## La Herencia en POO

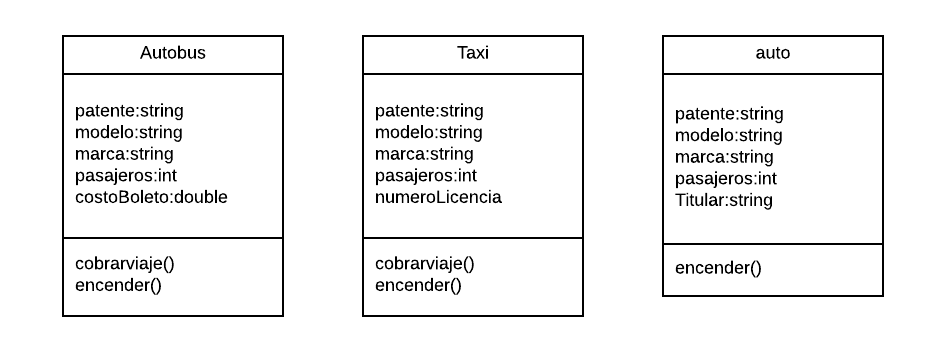
# HERENCIA

La herencia en programación nos permite definir una clase a partir de otra ya existente. Esta clase resultante de otra clase es denominada como “clase hija” y contiene todos los atributos y métodos de la clase existente conocida como “clase padre”. La principal ventaja de esto es que se evita duplicidad de los datos en clases que son prácticamente iguales o que posean los mismos atributos.

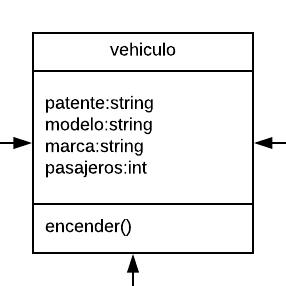
## Ventajas

1. Evita la **duplicidad**
2. favorecer la reutilización de código.
3. Facilitar el **mantenimiento** de aplicaciones. Podemos cambiar las clases que usamos fácilmente.
4. Facilitar la **extensión** de las aplicaciones. Podemos crear nuevas clases a partir de otras ya existentes.

Ejemplo:

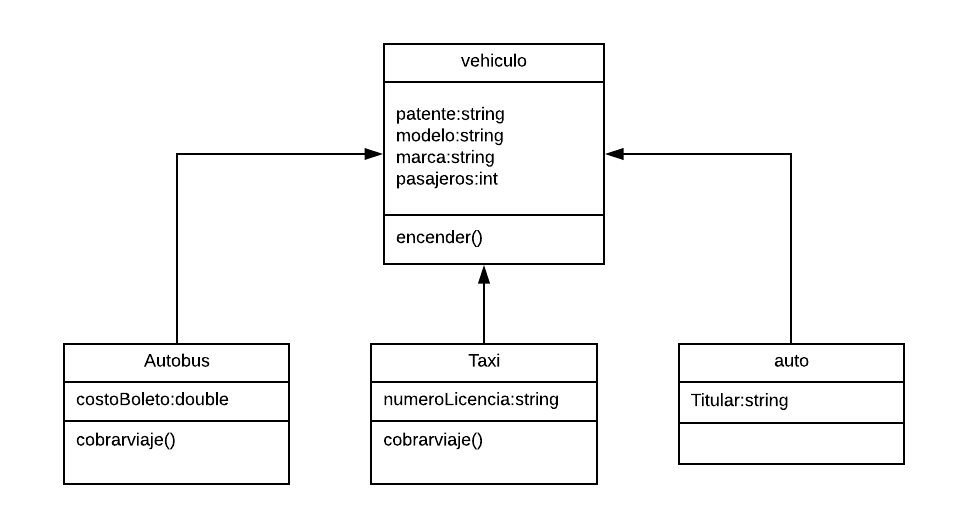
o

Como vemos en este ejemplo tenemos 3 clases que representan diferentes objetos de la vida real pero que son bastante similares en sus características. Para evitar tener que repetir los mismos datos en la creación de la clase, podemos crear una clase “auxiliar” que sirva para contener estos atributos y la cual pasaremos a conocer como clase **padre**. El nombre de esta clase **padre** va a estar definido por el grupo en común al que pertenecen las clases. Siguiendo con el ejemplo planteado podemos definir como clase ***padre*** la clase **vehiculo,** ya que todas las clases son vehículos, también se podría utilizar el nombre “automotor”.



Esta clase va a contener todos los atributos y métodos que sean comunes a todas las clases **hijas**.

Ahora que la clase padre contiene todos los atributos y métodos “repetidos” podemos quitarlos de las clases hijas, y solo indicar que dichas clases **heredan** de la clase padre **vehículo.** Esto se hace dibujando una flecha que parte desde cada clase hija hacia la clase padre.



Como vemos en el nuevo diagrama las clases autobús, taxi y auto se han simplificado bastante quedando con menos atributos y métodos, ya que los mismos son **heredados** de la clase padre **vehiculo.**

Las clases que **heredan** métodos y atributos de una clase **padre** se conocen como clases **hijas.**

Una clase padre puede ser también clase hija de otra clase. Por ejemplo:

**Producto → Ropa → Camiseta**

Donde producto es padre de la clase ropa y a la vez la clase ropa es padre de la clase camiseta.

**IMPORTANTE**: las flechas de herencia en el diagrama deben estar dibujadas partiendo de la clase hija hacia la clase padre y con la etiqueta “hereda de…”

## Herencia en c#

Ahora que entendemos las clases padres e hijas podemos pasar al código en c#.

Lo primero es definir nuestra clase padre, para la cual tomaremos la clase vehiculo del ejemplo anterior:

    class vehiculo {  
        **public** string *patente*;  
        **public** string *modelo*;  
        **public** string *marca*;  
        **public** **int** *pasajeros*;  
        **public** void **encender**(){}  
    }

Luego de definir la clase padre queda declarar las clases hijas, indicando la herencia hacia la clase padre de la siguiente forma:

    class taxi: vehiculo {  
        **public** **int** *numeroLicencia*;  
        **public** void **cobrarViaje**(){}  
  
    }      
    class autobus: vehiculo {  
        **public** **double** *costoboleto*;  
        **public** void **cobrarViaje**(){}  
  
    }  
  
    class auto: vehiculo {  
        **public** string *Titular*;  
    }

Los dos puntos indican que una clase hereda los atributos y métodos de otra.

ClaseHija : ClasePadre

Luego solo falta crear los objetos para cada una de las clases.

    class Program {  
        **public** static void **Main**(string[] args) {  
            taxi objetoTaxi = **new** taxi();  
            objetoTaxi.*patente* = Console.**ReadLine**();  
            objetoTaxi.*modelo* = Console.**ReadLine**();  
            objetoTaxi.*marca* = Console.**ReadLine**();  
            objetoTaxi.*pasajeros* = **int**.**Parse**(Console.**ReadLine**());  
            objetoTaxi.*numeroLicencia* = **int**.**Parse**(Console.**ReadLine**());  
            objetoTaxi.**encender**();  
            objetoTaxi.**cobrarViaje**();  
              
            autobus objetoAutobus = **new** autobus();  
            objetoAutobus.*patente* = Console.**ReadLine**();  
            objetoAutobus.*modelo* = Console.**ReadLine**();  
            objetoAutobus.*marca* = Console.**ReadLine**();  
            objetoAutobus.*pasajeros* = **int**.**Parse**(Console.**ReadLine**());  
            objetoAutobus.*costoboleto* = **double**.**Parse**(Console.**ReadLine**());  
            objetoAutobus.**encender**();  
            objetoAutobus.**cobrarViaje**();  
              
            auto objetoAuto = **new** auto();  
            objetoAuto.*patente* = Console.**ReadLine**();  
            objetoAuto.*modelo* = Console.**ReadLine**();  
            objetoAuto.*marca* = Console.**ReadLine**();  
            objetoAuto.*pasajeros* = **int**.**Parse**(Console.**ReadLine**());  
            objetoAuto.*Titular* = Console.**ReadLine**();  
            objetoAuto.**encender**();  
              
        }  
          
    }

**IMPORTANTE**: siempre tener en cuenta que en la mayoría de los casos las clases que van a ser utilizadas van a ser las clases hijas y no las clases padres.

## Actividad

Genera las siguientes aplicaciones utilizando herencia:

### 1.Empresa agropecuaria (Herencia)

se plantea desarrollar una aplicación que permita la gestión de una empresa agroalimentaria que trabaja con 2 tipos de productos, productos frescos, y productos congelados. los 2 productos tienen la siguiente información en común: nombre del producto y número de lote.

A su vez los productos frescos deben llevar el código de supervisión alimentaria y el país de origen.

Los productos congelados deben llevarla temperatura de mantenimiento recomendada y nombre de la empresa productora.

La aplicación debe permitir ingresar x cantidad de uno de estos productos y analizar el resultado dependiendo los productos ingresados:

para los productos congelados se debe poder determinar el porcentaje de productos donde la temperatura de mantenimiento es menor a 5 grados, y la cantidad de productos de la empresa Swift.

para los productos frescos se debe poder determinar la cantidad de productos que no tienen código de supervisión alimentaria

aplicar concepto de herencia en el desarrollo de la aplicación

### 2. Taxis (Herencia)

Crear una aplicación que sirva para registrar los recorridos de los taxis y autobuses de una empresa.

Para los taxis se necesita registrar el nombre y número de chofer y la patente, km de recorridos

Para los autobuses se registra nombre y número de chofer patente y la cantidad de pasajeros por recorrido.

Crear una aplicación que permita ingresar n recorridos de uno de los dos tipos de vehículos (taxi o autobuses) y calcular:

1. Para los taxis mostrar el nombre y patente del chofer que recorrió más km.
2. Para los autobuses calcular el promedio de pasajeros que viajaron ese día.