Unified Modeling Language

Class Diagram

- A class diagram can describe the static structure of a simple system.
- It shows how a system is structured rather than how it behaves.
- The static structure of a more complex system usually consists of a number of class diagrams.
- Each class diagram represents classes and their inter-relationships.
- Classes may be related to each other in four ways:
  - Generalization,
  - Aggregation,
  - Association,
  - Different dependencies.

Class Diagram

Classes:

- A class represents entities (objects) with common features. That is, objects having similar attributes and operations constitute a class.
- A class is represented by a solid outlined rectangle with compartments.
- Classes have a mandatory name compartment where the name is written centered in boldface.
- The class name is usually written using mixed case convention and begins with an uppercase (e.g., LibraryMember).
- Object names are written using a mixed case convention, but starts with a small case letter (e.g., studentMember).
- Class names are usually chosen to be singular nouns.

Class Diagram

Representations of Classes:

Classes can optionally have attributes and operations compartments. When a class appears in several diagrams, its attributes and operations are suppressed on all but one diagram.

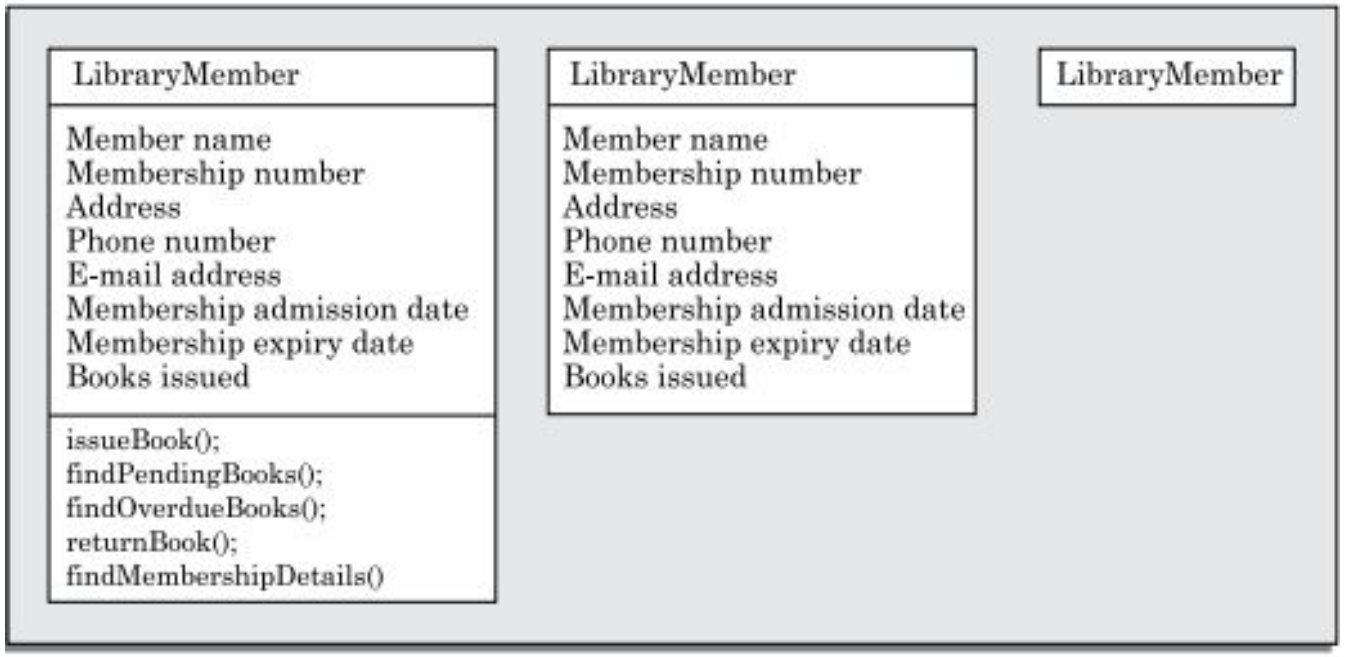

Figure: Different representations of the LibraryMember class.

At the start of the design process, only the names of the classes is identified. This is the most abstract representation for the class. Later in the design process the methods for the class and the attributes are identified and the more elaborate represents are used.

Class Diagram

Attributes:

- An attribute is a named property of a class. It represents the kind of data that an object might store.
- Attributes are listed by their names, and optionally their types (that is, their class, e.g., Int, Book, Employee, etc.), an initial value, and some constraints may be specified.
- Attribute names begin with a lower case letter, and are written left-justified using plain type letters.
- An attribute name may be followed by square brackets containing a multiplicity expression, e.g., sensorStatus[10]. The multiplicity expression indicates the number of attributes that would be present per instance of the class.
- The type of an attribute is written by following the attribute name with a colon and the type name, (e.g. sensorStatus:Int).

Class Diagram

Association:

- Association between two classes is represented by drawing a straight line between the concerned classes.
- The name of the association is written alongside the association line.
- An arrowhead may be placed on the association line to indicate the reading direction of the annotated association name.
- On each side of the association relation, the multiplicity is either noted as a single number or as a value range. The multiplicity indicates how many instances of one class are associated with one instance of the other class. Value ranges of multiplicity are noted by specifying the minimum and maximum value, separated by two dots, e.g., 1..5.

Class Diagram

Association:

- An asterisk is used as a wild card and means many (zero or more). The association of the given figure should be read as “Many books may be borrowed by a LibraryMember and a book may be borrowed by exactly one member”.

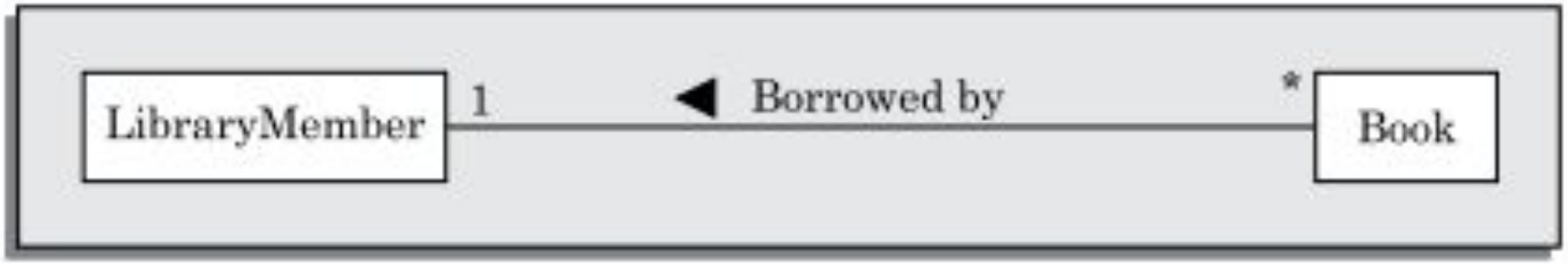

Figure: Association between two classes

Class Diagram

Aggregation:

Aggregation is a special type of association relation where the involved classes are not only associated to each other, but a whole-part relationship exists between them. That is, an aggregate object not only “knows” the ids of its parts (and therefore can invoke the methods of its parts), but also takes the responsibility of creating and destroying its parts.

Aggregation is represented by an empty diamond symbol at the aggregate end of a relationship.

Class Diagram

Aggregation:

An example of the aggregation relationship has been shown in the figure. The figure represents the fact that a document can be considered as an aggregation of paragraphs. Each paragraph can in turn be considered as an aggregation of sentences. Observe that the number 1 is annotated at the diamond end, and the symbol * is annotated at the other end. This means that one document can have many paragraphs. On the other hand, if we wanted to indicate that a document consists of exactly 10 paragraphs, then we would have written number 10 in place of the symbol *.

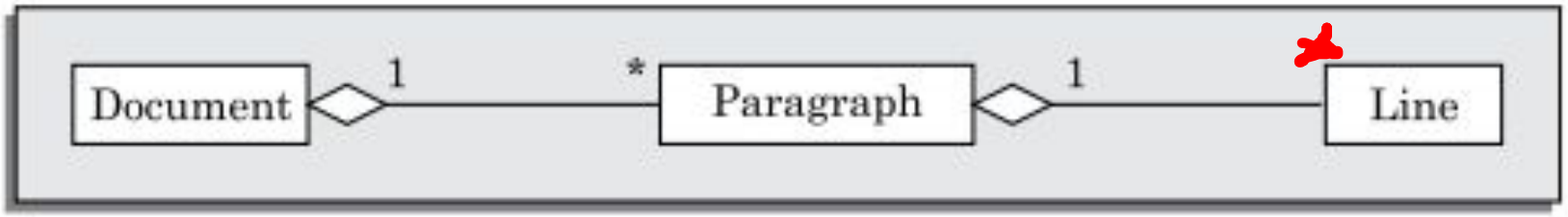

Figure: Representation of aggregation

Class Diagram

Composition:

Composition is a stricter form of aggregation, in which the parts are existence-dependent on the whole. This means that the lifeline of the whole and the part are identical. When the whole is created, the parts are created and when the whole is destroyed, the parts are destroyed.

Example:

- An example of composition is an order object where after placing the order, no item in the order can be changed.
- If any changes to any of the order items are required after the order has been placed, then the entire order has to be cancelled and a new order has to be placed with the changed items.

Class Diagram

Composition:

- As soon as an order object is created, all the order items in it are created and as soon as the order object is destroyed, all order items in it are also destroyed.
- The life of the components (order items) is identical to that of the aggregate (order). The composition relationship is represented as a filled diamond drawn at the composite-end.

Figure: Representation of composition

Class Diagram

Aggregation Vs. Composition:

Both aggregation and composition represent part/ whole relationships. If components can dynamically be added to and removed from the aggregate, then the relationship is expressed as aggregation.

If the components are not required to be dynamically added/delete, then the components have the same life time as the composite, then the relationship should be represented by composition. In the implementation of a composite relationship, the component objects are created by the constructor of the composite object. Therefore, when a composite object is created, the components are automatically created. No methods in the composite class for adding, deleting, or modifying the component is implemented.

Class Diagram

Aggregation Vs. Composition:

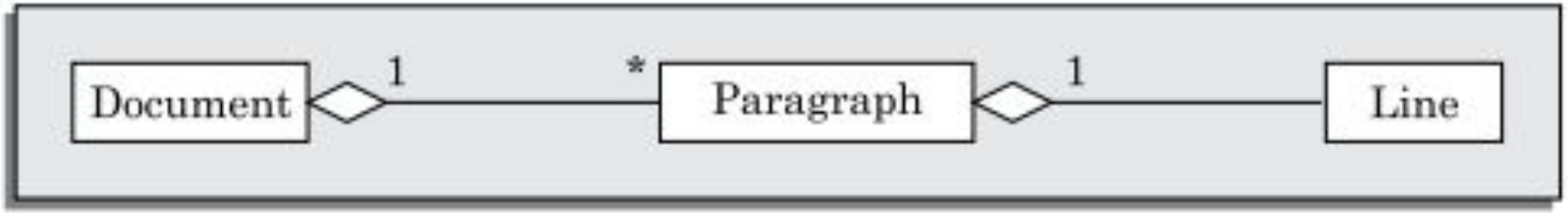

Figure: Representation of aggregation

Figure: Representation of composition

Class Diagram

Aggregation Vs. Composition:

We illustrate the subtle difference between the aggregation and composition relation between classes using the example of order and item objects. Consider that an order consists of many order items. If the order once placed, the items cannot be changed at all. In this case, the order is a composition of order items. However, if order items can be changed (added, delete, and modified) after the order has been placed, then aggregation relation can be used to model the relationship between the order and the item classes.

Class Diagram

Inheritance:

- The inheritance relationship is represented by means of an empty arrow pointing from the subclass to the superclass. The arrow may be directly drawn from the subclass to the superclass.
- When there are many subclasses of a base class, the inheritance arrow from the subclasses may be combined to form a single line (as shown in the figure) and is labelled with the specific aspect of the derived classes that is abstracted by the base class.
- The combined arrows emphasize the collectivity of the subclasses. This is useful when specialization for different groups derived classes has been done on the basis of some discriminators.

Class Diagram

Inheritance:

In the example of given figure, issuable and reference are the discriminators. This highlights the facts that some library book types are issuable, whereas other types are for reference.

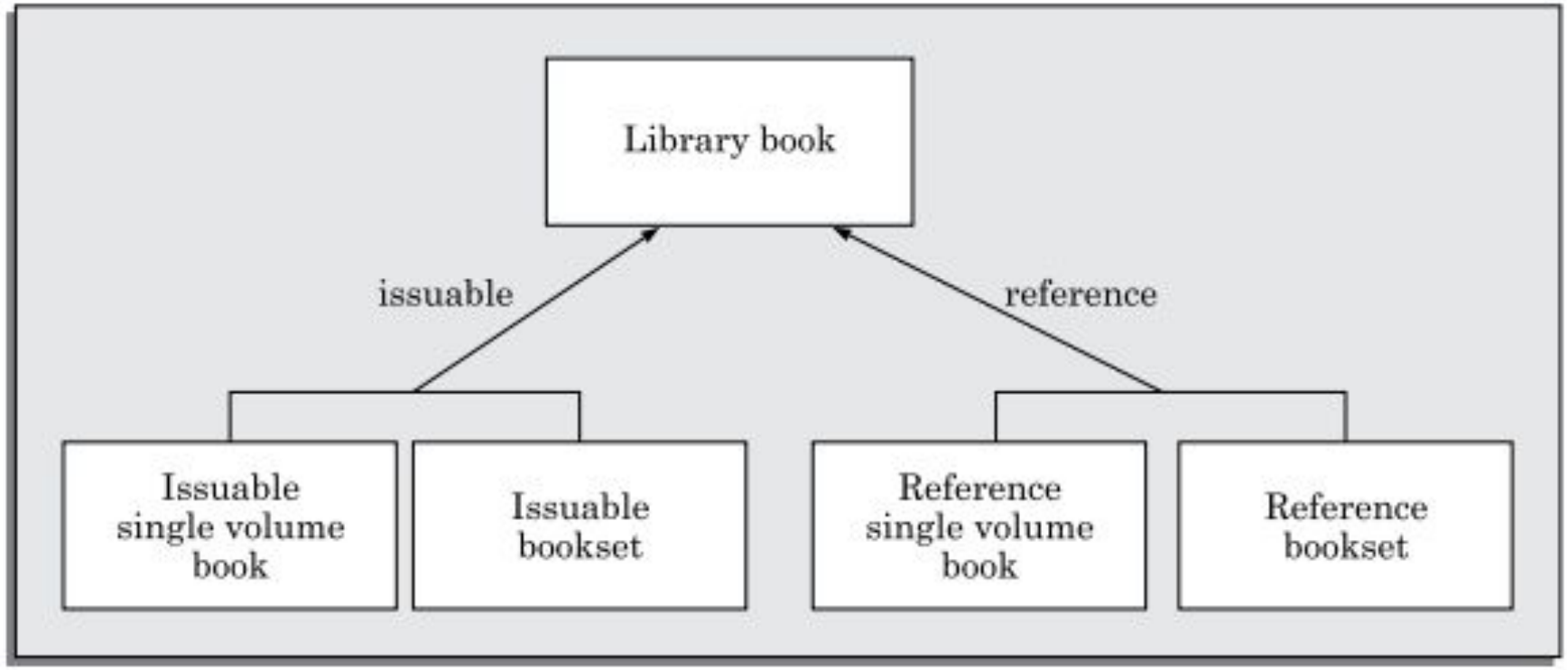

Figure: Representation of the inheritance relationship