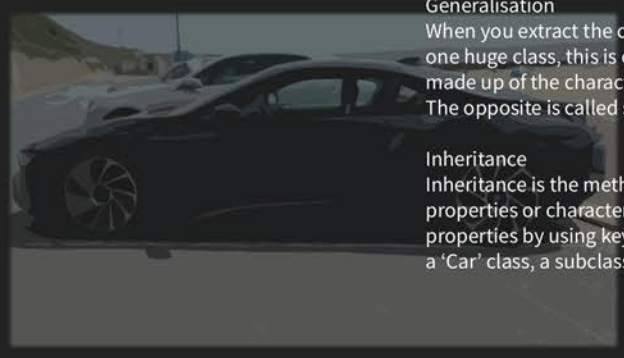
OBJECT ORIENTATED CLASS RELATIONSHIPS

BY LUKE BRUNI



GENERALISATION AND INHERITANCE

Generalisation

When you extract the characteristics from two or more classes and combine them to create one huge class, this is called generalisation. That huge class, also known as a superclass, is made up of the characteristics from the sub classes.
The opposite is called specialisation, splitting the characteristics to create subclasses.

Inheritance

Inheritance is the method of extending a base class to other existing classes to have properties or characteristics from said base class. That extended class then inherits the properties by using keywords to identify that extension. An example would be that if you had a 'Car' class, a subclass would be a make of car, i.e 'Nissan Skyline'.

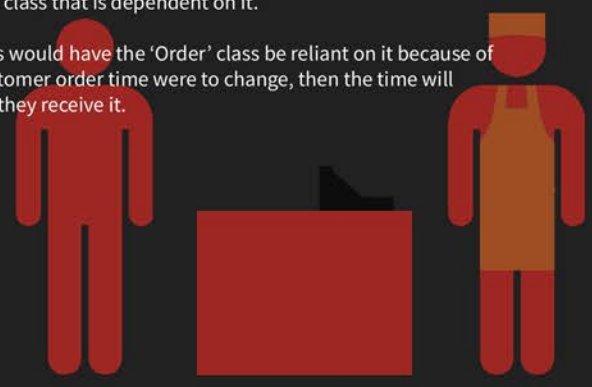
REALISATION

Realisation displays the relationship between a class and the object that has the characteristics, with said object realizing the blueprint class. An example would be a 'Cat' and a 'Dog' class being defined by a main class, known as 'Animals'.

DEPENDENCY

This is defined as where two classes are related in a specific way, as one class is reliant on another but that other class may or may not be as reliant. If one class where to change there would be a change in the class that is dependent on it.

For instance, a 'Customer' class would have the 'Order' class be reliant on it because of some of the features. If the customer order time were to change, then the time will change for the 'Order' class as they receive it.



AGGREGATION

First, we have association, where two components are linked to make a structural relationship. One class instance establishes another via their objects and can be one-to-many, many-to-many or one-to-one. Both classes can use the same entities and can exist without being reliant on one another.

Aggregation is a special variant of association but there is now an ownership of the entities and now the relationship morphs into a one-way association. Once both classes uses the same object, but there is an owner that holds that object.

LOOSE & TIGHT COUPLING

Coupling describes the amount of dependencies of a class and a subclass that it inherits. Coupling can be seen as loose or tight where each one affects the dependency of these classes.

Tight Coupling has the objects highly dependent on other interfaces and has to know about other objects. Should one object be changed, then the other objects will change with that object. Tight coupling is mainly used within a small application as the changes can be easily tracked.

Loose Coupling reduces the amount of dependencies between the objects and interfaces, making the code more flexible and reduces the risk of having multiple objects change alongside the original object. This makes loose coupling more desirable in larger applications.

```
namespace
TightCoupling
{
    public class Remote
    {
        private  Television Tv  { get; set;}
        protected Remote()
        {
            Tv = new Television();
        }

        static Remote()
        {
            _remoteController = new Remote();
        }
        static Remote _remoteController;
        public static Remote Control
        {
            get
            {
                return _remoteController;
            }
        }

        public void RunTv()
        {
            Tv.Start();
        }
    }
}
```

LOW & HIGH COHESION

Cohesion refers to what purpose a class or module has and how the functions within that class act on its purpose.

High Cohesion has the class have a clear purpose and is centred around that purpose, with the functions within that class being built to fulfil the class' purpose. This code is easy to understand as the objects with it being small and focused.

Low Cohesion is where the class has a mass amount of functions but doesn't fit the purpose of that class, so in turn, it leaves a load of code that is hard to read and can't be reused.

Low Cohesion

Staff
checkEmail() sendEmail() emailValidate() printLetter()

High Cohesion

Staff
-salary -emailAddr
setSalary(newSalary) getSalary() setEmailAddr(newEmail) getEmailAddr()

REFERENCES

Thorben Janssen (2017). OOP Concept for Beginners: What is Inheritance?. [online]. Stackify. Available from: <https://stackify.com/oop-concept-inheritance/>. [Accessed 29 November 2018].

Joydip Kanjilal (2016). Exploring generalization, specialization, and dependency in OOP. [online]. InfoWorld. Available from: <https://www.infoworld.com/article/3032175/application-development/exploring-generalization-specialization-and-dependency-in-oop.html>. [Accessed 29 November 2018].

Joe (2014). Association, Aggregation, Composition, Abstraction, Generalization, Realization, Dependency. [online]. JavaPapers. Available from: <https://javapapers.com/oops/association-aggregation-composition-abstraction-generalization-realization-dependency/>. [Accessed 29 November 2018].

Jasminder Singh (2015). Dependency, Generalization, Association, Aggregation, Composition in Object Oriented Programming. [online]. C# Corner. Available from: <https://www.c-sharpcorner.com/UploadFile/b1df45/dependency-generalization-association-aggregation-compos/>. [Accessed 29 November 2018].

Sarah El-Dawody (2016). Types Of Relationships In Object Oriented Programming (OOP). [online]. LinkedIn. Available from: <https://www.linkedin.com/pulse/types-relationships-object-oriented-programming-oop-sarah-el-dawody>. [Accessed 29 November 2018].

Thorben Janssen (2018). OOP Concepts for Beginners: What is Composition?. [online]. Stackify. Available from: <https://stackify.com/oop-concepts-composition/>. [Accessed 29 November 2018].

Anil Sharma (n.d). Understanding Loose Coupling and Tight Coupling. [online]. Net Stuff. Available from: <http://www.dotnet-stuff.com/tutorials/c-sharp/understanding-loose-coupling-and-tight-coupling>. [Accessed 29 November 2018].

Yusuf Karatoprak (2012). Difference Between Loose Coupling and Tight Coupling. [online]. C# Corner. Available from: <https://www.c-sharpcorner.com/UploadFile/yusufkaratoprak/difference-between-loose-coupling-and-tight-coupling/>. [Accessed 29 November 2018].

Radin Reth (2016). Difference Between Loose Coupling and Tight Coupling. [online]. Medium. Available from: <https://medium.com/@radinreth/what-does-low-coupling-and-high-cohesion-means-341498c3c97b>. [Accessed 29 November 2018].

Steven Lambert (2012). Quick Tip: The OOP Principle of Cohesion. [online]. EnvatoTuts. Available from: <https://gamedevelopment.tutsplus.com/tutorials/quick-tip-the-oop-principle-of-cohesion--gamedev-1811>. [Accessed 29 November 2018].