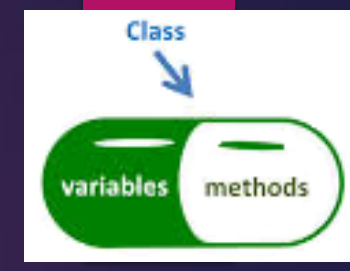
The Characteristics of an Object Oriented Paradigm

Polymorphism:

In object-oriented programming, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. So polymorphism is the ability (in programming) to present the same interface for differing underlying forms (data types). For example, in many languages, integers and floats are essentially polymorphic since you can add, subtract, multiply and so on, regardless of the fact that the types are different. They're rarely considered as objects in the usual term. So overall polymorphism is the ability to exist is various forms.



Encapsulation:

Encapsulation refers to the process of combining data and functions into a single unit. These data types are called 'classes', and one instance of a class is an 'object'. The data within is not accessed directly; it is accessed through the functions that are present inside the class. Encapsulation ensures good code modularity, which makes sure that routines are separate and less prone to conflict with each other. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. So, encapsulation makes the concept of data hiding possible.

Objects, Sub Objects & Containers:

Objects:
An Object is a component that is self-contained and it's properties and methods that are needed to make a certain type of data useful. An object's properties are what it knows and its methods are what it can do.

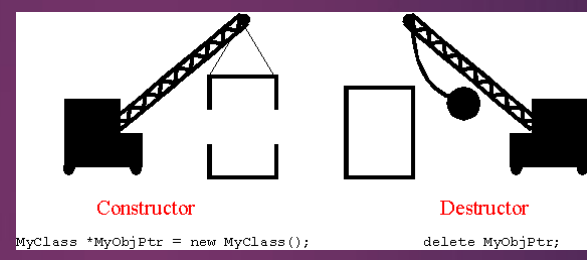
Sub Objects:
Sub Objects are known as the 'child object'. In relation to inheritance the sub object usually acquires all of the properties and behaviours of the parent object (except: constructors, destructor, overloaded operators and friend functions of the base class).

Containers:
A Container class is a class that was created to hold and categorise multiple instances of another type (either another class or a fundamental type). There is a huge range of different types of container classes and each have various advantages, disadvantages, and restrictions in their use. The most common used container is array.

Abstract & Concrete Classes:

Abstract:
Abstraction refers to the ability to filter out unnecessary or unimportant information; when you interact with objects in the world you only want to focus on that which is important for the task you are wanting to carry out. When applying abstraction it means that each object should only expose a high-level mechanism for using it. The mechanism should only be hiding internal implementation details and it should only be revealing operations that are relevant for the other objects.

Concrete:
Concrete classes are the types of classes that you have already encountered and worked with. They are used as the final implementation of a blueprint. The concrete class is complete in itself and can be extended by any class. The bottom-level classes must be concrete, otherwise no instances will ever be created that exhibit the defined behaviour.



Base & Derived Classes:

Base:
A base class is a class from which other classes are derived. It assists the creation of other classes that can reuse the code implicitly inherited from the base class apart from constructors and destructors.

Derived:
A class derived from a base class inherits both data and behaviour. For example a 'car' can be a base class from which 'engine' and 'steering' are derived. The derived class can communicate to the base class during instantiation by calling the base class constructor with a matching parameter.

Constructors & Destructors:

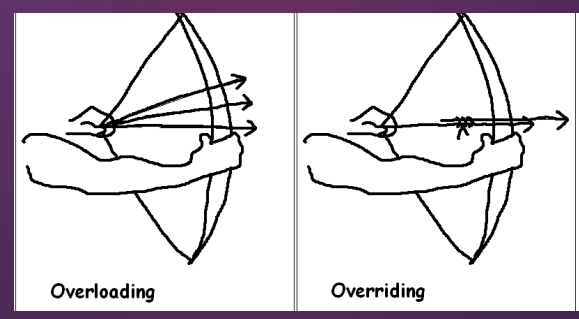
Constructors:
A constructor is a special class function which performs initialisation of every object. Whenever an object is being created the constructor is called upon by the compiler. The 3 different types of constructors are: Default, Copy and Parameterised. A constructor with no arguments is called a default constructor. It is the constructor that is defined essentially by the compiler. A copy constructor is called to make a copy of an object and finally parameterised constructors is a type of constructor that is used when we want to initialise the object with certain values.

Destructors
A destructor is a special method which is called upon during the destruction of an object. In OOP, a destructor gives an object a last chance to clean up any memory it allocated or perform any other tasks that must be completed before the object is destroyed.

Generic & Template:

The objective of generic programming is to write code that is independent of the data types. Within the C language, all codes are connected to a specific data type. For example, container data structures (such as array), you need to specify the type of the elements.

Template lets you program on generic type, instead of on a specific type. Template supports 'parameterized type' - i.e., you can use this type as an argument in building a class or a function. Template is extremely useful if a particular algorithm is to be applied to a variety of types, e.g., a container class which contains elements, possibly of various types.



Method Redefinition, Overriding & Overloading:

Method Overriding (Redefinition). It is useful when needed to extend the functionality of an inherited method. It is also a language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclass or parent classes. When implementing into the subclass it overrides(redefines) the implementation within the superclass which provides a method that has the same name, the same parameters and the same return type as the method within the parent class.

Method overloading means providing two separate methods in a class with the same name but different arguments, while the method return type may or may not be different, which allows us to reuse the same method name. A great advantage of method overloading is that it allows a programmer to use the function appropriately without having to know the inner-workings of that method.

References:

- *PC MAG* [online]. (1996-2018). Available from: <https://www.pcmag.com/encyclopedia/term/48238/object-oriented-programming>. [Accessed 26/11/18].
- Vangie Beal (2018). Polymorphism. [online]. *Webopedia*. Available from: <https://www.webopedia.com/TERM/P/polymorphism.html>. [Accessed 26/11/18].
- *Technopedia* [online]. (2018). Available from: <https://www.techopedia.com/definition/5656/constructor>. [Accessed 26/11/18].
- *Technopedia* [online]. (2018). Available from: <https://www.techopedia.com/definition/24284/destructor>. [Accessed 26/11/18].
- Alexander Petkov (2018). How to explain object-oriented programming concepts to a 6-year-old. [online]. *Freecodecamp*. Available from: <https://medium.freecodecamp.org/object-oriented-programming-concepts-21bb035f7260>. [Accessed 26/11/18].
- Aakash Mallik (2017). Interface vs Abstract class vs Concrete class. [online]. *Medium*. Available from: <https://medium.com/heuristics/interface-vs-abstract-class-vs-concrete-class-196f20c3af9a>. [Accessed 03/12/18].
- Michelle Yaiser (2011). Adobe [online]. Available from: <https://www.adobe.com/devnet/actionscript/learning/oop-concepts/objects-and-classes.html>. [Accessed 03/12/18].
- *Wikipedia* [online]. (2018). Available from: <https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)>. [Accessed 11/12/18].
- Alex (2018). *LearnCpp* [online]. Available from: <https://www.learncpp.com/cpp-tutorial/106-container-classes/>. [Accessed 11/12/18].
- *Technopedia* [online]. (2018). Available from: <https://www.techopedia.com/definition/26896/base-class>. [Accessed 11/12/18].
- *Wikipedia* [online]. (2018). Available from: <https://en.wikipedia.org/wiki/Method_overriding>. [Accessed 14/12/18].
- *Mytutor* [online]. (2018). Available from: <https://www.mytutor.co.uk/answers/3994/A-Level/Computing/What-is-method-Overloading-in-object-oriented-programming-OOP/>. [Accessed 14/12/18].
- *ntu* [online]. (2013). Available from: <http://www.ntu.edu.sg/home/ehchua/programming/cpp/cp8_Template.html>. [Accessed 14/12/18].