



Centro de Enseñanza Técnica Industrial  
Plantel Tonalá

Practica 2.3

ACTIVIDAD INTEGRADORA: CASTOR LAND (7F1)

Fecha de entrega:  
11/09/2025

Alumno(s):  
Lia Zoe Perez Villaseñor

Docente:  
Profesor: Jorge J. Camacho Cortés

## RESUMEN

El presente reporte documenta el desarrollo de la práctica "Miembros Escuelita", una aplicación de escritorio creada con C# y Windows Forms. El objetivo principal fue aplicar de manera práctica los conceptos fundamentales de la Programación Orientada a Objetos (POO), específicamente la herencia y el polimorfismo. El alcance del proyecto consistió en la implementación de una jerarquía de clases, con MiembroEscuela como clase base y tres clases derivadas (Profesor, Estudiante, Administrativo). Se diseñó una interfaz gráfica con formularios dedicados a la captura de datos para cada tipo de miembro. Como resultado, se obtuvo una aplicación funcional que permite registrar diferentes miembros de una escuela y mostrar su información específica a través de un método polimórfico en una ventana de mensaje, cumpliendo con todos los requisitos de la actividad.

*Palabras clave:*

### **C# (C Sharp)**

Es el **lenguaje de programación** moderno y orientado a objetos, creado por Microsoft, utilizado en esta actividad

### **Programación Orientada a Objetos (POO)**

Es un **estilo o paradigma de programación** que se basa en la idea de "objetos" para representar datos y funcionalidades. En lugar de tener lógica dispersa, se agrupan en objetos (como Profesor o Estudiante) que tienen sus propias propiedades (como Id, Nombre) y comportamientos (como MostrarDatos()).

### **Herencia:**

Es un pilar de la POO que permite que una clase (la "clase hija") **herede** características y comportamientos de otra clase (la "clase padre"). Esto evita repetir código.

### **Polimorfismo**

Del griego "muchas formas", es la capacidad que tienen los objetos de **comportarse de manera**

## ABSTRACT

### **ABSTRACT**

This report documents the development of the "Miembros Escuelita" project, a desktop application created with C# and Windows Forms. The main objective was to practically apply the fundamental concepts of Object-Oriented Programming (OOP), specifically inheritance and polymorphism. The scope of the project consisted of implementing a class hierarchy, with MiembroEscuela as the base class and three derived classes (Profesor, Estudiante, Administrativo). A graphical user interface was designed with dedicated forms for capturing data for each member type. As a result, a functional application was developed that allows for the registration of different school members and displays their specific information through a polymorphic method in a message box, fulfilling all the activity's requirements.

*Keywords:*

**C# (C Sharp)** It is the modern, object-oriented programming language, created by Microsoft, used in this activity.

**Object-Oriented Programming (OOP)** It is a programming style or paradigm based on the concept of "objects" to represent data and functionalities. Instead of having scattered logic, it is grouped into objects (like Profesor or Estudiante) that have their own properties (like Id, Nombre) and behaviors (like MostrarDatos()).

**Inheritance** It is a pillar of OOP that allows a class (the "child class" or subclass) to inherit characteristics and behaviors from another class (the "parent class" or base class). This avoids code repetition.

**Polymorphism** From the Greek for "many forms," it is the ability of objects to behave differently in response to the same command. This is exemplified by the MostrarDatos() method; although it's the same method call, it yields different results depending on the object's class.

**Windows Forms** It is the Microsoft framework (a set of tools) used to design and build the graphical interface of your desktop application. All visual elements such as windows (Form), buttons

**diferente** ante la misma orden. Como en el método `MostrarDatos()` que tienen distintas clases, Es el mismo método, pero con resultados distintos

### Windows Forms:

Es el **conjunto de herramientas** (framework) de Microsoft que te permitió diseñar y construir la interfaz gráfica de tu aplicación de escritorio. Todos los elementos visuales como las ventanas (Form), los botones (Button) y las cajas de texto (TextBox) son parte de Windows Forms.

### Interfaz Gráfica:

Es el **medio visual** a través del cual el usuario interactúa con tu programa. Se compone de todos los elementos gráficos como botones, ventanas y menús que se diseñan para que una persona pueda usar la aplicación de manera fácil e intuitiva, en lugar de usar comandos de texto.

(Button), and text boxes (TextBox) are part of Windows Forms.

**Graphical Interface (GUI)** It is the visual medium through which the user interacts with your program. It is composed of all the graphical elements, such as buttons, windows, and menus, that are designed so a person can use the application easily and intuitively, instead of using text-based commands.

## OBJETIVOS

- Distinguir las diferentes partes de un formulario .NET.
- Identificar los objetos y el uso de cada uno en un formulario .NET.
- Desarrollar una aplicación de escritorio con múltiples formularios, haciendo uso de herencia y polimorfismo.

## MARCO TEÓRICO

Este proyecto se fundamenta en el paradigma de la **Programación Orientada a Objetos (POO)**, utilizando el lenguaje **C#** y el framework **Windows Forms** para el desarrollo de una aplicación de escritorio. A continuación, se detallan los conceptos teóricos que sustentan esta práctica.

### Programación Orientada a Objetos (POO)

La **Programación Orientada a Objetos** es un paradigma de programación que organiza el diseño de software en torno a "objetos", en lugar de funciones y lógica. Un objeto es una entidad que agrupa un estado (datos o propiedades) y un comportamiento (procedimientos o métodos). El objetivo principal de la POO es modelar entidades del mundo real de una manera más intuitiva, lo que facilita la modularidad, la reutilización y el mantenimiento del código.

Los pilares fundamentales en los que se basa la POO son:

### 1. Herencia

La **herencia** es un mecanismo que permite a una clase (denominada *clase hija* o *subclase*) adquirir las propiedades y métodos de otra clase (*clase padre* o *súper clase*). Esto fomenta la reutilización de código y establece una relación jerárquica entre las clases.

- **Aplicación en la práctica:** En este proyecto, las clases Profesor, Estudiante y Administrativo **heredan** los atributos comunes Id y Nombre de la clase base MiembroEscuela, evitando así la necesidad de redefinirlos en cada una.

### 2. Polimorfismo

La palabra **polimorfismo** significa "muchas formas". En POO, se refiere a la capacidad de los objetos de diferentes clases para responder al mismo mensaje o método de maneras distintas y específicas.

- **Aplicación en la práctica:** El ejemplo más claro es el método MostrarDatos(). Aunque la llamada es idéntica para todos los objetos, su comportamiento cambia: un objeto de la clase Profesor mostrará su materia, mientras que un objeto Estudiante mostrará su promedio.

### 3. Encapsulamiento

El **encapsulamiento** consiste en agrupar en una sola entidad (la clase) los datos (propiedades) y los métodos que operan sobre esos datos. Además, permite ocultar la complejidad interna del objeto y exponer solo lo necesario al exterior, protegiendo la integridad de los datos.

- **Aplicación en la práctica:** Cada clase, como Estudiante, encapsula sus propios datos (Id, Nombre, CalificacionPromedio) y la lógica para mostrarlos (MostrarDatos()).

## Clases y Objetos

- **Clase:** Es una **plantilla o molde** para crear objetos. Define un conjunto de propiedades y métodos comunes. Por ejemplo, la clase Estudiante define que todos los estudiantes tendrán un ID, un nombre y una calificación promedio.
- **Objeto:** Es una **instancia de una clase**. Mientras que la clase es el plano, el objeto es la construcción real. Cuando en el programa se capturan los datos de un estudiante y se presiona "Agregar", se está creando un **objeto** específico de la clase Estudiante.

## Tecnologías y Frameworks

### Lenguaje C#

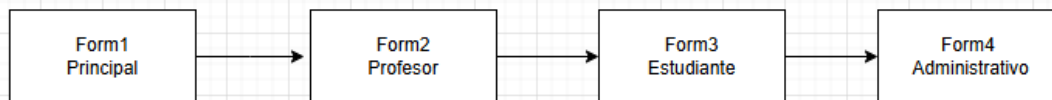
**C#** (pronunciado "C sharp") es un lenguaje de programación moderno, multiparadigma y fuertemente tipado, desarrollado por Microsoft. Es el lenguaje principal del ecosistema .NET y está diseñado específicamente para construir aplicaciones robustas y seguras, siendo ideal para la Programación Orientada a Objetos.

### Windows Forms

**Windows Forms** es un framework de interfaz gráfica de usuario (GUI) incluido en .NET, que permite crear aplicaciones de escritorio para Windows. Proporciona una amplia biblioteca de controles visuales, como botones (Button), cajas de texto (TextBox) y ventanas (Form), que facilitan el diseño de interfaces interactivas de manera rápida y eficiente.

El desarrollo de esta aplicación no se basa en fórmulas matemáticas, sino en **estructuras lógicas y algoritmos**. El sustento teórico radica en la correcta aplicación de la jerarquía de clases y los principios de la POO para modelar un sistema de gestión simple. Las operaciones realizadas, como la conversión de texto a número (`int.Parse()`), son procesos lógicos de manejo de datos, no aplicaciones de teorías matemáticas.

## DIAGRAMA A BLOQUES



## PROCEDIMIENTO

### Paso 1: Creación del Proyecto

Se inició creando un nuevo proyecto en Visual Studio, seleccionando la plantilla "**Aplicación de Windows Forms (.NET Framework)**" para el lenguaje C#. El proyecto fue nombrado "**Act\_Int\_P1**", de acuerdo con las especificaciones de la práctica.

**Figura 1:** Creación del proyecto de tipo Windows Forms en Visual Studio.

### Paso 2: Implementación de la Estructura de Clase

Se crearon cuatro archivos de clase (.cs) para implementar la estructura orientada a objetos:

1. **MiembroEscuela.cs:** La súper clase o clase base, que contiene las propiedades comunes `Id` y `Nombre`, y un método virtual llamado `MostrarDatos()`.

2. **Profesor.cs, Estudiante.cs y Administrativo.cs:** Las subclases o clases derivadas que heredan de `MiembroEscuela` y añaden sus propias propiedades. En cada una, se sobrescribió el método `MostrarDatos()` con la palabra clave `override` para mostrar su información específica.

A continuación, se muestra el código de la clase `Estudiante.cs` como ejemplo de una clase derivada que implementa herencia y polimorfismo.

C#

```
// Resultado: Código de la clase Estudiante
using System.Windows.Forms;
namespace Act_Int_P1
{
    class Estudiante : MiembroEscuela
    {
        public double CalificacionPromedio { get; set; }

        public Estudiante(int id, string nombre, double calificacionPromedio) : base(id, nombre)
        {
            this.CalificacionPromedio = calificacionPromedio;
        }

        public override void MostrarDatos()
        {
            MessageBox.Show("Id: " + Id +
                            "\nNombre: " + Nombre +
                            "\nPromedio: " + CalificacionPromedio);
        }
    }
}
```

### Paso 3: Diseño de la Interfaz Gráfica (GUI)

Se diseñaron un total de cuatro formularios:

- **Form1:** Actúa como el menú principal, con tres botones para navegar a los formularios de Profesor, Estudiante y Administrativo.
- **Form2, Form3 y Form4:** Formularios dedicados a la captura de datos de cada tipo de miembro. Cada uno contiene los `TextBox` necesarios para ingresar la información y tres botones de acción: "Agregar", "Limpiar" y "Cerrar".

**Figura 2:** Diseño de la interfaz en `Form3` para la captura de datos de un Estudiante.

### Paso 4: Programación de la Lógica de los Formularios

Se implementó la lógica para cada formulario.

- En **Form1**, se programó el evento `Click` de cada botón para instanciar y mostrar el formulario correspondiente.
- En los formularios secundarios, se programó la funcionalidad de los botones. El botón "**Agregar**" es el más importante, ya que lee los datos de los `TextBox`, crea un nuevo objeto de la clase correspondiente (ej. `new Estudiante(...)`) y finalmente llama a su método `MostrarDatos()` para exhibir la información en un `MessageBox`.

A continuación, se muestra el código del evento `Click` del botón "Agregar" en el formulario de Estudiante.

C#

```
// Resultado: Código del botón para agregar un nuevo estudiante
private void btnAgregar_Click(object sender, EventArgs e)
```

```

{
    if (!string.IsNullOrEmpty(txtId.Text) &&
        !string.IsNullOrEmpty(txtNombre.Text) &&
        !string.IsNullOrEmpty(txtPromedio.Text))
    {
        int id = int.Parse(txtId.Text);
        string nombre = txtNombre.Text;
        double promedio = double.Parse(txtPromedio.Text);

        // Se crea el objeto (instancia de la clase)
        Estudiante estudiante = new Estudiante(id, nombre, promedio);

        // Se invoca el método polimórfico
        estudiante.MostrarDatos();
    }
    else
    {
        MessageBox.Show("Llena todos los campos, castor ☹");
    }
}

```

#### Paso 5: Ejecución y Resultados Obtenidos

Finalmente, se ejecutó la aplicación para verificar su correcto funcionamiento. El flujo de operación fue el siguiente:

1. La aplicación inicia mostrando el menú principal (Form1).
2. Al hacer clic en el botón "Estudiante", se abre el formulario Form3.
3. Se ingresan los datos de prueba en los campos correspondientes.
4. Al presionar el botón "Agregar", el sistema crea el objeto Estudiante y muestra sus datos en una ventana de mensaje, confirmando que la herencia y el polimorfismo funcionan como se esperaba.

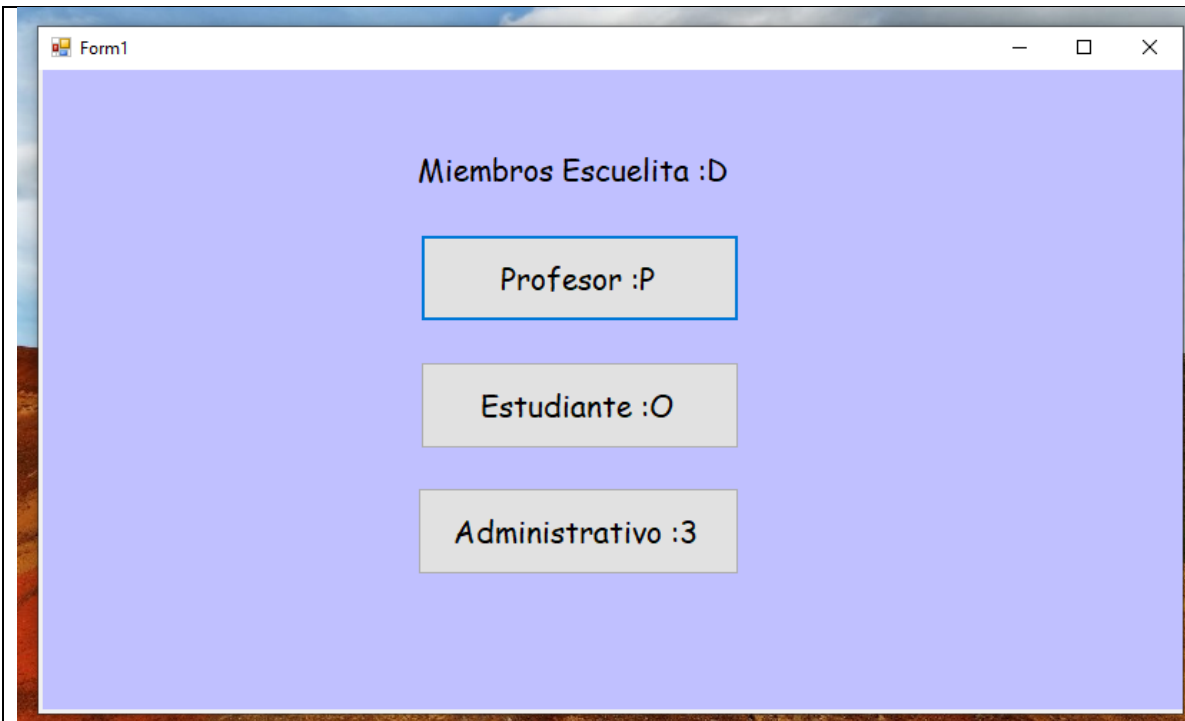
Las siguientes imágenes muestran la ejecución y el resultado final obtenido.

**Figura 3:** Menú principal de la aplicación en ejecución.

**Figura 4:** Formulario de Estudiante con datos de prueba listos para ser agregados.

**Figura 5: Resultado final obtenido.** La ventana de mensaje muestra los datos del objeto Estudiante, demostrando el funcionamiento del método MostrarDatos() sobrescrito.

Formulario 1 Principal:



Form1

Miembros Escuelita :D

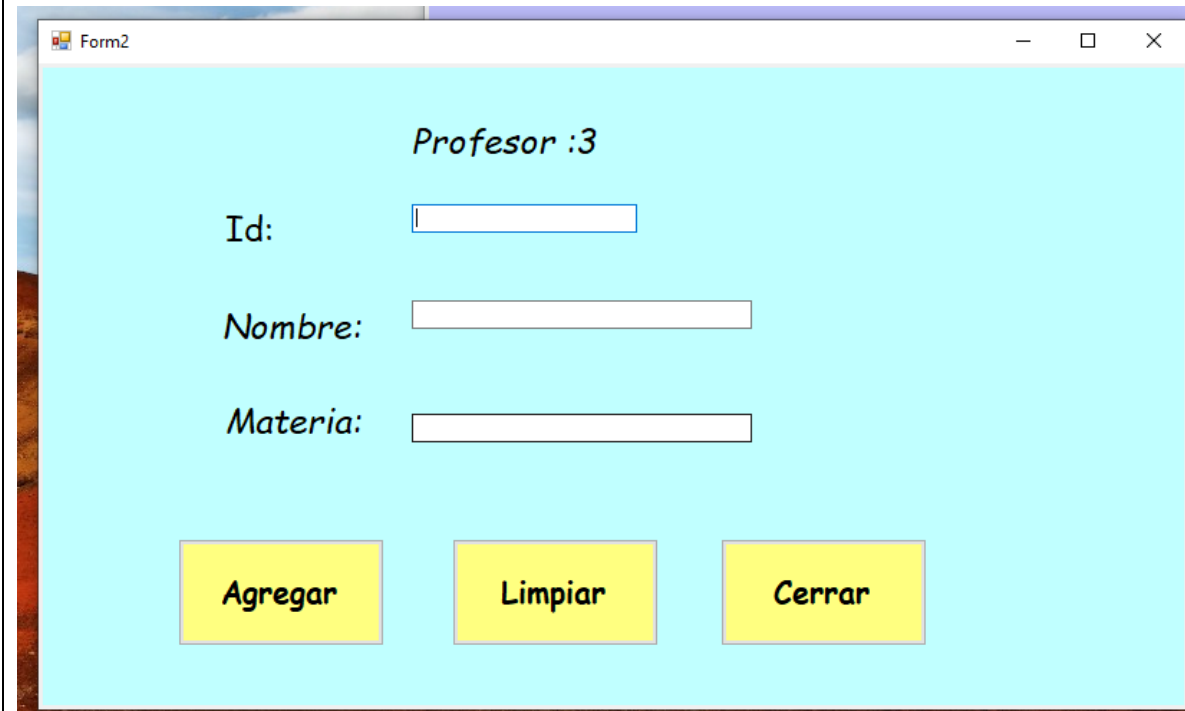
Profesor :P

Estudiante :O

Administrativo :3

This screenshot shows a Windows form titled 'Form1' with a light blue background. At the top center, the text 'Miembros Escuelita :D' is displayed. Below this text, there are three rectangular buttons stacked vertically. The top button is labeled 'Profesor :P', the middle button is labeled 'Estudiante :O', and the bottom button is labeled 'Administrativo :3'. All buttons have a light gray fill and a thin black border.

Formulario 2 Profesor:



Form2

*Profesor :3*

Id:

Nombre:

Materia:

Agregar Limpia Cerrar

This screenshot shows a Windows form titled 'Form2' with a light blue background. At the top center, the text 'Profesor :3' is displayed in an italicized font. Below this, there are three labels with corresponding text input fields: 'Id:' followed by a short text box, 'Nombre:' followed by a longer text box, and 'Materia:' followed by a text box of similar length to 'Nombre'. At the bottom of the form, there are three yellow rectangular buttons with black text: 'Agregar', 'Limpia', and 'Cerrar'.

Boton Agregar



Form2

Profesor :3

Id:

123

Nombre:

Paquito

Materia:

Matematicas

Id: 123

Nombre: Paquito

Materia: Matematicas

Aceptar

Agregar

Limpiar

Cerrar

Boton Limpiar:

Form2

Profesor :3

Id:

Nombre:

Materia:

Agregar

Limpiar

Cerrar

Formulario 3 Estudiante:

Programación Avanzada I

9

Form3

*Estudiante*

*Id*

*Nombre*

*Promedio*

*Agregar* *Limpiar* *Cerrar*

Boton Agregar:

Form3

*Estudiante*

*Id*

*Nombre*

*Promedio*

*Agregar* *Limpiar* *Cerrar*

Id: 300  
Nombre: Lia  
Calificacion Promedio: 100

*Aceptar*

Boton Limpiar:

Form3

*Estudiante*

*Id*

*Nombre*

*Promedio*

*Agregar* *Limpiar* *Cerrar*

Cerrar.

Formulario 4 Administrativo:

Form4

*Administrativo*

*Id:*

*Nombre:*

*Departamento:*

*Agregar* *Limpiar* *Cerrar*

Botón Agregar:

Form4

*Administrativo*

*Id:*

*Nombre:*

*Departamento:*

Id: 4500  
Nombre: Papupro  
Departamento: Papulandia

Botón Limpiar:

Form4

*Administrativo*

*Id:*

*Nombre:*

*Departamento:*

Cerrar.

## CONCLUSIÓN

se concluye que **se cumplieron exitosamente todos los objetivos planteados**. El propósito principal era aplicar los conceptos de **herencia** y **polimorfismo** de la Programación Orientada a Objetos en un entorno práctico, y la aplicación desarrollada es un reflejo directo de dicho aprendizaje.

La comparación entre los resultados esperados y los obtenidos es altamente satisfactoria. Se esperaba una aplicación de escritorio funcional con una jerarquía de clases clara y un método polimórfico que se comportara de manera diferente según el objeto, y **el resultado obtenido corresponde exactamente con esta expectativa**.

Sufrí bastante durante la realización de esta práctica, el señor castor superior no me quiso aplazar la fecha de entrega pero estoy feliz de que pude entregarla en tiempo y forma.

## BIBLIOGRAFÍA

Microsoft. (2024, 15 de julio). *Herencia (Guía de programación de C#)*. Microsoft Learn. Recuperado el 11 de septiembre de 2025, de <https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/inheritance>

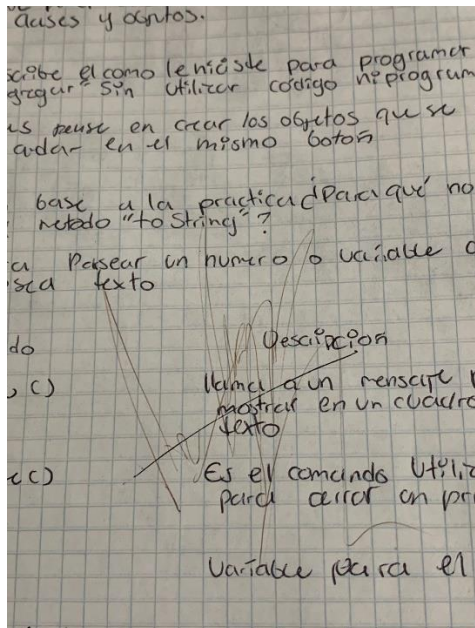
Microsoft. (2024, 23 de agosto). *Información general sobre Windows Forms (.NET Framework)*. Microsoft Learn. Recuperado el 11 de septiembre de 2025, de <https://learn.microsoft.com/es-es/dotnet/desktop/winforms/windows-forms-overview>

Microsoft. (n.d.). *Guía de programación de C#*. Microsoft Learn. Recuperado el 11 de septiembre de 2025, de <https://learn.microsoft.com/es-es/dotnet/csharp/programming-guide/>

Sharp, J. (2022). *C# 10 and .NET 6 – Modern Cross-Platform Development* (6th ed.). Packt Publishing.

Troelsen, A., & Japikse, P. (2022). *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming*. Apress.

## FIRMA DEL EXPERIMENTO



## AUTOEVALUACIÓN

11-09-25

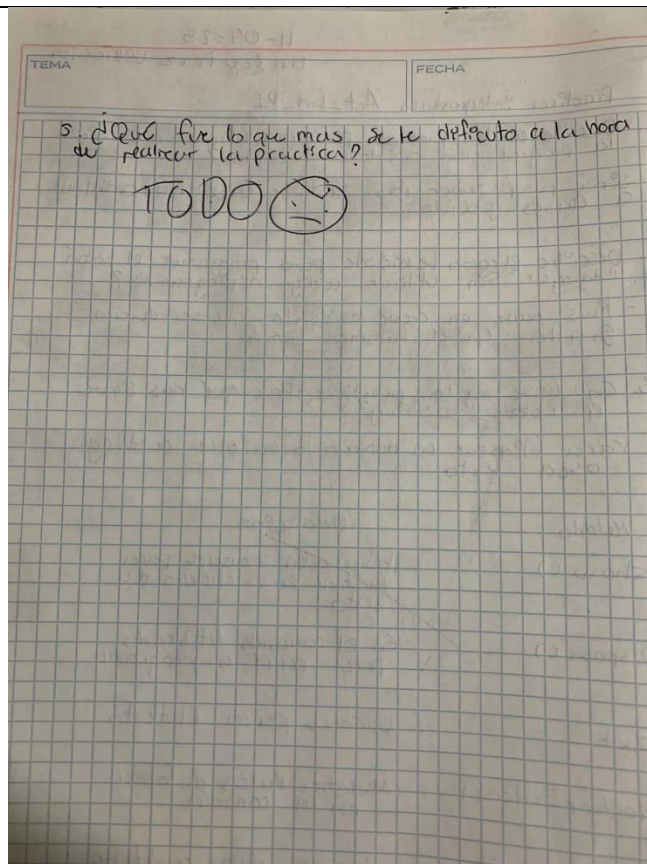
Lra Ezer Perez Villaseñor

TEMA

### Práctica Integradora Act-Lnt-P1

1. Describe con tus propias palabras ¿Para qué sirve la programación orientada a objetos?  
Sirve para saber como construir un sistema orientado a clases y objetos.
2. Describe el como le hiciste para programar el botón "Agregar" Sin utilizar código ni programas?  
- Plus pense en crear los objetos que se desean guardar en el mismo botón
3. En base a la practica ¿Para qué nos sirve el método "toString"?  
Para pasar un numero o variable a string o sea texto

Método	Descripción
Show ()	llama a un mensaje para mostrar en un cuadro de texto
Dispose ()	Es el comando utilizado para cerrar un programa
Text	Variable para el texto
Mostrar Datos ()	Muestra datos de forma en la consola
to string ()	Pasa datos a String o sea a texto.



5. La realización de los formularios y los objetos del mismo fueron algo confusos personalmente.