POO EN C#

Introducción

- En esta clase, vamos a hablar sobre los conceptos fundamentales de la programación orientada a objetos en C#. Cubriremos los siguientes temas:
  - Clases y objetos
  - Métodos y constructores
  - Herencia

Clases y objetos

- En C#, las clases son plantillas que se utilizan para crear objetos. Un objeto es una instancia de una clase. Las clases se definen utilizando la palabra clave class, seguida del nombre de la clase y un conjunto de llaves que contienen las propiedades y métodos de la clase.

Clases y objetos

```
class Persona {
    public string nombre;
    public int edad;

    public void Saludar() {
        Console.WriteLine("Hola, mi nombre es " + nombre + " y tengo " + edad + " años.");
    }
}

Persona p = new Persona();
p.nombre = "Juan";
p.edad = 30;
p.Saludar(); // Hola, mi nombre es Juan y tengo 30 años.
```

Métodos y constructores

- Los métodos son funciones que se definen en una clase y se utilizan para realizar acciones en objetos de esa clase. Los constructores son métodos especiales que se utilizan para inicializar objetos de una clase.

Métodos y constructores

```
class Circulo {
    public double radio;

    public Circulo(double r) {
        radio = r;
    }

    public double Area() {
        return Math.PI * radio * radio;
    }
}

Circulo c = new Circulo(5);
Console.WriteLine("El área del círculo es " + c.Area()); // El área del círculo es 78.53981633974483
```

Modificadores de acceso

- En C#, los modificadores de acceso son palabras clave que se utilizan para determinar el nivel de acceso a las propiedades y métodos de una clase. Los modificadores de acceso son public, private, protected, internal y protected internal.
  - public: Puede obtener acceso al tipo o miembro cualquier otro código del mismo ensamblado o de otro ensamblado que haga referencia a éste.
  - protected: Puede obtener acceso al tipo o miembro solo dentro de su propia clase o en las clases derivadas.
  - internal: Puede obtener acceso al tipo o miembro solo dentro del mismo ensamblado.
  - private: Puede obtener acceso al tipo o miembro solo dentro de la misma clase o estructura.
  - protected internal: Puede obtener acceso al tipo o miembro dentro del mismo ensamblado o en las clases derivadas fuera del ensamblado.

Modificadores de acceso

```
class Estudiante : Persona {
    public void CambiarNombre(string n) {
        SetNombre(n);
    }
}

Estudiante e = new Estudiante();
e.CambiarNombre("Juan");
Console.WriteLine(e.GetNombre()); // Juan
```

Modificadores de acceso

- Un ensamblado en C# es un archivo que contiene código compilado, recursos y metadatos que describen el contenido del ensamblado1. Los ensamblados se utilizan para agrupar y distribuir código de manera lógica y reutilizable.

- Una aplicación o una DLL son ejemplos de ensamblados en C#.
- El modificador de acceso "internal" se utiliza para establecer que un tipo o miembro solo puede ser accedido dentro del mismo ensamblado.
- El modificador de acceso protected internal es una combinación de los modificadores de acceso protected e internal. Un miembro con el modificador protected internal es accesible desde el ensamblado actual y desde cualquier tipo derivado de la clase contenedora.

Herencia

- La herencia es un concepto importante en la programación orientada a objetos que permite a las clases heredar propiedades y métodos de otras clases. La clase que se hereda se llama clase base o clase padre, y la clase que hereda se llama clase derivada o clase hija.

Herencia

```
class Animal {
    public void Comer() {
        Console.WriteLine("Estoy comiendo.");
    }
}

class Perro : Animal {
    public void Ladrar() {
        Console.WriteLine("Guau guau.");
    }
}

Perro p = new Perro();
p.Comer(); // Estoy comiendo.
p.Ladrar(); // Guau guau.
```

Override de métodos

- El override de métodos es un concepto importante en la programación orientada a objetos que permite a las clases derivadas reemplazar la implementación de un método de la clase base.

Override de métodos

```
class Animal {
    public virtual void Sonido() {
        Console.WriteLine("El animal hace un sonido.");
    }
}

class Perro : Animal {
    public override void Sonido() {
        Console.WriteLine("El perro ladra.");
    }
}

Animal a = new Animal();
a.Sonido(); // El animal hace un sonido.

Perro p = new Perro();
p.Sonido(); // El perro ladra.
```

Modificador base

- La palabra clave "base" se utiliza para acceder a los miembros de la clase base desde una clase derivada. Solo se permite el acceso a la clase base en un constructor, en un método de instancia y en un descriptor de acceso de propiedad de instancia. El uso de la palabra clave "base" desde dentro de un método estático dará un error.

Modificador base

```
class Animal {
    public string Nombre { get; set; }

    public Animal(string n) {
        Nombre = n;
    }
}

class Perro : Animal {
    public Perro(string n) : base(n) {
        // hacer algo más para inicializar la clase Perro
    }
}

Perro p = new Perro("Fido");
Console.WriteLine(p.Nombre); // Fido
```

Preguntas?

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash