"La Ciencia sin Moral es Vana"

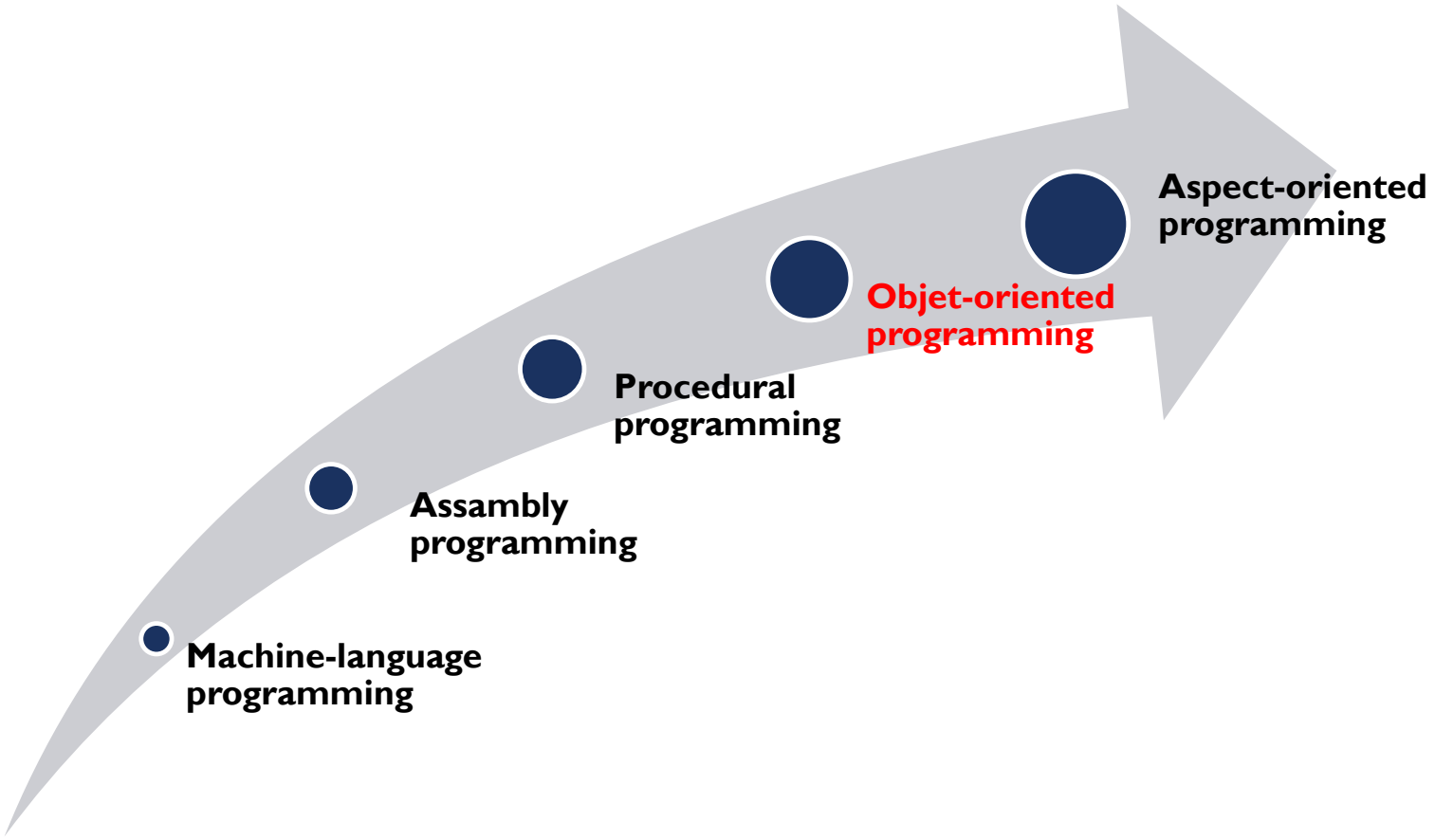
Universidad Católica de El Salvador
Facultad de Ingeniería y Arquitectura
Materia: Programación II
Docente: Master Giovanni Acosta

Tema 8:
Introducción a la Programación Orientada a Objetos

Objetivos:

- Conocer los elementos, características y ventajas de la programación orientada a objetos
- Definir clases y sus elementos
- Definir el uso de los métodos constructores
- Desarrollar y utilizar la instancia de una clase

EVOLUCIÓN DE LOS PARADIGMAS DE PROGRAMACIÓN



¿Qué es la programación orientada a objetos?

Elementos, características y ventajas

¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

- La Programación Orientado a Objetos (POO) es una forma particular de programar que se asemeja más a la forma como expresamos las cosas en la vida real.

- **La PROGRAMACIÓN ORIENTADA A OBJETOS (POO)** es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos o atributos) y comportamiento (es decir, funciones o métodos)

- La POO expresa un programa como un conjunto de objetos, que colaboran entre sí, para realizar una tareas específica. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

- Está basada en varias técnicas, incluyendo:

Abstracción

Encapsulamiento

Herencia

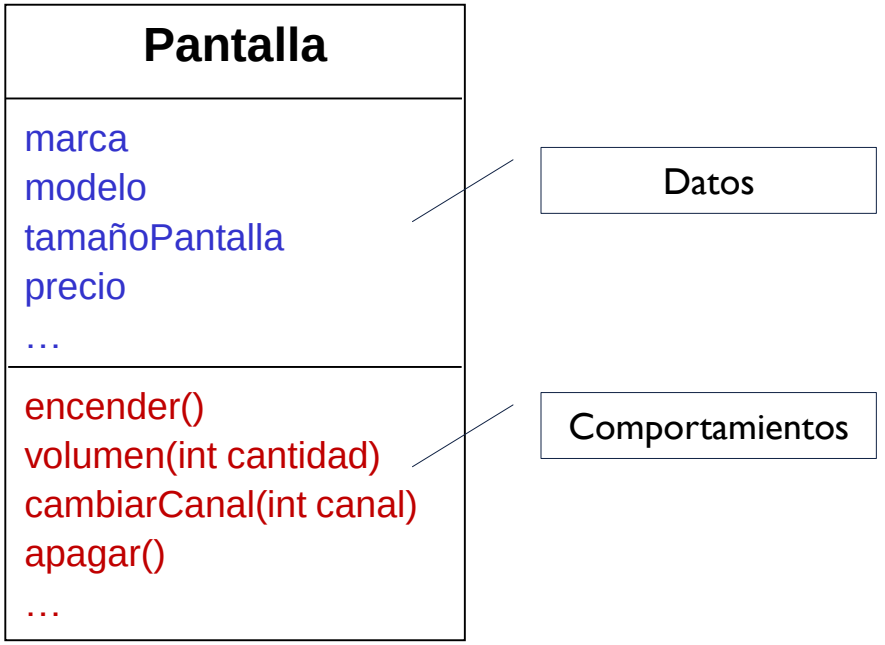
Polimorfismo

En la POO todo es un objeto, el cual contiene su estado y comportamiento

¿QUÉ ES DESARROLLO ORIENTADO A OBJETOS?

- Una nueva forma de pensar acerca del software basándose en abstracciones que existen en el mundo real, definiendo datos y comportamientos de cada objeto.

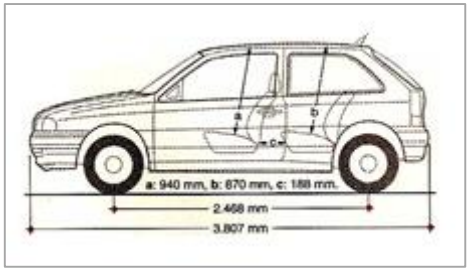
Objeto



Clase en notación UML

¿QUÉ ES UNA CLASE?

- **CLASE** es un modelo que define un conjunto de **datos** (el estado), y **métodos** apropiados para operar con dichos datos (el comportamiento). Cada objeto creado a partir de la clase se denomina instancia de la clase.



Clase

Instancias de clase (objetos)

- La palabra clase proviene de clasificación. Formar clases es el acto de clasificar.
- Por ejemplo, todos los automóviles comparten un mismo comportamiento y atributos comunes. Se usa la palabra **Auto** para referirse a todos estos comportamientos y propiedades comunes.
- Es posible modificar las propiedades (el estado) de una instancia de clase.

¿QUÉ ES UNA CLASE? (CONT..)

- **Todos los objetos de un mismo tipo comparten la misma clase.**
- Todo objeto es instancia de alguna clase.
- La clase describe las características de los objetos de la misma clase, qué atributos (datos) tienen y qué métodos (comportamientos)

- La clase constituye la plantilla de un mismo tipo de objetos.
- Constituye la base para poder crear un objeto.
- Cuando se crea un objeto de una clase se dice que se instancia un objeto de dicha clase.
- La clase es un concepto estático (***en tiempo de diseño***) mientras que el objeto es un concepto dinámico (***en tiempo de ejecución***)

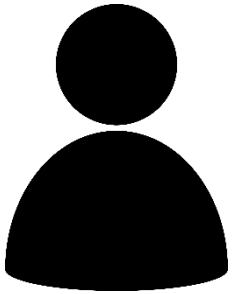
CARACTERÍSTICAS DE LA POO

¿QUÉ ES LA ABSTRACCIÓN EN POO?

- **ABSTRACCIÓN**: La abstracción consiste en captar las características esenciales de un objeto, así como su comportamiento, en un determinado contexto.

- **Abstracción**: capacidad del ser humano para entender una situación excluyendo detalles y sólo viéndola a alto nivel. El hombre ha comprendido el mundo con la abstracción.

- Ejercicio: realizar la abstracción de las clases Person y CellPhone



Person

CellPhone

¿QUÉ ES LA ENCAPSULACIÓN EN LA POO?

- El **ENCAPSULAMIENTO,** es la propiedad de ocultar los datos abstraídos y aislarlos o protegerlos de quién no se desee que tenga acceso a ellos; asegura que los aspectos externos de un objeto se diferencien de sus detalles internos.

- El encapsulamiento en un sistema orientado a objeto se representa en cada Clase u Objeto, definiendo sus atributos y métodos con los modos de acceso: public, private, protected

- Los objetos encapsulan en su interior lo necesario para su funcionamiento, su propio código, datos e incluso otros objetos que le sean necesarios, es decir, la encapsulación ***consiste en ocultar los detalles de implementación del objeto*****.**

¿QUÉ ES LA HERENCIA EN LA POO?

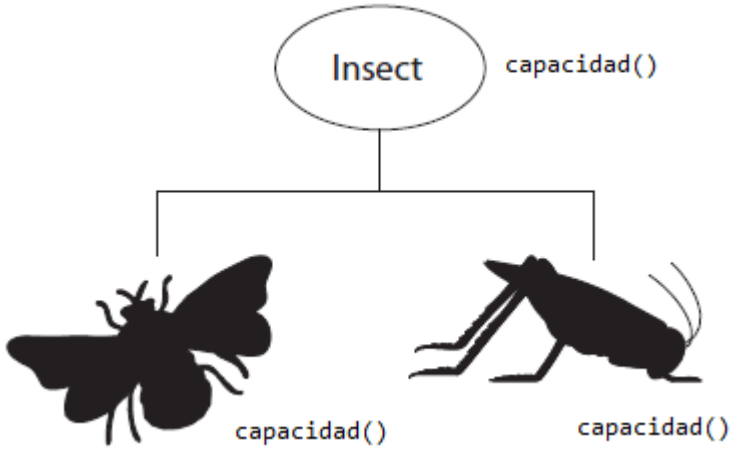
- La **HERENCIA** permite definir nuevas clases denominadas “clases derivadas, hijas o subclases”, a partir de clases ya existentes, llamadas “clase base, padre o superclase”. De esta manera los objetos pueden construirse en base a otros objetos ya creados.

- La herencia permite escribir y depurar una clase una vez, y después volver a utilizar ese código una y otra vez como base de nuevas clases. Las clases derivadas heredan y pueden extender: las propiedades y métodos propios.

¿QUÉ ES EL POLIMORFISMO EN LA POO?

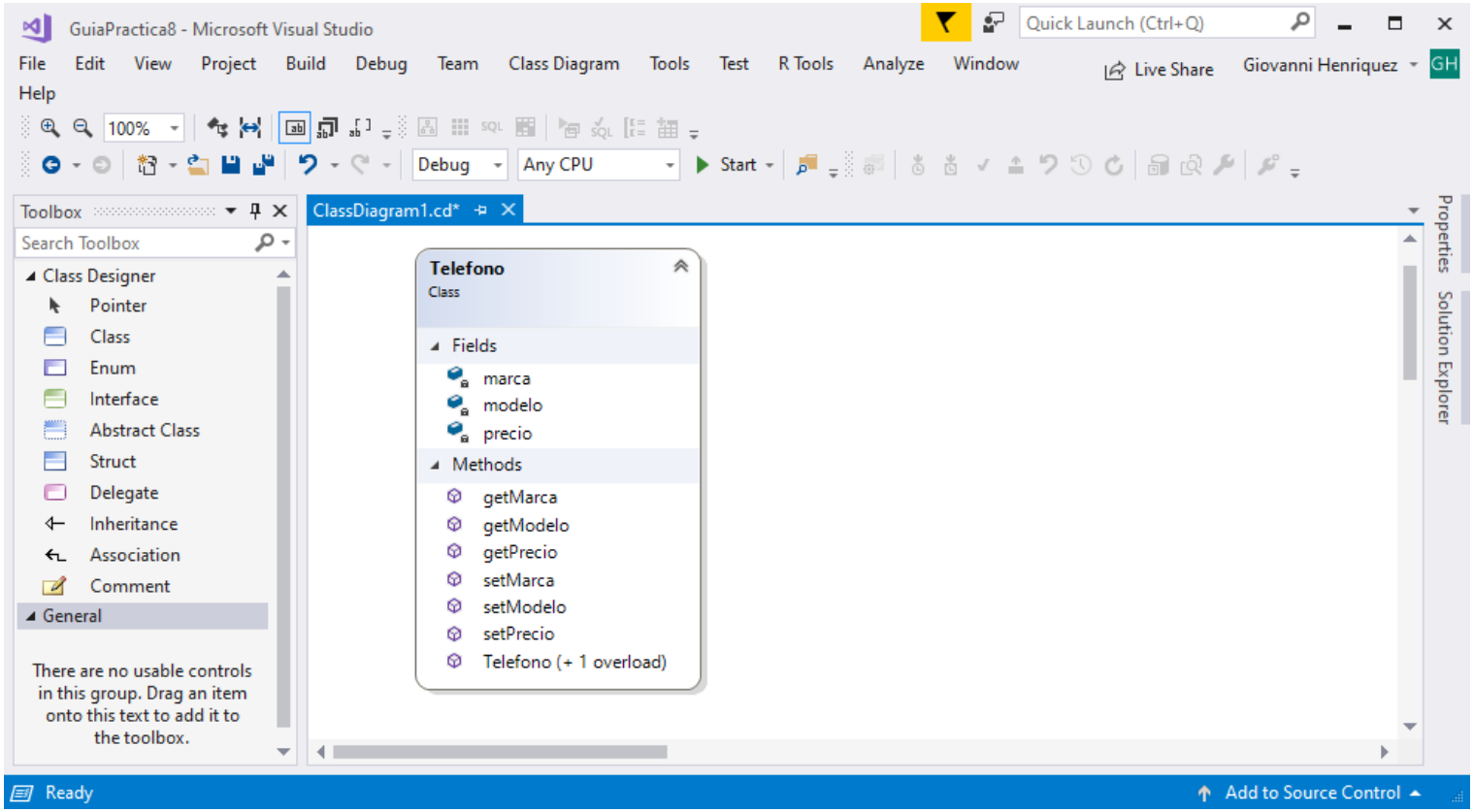
- **POLIMORFISMO** significa la cualidad de tener más de una forma. En el contexto de la POO, el polimorfismo se refiere al hecho de que un método de una clase base posea definiciones diferentes en clases derivadas que las hereden o implementen. En otras palabras, diferentes objetos reaccionan al mismo mensaje de modo diferente.

- El término polimorfismo se refiere a que una característica de una clase puede tomar varias formas:
  - El polimorfismo representa en nuestro caso la posibilidad de desencadenar operaciones distintas en respuesta a un mismo mensaje.
  - Cada subclase hereda las operaciones pero tiene la posibilidad de modificar localmente el comportamiento de estas operaciones.



¿CUÁLES SON LAS VENTAJAS DE LA POO?

- Fomenta la reutilización y extensión del código
- Permite crear sistemas más complejos
- Relacionar el sistema al mundo real
- Facilita la creación de programas visuales
- Construcción de prototipos
- Agiliza el desarrollo de software
- Facilita el trabajo en equipo
- Facilita el mantenimiento del software



Programación orientada a objetos en Visual C#

Sintaxis y programación

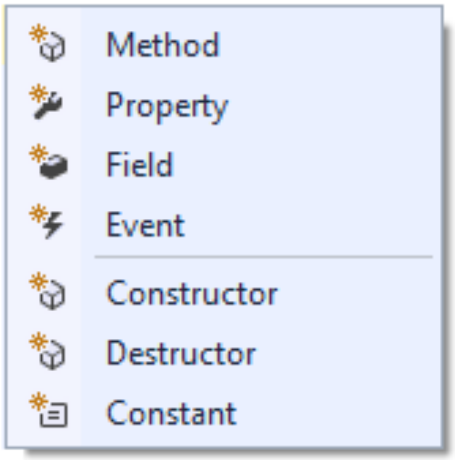
¿QUÉ ES UNA CLASE EN VISUAL C#?

- **Una clase** *se define como la descripción de los atributos (datos) y comportamientos (métodos) del objeto.*

- Con los atributos se define el estado del objeto en un momento dado, tales como: color, tamaño, longitud, etc., y con los métodos se define su comportamiento, tal como guardar(), leer(), insertar(), actualizar(), eliminar(), listar(), frenar(), arrancar(), etc. Los objetos son creados en tiempo de ejecución y son almacenados de manera temporal en la memoria de la computadora.

¿QUÉ PUEDE CONTENER UNA CLASE?

- En una clase se pueden contener los siguientes elementos:
  1. Métodos
  2. Propiedades
  3. Campos
  4. Eventos
  5. Constructores
  6. Destructores
  7. Constantes



COMPONENTES DE UNA CLASE

Métodos	• Acciones que un objeto puede realizar.
Propiedades	• Son miembros que ofrecen un mecanismo flexible para leer, escribir o calcular los valores de campos privados.
Campos	• Es una variable de cualquier tipo que se declara directamente en una clase.
Eventos	• Habilitan una clase u objeto para notificarlo a otras clases u objetos.
Constructores	• Son métodos de clase que se ejecutan automáticamente cuando se crea un objeto de un tipo determinado.
Destructores	• Se utilizan para eliminar instancias de clases.
Constantes	• Son valores que se conocen en tiempo de compilación y no cambian.

SINTAXIS PARA CREAR UNA CLASE

- La sintaxis para crear una clase en Visual C# es:

```
class NombreClase
{
    // Atributos (campos, propiedades y constantes)
    // Método constructor
    // Otros métodos
}
```

¿CÓMO CREAR CAMPOS?

- Los atributos de una clase incluyen: variables, propiedades y constantes miembro. Las variables miembro también se denominan campos.

- Sintaxis:

```
private string marca;
```

Modificador	Definición
public	Accesible en todas partes
private	Accesible sólo dentro de la clase
protected	Accesible dentro de la clase y desde las clases derivadas

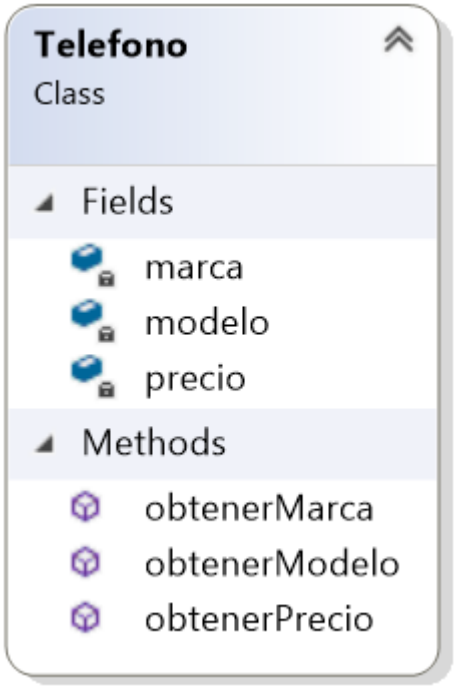
- Ejemplo:

```
class Telefono
{
    // Campos marca, modelo y precio del teléfono
    private string marca;
    private string modelo;
    private decimal precio;
}
```

En POO, los campos deben ser **private** o **protected** para encapsular y tener control de los datos de un objeto

¿CÓMO CREAR MÉTODOS DE ACCESO (GET)?

- El método de acceso retorna información sobre un determinado atributo (campo) de la clase



Modificador de acceso | Tipo de dato a retornar | Nombre del método

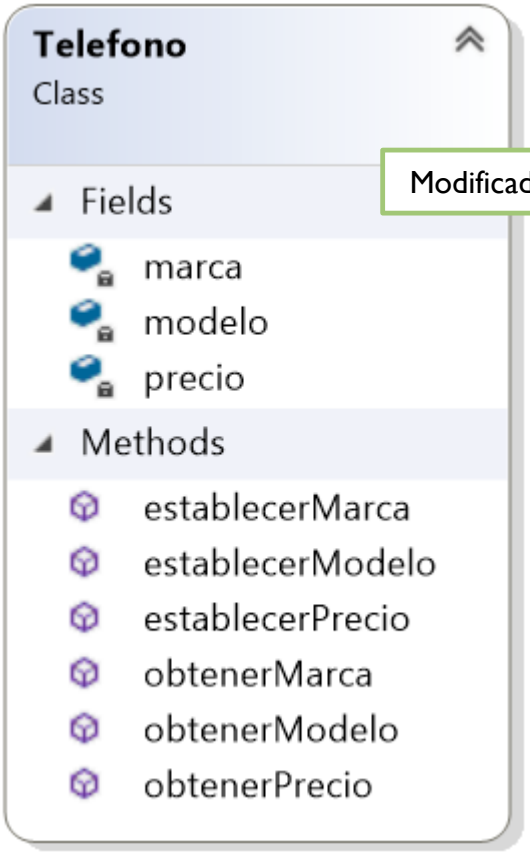
```
public string obtenerMarca()
{
    return marca;
}
```

Campo a retornar

En POO, los métodos se declaran como **public**, excepto que sea un método oculto para el uso interno de la clase, se declara como **private**

¿CÓMO CREAR MÉTODOS DE MODIFICACIÓN (SET)?

- El método de modificación cambia el valor de un determinado atributo (campo) de la clase



Modificador de acceso | No retorna | Nombre del método | Parámetro

```
public void establecerMarca(string nuevaMarca)
{
    marca = nuevaMarca;
}
```

Valor que se asignará al campo

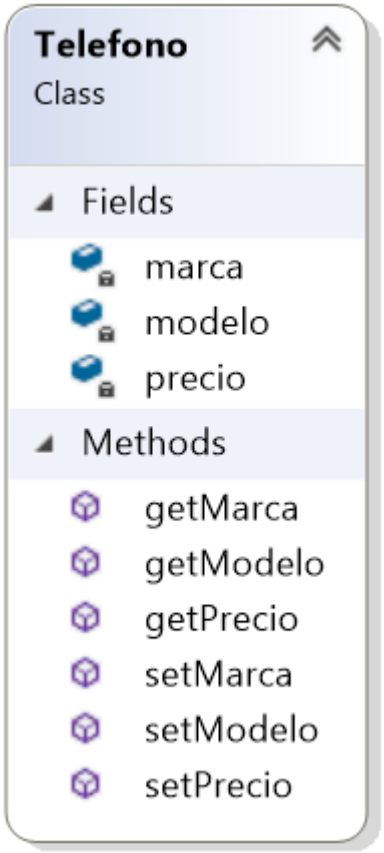
CONVENCIONES SOBRE NOMBRE DE MÉTODOS

- En POO, los nombres de los métodos de acceso suelen iniciar con la palabra **get** y luego el nombre del campo a retornar

```
// Método para retornar la marca
0 references
public string getMarca()
{
    return marca;
}
```

- En POO, los nombres de los métodos de modificación suelen iniciar con la palabra **set** y luego el nombre del campo a retornar

```
// Método para establecer la marca
0 references
public void setMarca(string marca)
{
    this.marca = marca;
}
```



get, para los métodos de acceso

set, para los métodos de modificación

¿CÓMO CREAR PROPIEDADES?

- Una **propiedad** es un atributo que proporciona un mecanismo flexible para leer, escribir o calcular el valor de un campo privado. Con el beneficio de un *miembro de dato público*. Pero a la vez proporciona la seguridad y la flexibilidad de los métodos.
- Ejemplo:

```
//campo
private int edad;
//propiedad
0 references
public int Edad
{
    get
    {
        return edad;
    }
    set
    {
        if (edad>18)
        {
            edad = value;
        }
    }
}
```

Campo privado para establecer y recuperar el valor de la propiedad
Nombre del campo con inicial **minúscula**

Nombre de la propiedad con inicial **mayúscula**

El descriptor de acceso **get** devuelve el valor del campo privado

El descriptor de acceso **set** puede realizar alguna validación de datos antes de asignar un valor al campo privado

¿CÓMO CREAR PROPIEDADES IMPLEMENTADAS AUTOMÁTICAMENTE USANDO CAMPOS?

- Ejemplos:

```
//campos
private string nombre;
private int edad;
private decimal sueldo;
//propiedades
0 references
public string Nombre { get => nombre; set => nombre = value; }
0 references
public int Edad { get => edad; set => edad = value; }
0 references
public decimal Sueldo { get => sueldo; set => sueldo = value; }
```

¿CÓMO CREAR PROPIEDADES IMPLEMENTADAS AUTOMÁTICAMENTE SIN USAR CAMPOS?

- Ejemplos:

```
//propiedades
0 references
public string Nombre { get; set; }
0 references
public int Edad { get; set; }
0 references
public decimal Sueldo { get; set; }
```

¿QUÉ ES UN MÉTODO CONSTRUCTOR?

- Un **constructor** es un método especial cuya función será inicializar los valores de los atributos (campos) en el momento en el que se cree el objeto.

- El constructor tiene las siguientes características:

1. • Tiene el mismo nombre de la clase
2. • Es el primer método que se ejecuta
3. • Se ejecuta en forma automática (al crear un nuevo objeto)
4. • No puede retornar datos (no posee tipo de retorno)
5. • Se ejecuta una única vez (al crear el objeto)

¿CÓMO SE CREA UN MÉTODO CONSTRUCTOR?

- Una clase puede contener un **método constructor sin parámetros**, para establecer valores fijos a los atributos (campos)

```
// Método constructor sin parámetros
0 references
public Telefono()
{
    marca = "";
    modelo = "";
    precio = 0;
}
```

El mismo nombre que la clase

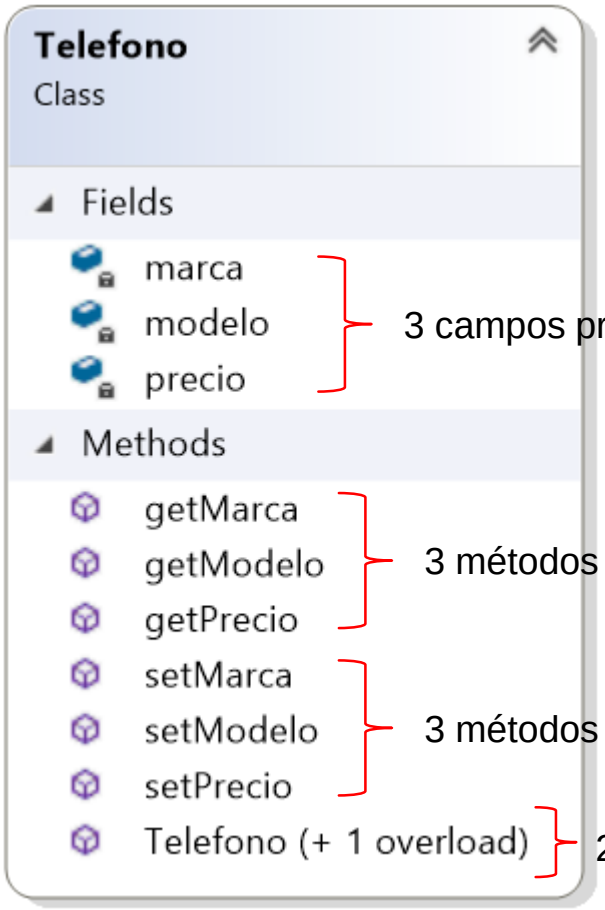
- Una clase puede contener también **métodos constructores con parámetros**, para que los campos sean inicializados al crear un nuevo objeto:

```
// Método constructor con parámetros
0 references
public Telefono(string marca, string modelo, decimal precio)
{
    this.marca = marca;
    this.modelo = modelo;
    this.precio = precio;
}
```

Si el nombre del campo es igual al nombre del parámetro, debe usar **this.campo** para diferenciarlo del nombre del parámetro

VISTA FINAL DE LA CLASE TELÉFONO

- La clase Teléfono posee:



3 campos privados

3 métodos de acceso

3 métodos de modificación

2 métodos constructores

¿CÓMO CREAR OBJETOS?

- Para crear un objeto a partir de una clase, utilizar la siguiente sintaxis:

```
nombreClase nombreObjeto = new nombreClase();
```

- Para crear un objeto utilizando un método constructor con parámetros, usar la siguiente sintaxis:

```
nombreClase nombreObjeto = new nombreClase( Lista de parámetros );
```

¿CÓMO INVOCAR LOS MÉTODOS DESDE EL OBJETO?

- Para ejecutar un método de acceso, usar la siguiente sintaxis:

```
string variable = nombreObjeto.getCampo();
```

El tipo de variable debe ser igual al tipo de dato del campo

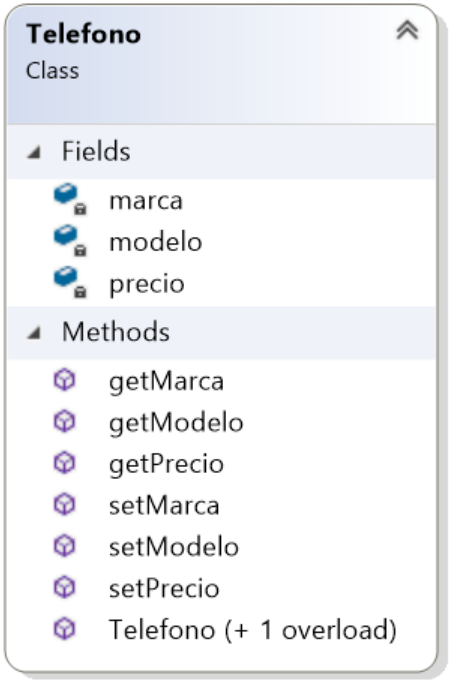
- Para ejecutar un método de modificación, utilizar la siguiente sintaxis:

```
nombreObjeto.setCampo("valor");
```

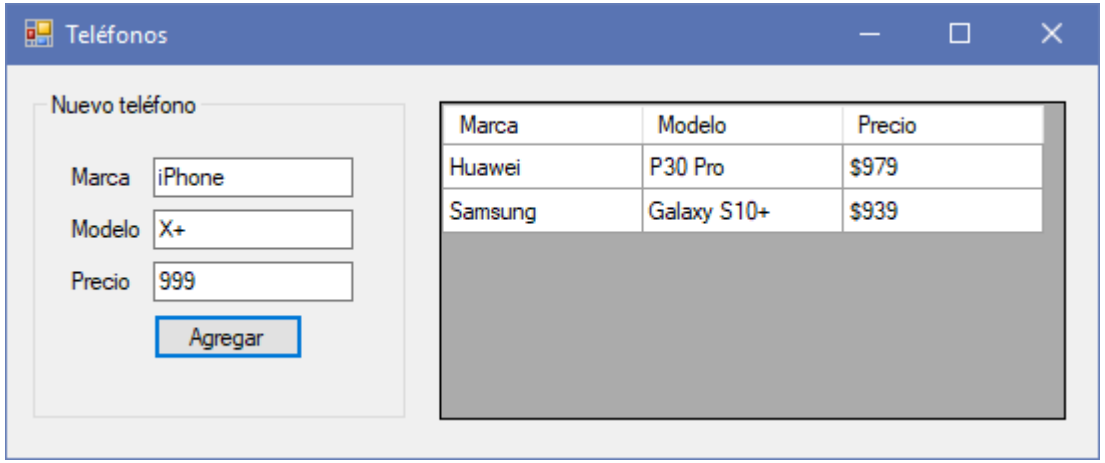
El valor debe ser del mismo tipo de dato que el campo

DEMO1 : USO DE LA CLASE TELÉFONO EN UNA APLICACIÓN

- Clase



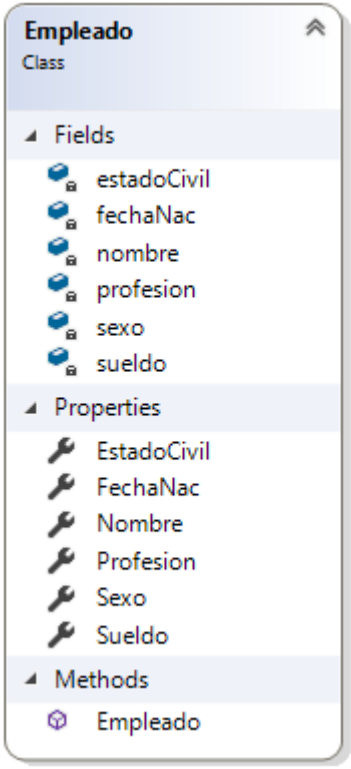
- Aplicación



DEMO 2: SERIALIZACIÓN BINARIA DE UN OBJETO

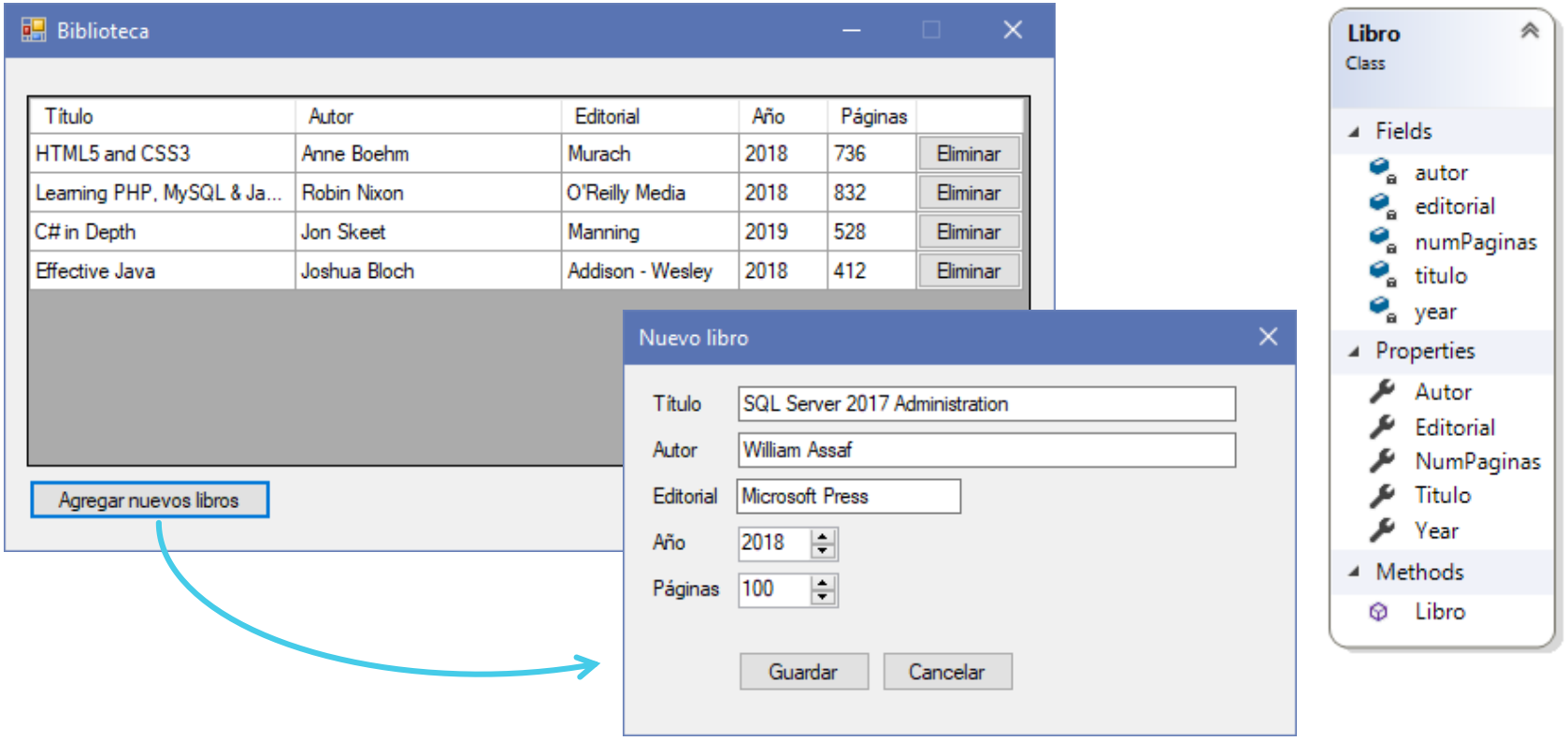
- Se debe preparar la clase para serializar sus objetos:

```
[Serializable]
5 references
class Empleado
{
    private string nombre;
    private DateTime fechaNac;
    private string sexo;
    private string estadoCivil;
    private string profesion;
    private decimal sueldo;
```



DEMO 3: SERIALIZAR UNA COLECCIÓN DE OBJETOS

- Se guardará una colección de objetos Libro en una lista y se serializará en un archivo binario.



BIBLIOGRAFÍA

- Asad, A. (2017). The C# Programmer's Study Guide (MCSD) Exam: 70-483. Pakistan: Apress.
- Ceballos, F. (2013). Microsoft C# Curso de programación. Segunda edición. México: Alfaomega.
- Putier, S. (2018). C# 7 y Visual Studio 2017. España: ENI.

- Programación Orientada a Objetos C# (Microsoft Docs):
- https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/object-oriented-programming