Clases → comportamiento → métodos
↘ Atributos → campos.

class <nombre>{
miembros} } x referencia.

Miembros de clase

- De instancia: del objeto
- Estáticos: de la clase

- Campos
- Métodos
- Constructores
- Constantes
- Propiedades
- Indizadores
- Finalizadores (o Destructores)
- Eventos
- Operadores
- Tipos anidados

ej: decir cómo se multiplican matrices y después usarlo con a x b en lugar de con una función.

Campo = var de instancia

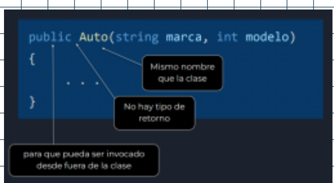
- Cada obj tiene su propio campo de instancia
- con un propio valor

Métodos de instancia

- Manipulan los datos de objetos
- Comportamiento
- Con métodos accedo a todos los campos del objeto

Métodos públicos (o privados), variables NO

Constructor de instancia → código q' se ejecuta al instanciar un objeto.
→ Establecen estado del nuevo objeto



→ sin valor de retorno

Por convención, var privadas van con _ → minúscula.

Encapsulamiento campos inaccesibles fuera de la clase

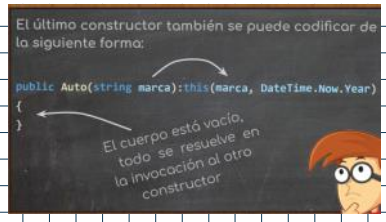
Puedo tener sobrecarga de constructores siempre y cuando cambie cant, tipo, orden y modificadores de parámetros.

Ojo: si creo un constructor con params, ya no puedo crear un objeto vacío.
Ej para no hacer lío y q' no queden cosas vacías

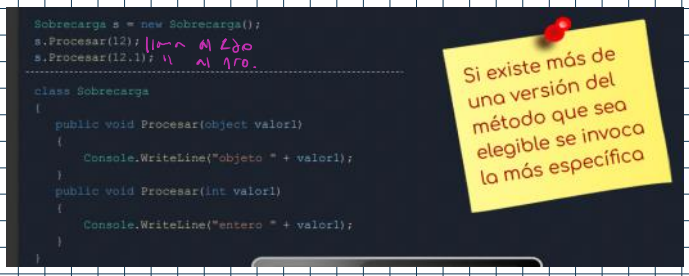
se puede invocar a otro constructor de la misma clase con : this.
se ejecuta ANTES q' las instrucciones del cuerpo del invocador

```
public Auto()
{
    _marca = "Fiat";
    _modelo = DateTime.Now.Year;
}
public Auto(string marca) : this()
{
    _marca = marca;
}
```

Llama al otro, q' setea marca con Fiat y el año actual
vuelve y se sobreescribe marca.



Métodos
- Tmb se pueden sobrecargar
- = reglas q' constructores para sobrecarga
- Valor de retorno no puede ser única diferencia




```
Sobrecarga s = new Sobrecarga();
object o = 12;
s.Procesar(o);
o = 12.1;
s.Procesar(o);
```

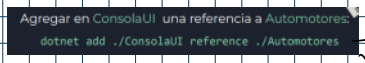
compilador intenta determinar estáticamente cuál usar
Por ello, llama las 2 veces al de object.
→ Dynamic se resuelve en ejecución

Arquitectura monolítica y modular

Monolítica → 1 solo módulo ejecutable implementado en 1 proyecto.

Modular → varios módulos, ejecutables y bibliotecas de clases.
- Cosas se pueden reutilizar en otros proyectos
- Poca dependencia

Bibliotecas de clase → dividir funcionalidad en módulos q' usan varias aplicaciones (.dll)
ejecutables y bibliotecas de clases → enlazan dinámicamente



→ Consola consume a automotores.
→ automotores es independiente