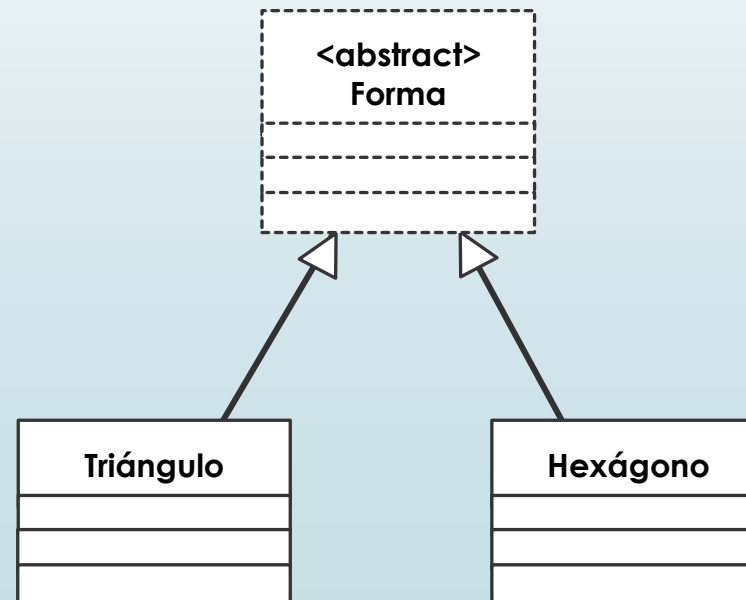


# Clases Abstractas

- Las clases abstractas son aquellas que por sí mismas no se pueden identificar con algo 'concreto' (no existen como tal en el mundo real), pero sí poseen determinadas características que son comunes en otras clases que pueden ser creadas a partir de ellas.





# Clases Abstractas



- Jerarquía de clases, donde los objetos que se instancian son de las clases hijas.
- No se pueden crear instancias de una clase abstracta.
- Existen métodos en la clase Padre que NO se implementan, sólo se hace en las clases hijas.
- Para declarar una clase o método como abstractos se utiliza la palabra clave **abstract**.
- Toda clase que contenga al menos un método abstracto, debe ser declarada a su vez como clase abstracta.



# Métodos Abstractos

- Los métodos abstractos no tienen ninguna implementación, de modo que la definición de método va seguida por un punto y coma en lugar de un bloque de método normal.

```
public abstract string Saludar();
```

- Las clases derivadas de la clase abstracta deben implementar todos los métodos abstractos.

# Clases Abstractas

```
abstract class Mascota
{
    protected string nombre;

    public Mascota(string nombre)
    {
        this.nombre = nombre;
    }

    public abstract string Saludar();
}

class Perro : Mascota
{
    protected string raza;

    public Perro(string nombre, string raza) : base(nombre)
    {
        this.raza = raza;
    }

    public override string Saludar()
    {
        return nombre + "," + raza + ": ¡Guau!";
    }
}
```

```
class Gato : Mascota
{
    protected string raza;

    public Gato(string nombre, string raza) : base(nombre)
    {
        this.raza = raza;
    }

    public override string Saludar()
    {
        return nombre + "," + raza + ": ¡Miau!";
    }
}

class Program
{
    static void Main(string[] args)
    {
        Mascota[] mascotas = new Mascota[2];

        mascotas[0] = new Perro("Neo", "Pequinés");
        mascotas[1] = new Gato("Fifi", "Persa");

        for (int i = 0; i < mascotas.Length; i++)
        {
            Console.WriteLine(mascotas[i].Saludar());
        }
    }
}
```



# Interfaces

- Los Interfaces son clases completamente abstractas que contiene sólo una colección de métodos abstractos y propiedades constantes.

```
interface NombreClase
```

- Las Interfaces son usadas para indicar qué métodos debe obligatoriamente implementar (contener) una Clase (aunque no tienen por qué comportarse del mismo modo).
- Estas son definiciones de métodos y valores sobre los cuales los objetos están de acuerdo para cooperar.

# Interfaces

```
class Mascota
{
    protected string nombre;

    public Mascota(string nombre)
    {
        this.nombre = nombre;
    }
}

interface IVolar
{
    void Volar();
}

class Gato : Mascota
{
    protected string raza;

    public Gato(string nombre, string raza) : base(nombre)
    {
        this.raza = raza;
    }
}
```

```
class Loro : Mascota, IVolar
{
    protected string raza;

    public Gato(string nombre, string raza) : base(nombre)
    {
        this.raza = raza;
    }

    public void Volar()
    {
        return nombre + "," + raza + ": ¡Estoy volando!";
    }
}
```