PROGRAMACIÓN ORIENTADA A OBJETOS

FACULTAD: INGENIERÍA MECÁNICA ELÉCTRICA Y ELECTRÓNICA

ESCUELA: INGENIERÍA ELECTRÓNICA

DOCENTE: ROMÁN MUNIVE WILDER ENRIQUE

ALUMNO: BERROCAL CHACALTANA JOSE ORLANDO

CICLO: IV EE - A

2025

Programación Orientada a Objetos – Visual Studio

I. INTRODUCCIÓN

La Programación Orientada a Objetos (POO) es un paradigma fundamental en el desarrollo de software moderno, especialmente en la creación de interfaces gráficas de usuario (GUI). En este informe, exploraremos cómo aplicar los principios de la POO para diseñar un menú dinámico utilizando contenedores y paneles en entornos como Visual Studio con tecnologías como Windows Forms (WinForms) o WPF (Windows Presentation Foundation).

II. CONCEPTOS CLAVE

2.1. Programación Orientada a Objetos (POO)

La POO se basa en cuatro pilares fundamentales:

Abstracción: Representación simplificada de un objeto.

Encapsulamiento: Ocultar detalles internos y exponer solo lo necesario.

Herencia: Reutilización de código mediante relaciones padre-hijo.

Polimorfismo: Capacidad de objetos diferentes de responder al mismo mensaje.

2.2. Contenedores y Paneles en Interfaces Gráficas

Los contenedores (como Panel, GroupBox, TabControl en WinForms o Grid, StackPanel en WPF) permiten organizar y agrupar controles de manera jerárquica.

Ejemplo de Jerarquía en POO para un Menú Dinámico:

```
public class MenuItem {
    public string Text { get; set; }
    public Action OnClick { get; set; }
}

public class DynamicMenu {
    private List<MenuItem> items = new List<MenuItem>();
    public void AddItem(MenuItem item) { /*...*/ }
    public void Render(Panel container) { /*...*/ }
}
```

III. DESARROLLO DE UN MENÚ DINÁMICO EN VISUAL STUDIO

*3.1. Implementación en Windows Forms (*WinForms*)*

WinForms utiliza el espacio de nombres System.Windows.Forms,

donde Panel y FlowLayoutPanel son esenciales para crear menús dinámicos.

Ejemplo de Código:

```
public class DynamicMenu {
    private FlowLayoutPanel panel;

    public DynamicMenu(FlowLayoutPanel targetPanel) {
        this.panel = targetPanel;
    }

    public void AddButton(string text, EventHandler onClick) {
        Button btn = new Button() { Text = text };
        btn.Click += onClick;
        panel.Controls.Add(btn);
    }
}
```

Uso en un Formulario:

```
public partial class MainForm : Form {
    public MainForm() {
        InitializeComponent();
        DynamicMenu menu = new DynamicMenu(flowLayoutPanel1);
        menu.AddButton("Inicio", (s, e) => MessageBox.Show("Inicio"));
        menu.AddButton("Ajustes", (s, e) => MessageBox.Show("Ajustes"));
    }
}
```

3.2. Implementación en WPF

WPF usa XAML para definir la interfaz y StackPanel o Grid como contenedores dinámicos.

Ejemplo en XAML:

```xaml
<StackPanel x:Name="MenuContainer">
    <!-- Los elementos se añaden dinámicamente -->
</StackPanel>
```

Código C# para Añadir Elementos:

```csharp
public class DynamicMenuWPF {
    private StackPanel container;

    public DynamicMenuWPF(StackPanel panel) {
        this.container = panel;
    }

    public void AddMenuItem(string text, RoutedEventHandler handler) {
        Button btn = new Button() { Content = text };
        btn.Click += handler;
        container.Children.Add(btn);
    }
}
```

IV. VENTAJAS DE USAR POO EN MENÚS DINÁMICO

Reutilización de Código: La clase DynamicMenu puede usarse en múltiples proyectos.

Mantenibilidad: Cambios en la lógica del menú se centralizan en una clase.

Escalabilidad: Fácil añadir nuevos tipos de ítems (ej: submenús desplegables

V. CONCLUSIÓN

La POO facilita el desarrollo de menús dinámicos al permitir una estructura modular y mantenible. Visual Studio, con WinForms o WPF, ofrece herramientas robustas para implementar contenedores y paneles que responden a interacciones dinámicas.

VI. BIBLIOGRAFÍA

1. **Microsoft Docs** (2023). *"Windows Forms Controls"*.
   Disponible en: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/
2. **Troelsen, A.** (2021). *"Pro C# 9 with .NET 5"*. Apress.
   - Capítulo 27: "Introducción a Windows Presentation Foundation (WPF)".
3. **Gamma, E. et al.** (1994). *"Design Patterns: Elements of Reusable Object-Oriented Software"*. Addison-Wesley.
   - Patrones como **Composite** (para jerarquías de menús).
4. **Liberty, J.** (2020). *"Programming C# 9.0"*. O'Reilly.
   - Sección sobre **eventos y delegados** en GUIs.
5. **Stack Overflow** (2023). *"Dynamic Menu Generation in C#"*.