

Cuando se trabaja con proyectos grandes, el hecho de repartir una clase entre archivos independientes permite que varios programadores trabajen en ella al mismo tiempo.

```
public partial class Clase1
{
    public void Metodo1()
    {
    }
}

public partial class Clase1
{
    public void Metodo2()
    {
    }
}
```

Clases par

Cuando se trabaja con código fuente generado automáticamente, se puede agregar código a la clase sin tener que volver a crear el archivo de código fuente. Visual Studio usa este enfoque al crear formularios Windows Forms, código de contenedor de servicio Web, etc. Puede crear código que use estas clases sin necesidad de modificar el archivo creado por Visual Studio.

```
class Employee
{
    void DoWork()
}

class Employee
{
    GoToLunch()
}
```

Los métodos parciales permiten que el implementador de un elemento de una clase defina un método, similar a un evento. El implementador del otro elemento de la clase puede decidir si quiere implementar el método o no. Si el método no se implementa, el compilador quita la firma del método y todas las llamadas al método. Las llamadas al método, incluidos los resultados que se producirían por la evaluación de los argumentos de las llamadas, no tienen efecto en tiempo de ejecución.

Comerciales

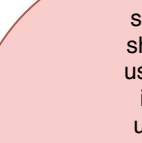
Métodos Partial

Tipos de Clase

Una **clase** es una estructura de datos que combina estados (campos) y acciones (métodos y atributos).

Tipos de Clase

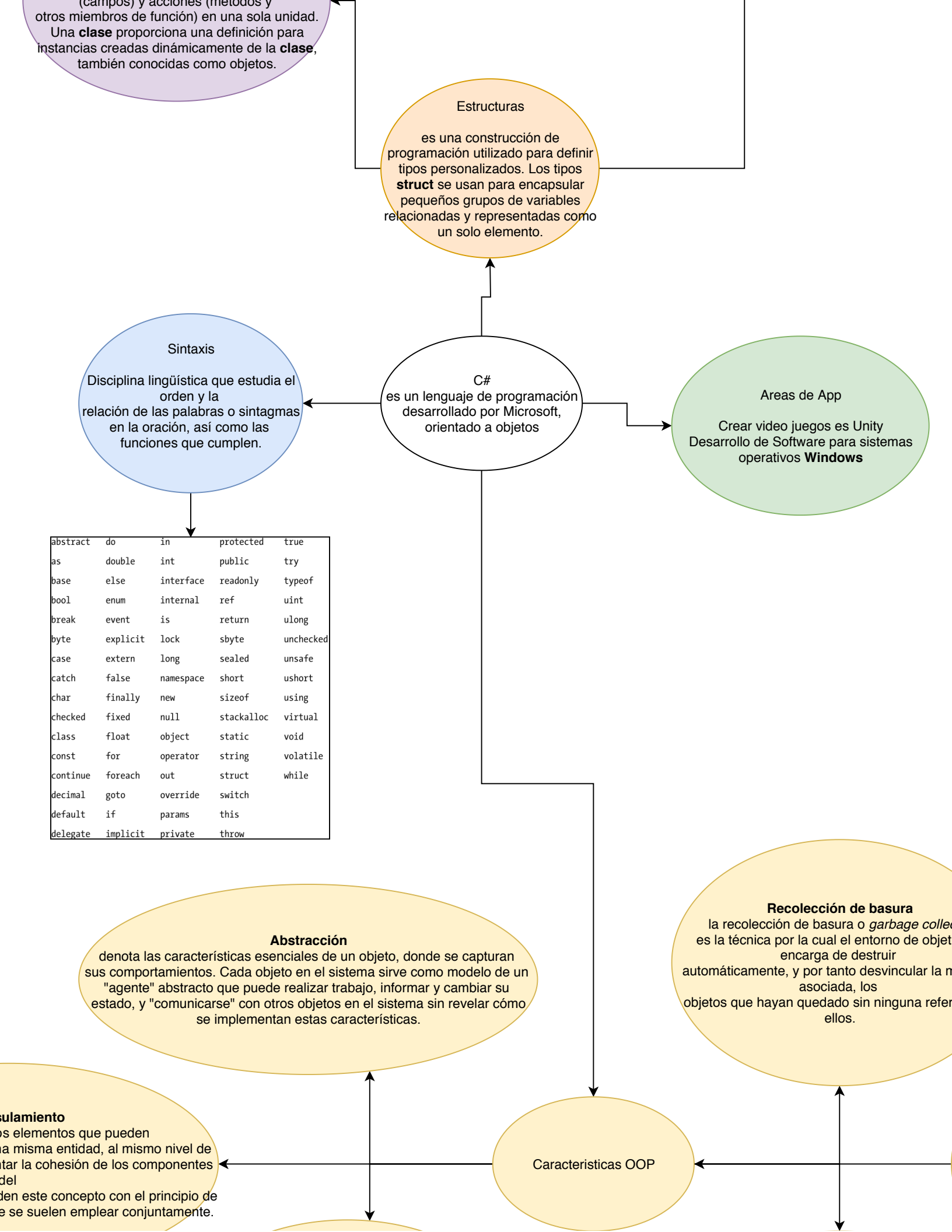
Una **clase** es una estructura de datos que combina estados (campos) y acciones (métodos y atributos).



byte
sbyte
short
ushort
int
unit
long
ulong
float
double
decimal
char
bool

Tipos de Variables

Encaps
significa reunir todos lo
considerarse pertenecientes a un
abstracción. Esto permite aument
sistema. Algunos autores confun
ocultación, principalmente porque



ctor
os se
memoria
erencia a

Herencia

las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo.

Modularidad

se denomina modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos

Polimorfismo

comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos con el mismo nombre se utilizará el comportamiento correspondiente al objeto que se está usando. O, dicho de otro modo, las referencias y las variables pueden contener objetos de diferentes tipos, y un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado

Principio de ocultación

cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado.

objetos
narlos por ese

eto que se esté
colecciones de
la invocación de
omportamiento

o.