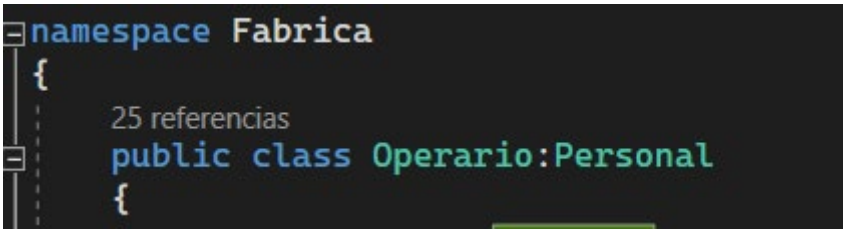
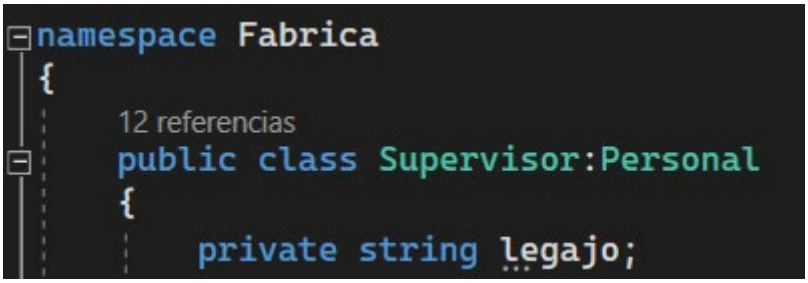
Uso de temas vistos en clase

Voy a dar un ejemplo de casos en el código donde use los temas vistos y pedidos para el parcial

En mi código no uso conversiones explicitas o implícitas y tampoco uso sobrecarga ya que no encontré ninguna funcionalidad para aplicarlo

1. Herencia: De una clase llamada “Personal” hay dos clases derivadas que se llaman “Supervisor” y “Operario”.





2. Polimorfismo: Uso polimorfismo en la clase “Personal” para que las clases derivadas puedan usar el método mostrar para cuando inicio sesión.

```
5 referencias
public abstract class Personal
{
    private string usuario;
    private string password;
    private Rango rango;

    2 referencias
    public Personal(string usuario, string password, Rango rango)
    {
        this.usuario = usuario;
        this.password = password;
        this.rango = rango;
    }

    4 referencias
    public string Usuario { get => usuario; set => usuario = value; }

    2 referencias
    public string Password { get => password; set => password = value; }

    2 referencias
    public Rango Rango { get => rango; set => rango = value; }

    4 referencias
    public abstract string Mostrar();
}
```

```
2 referencias
public override string Mostrar()
{
    StringBuilder sb=new StringBuilder();
    sb.AppendLine($"Nombre: {Usuario}");
    sb.AppendLine($"Legajo: {legajo}");
    sb.AppendLine($"Rango: {Rango}");
    return sb.ToString();
}
```

3. Colecciones: Use listas y diccionarios, diccionarios use para la parte donde le asigno el stock inicial a los productos y listas use por ejemplo para agregar a los supervisores y operarios a la lista de los que pueden entrar al sistema.

```
public static Dictionary<string, int> Materiales = new Dictionary<string, int>()
{
    {"Amargo", 100 },
    {"Vegano", 100 },
    {"Con mani", 100 },
    {"Chocolate Blanco", 50 },
    {"2x2", 50 },
    {"6x6", 50 },
    {"Glaseadas", 100 },
    {"Cubiertas", 100 },
    {"Con chispas", 100 },
    {"Cereza", 50 },
    {"Dulce de leche", 50 }
};
```

```
private static List<Operario> operarios = new List<Operario>();

8 referencias
public static List<Operario> ListaOperarios { get => operarios; set => operarios = value; }
```

```
Operario operario1 = new Operario("Martin", "12345", Rango.Operario, "Chacal");
Operario operario2 = new Operario("Federico", "12345", Rango.Operario, "Galarza");
Operario operario3 = new Operario("Paola", "12345", Rango.Operario, "Argento");
Operario operario4 = new Operario("Lauti", "profe", Rango.Operario, "Profe");
Operario operario5 = new Operario("Lucas", "profe", Rango.Operario, "Profe");
Operario operario6 = new Operario("Mathi", "profe", Rango.Operario, "Bustamante");

Operario.ListaOperarios.Add(operario1);
Operario.ListaOperarios.Add(operario2);
Operario.ListaOperarios.Add(operario3);
Operario.ListaOperarios.Add(operario4);
Operario.ListaOperarios.Add(operario5);
Operario.ListaOperarios.Add(operario6);
```

4. Formulario modal: Use formularios modales para la parte de simulación de la creación de un producto, donde cada formulario modal dura 3 segundos y se van cambiando uno al otro hasta el proceso final.

```
2 referencias
private DialogResult MostrarFormularioModal()
{
    using (FormModal1 formProduct = new FormModal1())
    {
        Hide();
        var result = formProduct.ShowDialog();
        if (result == DialogResult.Cancel)
        {
            Show();
        }
        return result;
    }
}
```

```
13 referencias
public partial class FormModal1 : Form
{
    private System.Windows.Forms.Timer timer;
    5 referencias
    public FormModal1()
    {
        InitializeComponent();

        progressBar1.Maximum = 100;
        progressBar1.Value = progressBar1.Maximum;
        timer = new System.Windows.Forms.Timer();
        timer.Interval = 2000;
        timer.Tick += timer1_Tick;
        timer.Start();
    }

    2 referencias
    private void timer1_Tick(object sender, EventArgs e)
    {

        if (progressBar1.Value > 0)
        {
            progressBar1.Value--;
        }

        timer.Stop();

        FormModal2 form2 = new FormModal2();

        this.Hide();
        form2.ShowDialog();

        this.Close();
    }
```

5. Clases estáticas: Hago la clase “Produccion” estatica para que el stock sea consistente en toda la aplicación, sin importar desde dónde accedas o modifiques esos valores.

```
namespace Fabrica
{
    15 referencias
    public static class Produccion
    {
```

6. Enumeradores: Use enumeradores para elegir el rango de supervisor y operario

```
18 referencias
public enum Rango
{
    Operario,
    Supervisor
}
```

7. Propiedades: Use propiedades para las clases “Operario” y “Supervisor” para que puedan ser leídas y modificadas desde otros lugares en el código, pero al mismo tiempo tener la opción de controlar o limitar este acceso en el futuro si es necesario.

```
8 referencias
public static List<Operario> ListaOperarios { get => operarios; set => operarios = value; }
6 referencias
public static List<Supervisor> ListaSupervisor { get => supervisor; set => supervisor = value; }
```