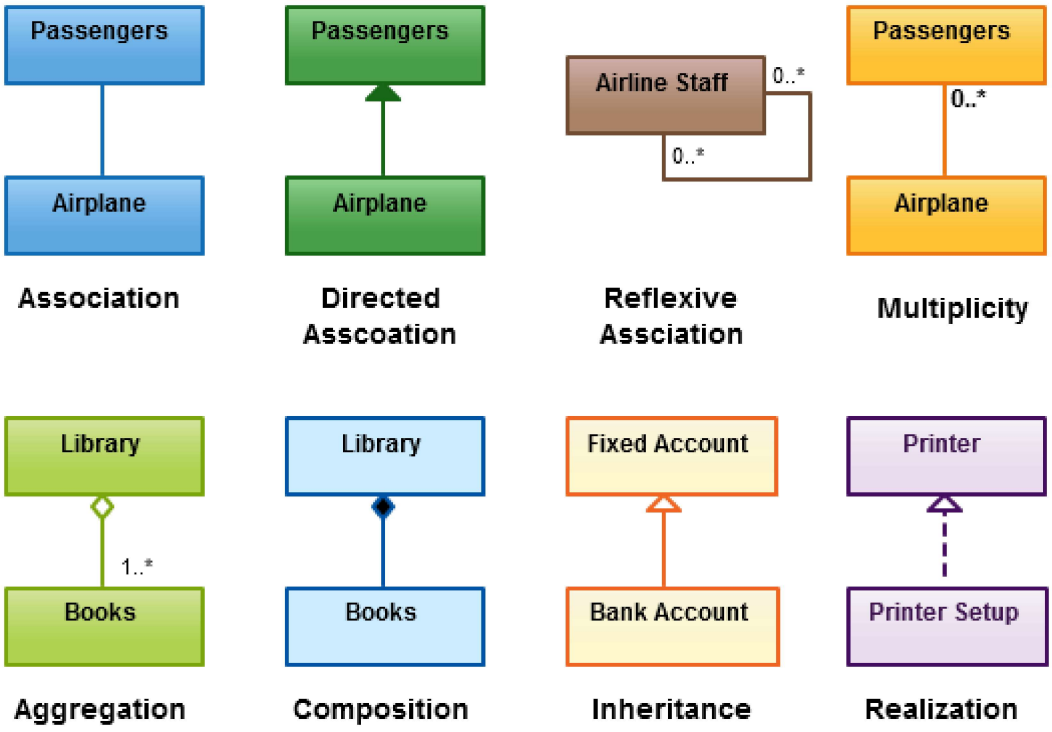


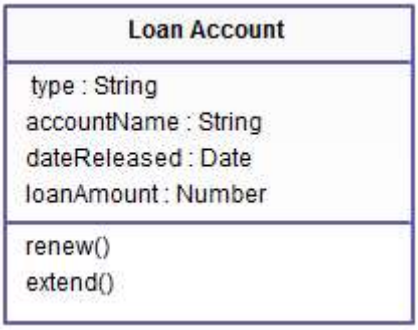
(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Class-Diagram-Relationships.png)

Relationships in UML class diagrams

What are Class Diagrams?

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

The following figure is an example of a simple class:



(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Class-Diagram.jpeg)

Simple class diagram with attributes and operations

In the example, a class called “loan account” is depicted. Classes in class diagrams are represented by boxes that are partitioned into three:

1. The top partition contains the name of the class.
2. The middle part contains the class’s attributes.
3. The bottom partition shows the possible operations that are associated with the class.

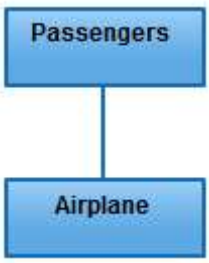
The example shows how a class can encapsulate all the relevant data of a particular object in a very systematic and clear way. A class diagram is a collection of classes similar to the one above.

Relationships in Class Diagrams

Classes are interrelated to each other in specific ways. In particular, relationships in class diagrams include different types of logical connections. The following are such types of logical connections that are possible in UML:

- **Association**
- **Directed Association**
- **Reflexive Association**
- **Multiplicity**
- **Aggregation**
- **Composition**
- **Inheritance/Generalization**
- **Realization**

Association

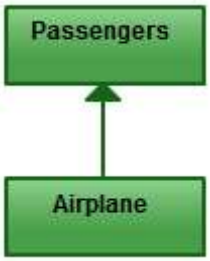


(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Association-Relationship.jpeg)

Association

is a broad term that encompasses just about any logical connection or relationship between classes. For example, passenger and airline may be linked as above:

Directed Association



(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Directed-Association-Relationship.jpeg)

Directed Association

refers to a directional relationship represented by a line with an arrowhead. The arrowhead depicts a container-contained directional flow.

Reflexive Association

(https://d3n817fwly711g.cloudfront.net/blog/wp-

content/uploads/2012/03/Reflexive-Association-Relationship.jpeg)

Reflexive Association

This occurs when a class may have multiple functions or responsibilities. For example, a staff member working in an airport may be a pilot, aviation engineer, a ticket dispatcher, a guard, or a maintenance crew member. If the maintenance crew member is managed by the aviation engineer there could be a managed by relationship in two instances of the same class.

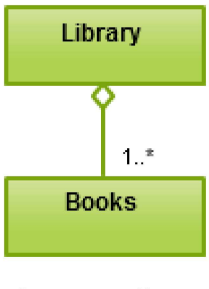
Multiplicity

(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Multiplicity-Relationship.jpeg)

Multiplicity

is the active logical association when the cardinality of a class in relation to another is being depicted. For example, one fleet may include multiple airplanes, while one commercial airplane may contain zero to many passengers. The notation 0..* in the diagram means “zero to many”.

Aggregation



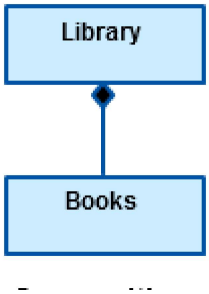
(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Aggregation-Relationship.png)

Aggregation

refers to the formation of a particular class as a result of one class being aggregated or built as a collection. For example, the class “library” is made up of one or more books, among other materials. In aggregation, the contained classes are not strongly dependent on the lifecycle of the container. In the same example, books will remain so even when the library is dissolved. To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.

To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.

Composition



(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Composition-Relationship-UML.png)

Composition

The composition relationship is very similar to the aggregation relationship. with the only difference being its key purpose of emphasizing the dependence of the contained class to the life cycle of the container class. That is, the contained class will be obliterated when the container class is destroyed. For example, a shoulder bag's side pocket will also cease to exist once the shoulder bag is destroyed.

To show a composition relationship in a UML diagram, use a directional line connecting the two classes, with a filled diamond shape adjacent to the container class and the directional arrow to the contained class.

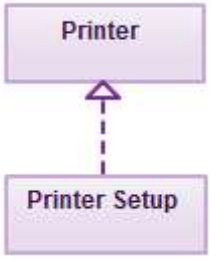
Inheritance / Generalization

(https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Inheritance-Relationship.jpeg)

Inheritance

refers to a type of relationship wherein one associated class is a child of another by virtue of assuming the same functionalities of the parent class. In other words, the child class is a specific type of the parent class. To show inheritance in a UML diagram, a solid line from the child class to the parent class is drawn using an unfilled arrowhead.

Realization



(https://d3n817fwly711g.cloudfront.net/blog/wp-

content/uploads/2012/03/Realization-Relationship.jpeg)

Realization

denotes the implementation of the functionality defined in one class by another class. To show the relationship in UML, a broken line with an unfilled solid arrowhead is drawn from the class that defines the functionality of the class that implements the function. In the example, the printing preferences that are set using the printer setup interface are being implemented by the printer.

Drawing class diagrams using Creately

We've given a lot of thought to relationships when we built our **class diagramming tools (https://creately.com/diagram-type/class-diagram?utm_source=classrelationships&utm_medium=blog&utm_campaign=tutorialposts)**. Our connectors adjust to the context and show only the most logical relationships when connecting classes. This significantly reduced your chances of making a mistake.

Drawing from scratch can be cumbersome. You can get started immediately using our professionally designed class diagrams. Browse our **class diagram templates (https://creately.com/diagram-type/templates/class-diagram?utm_source=classrelationships&utm_medium=blog&utm_campaign=tutorialposts)** and pick the one that's closely related to your system.

Any more questions about class diagram relationships?

I hope I've clearly explained the various relationships between class diagrams. They are not as complex as you think and can be mastered with some practice. And by using our tool you shouldn't have any trouble coming up with class diagrams. If you have any more questions don't hesitate to ask in the comments section. Also checkout this guide to **UML Diagram Types (https://creately.com/blog/diagrams/uml-diagram-types-examples/)** with Examples for further reading.

References: