Mark Biswell

Object Oriented Class Relationships

Generalisation & Inheritance

Inheritance is the process of objects taking different methods and properties from other objects. A class that is inherited from is called a parent class, the class that takes the methods/properties is called the child class.

Generalisation referes to the process of essentially combining properties that 2 or more classes share to create a superclass.

Dependancy

Dependancy referes to a relationsip between two or more objects that mean they cannot be implemented correctly without all of the dependant elements being present. Dependacy is shown on a UML diagram using dashed arrows.

Aggregation & Composition

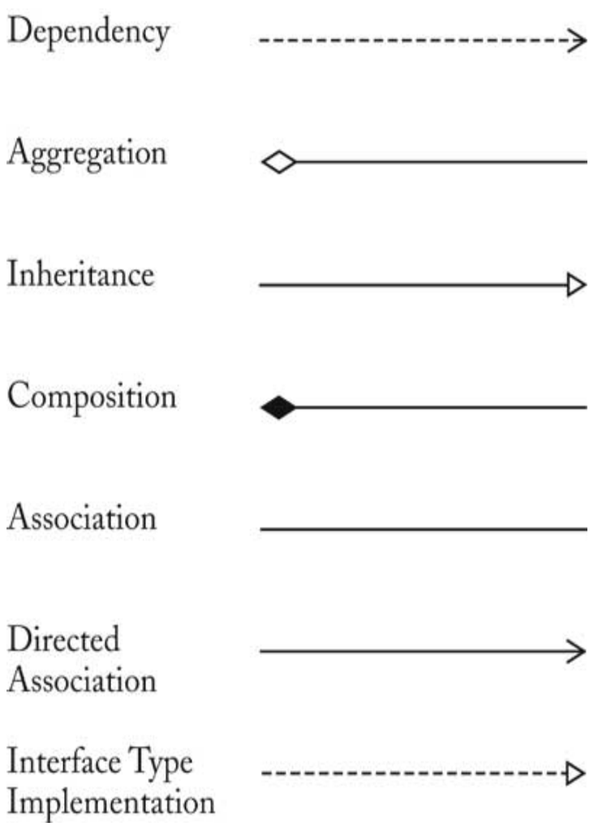
These terms can be seen as extentions of inheritance

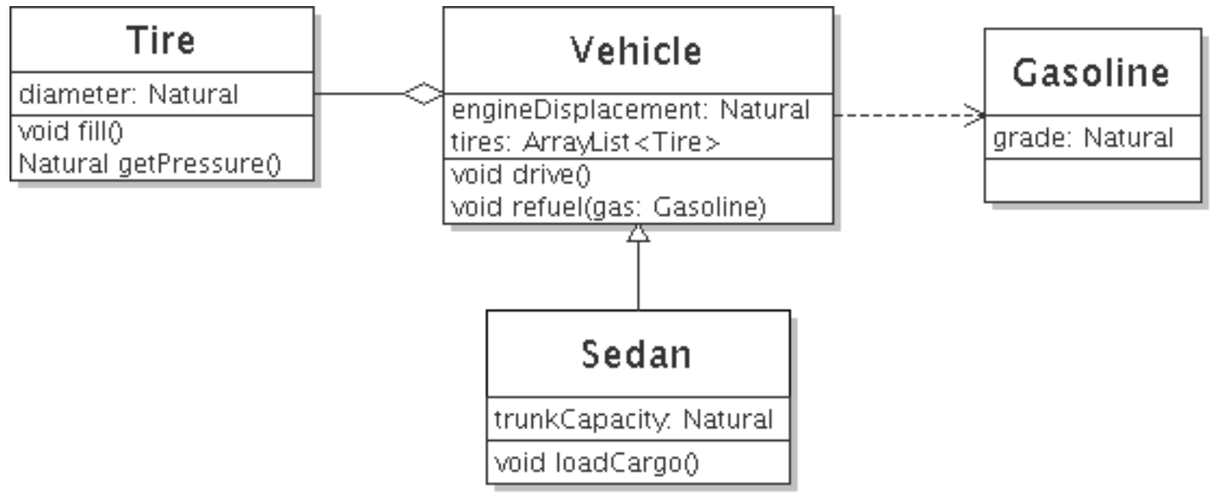
Aggregation means that the child object can be created without the need of the parent class already have beeing created.

Composition is the opposite, and referes to a relationship where the child object needs the parent object to be present. If the parent object were to be deleted, any child classes would also be deleted.

Realisation

A term used in UML modelling, Realisation describes a relationship between two elements in a model in which one element implements the behaviour that the other element specifies.





Loose & Tight Coupling

Coupling can be seen as a measurement of how dependand different classes are to eachother.

Loose coupling referes to the state of elements be more or less independent of one another and can function on their own.

Tight coupling is the opposite, refering to classes that are extremely depending on other classes to function. Tight coupling can be a result of trying to give one class too many methods.

Low & High Cohesion

Cohesion is a term used to describe how much different elements belong together. Essentailly a high cohesion means that the elements being refered to are very strongly related and low cohesion means the opposite. High cohesion is often seen as the signs of a reliable and robust program as all the elements of the program are connected and therefore can be easier to maintain/update.

Atomic Object. 2018. OOP Class Relationships. [ONLINE] Available at: https://atomicobject.com/resources/oo-programming/oo-class-relationships. [Accessed 25 November 2018].