# Object Oriented Programming

* It is a methodology we use to design our programs just using classes and objects.
* It makes it easier to develop and maintain your project as it gets bigger.

## Object

* It is any entity that has a state and behavior

## Class

* Template that are used to create objects and define the object data types and methods
* They are a blueprint from which you can create multiple objects from

# Inheritance

* It is just a mechanism in which a class can acquire all the properties and methods of another class.
* It allows us to have code reusability and organization
* It allows us to use one of the tools in polymorphism (Method Overriding)

## Different types of inheritance

* Single inheritance
  + Where the child class inherits one parent class
* **Multilevel** inheritance
  + Like the single inheritance but the child class inherits from another child class that inherits from the parent class
* Hierarchical inheritance
  + Where the parent class has multiple child classes.

# Polymorphism

* The ability of an object to take on many forms
* It allows you to substitute different implementation/behaviors for different needs
* It can also be used to add more functionality to pre-existing code

## Method Overriding

* When a derived/child class changes the implementation details of a method from the base/parent class

## Method Overloading

* When there is multiple method but with different parameters and most of the time, different implementation/behavior details

# Encapsulation

* The process of wrapping code and data together into a single unit.
* So essentially, the validation/how you set a value and any processing of your data in your class will be handle by the class itself.
* They prevent unauthorized access to your object’s properties and setting values that shouldn’t be there.

# Abstraction

* The process of hiding the implementation/how something is done and only showing the functionality to the user.
* Simplify complex written code and just worrying about how it will function in your application
* Ex: You know how to send a text message in your phone, but do you really know how it is able to do that?

## Interface

* It contains nothing but **abstract** methods and properties
* You can add a method and not give it any code/implementation/function/behavior
* You can implement multiple interfaces (THIS IS NOT MULTIPLE INHERITANCE)
* You cannot create a constructor in an interface

## Abstract class

* May contain some methods and properties with implementation details
* You can create a constructor in an abstract class

# Access Modifiers

* They are a way to restrict access.
  + If you need a way to not have a method, be inherited from a parent class
  + If you need a property to only be accessible within the class

## Public

* Everything has access to it

## Internal

* Access within the class
* Access within the child class
* Access within the same project/assembly
* Default access modifiers for classes

## Protected

* Access within the class
* Access within the child class

## Private

* Access within the class
* This is the default access modifiers for class members (fields, methods, constructor, properties)

# Non-access Modifiers

## Abstract

* Enables you to create incomplete implementation of whatever you applied to, and it must be implemented by the child class.

## Static

* The class members belong to the class itself rather than a specific object
  + If one object changes the value of that static field, every object will also change that value.
* Static classes cannot be instantiated or inherited; its members must also be static.

## Const

* Constant fields may not be modified.
* Constants can be numbers, Boolean, values string, null reference.
* Don’t create a constant to represent something that will always change in the future.
* Must be initialized at declaration.
* They are implicitly static.

## Readonly

* That field can only be instantiated/set a value once
  + Can be instantiated/set a value later (mostly inside of a constructor)

## Sealed

* Sealed classes cannot be inherited by other classes
* Sealed methods, properties cannot be inherited by other classes

## Virtual

* Allows for a class member to be overridden in the child class

## Override

* It will look for the class member in the parent class and checks if the child class overrides that method.
* If you put override non-access modifier then that method must override something or else, it will give you an exception.

## Partial

* When you split a class into two or more files.
* When you want two or more developers work on the same class in C#.
* At compile time, it will “merge” those partial classes into one.