Association is the link between two classes within a model that need to communicate with each other. When we need two classes within a model to speak to each other we need to create a link and to do this we represent this link as an association (connector).

Realisation is the implementation of the functionality defined in one class by another class. Additionally, to show this within a UML diagram you will need to use a broken line with an unfilled solid arrowhead which is drawn from the class that defines the functionality of the class that implements the function.

Within object-orientated design (OOD), Coupling refers to the degree of direct information that one element has about another, Additionally, how often do changes in class A force related changes within class B. Tight coupling is when two classes often change together. In simplifier terms if class A knows more than it should about how class B was implemented then class A and class B are tightly coupled. On the other hand, if the only information that class A has about class B is what class B shows through its interface then class A and class B are loosely coupled.

Dependency is a relationship where two or more objects, where an object will depend on other objects or object for its implementation. Additionally, if one of these objects change the other objects may be affected. To show this relationship within a UML diagram programmers and developers will use the dashed arrows.

Cohesion refers to what the class or module can do. A class with low cohesion would mean that this class does a lot of actions, it is a broad, unfocused class and does not know what it should do specifically, so it does a lot instead. A high cohesion class would mean that the class has a good idea about what its meant to be achieving and that it is focused. For example, only using/performing methods that relate to the intention of the class.

Composition is a fundamental concept within object-orientated programming (OOP). It depicts a class that references one or more objects within other classes in instance variables. Additionally, it has a lot of uses within the real world and is commonly used in carefully designing software components. Some benefits of this concept are it allows you to reuse existing code, design professional APIs and change the implementation of a class used in a compostion without adapting any external clients.

Generalization is the act of extracting shared characteristics from two or more classes; and putting them into one, generalized superclass. These essential characteristics can be properties, methods and attributes. On the contrary to Generalization, specialization is used to represent the ‘type of’ relationship by making subclasses from existing base classes.

Inheritance is an ability of a class that allows said class to extend one or more base classes. Additionally, Generalization is the strongest from of class relationships as the classes within a Generalization relationship are tightly grouped with one another. There are also 5 different types of Inheritance and they are single, multiple, multilevel and hierarchical and hybrid.

Single – one class extending another

Multiple - multiple base classes from which a class is derived

Multilevel – classes inherited from one another to from a chain

Hierarchical – classes that represent a hierarchical structure through inheritance

Hybrid – type of inheritance in which two or more forms of inheritance are combined into one

**Object Orientated Class Relationships**