Patrón Singleton

Propósito de los patrones

- ¿Qué es un patrón de diseño?
- ¿Por qué usarlos?

¿Qué es un patrón de diseño?

- Ante un problema reiterado ofrece una solución que lo resuelve.
- Describe el problema en forma sencilla.
- Describe el contexto en que ocurre.
- Describe los pasos a seguir.
- Describe los puntos fuertes y débiles de la solución.
- Describe otros patrones asociados.

¿Por qué usarlos?

- Mejora en la comunicación y documentación.
- Mejora la ingeniería de software.
- Previene “reinventar la rueda” en diseño.
- Mejora la calidad y estructura.

Singleton

Problema:

- No se puede tener más de una instancia de una clase.

```
public class Administradora
{
    private static Administradora _instancia;

    public static Administradora Instancia
    {
        get
        {
            // verifica que el objeto esté en null para crear una nueva
            // instancia o devolver la que ya existe
            if (_instancia == null) _instancia = new Administradora();
            return _instancia;
        }
    }

    // es muy importante poner el constructor como privado
    private Administradora() { }


class Program
{
    static void Main(string[] args)
    {
        Administradora sistema = Administradora.Instancia;

    }
}
```