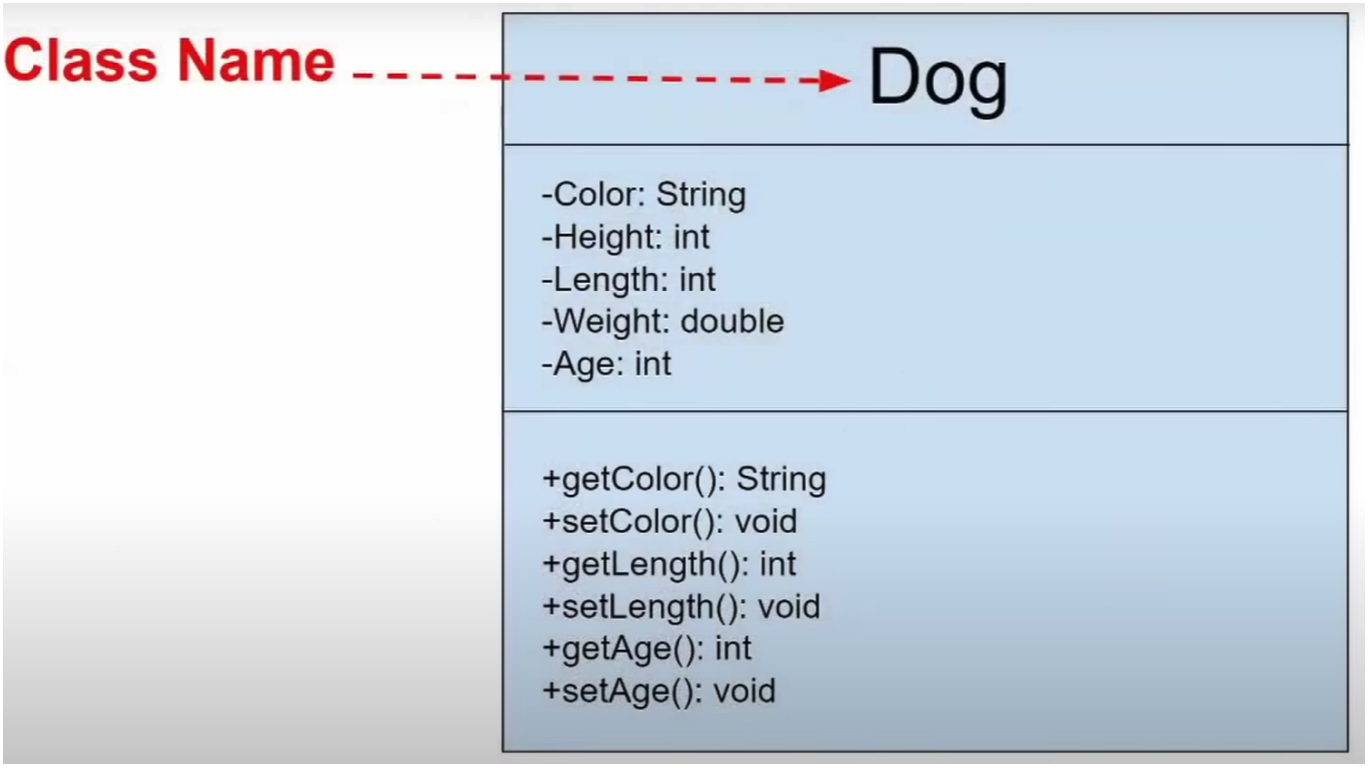
UML - Class Diagrams

- 1 or more Pets associated with 1 PetOwner
- Each pet has exactly one PetOwner

This is a way to use diagrams to visualize the structure of your code, this is used to plan out software systems.

In the class diagram, the classes are represented by boxes that contain three parts:

- The top part contains the name of the class
- The middle part contains the attributes of the class (variables)
- The bottom part contains the methods of the class



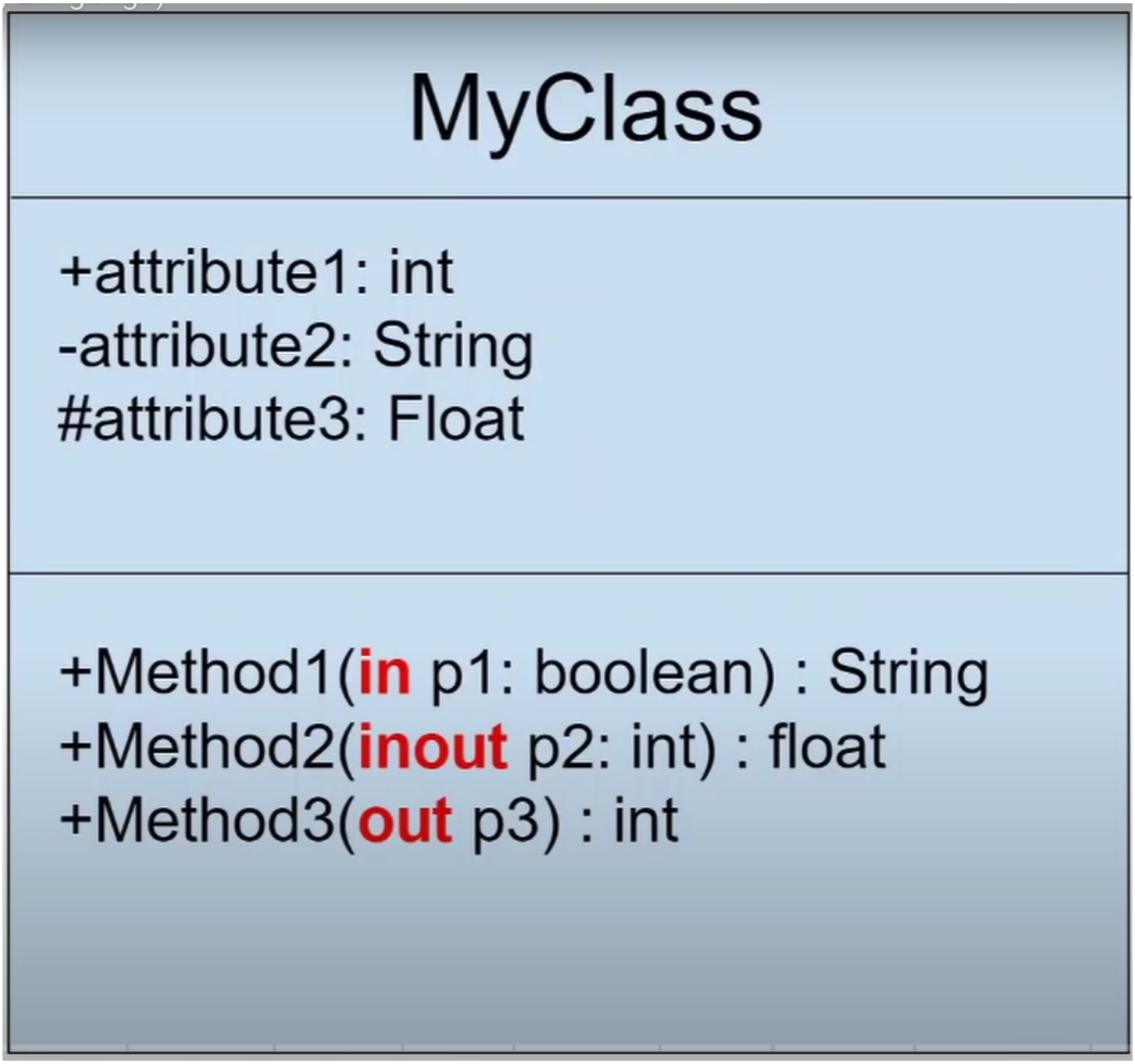
The variable types are written after the name of the variable, the method return type are written after the name of the method.

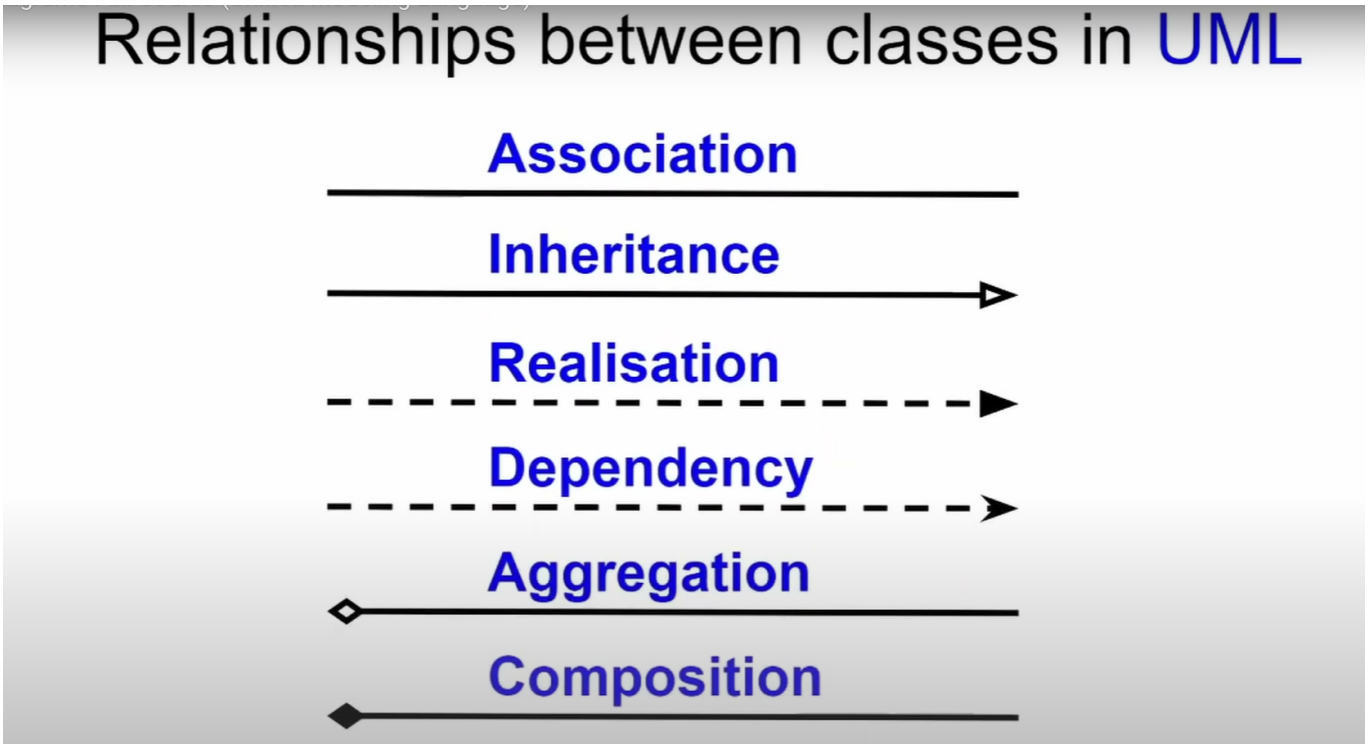
The minus and plus mean are the access modifiers.

Just like in java there are 4 access modifiers:

- (+) public
- (-) private
- (#) protected
- (~) package / no modifier

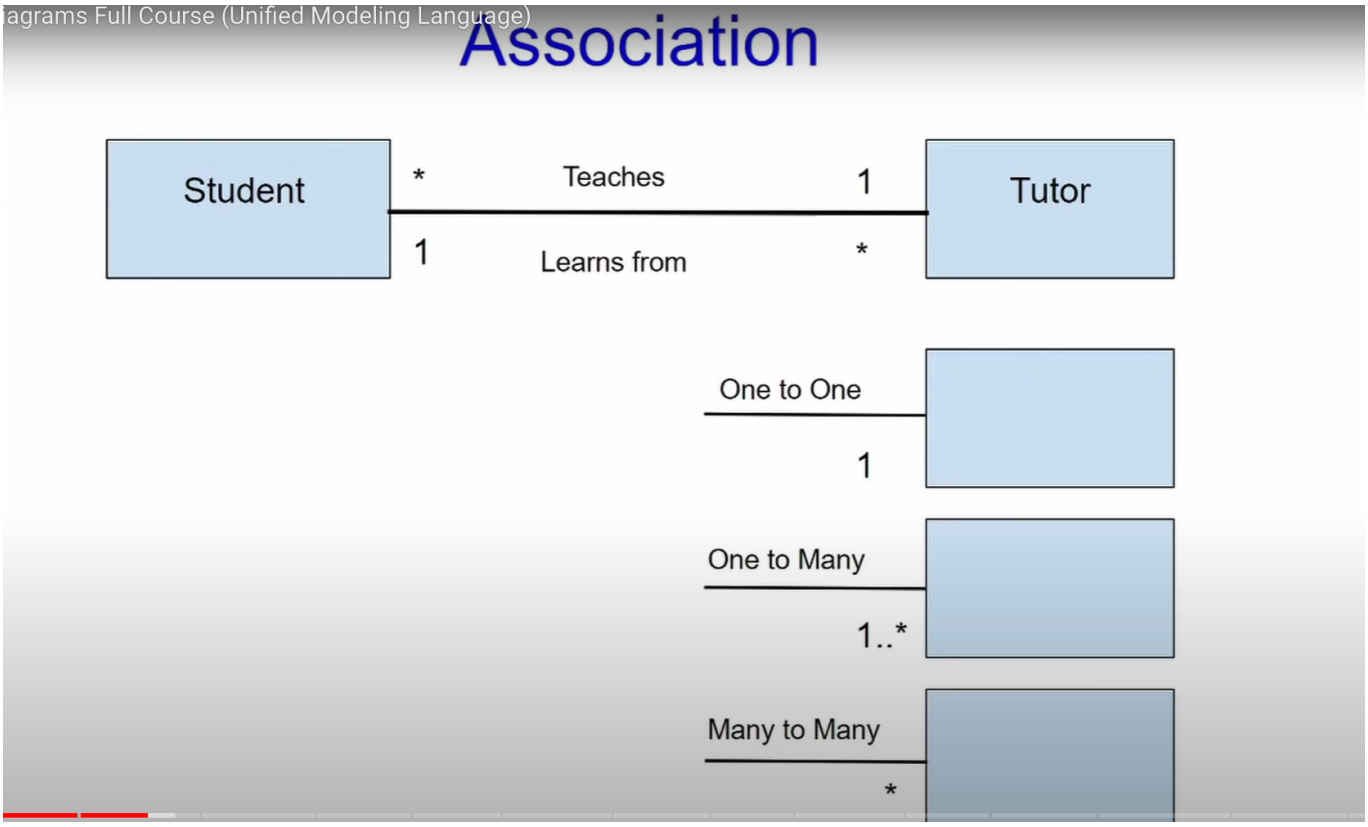
The parameters of the methods are written in the (), and how it affects the method





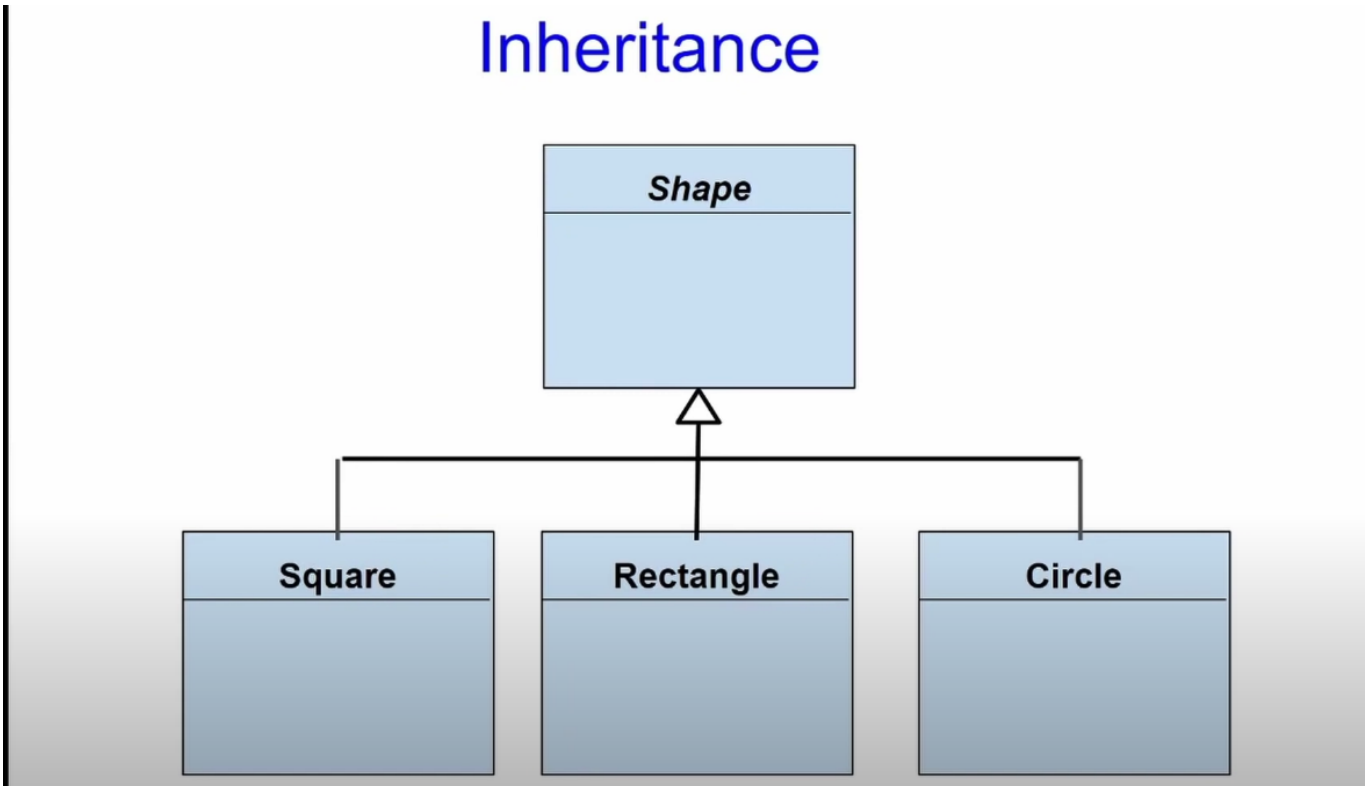
Associations

- Association - A relationship between two classes, a person works for a company.



This says on student can have many teachers, and a teacher can have many students. (Which is the object that are created)

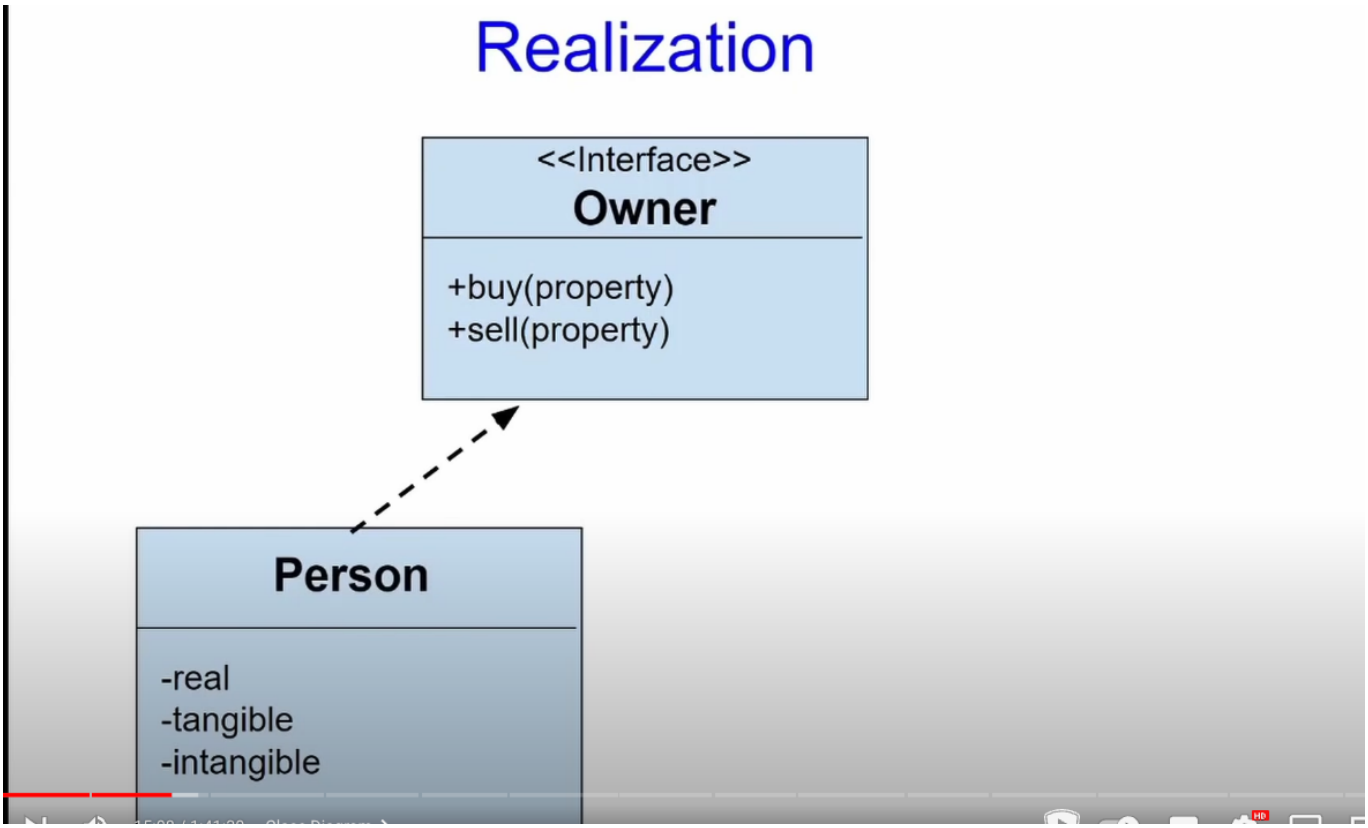
Inheritance



Inheritance is a relationship between two classes where one class is a child of the other class, a car is a vehicle. If it is an abstract class it is represented by an italicized name.

Realization

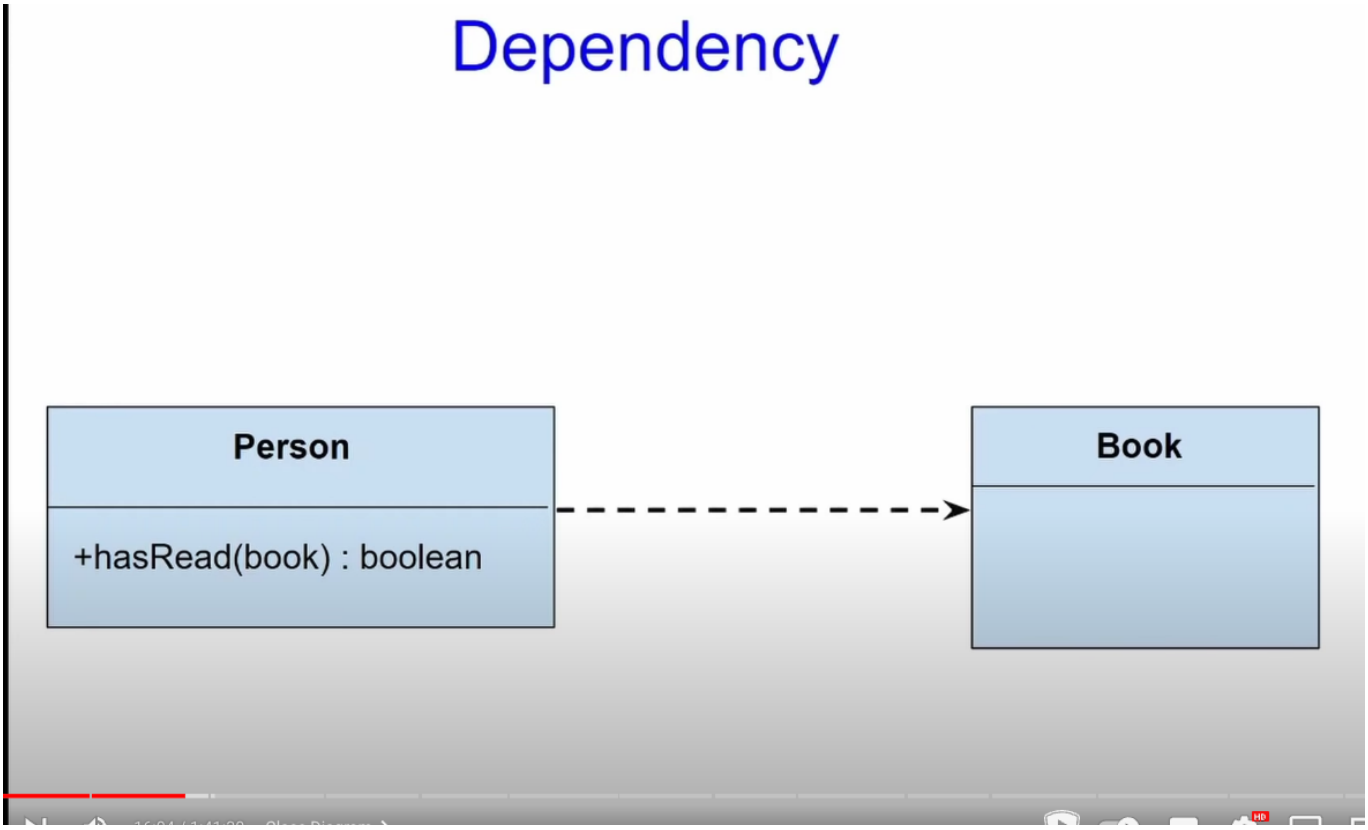
This is a relationship between two classes, where one class is an interface and the other class is an implementation of the interface.



Dependency

This is a relationship between two classes, where one class uses the other class.

When an object of one class uses an object of another class, not stored in any field, aka a septal type of association.

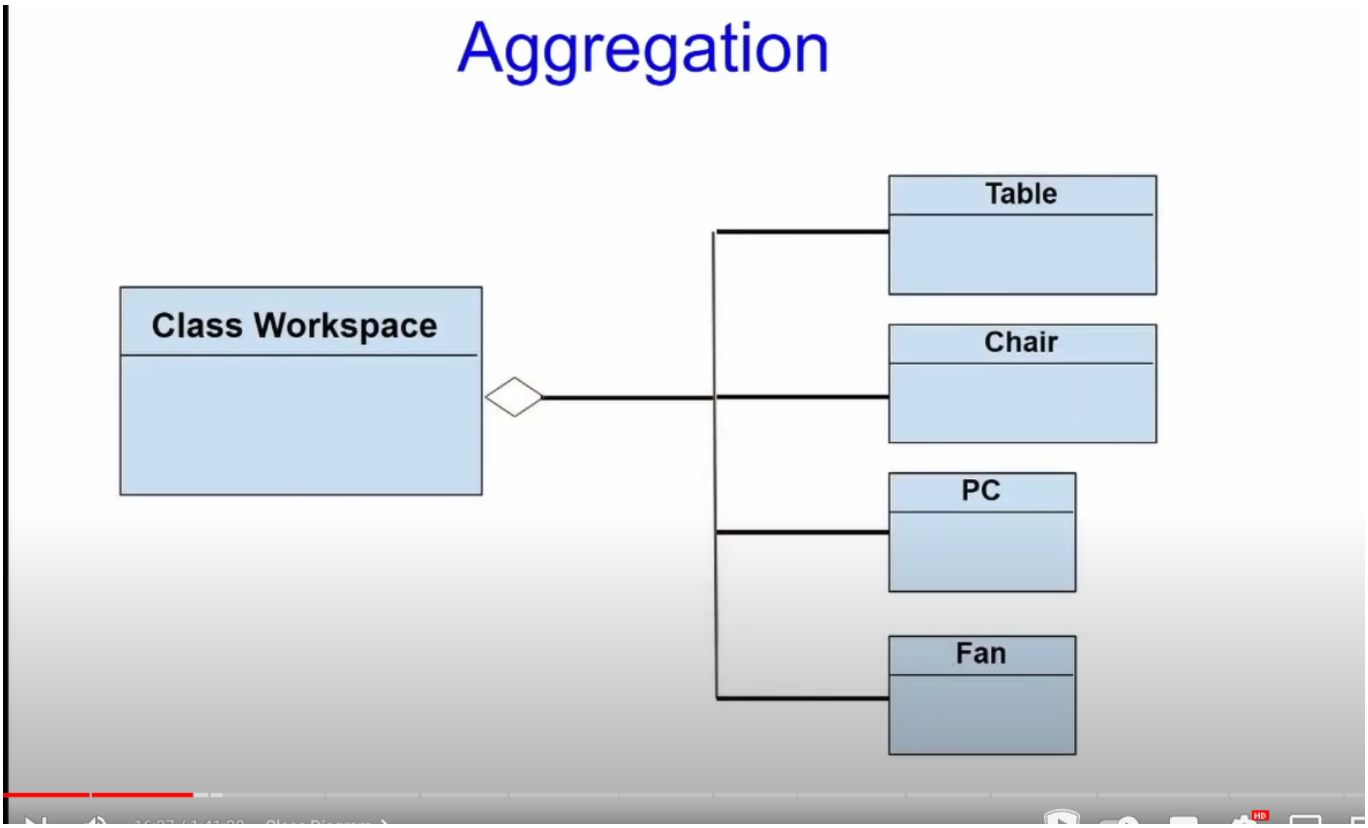


The person class, has a hasRead method that returns true, if the book has been read. Arrow point to the class of the object that is used.

Aggregation

This is a special type of relationship between two classes, where one class is a container for another class.

For example a programmer has a chair, desk and computer. When we deleted the workstation class, we still have all of these classes but on there own.



Composition

This is a special type of aggregation, where the container class owns the contained class.

For example our body is made up of organs, if we delete the body, we delete the organs.