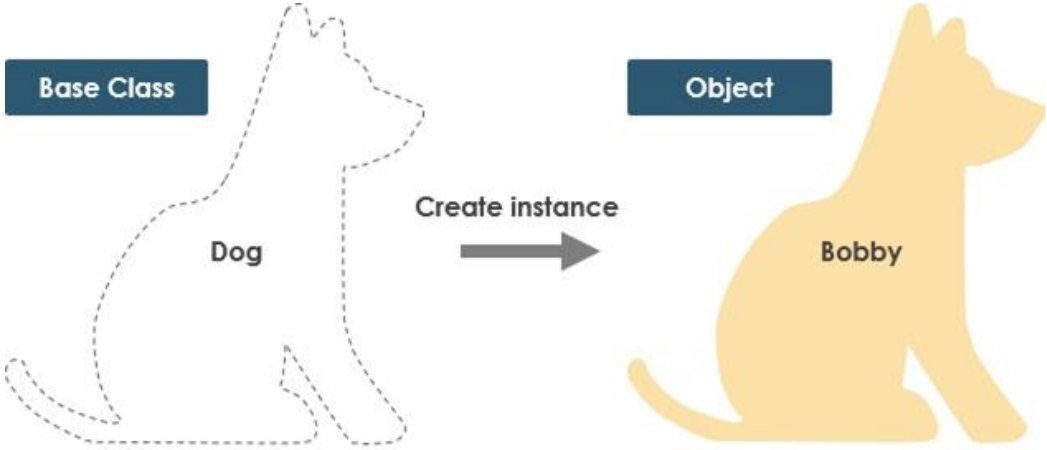
# What are class Diagrams?

- A type of UML diagram used in software engineering to visually represent the structure and relationships of classes within a system
- i.e. used to construct and visualize object-oriented systems.

# Classes and Object

- A **class** is a kind of blueprint or template for creating objects.
- In other words, a class refers to the category or type of objects.
- An **object** is called an instance of a class.
- A class (concept) describes a group of objects with
  - similar properties (attributes),
  - common behavior (operations),
  - common relationships to other objects,
  - and common meaning (“semantics”).

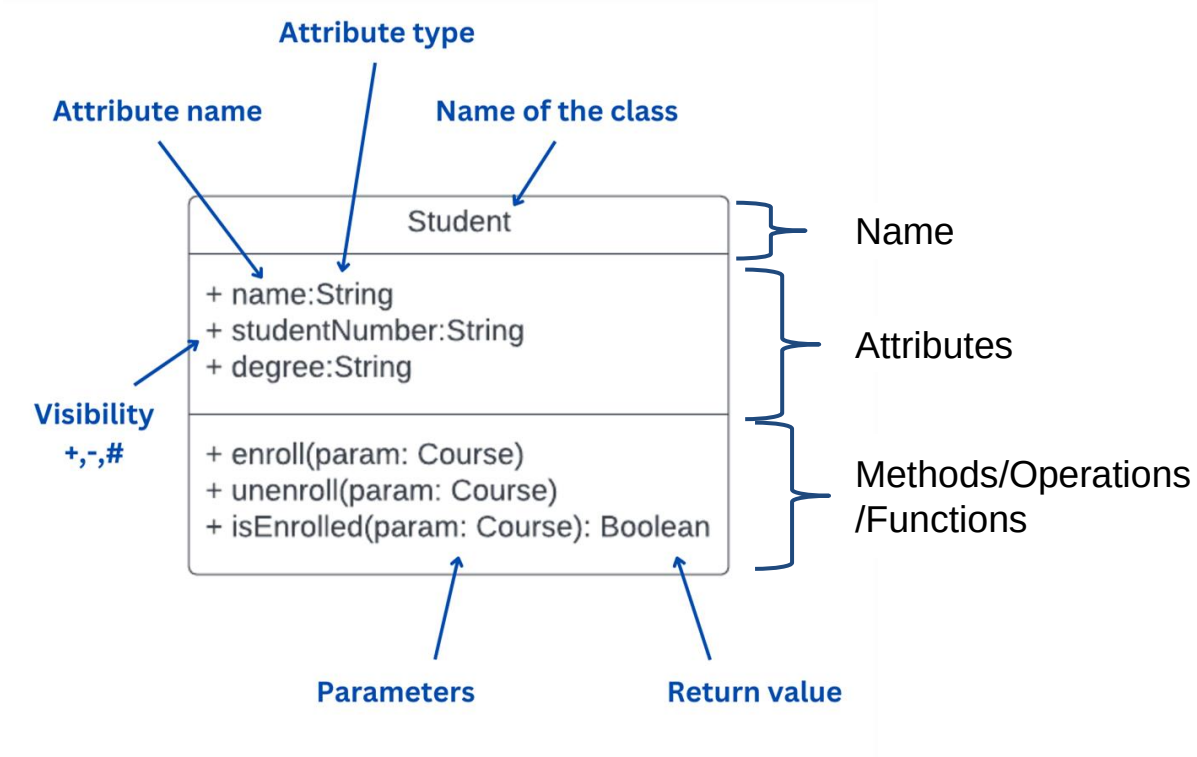
# Classes and Object




| Properties | Methods |
| --- | --- |
| Color | Sit |
| Eye Color | Lay Down |

| Property Values | Methods |
| --- | --- |
| Color: Yellow | Sit |
| Eye Color: Brown | Lay Down |

# UML Class Notation



# Member access modifiers

- All classes have different access levels depending on the **access modifier (visibility)**.
- Here are the access levels with their corresponding symbols:
  - Public (+) -visible to all classes
  - Private (-) - visible only within the class
  - Protected (#) -visible to subclasses
  - Package (~) - for package or default visibility (visible to classes in the same package)

only childs

# Relationships

- Objects do not exist in isolation from one another.

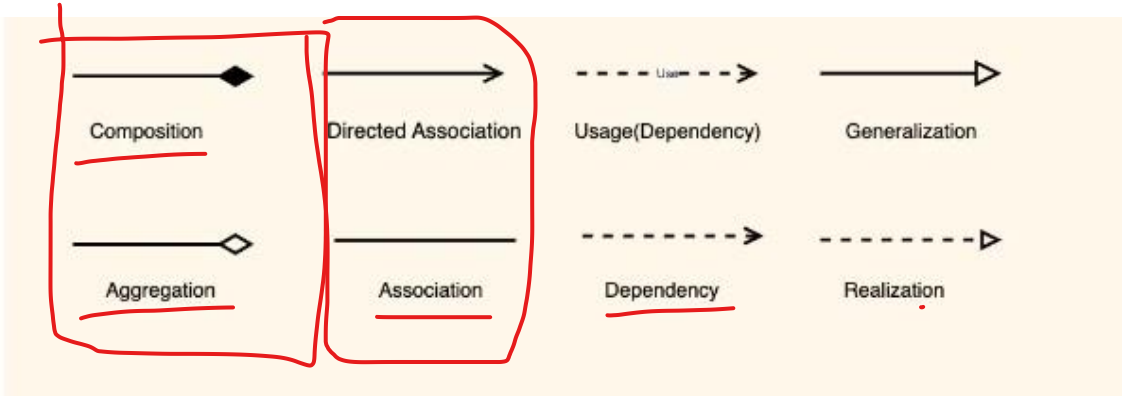
- A **relationship** represents a connection among things.
  - E.g. **Mineth:Student** is enrolled to Maths:Course object

- We will capture these relationships at the class level because,
  - It makes the model more reusable.
  - It makes the model more maintainable.
  - It makes the model more understandable.

- Class diagrams show classes and their relationships In UML.

# Relationships

- Relationships between classes describe,
  - how classes are connected or interact with each other within a system
- There are several types of relationships in object-oriented modeling, each serving a specific purpose.
- Some common types of relationships in class diagrams:

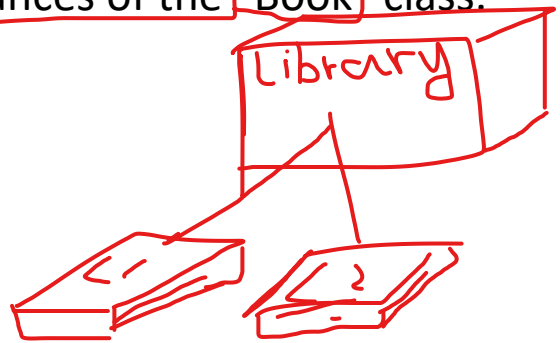


# Association

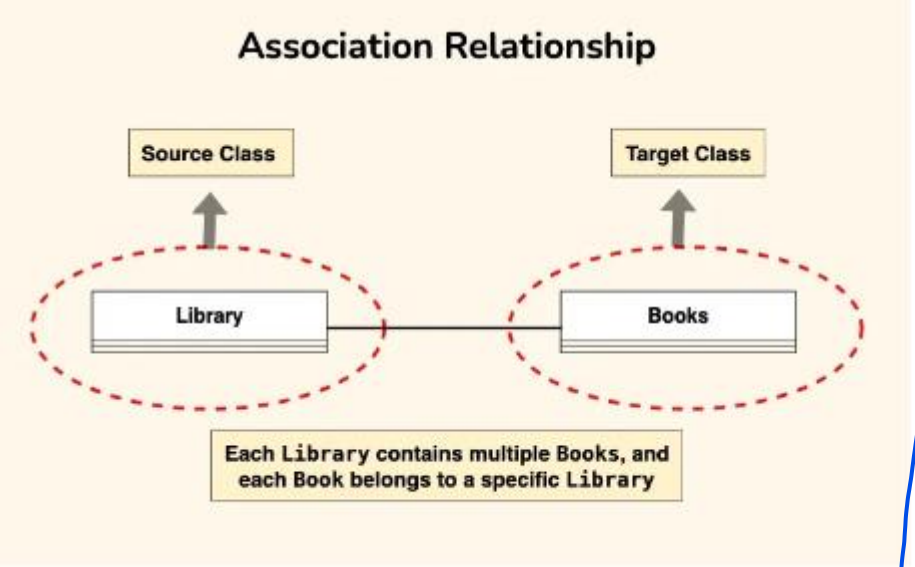
- An association represents a bi-directional relationship between two classes.
- It indicates that instances of one class are connected to instances of another class.
- Associations are typically depicted as a solid line connecting the classes, (with optional arrows indicating the direction of the relationship).

# Association

- Example: A simple system for managing a library.
  - System have two main entities: Book and Library
  - Each Library contains multiple Books, and each Book belongs to a specific Library.
  - This relationship between Library and Book represents an association.
- "Library" class can be considered the source class
  - Reason: it contains a reference to multiple instances of the "Book" class.
- "Book" class would be considered the target class
  - Reason: it belongs to a specific library.



# Association




Association:

General connection between classes.
Example: Teacher-Student.
Aggregation:

Whole-part relationship where parts can exist independently.
Example: Library-Book.
Composition:

Whole-part relationship where parts cannot exist independently.
Example: House-Room.

Associations Without Arrowheads
Bidirectional Association
Meaning: When an association line between two classes does not have arrowheads, it generally represents a bidirectional association. This means that both classes are aware of each other and can navigate to one another.

# Association Multiplicity

- Can a company exist without any employee?
  - If yes, then the association is optional at the Employee end - zero or more (0..*)
  - If no, then it is not optional -one or more (1..*)
  - If it must have only one employee - exactly one (1)
- What about the other end of the association?
  - Can an employee work for more than one company?
  - No. So the correct multiplicity is one.

# Association Multiplicity

- Some examples of specifying multiplicity:
  - Optional (0 or 1) : **0..1**
  - Exactly one : **1** or **1..1**
  - Zero or more : **0..*** or ***
  - One or more : **1..***
  - A range of values (between) : **2..6**

One-to-Many (1:N): A single instance of one class is associated with multiple instances of another class.
Example: A teacher teaches multiple students.

Multiple instances of one class are associated with a single instance of another class.
Example: Multiple students are taught by one teacher.

# Directed Association

- Represents a relationship between two classes where the association has a direction, indicating that one class is associated with another in a specific way.
- An arrowhead is added to the association line to indicate the direction of the relationship.
- The arrow points from the class that initiates the association to the class that is being targeted or affected by the association.
- Directed associations are used when the association has a specific flow or directionality,
  - such as indicating which class is responsible for initiating the association or which class has a dependency on another

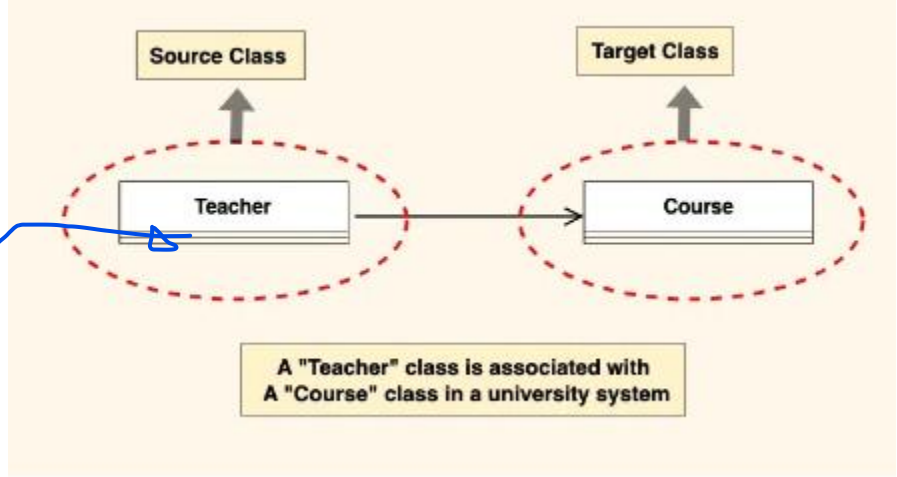
# Directed Association

- Example: university system
- "Teacher" class is associated with a "Course" class
- Arrow may point from the "Teacher" class to the "Course" class, indicating ,
  - a teacher is associated with or teaches a specific course.
- Source class: the "Teacher" class.
  - The "Teacher" class initiates the association by teaching a specific course.
- Target class: the "Course" class.
  - The "Course" class is affected by the association as it is being taught by a specific teacher.

in teaacher class there is method called teach(course)

# Directed Association

Associations With Arrowheads
Unidirectional Association
Meaning: When an association line has an arrowhead, it indicates a unidirectional association. This means that the class at the tail end of the arrow knows about the class at the head end, but not necessarily vice versa.



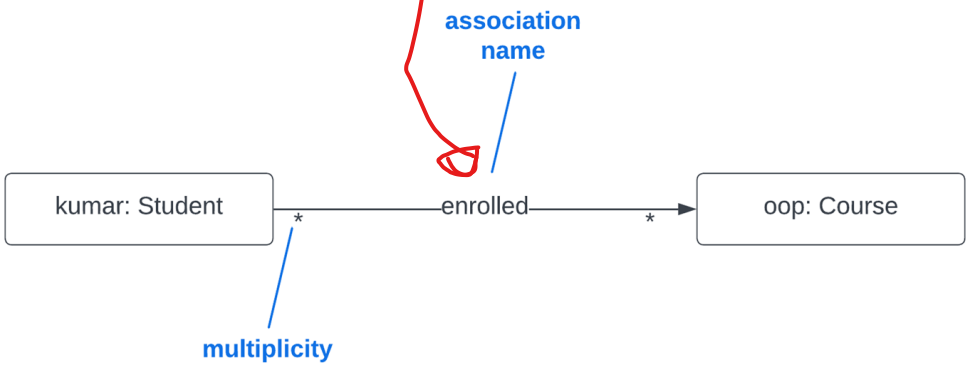

teacher knows there is a course class,but  course doesnt know it

# Association

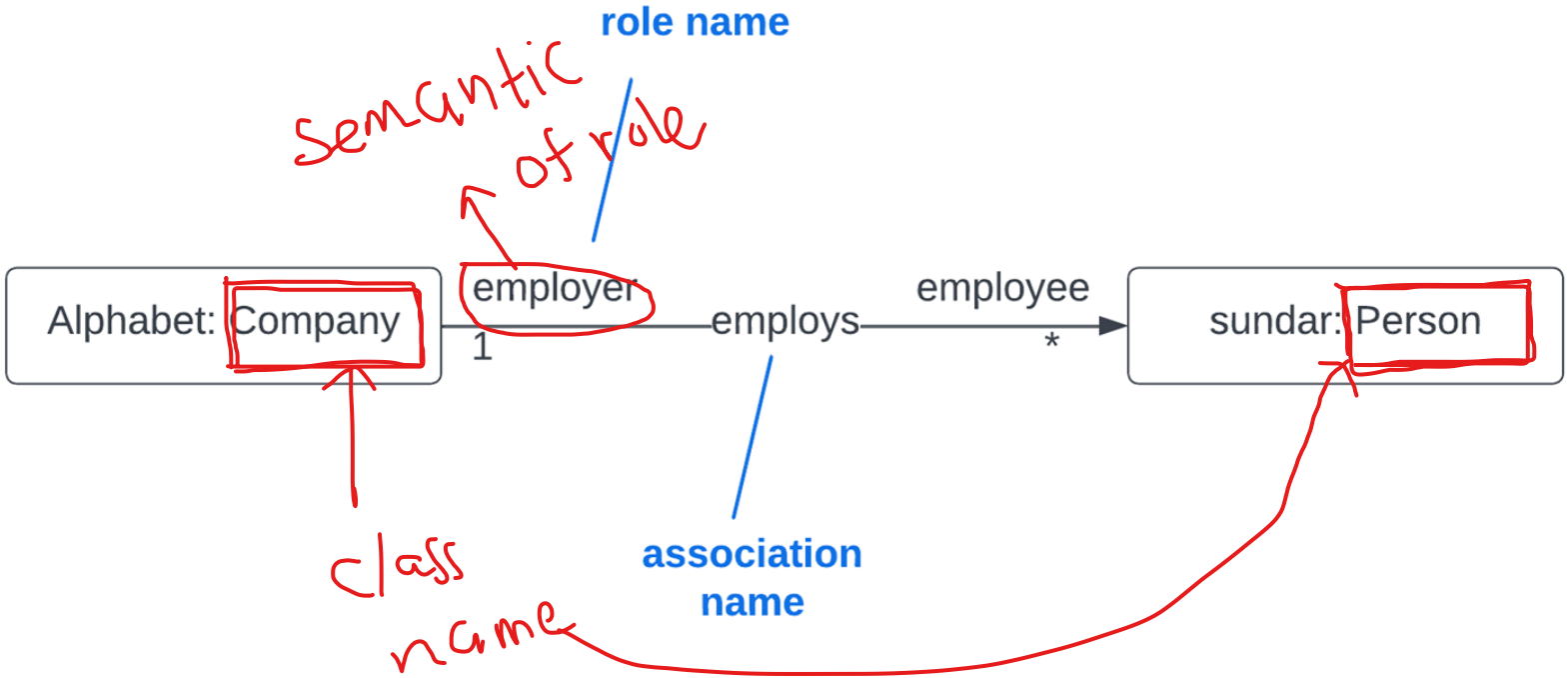
- Associations may optionally have the following:
  - Association name
  - Role names
  - Multiplicity

# Association

- **Association name** should be a verb or verb phrase.
- **Role names** should be a noun or noun phrase describing the semantics of the role.
- **Multiplicity** is the number of objects that can participate in an instantiated relation



# Association



# Aggregation

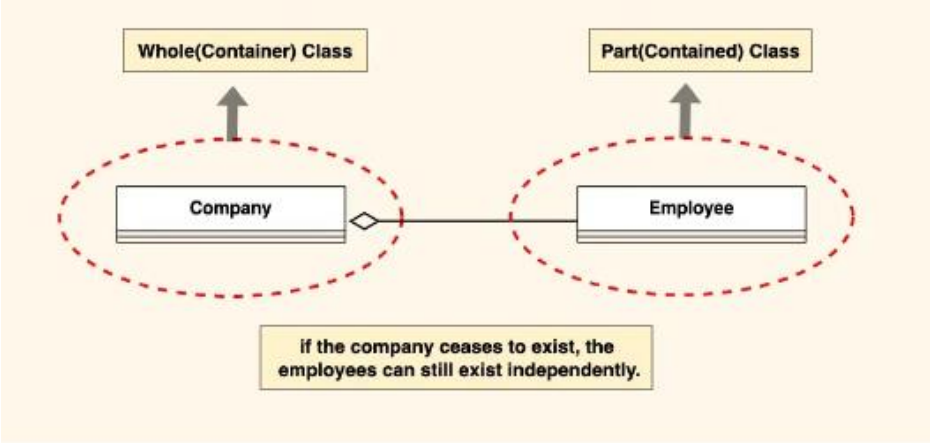
- A specialized form of association that represents a “whole-part” relationship
- Denotes a stronger relationship where one class (the whole) contains or is composed of another class (the part).
- Represented by a diamond shape on the side of the whole class.
- Child class can exist independently of its parent class.

It means that one class (the whole) contains other classes (the parts), but the parts can exist independently of the whole.

# Aggregation

- Example: The company can be considered as the whole, while the employees are the parts.
- Employees belong to the company, and the company can have multiple employees.
- However, if the company ceases to exist, the employees can still exist independently.

# Aggregation



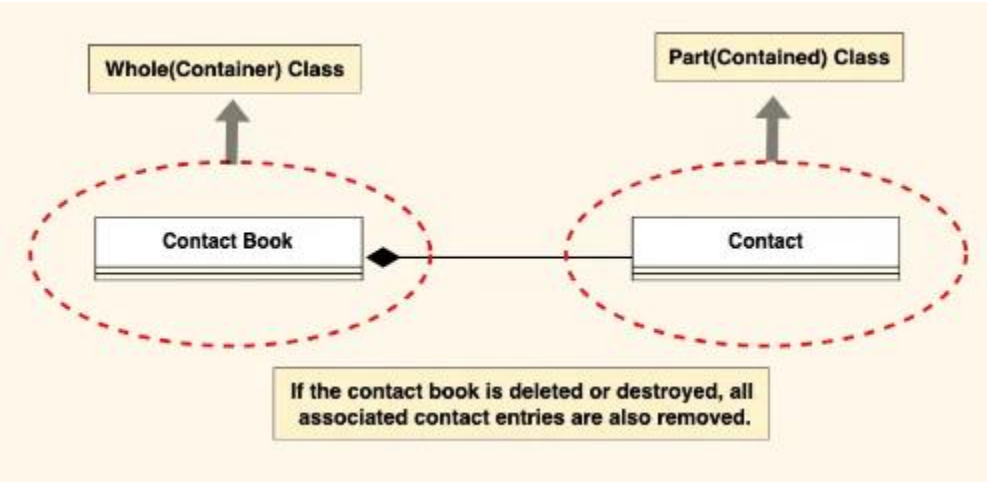
# Composition

- A stronger form of aggregation, indicating a more significant ownership or dependency relationship.
- In composition, the part class cannot exist independently of the whole class.
- Composition is represented by a filled diamond shape on the side of the whole class.

# Composition

- Example: A digital contact book application
- The contact book is the whole, and each contact entry is a part.
- Each contact entry is fully owned and managed by the contact book.
- If the contact book is deleted or destroyed, all associated contact entries are also removed.
- This illustrates composition because the existence of the contact entries depends entirely on the presence of the contact book.
- Without the contact book, the individual contact entries lose their meaning and cannot exist on their own.

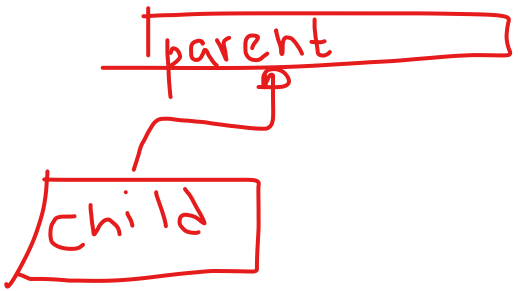
# Composition



# Generalization(Inheritance)

Represents an “is-a” relationship between classes.

- One class (the subclass or child) inherits the properties and behaviors of another class (the superclass or parent).
- Inheritance is depicted by a solid line with a closed, hollow arrowhead pointing from the subclass to the superclass.

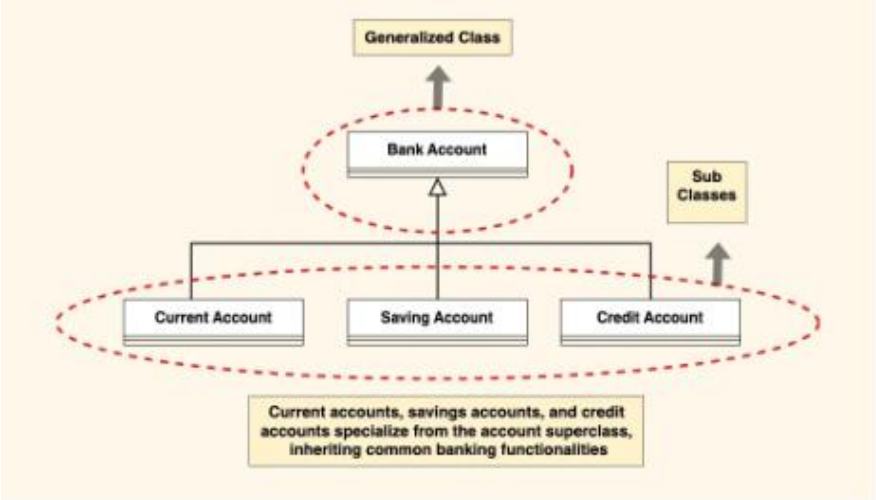



having a hole or empty space inside

# Generalization(Inheritance)

- Example: Bank accounts
- Can use generalization to represent different types of accounts such as current accounts, savings accounts, and credit accounts.
- Bank Account class serves as the generalized representation of all types of bank accounts, while the subclasses (Current Account, Savings Account, Credit Account) represent specialized versions that inherit and extend the functionality of the base class.

# Generalization(Inheritance)



# Dependency Relationship

Exists between two classes when one class relies on another.

- But the relationship is not as strong as association or inheritance.
- Represents a more loosely coupled connection between classes.
- Dependencies are often depicted as a dashed arrow

Dependency represents a "uses-a" relationship where one class relies on another class to function.
Typically seen when one class uses another class as a parameter in a method or creates an instance of another class within a method.

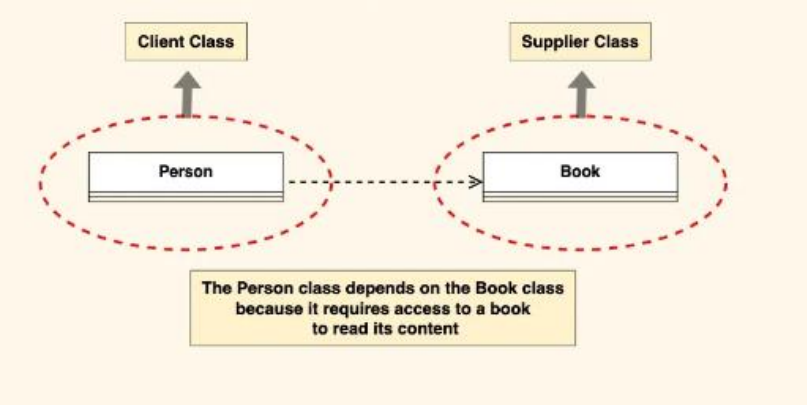
# Dependency Relationship

Example:  A scenario where a Person depends on a Book

- **Person Class:**
  - Represents an individual who reads a book.
  - The Person class depends on the Book class to access and read the content.
- **Book Class:**
  - Represents a book that contains content to be read by a person.
  - The Book class is independent and can exist without the Person class.

# Dependency Relationship

- The Person class depends on the Book class because it requires access to a book to read its content.
- However, the Book class does not depend on the Person class; it can exist independently and does not rely on the Person class for its functionality.



# How to draw Class Diagrams?

- Involves visualizing the structure of a system, including classes, their attributes, methods, and relationships.
- Steps to draw class diagrams
  1. Identify Classes:
  2. List Attributes and Methods:
     - For each class, list its attributes (properties, fields) and methods (functions, operations).
     - Include information such as data types and visibility (public, private, protected).

# How to draw Class Diagrams?

- Steps to draw class diagrams

3. Identify Relationships
   - Common relationships include associations, aggregations, compositions, inheritance, and dependencies.
   - Understand the nature and multiplicity of these relationships.
4. Create Class Boxes:
   - Draw a rectangle (class box) for each class identified.
   - Place the class name in the top compartment of the box.
   - Divide the box into compartments for attributes and methods.

# How to draw Class Diagrams?

- Steps to draw class diagrams

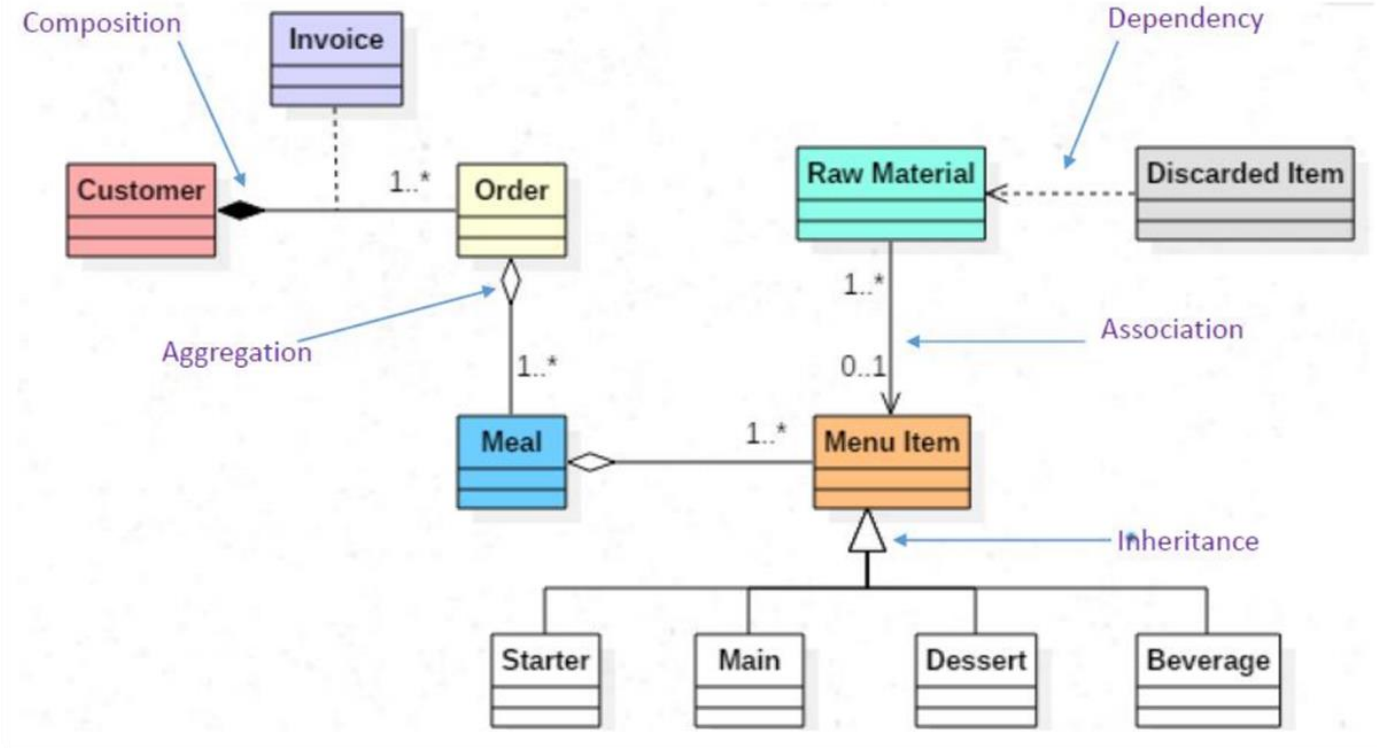
5. Add Attributes and Methods:
   - Inside each class box, list the attributes and methods in their respective compartments.
   - Use visibility notations (+ for public, – for private, # for protected, ~ for package/default).
6. Draw Relationships:
   - Draw lines to represent relationships between classes.
   - Use arrows to indicate the direction of associations or dependencies.
   - Different line types or notations may be used for various relationships.

# How to draw Class Diagrams?

- Steps to draw class diagrams

7. Label Relationships:
   - With multiplicity and role names if needed.
   - Multiplicity: the number of instances involved in the relationship
   - Role names: the role of each class in the relationship.
8. Review and Refine:
   - To ensure it accurately represents the system's structure and relationships.
   - Refine the diagram as needed based on feedback and requirements.
9. Use Tools for Digital Drawing

# Class relationships - Example



# Activity

- Draw a Class diagram for an ATM system using the details provided in the Last lecture.

# Thank you