DIPLOMADO: Fundamentos de Programación Web
MODULO: Lógica de programación
INSTRUCTOR: Tec. Darwin Flores

Objetivos:

1. Ampliar los conocimientos sobre el lenguaje de programación C#.
2. Fortalecer las habilidades de análisis, lógica y resolución de problemas aplicando un
3. lenguaje de programación.
4. Construir aplicaciones utilizando los pilares de la Programación Orientada a Objetos.

Vocabulario de la POO

- Clase
- Objeto.
- Ejemplar de clase. Instancia de clase. Ejemplarizar una clase. Instanciar una clase.
- Modularización
- Encapsulamiento / encapsulación.
- Herencia
- Polimorfismo

Paradigma

Un paradigma es una forma de afrontar la construcción de código de software

- No hay paradigmas mejores ni peores
- Todos tienen sus ventajas e inconvenientes

Hay distintos paradigmas:

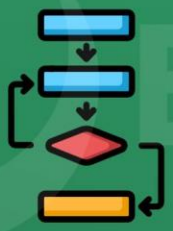
- POO, Estructurado, funcional, Lógico, etc.

¿QUÉ SON LOS PARADIGMAS DE PROGRAMACIÓN?

Los paradigmas son los diferentes estilo de usar la programación para resolver un problema.

PROGRAMACIÓN ESTRUCTURADA

Programación secuencial con la que todos aprendemos a programar. Usa ciclos y condicionales.



PROGRAMACIÓN ORIENTADA A OBJETOS

Divide los componentes del programa en objetos que tienen datos y comportamiento y se comunican entre sí.

PROGRAMACIÓN REACTIVA

Observa flujos de datos asincrónicos y reacciona frente a sus cambios.

PROGRAMACIÓN FUNCIONAL

Divide el programa en tareas pequeñas que son ejecutadas por funciones.

Programación Orientada a Objetos

- Facilidad de diseño y relación con el mundo real (UML)
- Reutilización de piezas de código (no copy/paste)
- Encapsulamiento (ocultar el estado de los objetos)

Elementos de la POO

Los elementos principales son:

clases: Especificación de un conjunto de elementos

objetos: Elemento autónomo y con una funcionalidad concreta.

Instancias concretas de una clase

Elementos de la POO

Objetos

Elementos con comportamiento y estado. Métodos y atributos concretos

Instancias de clase

Interactúan por medio de mensajes

Elementos de la POO

Clases

Plantillas para definir elementos (objetos).

Pueden estar directamente relacionadas unas con otras

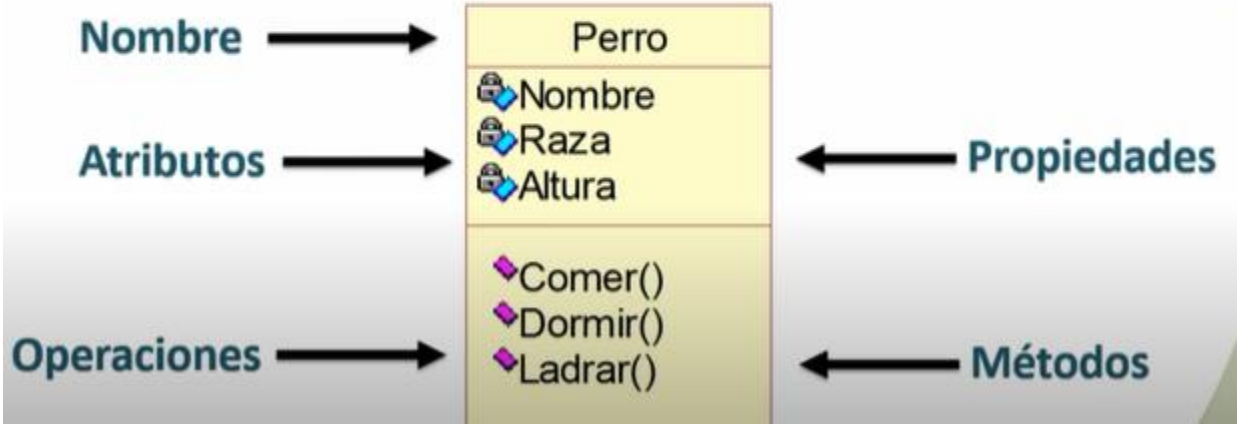
Elementos de la POO

Clase

Objetos

Elementos de la POO

Clase



Elementos de la POO

Encapsulación

Puede (y suele) haber distintos niveles de visibilidad:

public: se puede acceder desde cualquier lugar

private: sólo se puede acceder desde la propia clase

protected: sólo se puede acceder desde la propia clase o desde una clase que herede de ella.

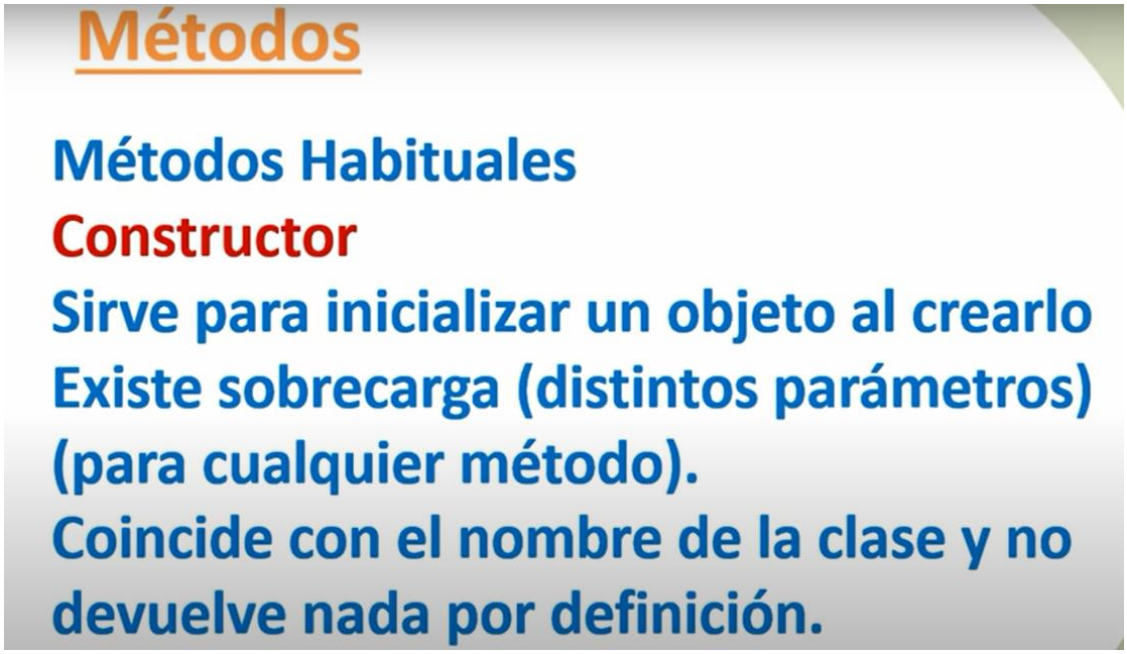
Instanciación de Objetos

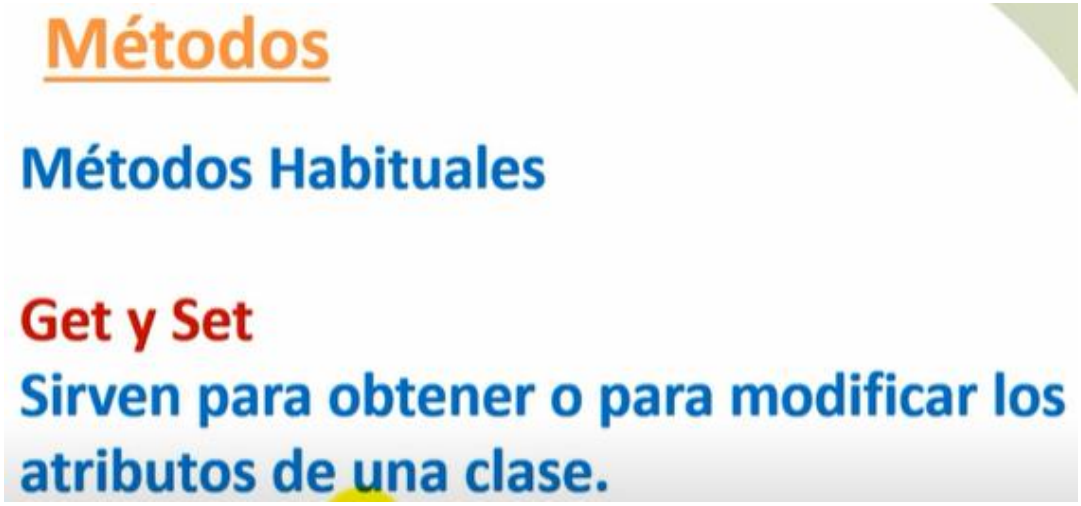
Antes de utilizar un objeto debemos de crearlo.

```
Tipo identificador=New Tipo();
```

EJEMPLO 1:

```
      using System;

namespace Ejemplo1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Perro perrito1  = new Perro(); // Objeto/Instancia a la clase Perro
            perrito1.nombre = "Gazu";
            perrito1.raza = "Pitbull";
            perrito1.altura = 1.7;


            Perro perrito2 = new Perro(); // Objeto/Instancia a la clase Perro
            perrito2.nombre = "Terri";
            perrito2.raza = "Chihuahua";
            perrito2.altura = 0.60;

            Console.WriteLine(perrito1.Comer("Pollo"));

            Console.WriteLine(perrito2.Comer("Carne"));

            Console.ReadKey();
        }//Fin main

    }//Fin clase Program

    class Perro
    {
        //Atributos
        public string nombre;
        public string raza;
        public double altura;

        //Metodos o Procedimientos
        public string Comer(string comida)
        {
            return "Nombre del perro: "+nombre+", raza: "+raza+" y mide: "+altura+" y esta
comiendo "+comida;
        }

        public void Dormir()
        {

        }

        public void Ladrar()
        {

        }
    }
}
```





Clase Perro.cs

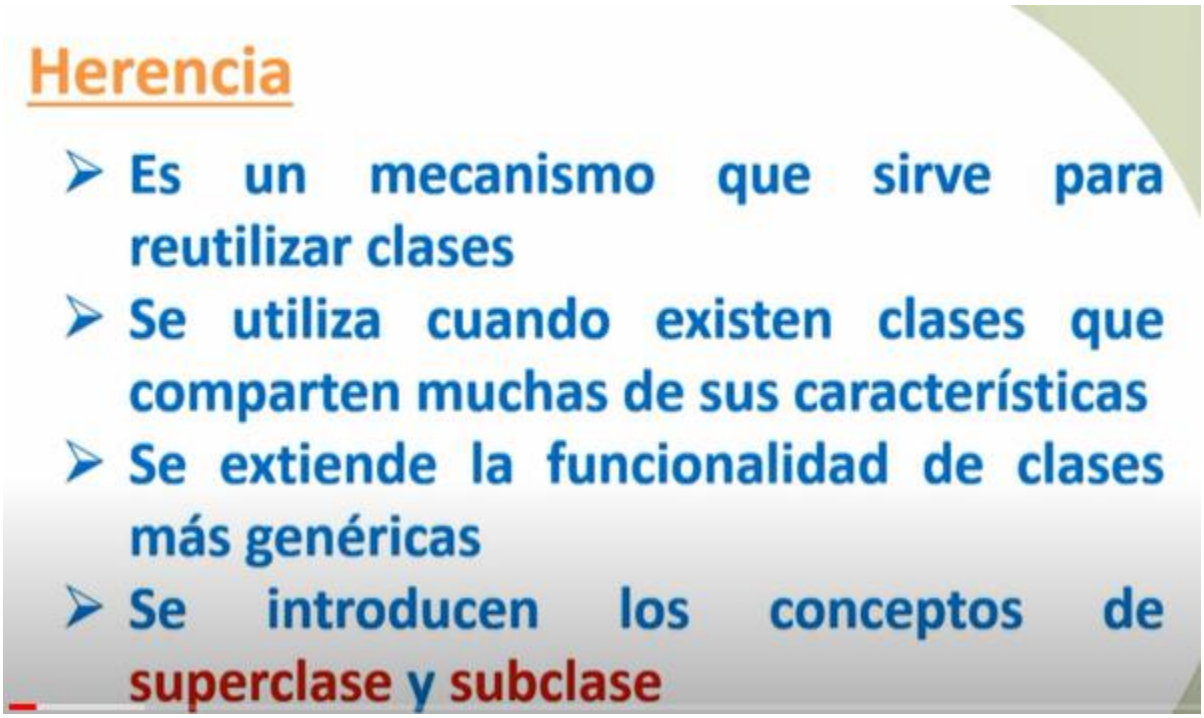
```
using System;

namespace Ejemplo1
{
    internal class Perro
    {
        //Atributos
        private string nombre;
        private string raza;
        private double altura;

        //Constructores
        public Perro() //Constructor vacío
        {
            this.nombre = "Chizu";
            this.raza = "";
            this.altura = 0.0;
        }

        //Metodos get en setter
```

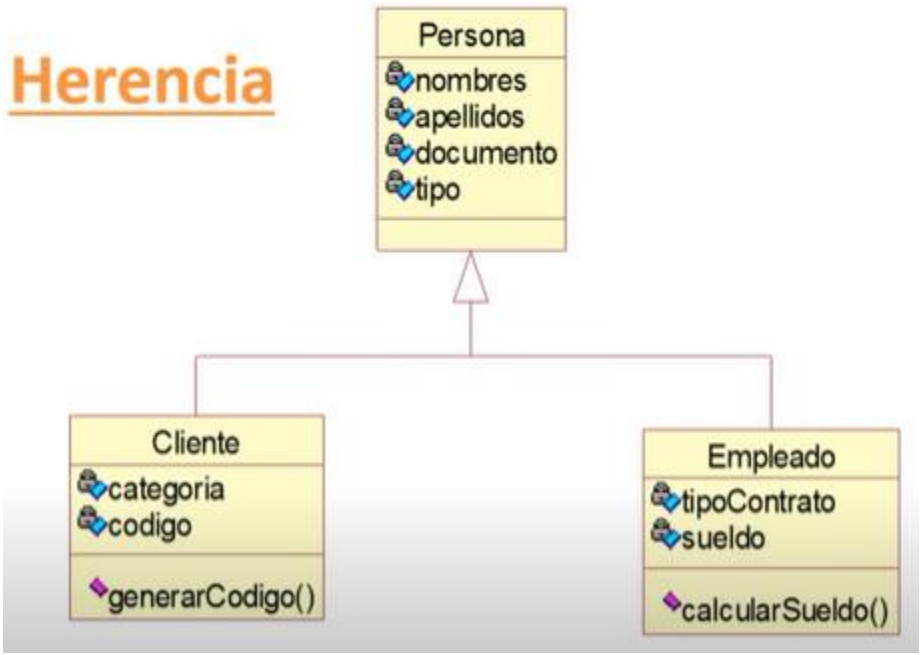
```
        public string Nombre
        {
            get { return this.nombre; }
            set { this.nombre = value; }
        }

        public string Raza
        {
            get { return this.raza; }
           set { this.Raza = value; }
        }

        //Metodos o Procedimientos
        public string Comer(string comida)
        {
            return "Nombre del perro: " + nombre + ", raza: " + raza + " y mide: " + altura + " y
esta comiendo " + comida;
        }

        public void Dormir()
        {

        }

        public void Ladrar()
        {

        }


    }
}
```

Main()

```
using System;

namespace Ejemplo1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Perro perrito1  = new Perro(); // Objeto/Instancia a la clase Perro
            perrito1.Nombre = "Gazu";

            Console.WriteLine(perrito1.Nombre);


            Perro perrito2 = new Perro(); // Objeto/Instancia a la clase Perro
            perrito2.Nombre = "Terri";


            Console.WriteLine(perrito1.Comer("Pollo"));

            Console.WriteLine(perrito2.Comer("Carne"));

            Perro perro = new Perro();
            Console.WriteLine(perro.Comer("Bistec"));

            Console.ReadKey();
```

```
        }//Fin main

    }//Fin clase Program


}
```





CLASE: Persona.cs

```
using System;

namespace Ejemplo1
{
    internal class Persona
    {
```

```
        private string nombre;
        private string apellido;
        private string dui;
        private string tipo;
        public string Nombre { get; set; }
        public string Apellido { get; set; }
        public string Dui { get; set; }
        public string Tipo { get; set; }
    }
}
```

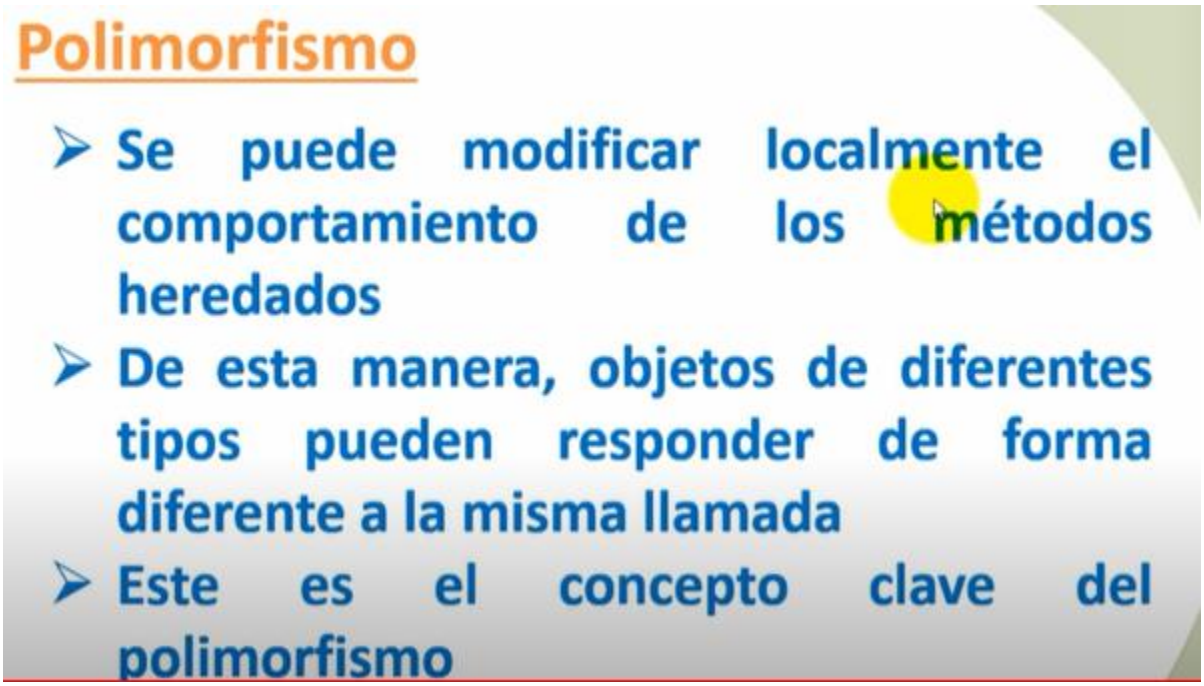
Clase: Cliente.cs

```
using System;

namespace Ejemplo1
{
    internal class Cliente:Persona
    {
        private string categoria;
        private string codigo;

        public string Categoria { set; get; }
        public string Codigo { set; get; }

        public void GenerarCodigo()
        {
            this.Codigo = "C" + Apellido.Substring(0, 3)+"2022";
        }
    }
}
```

Main()

```
using System;

namespace Ejemplo1
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Cliente cliente = new Cliente();
            cliente.Nombre = "DARWIN ";
            cliente.Apellido = "FLORES COLINDRES";
            cliente.Dui = "0508080-0";
            cliente.Categoria = "A";
            cliente.Tipo = "Frecuente";
            cliente.GenerarCodigo();

            Console.WriteLine("NOMBRE: "+cliente.Nombre);
            Console.WriteLine("APELLIDO: " + cliente.Apellido);
            Console.WriteLine("DUI: " + cliente.Dui);
            Console.WriteLine("CÓDIGO: " + cliente.Codigo);


            Console.ReadKey();
        }//Fin main
```

```
    }//Fin clase Program


}
```



Ejemplo:

Clase: Mamifero.cs

```
using System;

namespace Ejemplo1
{
    internal class Mamifero
    {
        private string nombre;

        public string Nombre { get { return nombre; } set { nombre = value; } }

        public Mamifero(string nombre)
        {
            this.Nombre = nombre;
        }
        public virtual void Pensar()
        {
            Console.WriteLine("ASI PIENSA LOS MAMÍFEROS");
        }
    }
}
```

Clase: Humano.cs

```
using System;

namespace Ejemplo1
{
    internal class Humano:Mamifero
    {
        public Humano(string nombreHumano) : base(nombreHumano) { }
        public override void Pensar()
        {
            Console.WriteLine("ESTOY PENSANDO COMO HUMANO");
        }
    }
}
```

Clase: Gorila.cs

```
using System;


namespace Ejemplo1
{
    internal class Gorila:Mamifero
    {
        public Gorila(string nombreGorila) : base(nombreGorila) { }
        public override void Pensar()
        {
            Console.WriteLine("Hay que pensar como un gorila");
        }
    }
}
```

Ejercicios:

1. Crea una clase Coche con las siguientes propiedades:
   - ID
   – Marca
   – Modelo
   – KM
   – Precio

   Debemos crear un constructor donde le pasemos los valores.
   Crea sus get y set de cada propiedad.
   Crea el metodo toString.
2. Cree un nuevo proyecto en C# con tres clases. Las clases principales del programa son las siguientes clases:
   Persona
   Estudiante
   Profesor
   Las clases de Estudiante y Profesor heredan de la clase Persona.
   La clase Estudiante incluirá un método público Estudiar() que escribirá en pantalla Estoy estudiando.
   La clase Persona debe tener dos métodos público Saludar() y SetEdad(int edad) que asignará la edad de la persona.
   La clase Profesor incluirá un método público Explicar() que escribirá en pantalla Estoy explicando.

Además crea un método público VerEdad() en la clase Estudiante que escriba en pantalla Mi edad es: x años.

3. Crear una clase Libro que contenga los siguientes atributos:
   - ISBN
   - Titulo
   - Autor
   - Número de páginas

   Crear sus respectivos métodos get y set correspondientes para cada atributo.

   Crear el método toString() para mostrar la información relativa al libro con el siguiente formato:

   "El libro su_titulo con ISBN su_ISBN creado por el autor su_autor tiene num_paginas páginas"

   En el fichero main, crear 2 objetos Libro (los valores que se quieran) y mostrarlos por pantalla.

   Por último, indicar cuál de los 2 tiene más páginas.