Inheritance

Inheritance is the ability that one class has of obtaining the attributes and methods from another class directly, excluding the need of declaring some objects. It provides a way to create a relationship between classes where the derived classes can reuse and extend the functionality of the base class, or superclass, helping the program to be better structured, providing code reuse, and avoiding object repetition. Inheritance allows the programmer to structure the code by using a superclass or a base class that will retain some attributes and methods that can be used by its derived classes or child classes, thus avoiding repetitions and making the even more organized.

Although Inheritance is very useful, we must be aware that overusing it might cause some problems to our code, consider yourself making an Inheritance chain or a super class that has a large number of derived classes such as thirty, and figure out that doing it the program would probably cause some error at a certain point becoming less organized due to the number of classes related to one super class. Therefore, if the plan is to create many classes, stead of having just one super class, reconsider creating different ones so the derived can be better divided, for example, if the program has 30 classes, divide them into 6 super classes so each super class would have 5 derived class or child classes, summing 30.

For example, let's say we want to create a program that displays the information of a car.

```
class  Vehicle {
        //This will be the super class or the base class//
        protected string _manufacturer{get; set;}
        protected int _Year{ get; set }
        public Vehicle(string manufacturer, int year)
          {
            _manufacturer = manufacturer;
            _year = year;
          }
}
```

```
class Car: Vehicle{
        //This will be the derived class or the child class
        private string _carName{get; set;};
        Public Vehicle(string _manufacturer, int _Year , string carName) base(string
_manufacturer, int _year)
         {
                _carName= carName;
        }
        public void renderCarinfo(){
        Console.Writeline($"{_manufacturer} - {_Year} – {_carName} ")
}


Using system
Class program{
        Static void Main(string[] args)
        carInfo._manufacturer = "Ford";
         carInfo._year = 2022;
        carInfo._carName = "Mustang"
        carInfo.RenderCarInfo();}
}
```

The example above shows how the Car class is reusing elements from the Vehicle super class without having to repeat its attributes.