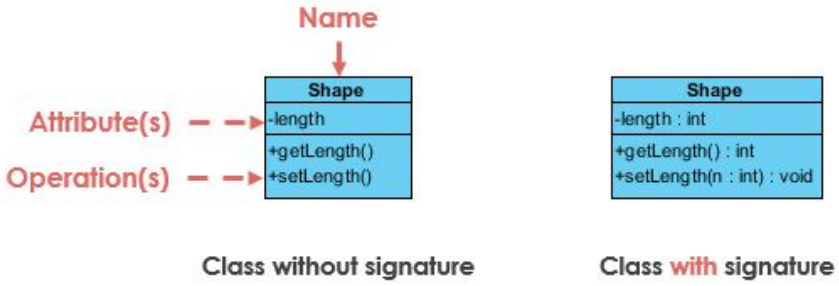
UML Class Diagram Tutorial

Aggregation and Composition

By Kevin O’Brien

UML Class Notation



Each Class encapsulates its attributes and operations.

Each Attribute has a type.

Each Operation has a signature.

For a UML Class Diagram, only the name of the class is required, but it is recommended that you show the class with its operations and attributes.

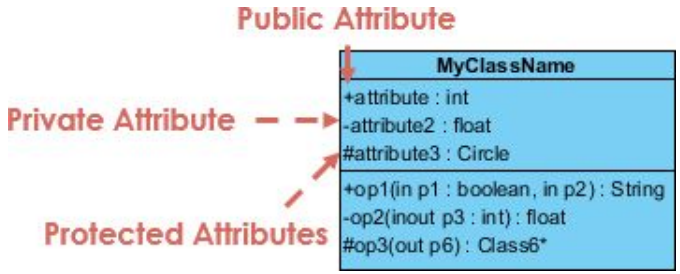
Class Visibility

The +, -, and # symbols are used to denote the visibility of attributes and operations.

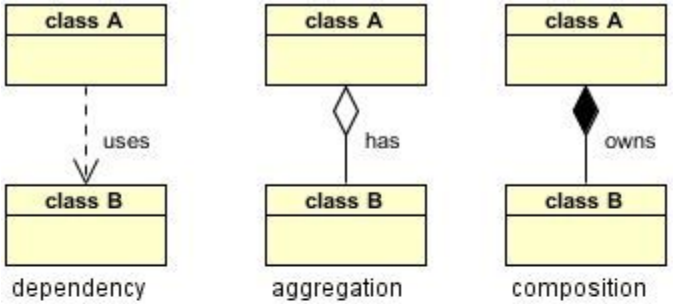
Public: +

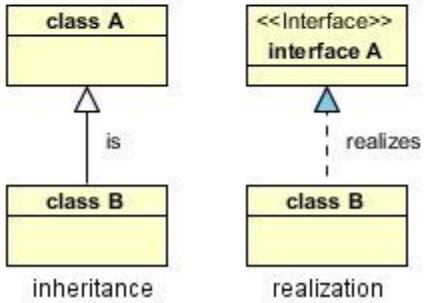
Private: -

Protected: #



Relationships between classes

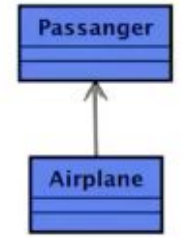




Association

Classes have a direct relationship with each other, but don't necessarily have attributes.

If two classes in a model need to communicate with each other, there must be a link between them, and that can be represented by an association (connector).



Association

In Java, Association can occur when an object of one class is declared INSIDE a method of another class. Because the object is declared inside a method, it is not an attribute. Thus, the relationship between the classes is just Association.

Aggregation and Composition

Aggregation and Composition are both subsets of Association.

In both aggregation and composition, an object of one class "owns" an object of another class, but there is a key difference.

In aggregation, the child can exist independently of the parent.

In composition, the child cannot exist independent of the parent.

Aggregation

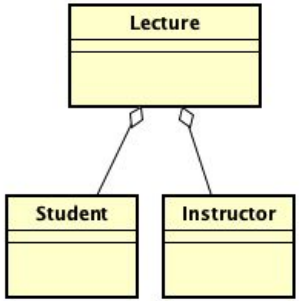
Aggregation - a relationship where the child can exist independently of the parent.

The aggregate class contains a reference to another class and is said to have ownership of that class.

Aggregation is the design technique to implement a *Has-A* relationship in classes.

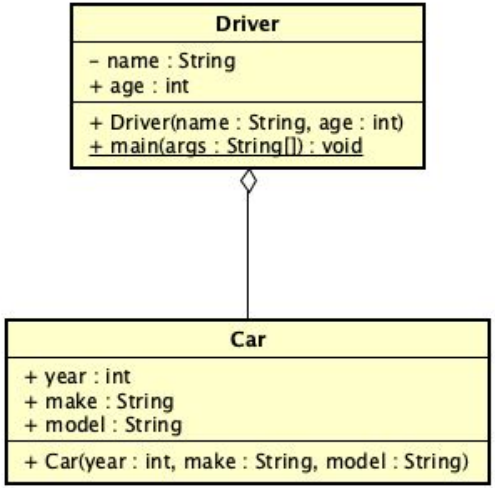
Weak Association

For example: Lecture (parent) and Student (child). Without the Lecture, the Instructor and the Students can still exist.



Aggregation

See Driver.java and Car.java



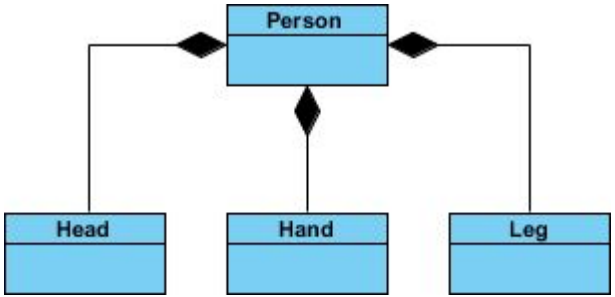
Composition

Composition - a relationship where the child cannot exist independent of the parent.

Composition is the design technique to implement a *Part-Of* relationship in classes.

Strong Association

Ex: Person (parent) and Head, Hand, and Leg (children). If you delete the Person, you delete all of their body. Therefore, the Head, Hand, and Leg cannot exist.



Composition

See Person.java and Hand.java