OOP concepts

- delegation
- aggregation vs composition vs association
- coupling and cohesion
- inheritance vs composition

Delegation

Don't do all stuff by yourself, delegate it to respective class.

Classical example of delegation design principle is equals() and hashCode() methods. In order to compare two object for equality we ask class itself to do comparison instead of Client class doing that check.

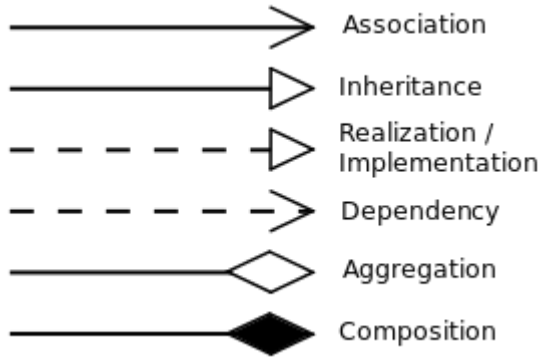
Benefit of this design principle is no duplication of code and pretty easy to modify behavior.

aggregation vs composition vs association

Aggregation implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.

Association is a relationship between classes of objects that allows one object instance to cause another to perform an action on its behalf (class and field).



coupling and cohesion

Coupling - A measure of how much a module (package, class, method) relies on other modules. It is desirable to reduce coupling, or reduce the amount that a given module relies on the other modules of a system.

If two modules communicate, they should exchange as little information as possible.

(for example, if you can return int[], you should not return List<Integer>)

Cohesion - A measure of how closely related the members (classes, methods, functionality within a method) of a module are to the other members of the same module. It is desirable to increase cohesion as that indicates that a module has a very specific task and does only that task.

(for example, if a class reads from the file, it should not read from the DB)

inheritance vs composition

Inheritance and composition are core OOP principles. Both about core reuse.

The question is: when we should use inheritance and when composition?

Is relation for inheritance

Has relation for composition

For example:

Dog has 4 legs. It is “has relation”.

We should not create a super class Quadruped, but class Leg and use *composition*.

Each object in Java is Object. It is “is relation”.

So, all classes in Java have methods of java.lang.Object