

notas de estudio

# Programación Orientada a Objetos con Gatitos.

Paulina Carolina Guevara Nieves



by @paulinaboleana

¡Hola! mi nombre es Paulina, aka paulinaboolena, una profesional en tecnología con 5 años de experiencia en la industria.

He trabajado en diferentes proyectos de empresas de diferente rubro desde aeroespaciales, consultoras y en la actualidad que estoy en una empresa de fintech llamada Nu, mi especialidad es el desarrollo backend.

También he explorado el desarrollo frontend como pasatiempo y he sido parte de comunidades como PHP MX y Technolatinas donde promuevo la inclusión y el empoderamiento de las mujeres en la tecnología

Este cuaderno de notas empezó por mi necesidad de repasar los conceptos básicos de POO (programación orientada en objetos) y lo publico con la esperanza de ayudar más programadores allá afuera.  
Los ejemplos de código están en c# o .net



# Conceptos Básicos | Clases

## Clases :

Una clase es una plantilla o modelo para crear objetos. En una clase se define el comportamiento y las propiedades que tendrán los objetos que se creen a partir de ella.

### Ejemplo con gatitos:

Imagina que los gatos son clases en un programa de software.

Cada tipo de gato (por ejemplo, siamés, persa, siameses de pelo largo) representa una clase diferente, y cada gato individual de una raza representa un objeto o instancia de esa clase.

Todos los gatos tienen patas, oídos y un hocico. Estas características compartidas podrían ser consideradas las **propiedades** de una clase.

Mi gato que es un siamés rojo y es muy gritón, pues así se comparte su raza y esa podría ser una propiedad de la clase "Siamés Rojo".



# Conceptos Básicos | Clases

## Ejemplo en código de la clase Siamés :

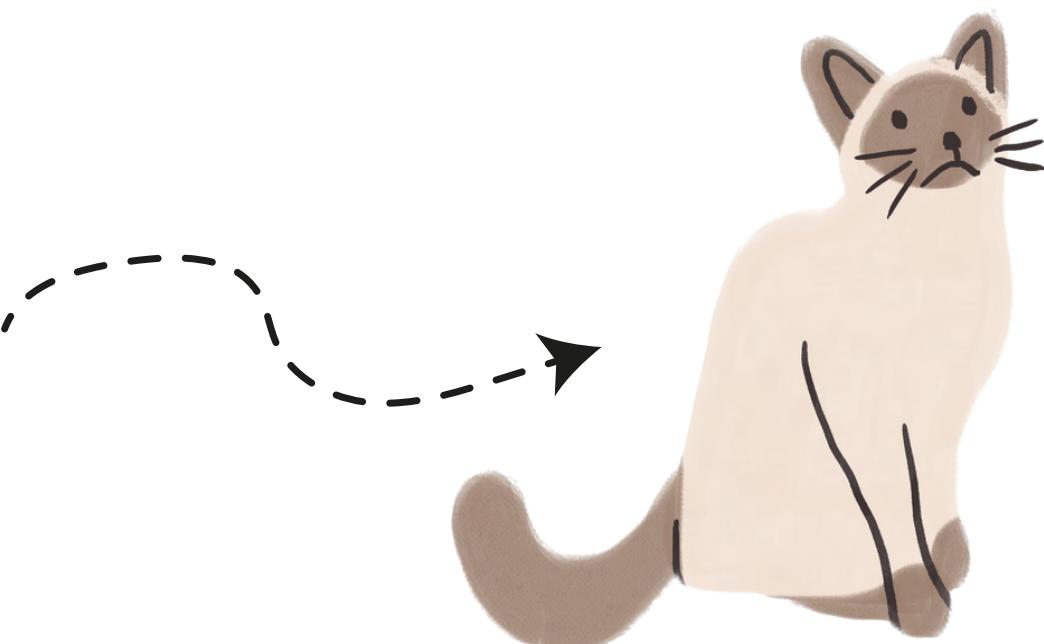
```
public class Siamese
{
    // Propiedades
    public string Name { get; set; }
    public int Age { get; set; }
    public string Color { get; set; }
    public bool IsLoud { get; set; }

    // Constructor
    public Siamese(string name, int age, string color, bool isLoud)
    {
        Name = name;
        Age = age;
        Color = color;
        IsLoud = isLoud;
    }

    // Métodos
    public void Meow()
    {
        if (IsLoud)
        {
            Console.WriteLine("¡Miau! ¡Miau! ¡Miau!");
        }
        else
        {
            Console.WriteLine("Miau");
        }
    }

    public void Scratch()
    {
        Console.WriteLine($"{Name} rasguña el sofá.");
    }
}
```

Este es podría  
ser un Gato  
(Objeto)  
creado de una  
clase Siamese



no te preocupes si aun no entiendes  
del todo, en las siguientes páginas  
se clarificarán tus dudas



# Conceptos Básicos | Clases

## Objetos:

Un objeto es una instancia de una clase. Cada objeto tiene su propio conjunto de propiedades y puede realizar acciones (métodos) definidas en la clase.

## Ejemplo con gatitos:

Retomando el ejemplo de código anterior podría decirse que al crear un objeto de la clase Siamese podemos crear muchos gatos siameses.

## Ejemplo con código:

```
● ● ●  
// Crear un objeto de la clase siames  
Siamese miGatoSiames = new Siamese("Mittens", 3, "blanco", true);
```

## CREAMOS UN SIÁMES



# Conceptos Básicos | Clases

## Encapsulación :

La encapsulación es uno de los principales conceptos de la programación orientada a objetos (POO). Se refiere a la idea de ocultar los detalles internos de un objeto y exponer solo una interfaz pública para interactuar con él.

En términos prácticos, esto significa que la mayoría de las propiedades y métodos de un objeto deben ser privados y solo se pueden acceder a través de métodos públicos que han sido cuidadosamente diseñados y probados.

Al ocultar los detalles internos de un objeto, los usuarios de la clase solo pueden interactuar con él a través de los métodos públicos que se han definido

### Ejemplo con gatitos:

Imagina que nuestro sistema de gatitos tiene la información "personal" ( XD ) de ellos, como su dirección, tipo de sangre y demás. Para proteger su información utilizaremos técnicas de diseño de software para mantener el código seguro y también los datos de nuestros meowusuarios



# Conceptos Básicos | Clases

## Ejemplo en código en nuestro meowsistema:

Definimos nuestra **superclase** Gato, cat aquí por #bilinguals

```
● ● ●

public class Cat
{
    private string _name;
    private int _age;
    private string _color;

    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    public int Age
    {
        get { return _age; }
        set { _age = value; }
    }

    public string Color
    {
        get { return _color; }
        set { _color = value; }
    }

    public Cat(string name, int age, string color)
    {
        Name = name;
        Age = age;
        Color = color;
    }
}
```



# Conceptos Básicos | Clases

Definimos una clase "Siamese" que herede la clase "Cat" y agregamos campos privados y propiedades públicas (IsLoud)

```
public class Siamese : Cat
{
    private bool _isLoud;

    public bool IsLoud
    {
        get { return _isLoud; }
        set { _isLoud = value; }
    }

    public Siamese(string name, int age, string color, bool isLoud) : base(name, age, color)
    {
        IsLoud = isLoud;
    }
}
```

Al encapsular las propiedades de nuestros gatos en campos privados y exponerlos solo a través de propiedades públicas, podemos controlar cómo se accede y se modifica la información de nuestros objetos. Además, la encapsulación nos permite cambiar la implementación interna de una clase sin afectar a los usuarios de la clase, siempre y cuando la interfaz pública se mantenga sin cambios.



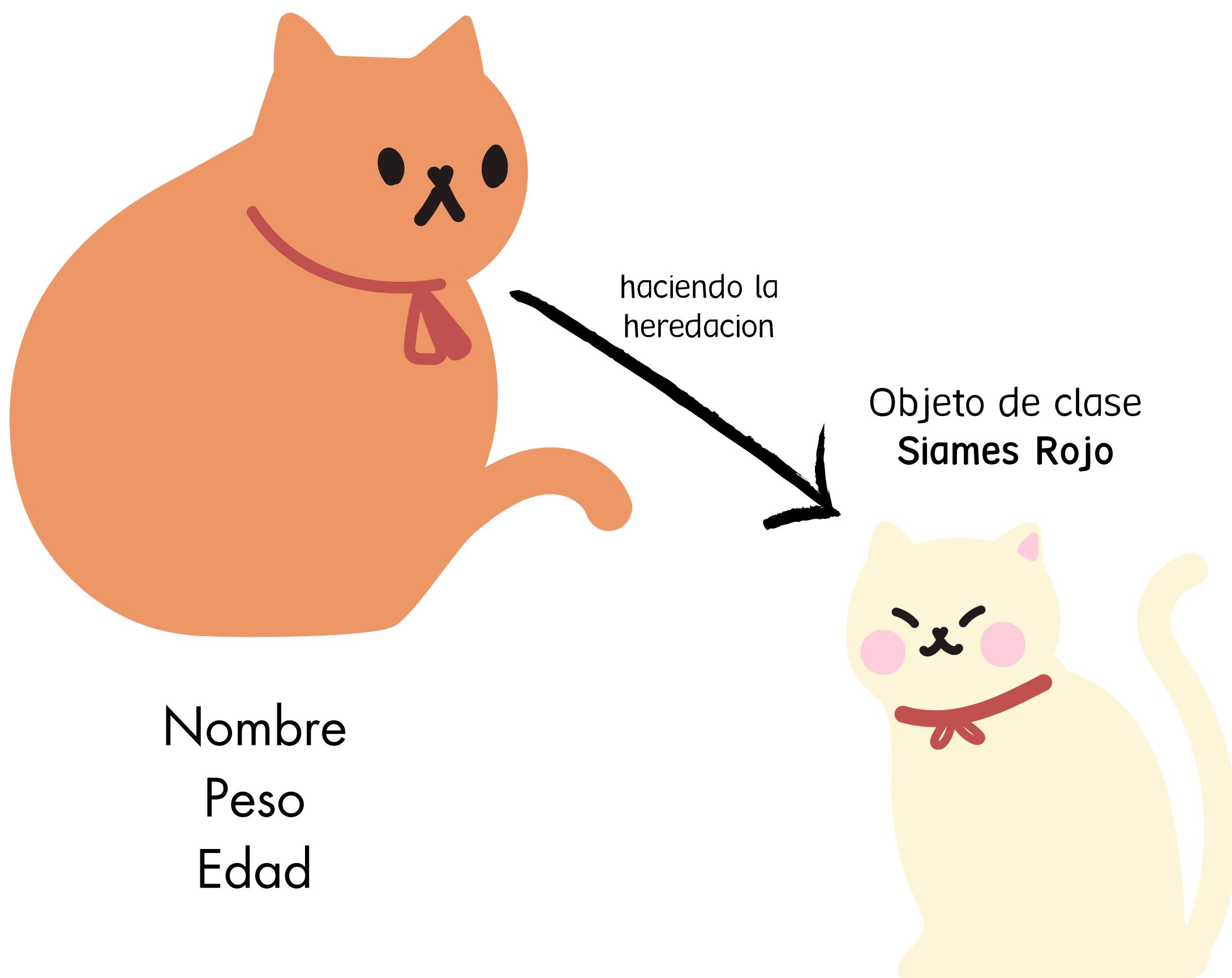
# Conceptos Básicos | Clases

## Herencia :

La herencia es un mecanismo que permite crear una nueva clase a partir de una clase existente, heredando sus propiedades y métodos. La nueva clase se llama subclase o clase hija, y la clase existente se llama clase padre o superclase.

### Ejemplo con gatitos:

En nuestro caso nuestra superclase sería **Gato** y nuestra clase que recibe los atributos y métodos seria la clase (Gato) **Siames Rojo**.



Nombre  
Peso  
Edad  
**EsGriton**



# Conceptos Básicos | Clases

**Polimorfismo:** El polimorfismo permite a los objetos de diferentes clases responder a un mismo mensaje o método de manera distinta. Esto se logra a través del uso de interfaces y clases abstractas.

## **Ejemplo con gatitos:**

Imaginemos que tenemos diferentes razas de gatos, como siameses, persas y calicos. Cada raza de gato tiene sus propias características y comportamientos únicos. Sin embargo, todos los gatos comparten algunas características en común, como su capacidad para maullar y ronronear.

De manera similar, el polimorfismo nos permite definir una interfaz común, como un método que acepta un objeto de tipo **Gato**, y luego podemos crear diferentes clases que heredan de **Gato** y responden a ese método de manera diferente. Por ejemplo, podemos tener una clase **Siamés** que responde al método maullando de manera diferente que una clase **Persian** o una clase **Calico**.



# Conceptos Básicos | Clases

## Ejemplo en código en nuestro meowsistema:

Podemos crear instancias de las clase **Siamese** y utilizar el **polimorfismo** para hacer que respondan al mismo mensaje de manera diferente. Un gato puede no maullar fuerte y otro sí

```
public class Cat
{
    public string Name { get; set; }

    public virtual void MakeSound()
    {
        Console.WriteLine("El gato hace un sonido");
    }
}

public class Siamese : Cat
{
    public bool IsLoud { get; set; }

    public override void MakeSound()
    {
        if (IsLoud)
        {
            Console.WriteLine("El gato siamés rojo maúlla muy fuerte");
        }
        else
        {
            Console.WriteLine("El gato siamés rojo maúlla suavemente");
        }
    }
}
```



```
siamese.MakeSound();
// "El gato siamés rojo maúlla
// muy fuerte"
```



# Conceptos Básicos | Clases

**Abstracción:** La abstracción en programación orientada a objetos se refiere a la idea de enfocarse solo en las características y comportamientos esenciales de un objeto y omitir los detalles no esenciales. En otras palabras, la abstracción nos permite simplificar el diseño de un objeto y centrarnos en lo que realmente importa.

## **Ejemplo con gatitos:**

Imaginemos que queremos crear un programa que simula la interacción entre los gatos y los humanos. En lugar de incluir cada detalle de cada gato individual, como su color de pelaje o su peso, podemos enfocarnos solo en los comportamientos y necesidades esenciales de los gatos. Por ejemplo, podemos incluir características como si el gato necesita comida, agua o atención médica, si está durmiendo o despierto, si está jugando o cazando, etc.



Estos son solo los conceptos básicos de una forma sencilla y con analogía con gatitos



Gracias especiales a mis papás, a mi esposo y bebé por apoyarme e inspirarme para crear esta pequeña guía

Puedes encontrarme en mis redes sociales como  
[@paulinaboolena](https://www.instagram.com/paulinaboolena)



# **Whats next?**

Fundamentos de C# con gatitos

