Object-Oriented Software Design

Topics covered in the previous Lecture

- Introduction to UML
- Use case diagram

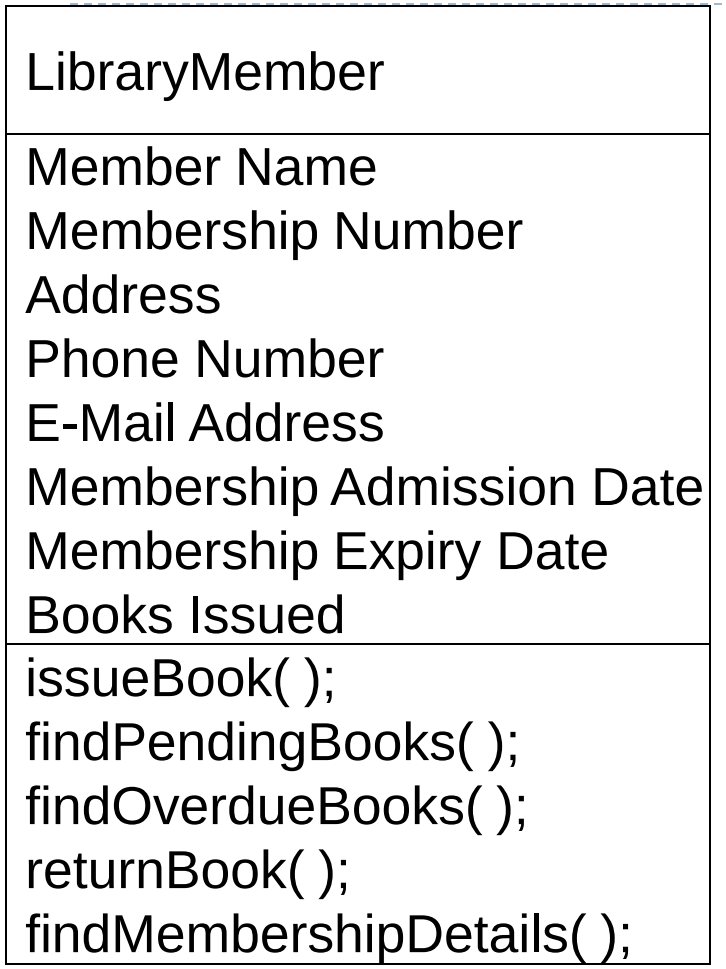
Class diagram

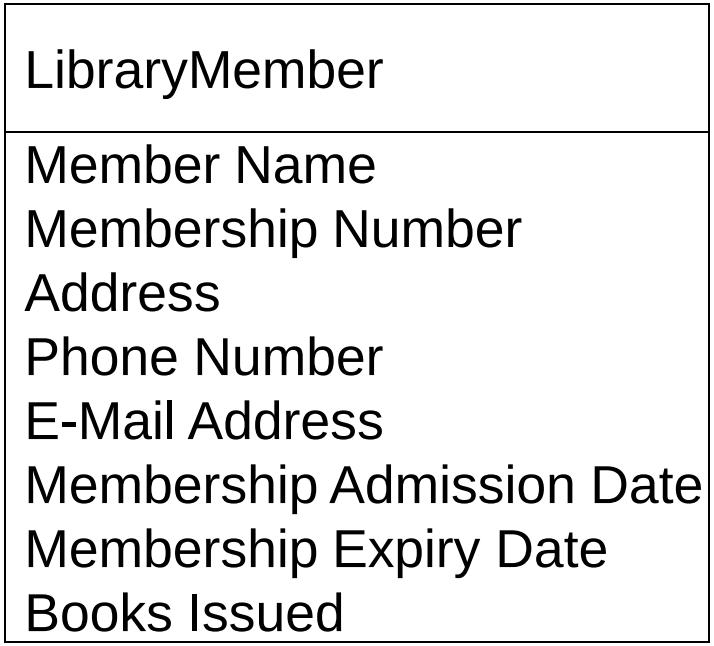
- Describes static structure of a system
- Main constituents are classes and their relationships:
  - Generalization
  - Aggregation
  - Association
  - Various kinds of dependencies

Class diagram

- Entities with common features, i.e. attributes and operations
- Classes are represented as solid outline rectangle with compartments
- Compartments for name, attributes & operations
- Attribute and operation compartment are optional for reuse purpose

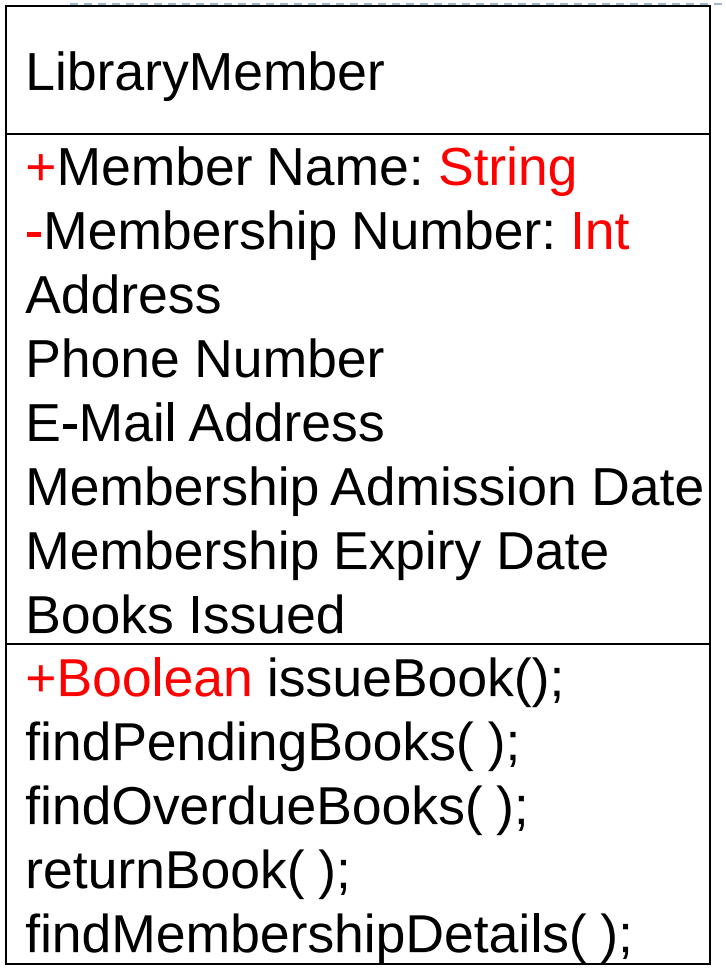
Example of Class diagram

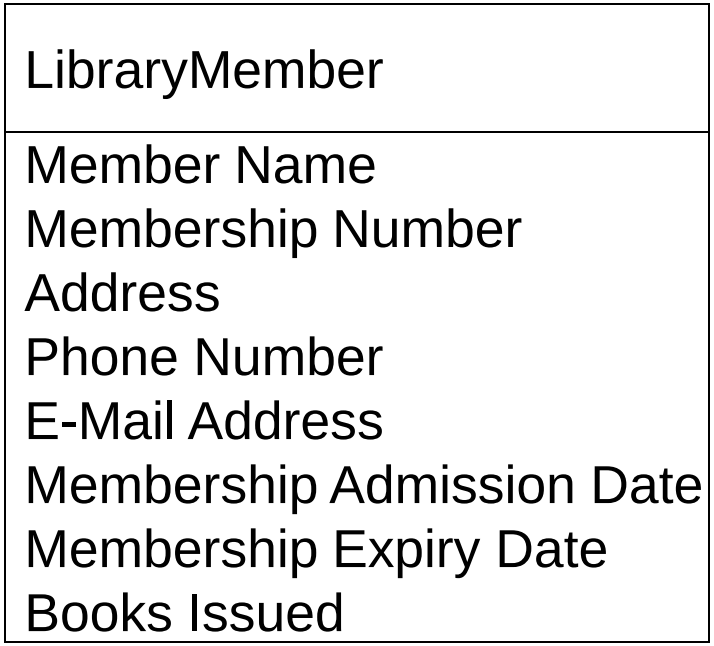




Different representations of the LibraryMember class

Example of Class diagram



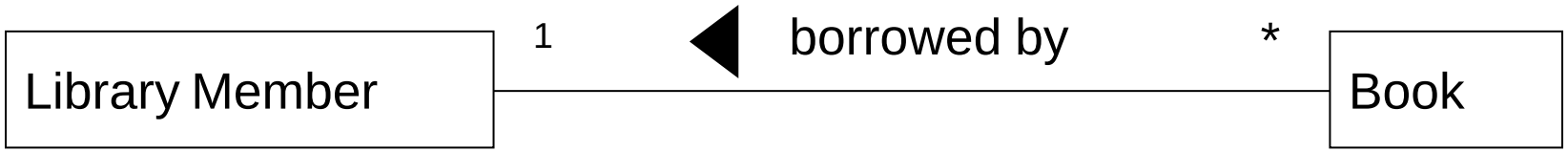


Different representations of the LibraryMember class

Association Relationship

- Enable objects to communicate with each other
- Usually binary but more classes can be involved
- Class can have relationship with itself (recursive association)
- Arrowhead used along with name, indicates direction of association
- Multiplicity indicates # of instances

Association Relationship

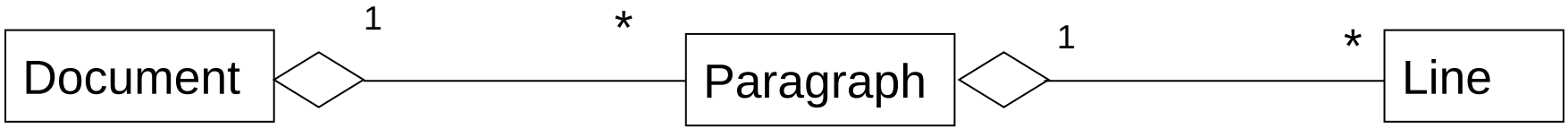


Association between two classes

Aggregation Relationship

- Represent a whole-part relationship
- Represented by diamond symbol at the composite end
- Cannot be reflexive(i.e. recursive)
- Not symmetric
- It can be transitive

Aggregation Relationship

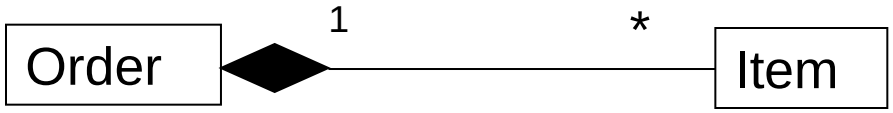


Representation of aggregation

Composition Relationship

- Composition is a stricter form of aggregation, in which the parts are existence-dependent on the whole.
- Life of a parts cannot exist outside the whole.
- Lifeline of both are identical.
- When the whole is created, the parts are created and when whole is destroyed, the parts are destroyed.

- Life of **item** is same as the **order**

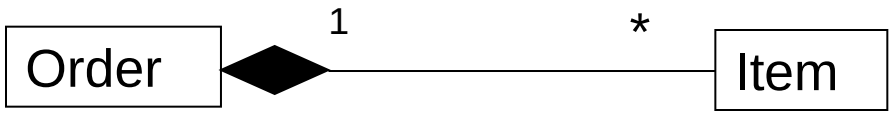


Representation of composition

Aggregation versus Composition

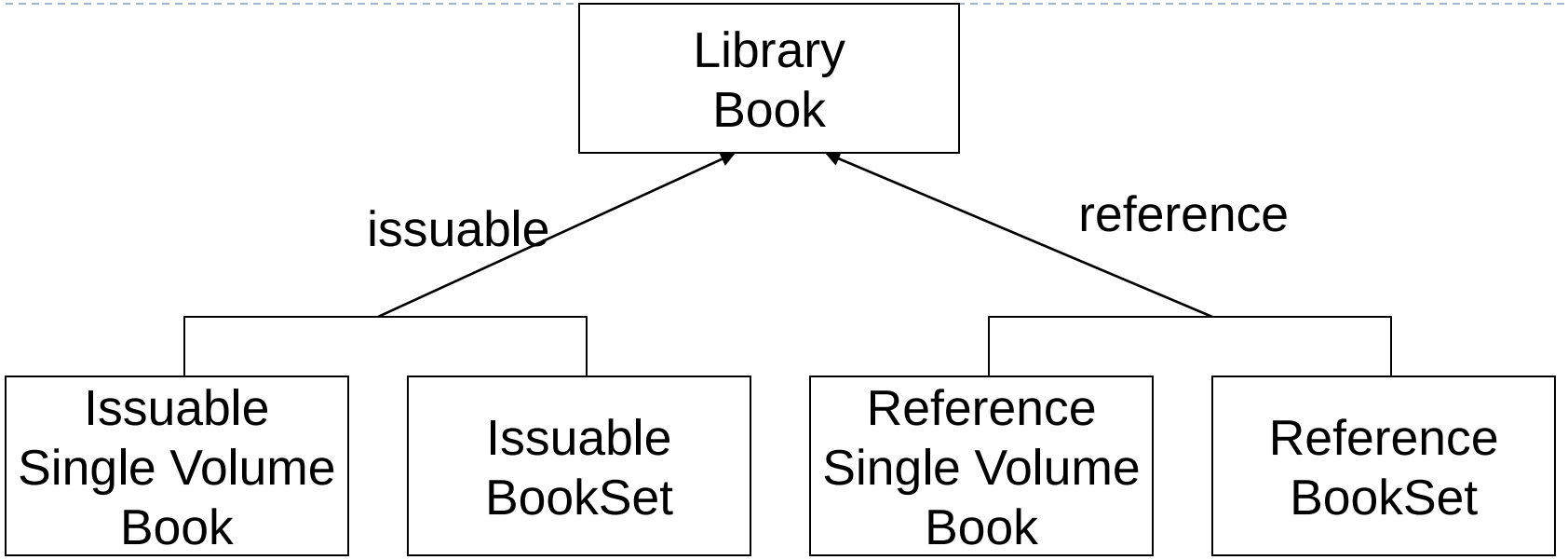
- Both represent part/whole relationships.
- When the components can dynamically be added and removed from the aggregate, then the relationship is aggregation.
- If the components can not be dynamically added or removed, then the relationship is composition.

- Life of **item** is same as the **order**



Representation of composition

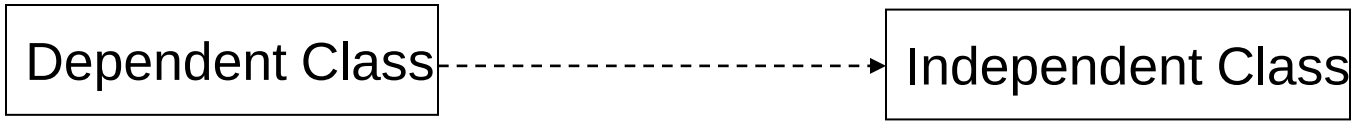
Inheritance Relationship



Representation of the inheritance relationship

- *Issuable* and *reference* here are discriminators.
- The set of subclasses of a class having the same discriminator is called a partition.

Class Dependency



Representation of dependence between class