What is a Class?

A Class is a blueprint for an object while objects are instances of classes. Each Object is built from the same set of blueprints and therefore contains the same components (properties and methods).

A class diagram is Unified Modeling Language (UML) diagram is type of static structure that describes the structure of the system by showing the system's –

- classes,
- their attributes,
- operations (or methods),
- relationships among objects.

UML Class Notation

A class represents a concept which encapsulates state (attributes) and behavior (operations). Each attribute has a type. Each operation has a signature.

Class Name:

- appears in the first partition.

Class Attributes:

- are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

Class Operations (Methods):

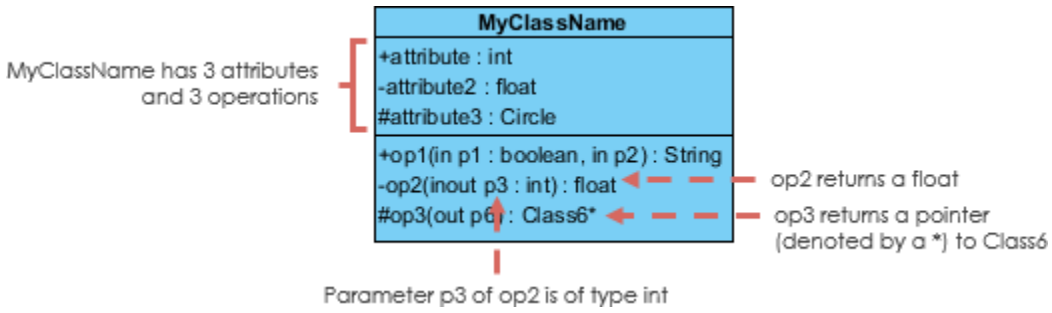
- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.

- The return type of method parameters are shown after the colon following the parameter name.

Class Visibility-

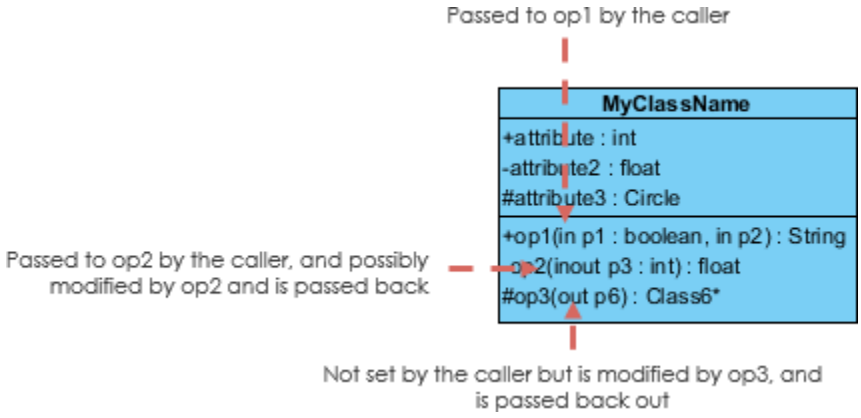
The +, - and # symbols before an attribute and operation name denote the visibility of the attribute and operation.

+ denotes public attributes or operations
- denotes private attributes or operations
# denotes protected attributes or operations



Parameter Directionality

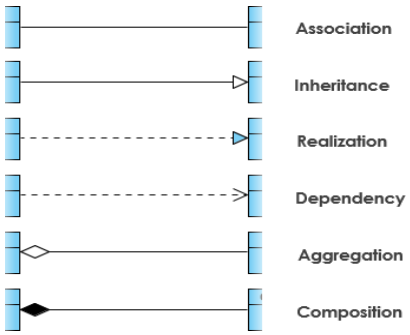
Each parameter in an operation (method) may be denoted as in, out or inout which specifies its direction with respect to the caller. This directionality is shown before the parameter name.



Relationships between Classes-

UML class diagram relationships show how one class affects another .

Different kinds of relationship annotations-



Association-

Association is the most basic of relationships. Association means any type of relationship or connection between classes.

Types of association relation-

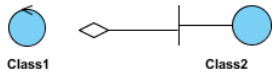
Directed association

The directed association is related to the direction of flow within classes. In a directed association, the flow is directed. The association from one class to another class flows in a single direction only.

The directed association between two classes is represented by a solid line with an arrowhead, which indicates the navigation direction. The flow of association from one class to another is always in one direction. For example, a person working in a company.

Aggregation

This arrow conveys that two classes are associated ,but not as close as in direct association. The child class can exist independent of the parent element. For example, a book still exists if somebody checks it out from the library



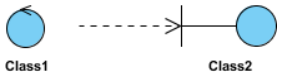
Composition

A special type of aggregation where parts are destroyed when the whole is destroyed.
Composition associations show relationships where the sub-object exists only as long as the container class exists. The classes have a common lifecycle. For example, a pocket on the front of a shirt cannot exist if we destroy the shirt. It is denoted by an arrow with filled Diamond Head.

Dependency

Dependency arrows show us where two elements depend on each other, but in a strong relationship than a basic association. Changes to the parent class also affect the child class. It shows supplier-client type relationship.

The relationship is displayed as a dashed line with an open arrow.

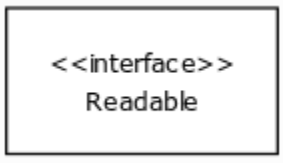


Realization/Implementation

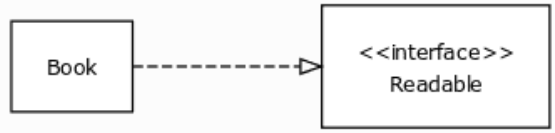
Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class. In other words, you can understand this as the relationship between the interface and the implementing class.

Describing an Interface

Interfaces in class diagrams are written <<interface>> Name of interface



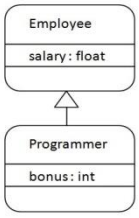
Methods are described just like they are for class.
Implementing an interface is shown as dashed arrow with a triangle arrowhead.



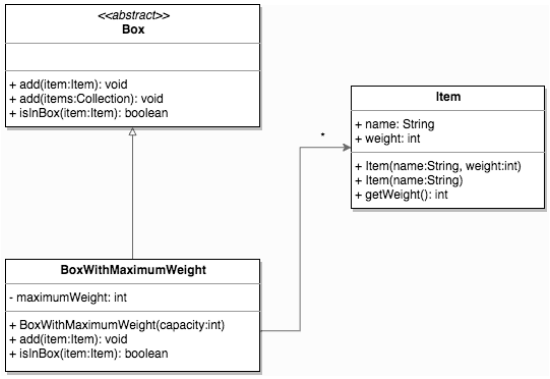
Inheritance **(or Generalization):**

Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific

classifier inherits the features of the more general classifier.
Represents an "is-a" relationship.
In a class diagram inheritance is described by an arrow with a triangle head. The triangle points to the class being inherited from.

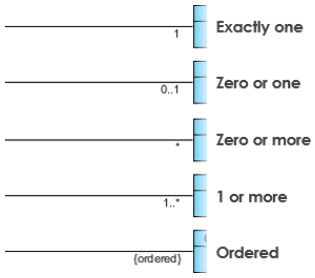


Inheritance of abstract classes is described almost the same way as regular classes. However we add the description <<abstract>> above the name of the class or class name is shown in italics. The name of the class and its abstract methods are also written in cursive



Multiplicity--Multiplicity or cardinality arrows show a place in our UML diagram where a class might contain many (or none) items. For example, a city bus might have any number of riders at a given time. Cardinality is expressed in terms of:

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



S

Example of a UML diagram for Mall Management System