**Laboratorio de Programación Orientada a Objetos**

Práctica 8:

Polimorfismo.



#### Equipo No:\_\_3\_\_

#### Integrantes:

|  |  |
| --- | --- |
| N.L. | Nombre |
| 1 | Antonio Martínez Rodrigo. |
| 3 | Briseño Vázquez Angel Geovany. |
| 8 | Nishimura Guerrero Christian Jesús. |
|  |  |

#### Fecha de realización:

04/05/2023

Práctica 8. Polimorfismo.

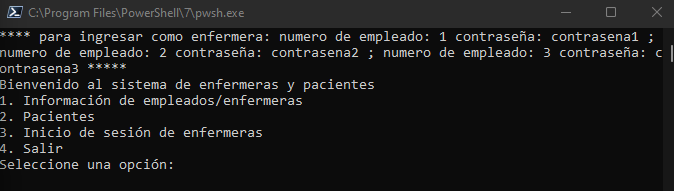
**PROBLEMAS PARA PRESENTAR:**

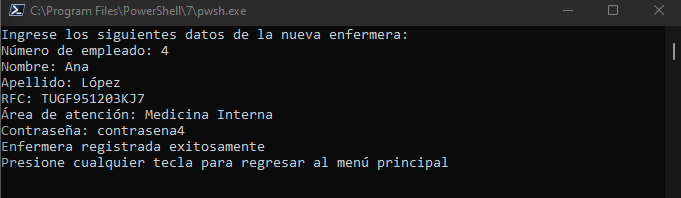
Realizar los siguientes programas en c#:

Texto

Descripción generada automáticamente con confianza media

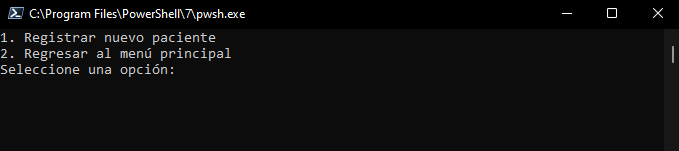
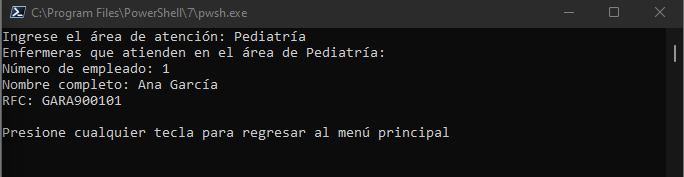
**Captura:**

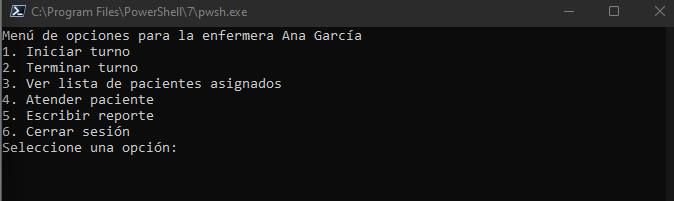
**Interfaz de usuario gráfica, Texto

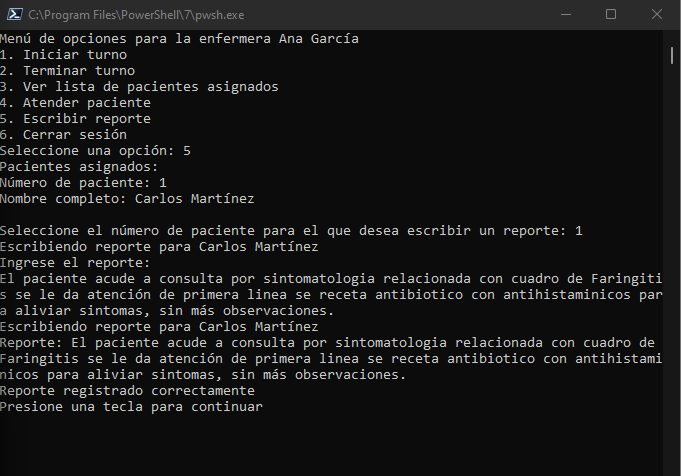
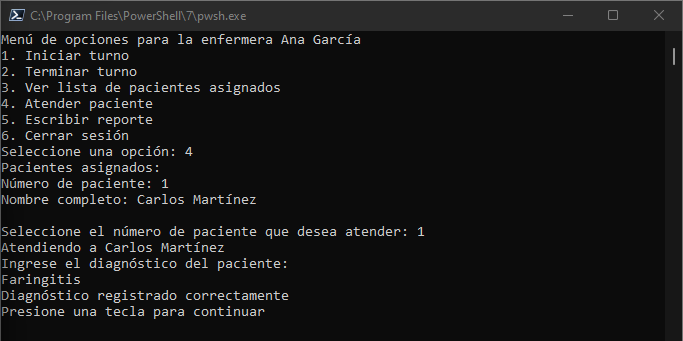
Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Código:**

**Archivo Empleado.cs:**

using System;

namespace ProyectoEnfermeras

{

    public class Persona

    {

        public string Nombre { get; set; }

        public string Apellido { get; set; }

        public Persona(string nombre, string apellido)

        {

            Nombre = nombre;

            Apellido = apellido;

        }

    }

    public class Empleado : Persona

    {

        public int NumeroEmpleado { get; set; }

        public string RFC { get; set; }

        public Empleado(int numeroEmpleado, string nombre, string apellido, string rfc) : base(nombre, apellido)

        {

            NumeroEmpleado = numeroEmpleado;

            RFC = rfc;

        }

        public virtual void RealizarTarea()

        {

            Console.WriteLine("Realizando tarea...");

        }

    }

    public class Enfermera : Empleado

    {

        public string Area { get; set; }

        public string Contrasena { get; set; }

        public Enfermera(int numeroEmpleado, string nombre, string apellido, string rfc, string area, string contrasena) : base(numeroEmpleado, nombre, apellido, rfc)

        {

            Area = area;

            Contrasena = contrasena;

        }

        public override void RealizarTarea()

        {

            Console.WriteLine("Realizando tarea de enfermería...");

        }

        public void EscribirReporte(Paciente paciente, string reporte)

        {

            Console.WriteLine("Escribiendo reporte para " + paciente.Nombre + " " + paciente.Apellido);

            Console.WriteLine("Reporte: " + reporte);

        }

    }

}

**Archivo Paciente.cs:**

using System;

namespace ProyectoEnfermeras

{

public class Paciente : Persona

    {

        public int NumeroPaciente { get; set; }

        public string CURP { get; set; }

        public string Direccion { get; set; }

        public Enfermera EnfermeraAsignada { get; set; }

        public string Diagnostico { get; set; }

        public Paciente(int numeroPaciente, string nombre, string apellido, string curp, string direccion, Enfermera enfermeraAsignada) : base(nombre, apellido)

        {

            NumeroPaciente = numeroPaciente;

            CURP = curp;

            Direccion = direccion;

            EnfermeraAsignada = enfermeraAsignada;

        }

        public void AsignarDiagnostico(string diagnostico)

        {

            Diagnostico = diagnostico;

            Console.WriteLine("Diagnóstico asignado: " + diagnostico);

        }

        public void AsignarDiagnostico(string diagnostico, Empleado medico)

        {

            Diagnostico = diagnostico;

            Console.WriteLine("Diagnóstico asignado por el médico " + medico.Nombre + " " + medico.Apellido + ": " + diagnostico);

        }

    }

}

**Archivo Program.cs:**

using System;

using System.Collections.Generic;

namespace ProyectoEnfermeras

{

    class Program

    {

        static List<Enfermera> enfermeras = new List<Enfermera>()

        {

            new Enfermera(1, "Ana", "García", "GARA900101", "Pediatría", "contrasena1"),

            new Enfermera(2, "Juan", "Pérez", "PEJU920202", "Oncología", "contrasena2"),

            new Enfermera(3, "María", "López", "LOMA850414", "Cirugía", "contrasena3")

        };

        static List<Paciente> pacientes = new List<Paciente>()

        {

            new Paciente(1, "Carlos", "Martínez", "MARC950101", "Av. Reforma #123", enfermeras[0]),

            new Paciente(2, "Laura", "González", "GOAL930202", "Calle 10 #456", enfermeras[1]),

            new Paciente(3, "Pedro", "Ramírez", "RAPD910303", "Paseo de la Reforma #789", enfermeras[2])

        };

        static void Main(string[] args)

        {

            MostrarMenu();

        }

        static void MostrarMenu()

        {

            Console.Clear();

            Console.WriteLine("\*\*\*\* para ingresar como enfermera: numero de empleado: 1 contraseña: contrasena1 ; numero de empleado: 2 contraseña: contrasena2 ; numero de empleado: 3 contraseña: contrasena3 \*\*\*\*\*");

            Console.WriteLine("Bienvenido al sistema de enfermeras y pacientes");

            Console.WriteLine("1. Información de empleados/enfermeras");

            Console.WriteLine("2. Pacientes");

            Console.WriteLine("3. Inicio de sesión de enfermeras");

            Console.WriteLine("4. Salir");

            Console.Write("Seleccione una opción: ");

            string opcion = Console.ReadLine();

            switch (opcion)

            {

                case "1":

                    MostrarMenuEmpleados();

                    break;

                case "2":

                    MostrarMenuPacientes();

                    break;

                case "3":

                    IniciarSesionEnfermera();

                    break;

                case "4":

                    Console.WriteLine("Gracias por utilizar el sistema");

                    break;

                default:

                    Console.WriteLine("Opción inválida");

                    break;

            }

            if (opcion != "4")

            {

                Console.WriteLine("Presione cualquier tecla para regresar al menú principal");

                Console.ReadKey();

                MostrarMenu();

            }

        }

        static void MostrarMenuEmpleados()

        {

            Console.Clear();

            Console.WriteLine("1. Registrar nueva enfermera");

            Console.WriteLine("2. Lista de enfermeras registradas");

            Console.WriteLine("3. Buscar enfermeras por área");

            Console.WriteLine("4. Regresar al menú principal");

            Console.Write("Seleccione una opción: ");

            string opcion = Console.ReadLine();

            switch (opcion)

            {

                case "1":

                    RegistrarEnfermera();

                    break;

                case "2":

                    MostrarListaEnfermeras();

                    break;

                case "3":

                    BuscarEnfermerasPorArea();

                    break;

                case "4":

                    break;

                default:

                    Console.WriteLine("Opción inválida");

                    break;

            }

        }

        static void RegistrarEnfermera()

        {

            Console.Clear();

            Console.WriteLine("Ingrese los siguientes datos de la nueva enfermera:");

            Console.Write("Número de empleado: ");

            int numeroEmpleado = int.Parse(Console.ReadLine());

            Console.Write("Nombre: ");

            string nombre = Console.ReadLine();

            Console.Write("Apellido: ");

            string apellido = Console.ReadLine();

            Console.Write("RFC: ");

            string rfc = Console.ReadLine();

            Console.Write("Área de atención: ");

            string area = Console.ReadLine();

            Console.Write("Contraseña: ");

            string contrasena = Console.ReadLine();

            Enfermera nuevaEnfermera = new Enfermera(numeroEmpleado, nombre, apellido, rfc, area, contrasena);

            enfermeras.Add(nuevaEnfermera);

            Console.WriteLine("Enfermera registrada exitosamente");

        }

        static void MostrarListaEnfermeras()

        {

            Console.Clear();

            Console.WriteLine("Lista de enfermeras registradas:");

            foreach (Enfermera enfermera in enfermeras)

            {

               Console.WriteLine("Número de empleado: " + enfermera.NumeroEmpleado);

Console.WriteLine("Nombre completo: " + enfermera.Nombre + " " + enfermera.Apellido);

Console.WriteLine("RFC: " + enfermera.RFC);

Console.WriteLine("Área de atención: " + enfermera.Area);

                Console.WriteLine();

            }

        }

        static void BuscarEnfermerasPorArea()

        {

            Console.Clear();

            Console.Write("Ingrese el área de atención: ");

            string area = Console.ReadLine();

           Console.WriteLine("Enfermeras que atienden en el área de " + area + ":");

            bool seEncontraronEnfermeras = false;

            foreach (Enfermera enfermera in enfermeras)

            {

                if (enfermera.Area == area)

                {

                   Console.WriteLine("Número de empleado: " + enfermera.NumeroEmpleado);

Console.WriteLine("Nombre completo: " + enfermera.Nombre + " " + enfermera.Apellido);

Console.WriteLine("RFC: " + enfermera.RFC);

                    Console.WriteLine();

                    seEncontraronEnfermeras = true;

                }

            }

            if (!seEncontraronEnfermeras)

            {

                Console.WriteLine("No se encontraron enfermeras que atiendan en esta área");

            }

        }

        static void MostrarMenuPacientes()

        {

            Console.Clear();

            Console.WriteLine("1. Registrar nuevo paciente");

            Console.WriteLine("2. Regresar al menú principal");

            Console.Write("Seleccione una opción: ");

            string opcion = Console.ReadLine();

            switch (opcion)

            {

                case "1":

                    RegistrarPaciente();

                    break;

                case "2":

                    break;

                default:

                    Console.WriteLine("Opción inválida");

                    break;

            }

        }

        static void RegistrarPaciente()

        {

            Console.Clear();

            Console.WriteLine("Ingrese los siguientes datos del nuevo paciente:");

            Console.Write("Nombre: ");

            string nombre = Console.ReadLine();

            Console.Write("Apellido: ");

            string apellido = Console.ReadLine();

            Console.Write("CURP: ");

            string curp = Console.ReadLine();

            Console.Write("Dirección: ");

            string direccion = Console.ReadLine();

            Random random = new Random();

            int numeroPaciente = random.Next(10000, 99999);

            Paciente nuevoPaciente = new Paciente(numeroPaciente, nombre, apellido, curp, direccion, enfermeras[random.Next(0, enfermeras.Count)]);

            pacientes.Add(nuevoPaciente);

            Console.WriteLine("Paciente registrado exitosamente con número de paciente " + numeroPaciente);

        }

        static void IniciarSesionEnfermera()

{

    Console.Clear();

    Console.Write("Ingrese su número de empleado: ");

    int numeroEmpleado = int.Parse(Console.ReadLine());

    Console.Write("Ingrese su contraseña: ");

    string contrasena = Console.ReadLine();

    Enfermera enfermera = enfermeras.Find(e => e.NumeroEmpleado == numeroEmpleado && e.Contrasena == contrasena);

    if (enfermera == null)

    {

        Console.WriteLine("Número de empleado o contraseña incorrectos");

        return;

    }

    Console.WriteLine("Bienvenido(a) " + enfermera.Nombre + " " + enfermera.Apellido);

    MostrarMenuEnfermera(enfermera);

}

static void MostrarMenuEnfermera(Enfermera enfermera)

{

    bool turnoIniciado = false;

    List<Paciente> pacientesAsignados = pacientes.FindAll(p => p.EnfermeraAsignada == enfermera);

    while (true)

    {

        Console.Clear();

       Console.WriteLine("Menú de opciones para la enfermera " + enfermera.Nombre + " " + enfermera.Apellido);

        Console.WriteLine("1. Iniciar turno");

        Console.WriteLine("2. Terminar turno");

        Console.WriteLine("3. Ver lista de pacientes asignados");

        Console.WriteLine("4. Atender paciente");

        Console.WriteLine("5. Escribir reporte");

        Console.WriteLine("6. Cerrar sesión");

        Console.Write("Seleccione una opción: ");

        string opcion = Console.ReadLine();

        switch (opcion)

        {

            case "1":

                if (turnoIniciado)

                {

                    Console.WriteLine("Ya ha iniciado su turno");

                }

                else

                {

                    Console.WriteLine("Ha iniciado su turno");

                    turnoIniciado = true;

                }

                break;

            case "2":

                if (turnoIniciado)

                {

                    Console.WriteLine("Ha terminado su turno");

                    turnoIniciado = false;

                }

                else

                {

                    Console.WriteLine("Aún no ha iniciado su turno");

                }

                break;

            case "3":

                if (pacientesAsignados.Count == 0)

                {

                    Console.WriteLine("No tiene pacientes asignados");

                }

                else

                {

                    Console.WriteLine("Pacientes asignados:");

                    foreach (Paciente paciente in pacientesAsignados)

                    {

                        Console.WriteLine("Número de paciente: " + paciente.NumeroPaciente);

Console.WriteLine("Nombre completo: " + paciente.Nombre + " " + paciente.Apellido);

Console.WriteLine("CURP: " + paciente.CURP);

Console.WriteLine("Dirección: " + paciente.Direccion);

                        Console.WriteLine();

                    }

                }

                break;

            case "4":

                if (pacientesAsignados.Count == 0)

                {

                    Console.WriteLine("No tiene pacientes asignados");

                }

                else

                {

                    Console.WriteLine("Pacientes asignados:");

                    foreach (Paciente paciente in pacientesAsignados)

                    {

                        Console.WriteLine("Número de paciente: " + paciente.NumeroPaciente);

Console.WriteLine("Nombre completo: " + paciente.Nombre + " " + paciente.Apellido);

                        Console.WriteLine();

                    }

                    Console.Write("Seleccione el número de paciente que desea atender: ");

                    int numeroPaciente = int.Parse(Console.ReadLine());

                    Paciente pacienteSeleccionado = pacientesAsignados.Find(p => p.NumeroPaciente == numeroPaciente);

                    if (pacienteSeleccionado == null)

                    {

                        Console.WriteLine("Número de paciente incorrecto");

                    }

                    else

                    {

                        Console.WriteLine("Atendiendo a " + pacienteSeleccionado.Nombre + " " + pacienteSeleccionado.Apellido);

                        Console.WriteLine("Ingrese el diagnóstico del paciente:");

                        string diagnostico = Console.ReadLine();

                        if (string.IsNullOrEmpty(diagnostico))

                        {

                            Console.WriteLine("No se ha ingresado un diagnóstico válido");

                        }

                        else

                        {

                            pacienteSeleccionado.AsignarDiagnostico(diagnostico);

                            Console.WriteLine("Diagnóstico registrado correctamente");

                        }

                    }

                }

                break;

            case "5":

                if (pacientesAsignados.Count == 0)

                {

                    Console.WriteLine("No tiene pacientes asignados");

                }

                else

                {

                    Console.WriteLine("Pacientes asignados:");

                    foreach (Paciente paciente in pacientesAsignados)

                    {

                       Console.WriteLine("Número de paciente: " + paciente.NumeroPaciente);

Console.WriteLine("Nombre completo: " + paciente.Nombre + " " + paciente.Apellido);

                        Console.WriteLine();

                    }

                    Console.Write("Seleccione el número de paciente para el que desea escribir un reporte: ");

                    int numeroPaciente = int.Parse(Console.ReadLine());

                    Paciente pacienteSeleccionado = pacientesAsignados.Find(p => p.NumeroPaciente == numeroPaciente);

                    if (pacienteSeleccionado == null)

                    {

                        Console.WriteLine("Número de paciente incorrecto");

                    }

                    else

                    {

                        Console.WriteLine("Escribiendo reporte para " + pacienteSeleccionado.Nombre + " " + pacienteSeleccionado.Apellido);

                        Console.WriteLine("Ingrese el reporte:");

                        string reporte = Console.ReadLine();

                        if (string.IsNullOrEmpty(reporte))

                        {

                            Console.WriteLine("No se ha ingresado un reporte válido");

                        }

                        else

                        {

                            enfermera.EscribirReporte(pacienteSeleccionado, reporte);

                            Console.WriteLine("Reporte registrado correctamente");

                        }

                    }

                }

                break;

            case "6":

                Console.WriteLine("Ha cerrado sesión");

                return;

            default:

                Console.WriteLine("Opción no válida");

                break;

        }

        Console.WriteLine("Presione una tecla para continuar");

        Console.ReadKey();

    }

}

    }

}

1. **¿Qué beneficio encuentras al implementar el concepto de Polimorfismo al programa**

**presentado?**

El principal beneficio de implementar el concepto de polimorfismo en el programa presentado es que permite escribir código más flexible y escalable. El polimorfismo permite que las clases sean utilizadas de manera polimórfica, es decir, que puedan ser tratadas como objetos de su clase base o de cualquier clase que herede de ella. Esto significa que se pueden agregar nuevas funcionalidades a las clases hijas sin tener que modificar el código de la clase base, lo cual puede ahorrar tiempo y reducir la complejidad del código.

1. **¿Qué problemas tuviste al actualizar tus clases conociendo y aplicando el concepto**

**de Polimorfismo?**

Uno de los problemas que se pueden presentar al actualizar clases existentes para implementar el polimorfismo es que se pueden introducir errores en el código existente. Además, puede ser necesario realizar cambios significativos en la estructura de las clases para implementar el polimorfismo de manera efectiva, lo cual puede requerir un mayor esfuerzo de desarrollo y pruebas.

Se utiliza la herencia y la sobrecarga de métodos para permitir que las clases hijas (como "Enfermera" y "Paciente") sobrescriban o agreguen comportamientos específicos a los métodos heredados de la clase base ("Empleado" y "Persona", respectivamente).

Por ejemplo, en la clase "Enfermera", se sobrescribió el método "RealizarTarea" para que muestre un mensaje específico para las enfermeras, mientras que en la clase "Paciente", se sobrecargó el método "AsignarDiagnostico" para permitir que diferentes tipos de empleados puedan asignar diagnósticos a los pacientes.

De esta manera, se puede lograr un código más flexible y escalable, permitiendo que las clases puedan ser utilizadas de manera polimórfica en diferentes contextos y situaciones.

1. **¿En qué momento del desarrollo se debe de aplicar el concepto de Polimorfismo?**

El concepto de polimorfismo se puede aplicar en cualquier momento del desarrollo de software, pero es especialmente útil cuando se anticipa que se necesitarán nuevas funcionalidades en el futuro o cuando se desea escribir código más flexible y escalable desde el principio. El polimorfismo también puede ser útil cuando se trabaja con interfaces o clases abstractas, ya que permite implementar diferentes comportamientos para diferentes implementaciones de la interfaz o clase abstracta.

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

Texto

Descripción generada automáticamente

**Captura:**

**Captura de pantalla de un celular

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Código:**

**Archivo Empleado.cs:**

using System;

using System.Collections.Generic;

public class Empleado {

    public int NumeroTrabajador { get; set; }

    public string Nombre { get; set; }

    public string Apellido { get; set; }

    public string Area { get; set; }

    private List<Producto> productos;

    public Empleado(int numeroTrabajador, string nombre, string apellido, string area) {

        NumeroTrabajador = numeroTrabajador;

        Nombre = nombre;

        Apellido = apellido;

        Area = area;

        productos = new List<Producto>();

    }

    public void AgregarProducto(Producto producto) {

        productos.Add(producto);

    }

    public void IniciarSesion() {

        Console.WriteLine("Iniciando sesión para el empleado " + Nombre + " " + Apellido + "...");

    }

    public void MostrarListaProductos() {

        foreach (Producto producto in productos) {

            producto.Informacion();

        }

    }

    public void VenderProducto(string numSerie) {

        Producto producto = productos.Find(p => p.NumSerie == numSerie);

        if (producto != null) {

            productos.Remove(producto);

            Console.WriteLine("Se vendió el producto con número de serie " + numSerie + " correctamente.");

        } else {

            Console.WriteLine("No se encontró ningún producto con el número de serie especificado.");

        }

    }

}

**Archivo Lavadora.cs:**

using System;

public class Lavadora : Producto {

    public double KgMax { get; set; }

    public string Dimensiones { get; set; }

    public string Color { get; set; }

    public Lavadora(string marca, string modelo, string numSerie, double kgMax, double costo, string dimensiones, string color) : base(marca, modelo, numSerie, costo) {

        KgMax = kgMax;

        Dimensiones = dimensiones;

        Color = color;

    }

    public override void Informacion() {

        Console.WriteLine("Tipo de producto: Lavadora");

        Console.WriteLine(this.ToString());

        Console.WriteLine("Kilogramos máximos: " + KgMax);

        Console.WriteLine("Dimensiones: " + Dimensiones);

        Console.WriteLine("Color: " + Color);

    }

}

**Archivo Producto.cs:**

using System;

public abstract class Producto {

    public string Marca { get; set; }

    public string Modelo { get; set; }

    public string NumSerie { get; set; }

    public double Costo { get; set; }

    public Producto(string marca, string modelo, string numSerie, double costo) {

        Marca = marca;

        Modelo = modelo;

        NumSerie = numSerie;

        Costo = costo;

    }

    public abstract void Informacion();

    public override string ToString() {

        return "Marca: " + Marca + ", Modelo: " + Modelo + ", Número de serie: " + NumSerie + ", Costo: " + Costo;

    }

}

**Archivo Televisor.cs:**

using System;

public class Televisor : Producto {

    public double Pulgadas { get; set; }

    public string Resolucion { get; set; }

    public Televisor(string marca, string modelo, string numSerie, double pulgadas, string resolucion, double costo) : base(marca, modelo, numSerie, costo) {

        Pulgadas = pulgadas;

        Resolucion = resolucion;

    }

    public override void Informacion() {

        Console.WriteLine("Tipo de producto: Televisor");

        Console.WriteLine(this.ToString());

        Console.WriteLine("Pulgadas: " + Pulgadas);

        Console.WriteLine("Resolución: " + Resolucion);

    }

}

**Archivo Program.cs:**

using System;

using System.Collections.Generic;

public class Program {

    public static void Main() {

        List<Empleado> listaEmpleados = new List<Empleado>();

        listaEmpleados.Add(new Empleado(1234, "Juan", "Pérez", "Ventas"));

        listaEmpleados.Add(new Empleado(5678, "María", "García", "Compras"));

        // Agregar algunos productos de ejemplo

        Lavadora lavadora1 = new Lavadora("LG", "WASH1", "1234", 10.5, 4000.0, "60x60x85cm", "Blanco");

        Lavadora lavadora2 = new Lavadora("Samsung", "ECO2", "5678", 8.0, 3500.0, "55x55x80cm", "Negro");

        Lavadora lavadora3 = new Lavadora("Mabe", "AQUA3", "9101", 12.0, 4500.0, "70x70x90cm", "Gris");

        Televisor televisor1 = new Televisor("Sony", "Bravia1", "2345", 55.0, "4K Ultra HD", 15000.0);

        Televisor televisor2 = new Televisor("Samsung", "QLED2", "6789", 65.0, "8K Ultra HD", 25000.0);

        Televisor televisor3 = new Televisor("LG", "NanoCell3", "1122", 49.0, "4K Ultra HD", 12000.0);

        Empleado empleado = listaEmpleados[0];

        empleado.AgregarProducto(lavadora1);

        empleado.AgregarProducto(lavadora2);

        empleado.AgregarProducto(lavadora3);

        empleado.AgregarProducto(televisor1);

        empleado.AgregarProducto(televisor2);

        empleado.AgregarProducto(televisor3);

        IniciarSesion(listaEmpleados);

    }

    public static void IniciarSesion(List<Empleado> listaEmpleados) {

        Console.WriteLine("\*\*\*\*Para fines prácticos los numeros de trabajador son: '1234' ó '5678' FIJATE BIEN EN EL AREA DEL TRABAJADOR UNO ES DE VENTAS Y EL OTRO DE COMPRAS, SÓLO EL DE VENTAS PUEDE VENDER\*\*\*\*\*");

        Console.WriteLine("Ingrese su número de trabajador:");

        int numTrabajador = Convert.ToInt32(Console.ReadLine());

        Empleado empleado = listaEmpleados.Find(e => e.NumeroTrabajador == numTrabajador);

        if (empleado != null) {

            empleado.IniciarSesion();

            Console.WriteLine("Nombre: " + empleado.Nombre);

            Console.WriteLine("Apellido: " + empleado.Apellido);

            Console.WriteLine("Área: " + empleado.Area);

            MostrarMenu(empleado);

        } else {

            Console.WriteLine("No se encontró ningún empleado con el número de trabajador especificado.");

        }

    }

    public static void MostrarMenu(Empleado empleado) {

        while (true) {

            Console.WriteLine("----- Menú -----");

            Console.WriteLine("1. Mostrar lista de productos");

            Console.WriteLine("2. Vender producto");

            Console.WriteLine("3. Salir");

            Console.WriteLine("Ingrese una opción:");

            int opcion = Convert.ToInt32(Console.ReadLine());

            switch (opcion) {

                case 1:

                    empleado.MostrarListaProductos();

                    break;

                case 2:

                    Console.WriteLine("Ingrese el número de serie del producto a vender:");

                    string numSerie = Console.ReadLine();

                    empleado.VenderProducto(numSerie);

                    break;

                case 3:

                    Console.WriteLine("Sesión cerrada.");

                    return;

                default:

                    Console.WriteLine("Opción no válida.");

                    break;

            }

        }

    }

}

**1. ¿Qué beneficio encuentras al implementar el concepto de Polimorfismo al problema**

**presentado?**

El beneficio principal de implementar el concepto de polimorfismo en el programa es que permite que el código sea más flexible y escalable. Al utilizar una clase abstracta o una interfaz con un método abstracto que se implementa en diferentes clases con diferentes implementaciones, se pueden agregar más tipos de productos en el futuro sin tener que modificar la lógica existente. Además, el código se vuelve más fácil de entender y mantener, ya que se pueden utilizar los mismos métodos y propiedades en diferentes objetos con diferentes comportamientos. Implementamos polimorfismo al utilizar una clase abstracta Producto y sus clases hijas Lavadora y Televisor, las cuales implementan el método abstracto Informacion de manera diferente. De esta manera, cuando se llama al método Informacion de un objeto Producto, se ejecuta la implementación correspondiente de la clase hija a la que pertenece el objeto. Esto es un ejemplo de polimorfismo en C#.

**2. ¿Qué problemas tuviste al actualizar tus clases conociendo y aplicando el concepto**

**de Polimorfismo?**

Al actualizar las clases para implementar el concepto de polimorfismo, el principal problema que se puede presentar es la necesidad de modificar la lógica existente para adaptarse a las nuevas clases y métodos abstractos. También puede haber un aumento en la complejidad del código debido a la necesidad de definir clases abstractas e interfaces y proporcionar implementaciones de métodos abstractos en las clases hijas.

**3. ¿En qué momento del desarrollo se debe de aplicar el concepto de Polimorfismo?**

El concepto de polimorfismo se debe aplicar en el momento en que se necesita crear un código más flexible y escalable. Si se espera que el código tenga que manejar diferentes tipos de objetos con diferentes comportamientos, entonces es recomendable utilizar clases abstractas o interfaces con métodos abstractos para implementar el polimorfismo. También es importante tener en cuenta que la implementación del polimorfismo puede agregar complejidad al código, por lo que se debe considerar si es necesario para el problema específico que se está resolviendo.

**Diagrama de clases ejercicio 2:**

**Imagen que contiene Texto

Descripción generada automáticamente**

*Tabla

Descripción generada automáticamente con confianza baja*

**Captura:**

**Texto

Descripción generada automáticamente con confianza bajaTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Código:**

**Archivo Cajero.cs:**

using System;

using System.Collections.Generic;

class Cajero : Empleado

{

    public int numVentanilla;

    public float comisiones;

    public Cajero(string nombre, string apellido, int numTrabajador, int numVentanilla) : base(nombre, apellido, numTrabajador)

    {

        this.numVentanilla = numVentanilla;

    }

    public override void CalcularComisiones(float monto)

    {

        this.comisiones += monto \* 0.0001f;

        Empleado.totalComisiones += this.comisiones;

    }

    public string AtenderCliente()

    {

        // Lógica para atender al cliente

        return "Cliente atendido";

    }

    public string RealizarDeposito(int numCuentaCliente, float cantidad)

    {

        // Lógica para realizar un depósito

        CalcularComisiones(cantidad);

        return "Depósito realizado";

    }

    public string RealizarRetiro(int numCuentaCliente, float cantidad)

    {

        // Lógica para realizar un retiro

        CalcularComisiones(cantidad);

        return "Retiro realizado";

    }

}

**Archivo Ejecutivo.cs:**

using System;

using System.Collections.Generic;

class Ejecutivo : Empleado

{

    public string numOficina;

    public float comisiones;

    public Ejecutivo(string nombre, string apellido, string numOficina, int numTrabajador) : base(nombre, apellido, numTrabajador)

    {

        this.numOficina = numOficina;

    }

    public override void CalcularComisiones(float monto)

    {

        this.comisiones += monto \* 0.001f;

        Empleado.totalComisiones += this.comisiones;

    }

    public string AbrirCuentaDebito(int numCuentaCliente, float cantidad)

    {

        // Lógica para abrir una cuenta de débito

        CalcularComisiones(cantidad);

        return "Cuenta de débito abierta";

    }

    public string AbrirCuentaCredito(int numCuentaCliente, float cantidad)

    {

        // Lógica para abrir una cuenta de crédito

        CalcularComisiones(cantidad);

        return "Cuenta de crédito abierta";

    }

    public string IniciarFondoInversion(int numCuentaCliente, float cantidad)

    {

        // Lógica para iniciar un fondo de inversión

        CalcularComisiones(cantidad);

        return "Fondo de inversión iniciado";

    }

}

**Archivo Empleado.cs:**

using System;

using System.Collections.Generic;

abstract class Empleado

{

    public string nombre;

    public string apellido;

    public int numTrabajador;

    public static float totalComisiones;

    public Empleado(string nombre, string apellido, int numTrabajador)

    {

        this.nombre = nombre;

        this.apellido = apellido;

        this.numTrabajador = numTrabajador;

    }

    public abstract void CalcularComisiones(float monto);

}

**Archivo Program.cs:**

using System;

using System.Collections.Generic;

class Program

{

    static void CalcularComisiones(float monto)

{

    foreach (Empleado empleado in listaEmpleados)

    {

        empleado.CalcularComisiones(monto);

    }

}

    static List<Empleado> listaEmpleados = new List<Empleado>();

    static void Main(string[] args)

    {

        listaEmpleados.Add(new Cajero("Juan", "Pérez", 123, 1));

        listaEmpleados.Add(new Cajero("Ana", "García", 124, 2));

       listaEmpleados.Add(new Ejecutivo("María", "Gómez", "456", 3));

        while (true)

        {

            Console.WriteLine("Bienvenido al sistema bancario");

            Console.WriteLine("1. Cajero");

            Console.WriteLine("2. Ejecutivo");

            Console.WriteLine("3. Consultar empleados");

            Console.WriteLine("4. Total de comisiones");

            Console.WriteLine("5. Salir");

            int opcion = int.Parse(Console.ReadLine());

            switch (opcion)

            {

                case 1:

    Console.WriteLine("Bienvenido, cajero");

    Console.WriteLine("1. Atender cliente");

    Console.WriteLine("2. Realizar depósito");

    Console.WriteLine("3. Realizar retiro");

    int subOpcionCajero = int.Parse(Console.ReadLine());

    switch (subOpcionCajero)

    {

        case 1:

            Console.WriteLine(((Cajero)listaEmpleados[0]).AtenderCliente());

            break;

        case 2:

            Console.WriteLine(((Cajero)listaEmpleados[0]).RealizarDeposito(123456, 1000));

            CalcularComisiones(1000);

            break;

        case 3:

            Console.WriteLine(((Cajero)listaEmpleados[0]).RealizarRetiro(234567, 500));

            CalcularComisiones(500);

            break;

        default:

            Console.WriteLine("Opción inválida");

            break;

    }

    break;

case 2:

    Console.WriteLine("Bienvenido, ejecutivo");

    Console.WriteLine("1. Abrir cuenta de débito");

    Console.WriteLine("2. Abrir cuenta de crédito");

    Console.WriteLine("3. Iniciar fondo de inversión");

    int subOpcionEjecutivo = int.Parse(Console.ReadLine());

    switch (subOpcionEjecutivo)

    {

        case 1:

            Console.WriteLine(((Ejecutivo)listaEmpleados[2]).AbrirCuentaDebito(345678, 5000));

            CalcularComisiones(5000);

            break;

        case 2:

            Console.WriteLine(((Ejecutivo)listaEmpleados[2]).AbrirCuentaCredito(456789, 10000));

            CalcularComisiones(10000);

            break;

        case 3:

            Console.WriteLine(((Ejecutivo)listaEmpleados[2]).IniciarFondoInversion(567890, 50000));

            CalcularComisiones(50000);

            break;

        default:

            Console.WriteLine("Opción inválida");

            break;

    }

    break;

case 4:

    float comisionesCajeros = 0;

    float comisionesEjecutivos = 0;

    foreach (Empleado empleado in listaEmpleados)

    {

        empleado.CalcularComisiones(0);

        if (empleado is Cajero)

        {

            comisionesCajeros += ((Cajero)empleado).comisiones;

        }

        else if (empleado is Ejecutivo)

        {

            comisionesEjecutivos += ((Ejecutivo)empleado).comisiones;

        }

    }

    Console.WriteLine("Comisiones totales de cajeros: " + comisionesCajeros);

    Console.WriteLine("Comisiones totales de ejecutivos: " + comisionesEjecutivos);

    break;

                case 5:

                    Console.WriteLine("¡Hasta luego!");

                    return;

                default:

                    Console.WriteLine("Opción inválida");

                    break;

            }

        }

    }

}

**1. ¿Qué beneficio encuentras al implementar el concepto de Polimorfismo al problema**

**presentado?**

El beneficio principal de implementar el concepto de polimorfismo en el problema presentado es la flexibilidad y la escalabilidad que proporciona. Al utilizar el polimorfismo, se puede definir una clase base que tenga un conjunto de comportamientos y luego extender esa clase base para crear subclases que implementen comportamientos específicos. Esto permite que la aplicación sea más modular y fácil de mantener, ya que los cambios en una clase base se reflejarán en todas las subclases que la utilicen.

**2. ¿Qué problemas tuviste al actualizar tus clases conociendo y aplicando el concepto**

**de Polimorfismo?**

Al actualizar las clases para hacer uso del polimorfismo, puede haber problemas si no se planifica adecuadamente. Si las clases se han diseñado originalmente sin tener en cuenta la posibilidad de la herencia y el polimorfismo, puede ser necesario realizar cambios significativos en la estructura de la clase para implementar estas técnicas. Además, es posible que se deban actualizar otros componentes de la aplicación que dependan de las clases actualizadas. Por lo tanto, es importante planificar cuidadosamente cualquier actualización de clases para que se implemente el polimorfismo de manera efectiva.

**3. ¿En qué momento del desarrollo se debe de aplicar el concepto de Polimorfismo?**

El concepto de polimorfismo se debe aplicar en el momento de diseñar y desarrollar la estructura de clases de la aplicación. Es importante tener en cuenta la posibilidad de la herencia y el polimorfismo al diseñar una clase base y cualquier subclase que se necesite. Al aplicar el polimorfismo en el diseño, se puede asegurar que la aplicación sea modular, escalable y fácil de mantener en el futuro. Es importante tener en cuenta que la implementación del polimorfismo puede requerir cambios significativos en la estructura de las clases existentes, por lo que es importante planificar cuidadosamente cualquier actualización de clases para incorporar el polimorfismo de manera efectiva.

**Diagrama de clases ejercicio 3:**

**Texto

Descripción generada automáticamente**

*Tabla

Descripción generada automáticamente*

**Captura:**

**Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Código:**

**Archivo CineTeatro.cs:**

using System;

using System.Collections.Generic;

namespace CineTeatroHermanosRodriguez

{

    class CineTeatro

    {

        public string Nombre { get; set; }

        public List<Pelicula> Peliculas { get; set; }

        public List<ObraTeatro> ObrasTeatro { get; set; }

        private List<Venta> Ventas { get; set; }

        public CineTeatro(string nombre)

        {

            Nombre = nombre;

            Peliculas = new List<Pelicula>();

            ObrasTeatro = new List<ObraTeatro>();

            Ventas = new List<Venta>();

        }

        public void AgregarPelicula(Pelicula pelicula)

        {

            Peliculas.Add(pelicula);

        }

        public void AgregarObraTeatro(ObraTeatro obraTeatro)

        {

            ObrasTeatro.Add(obraTeatro);

        }

        public void IniciarSesionDueño(string usuario, string contrasena)

        {

            if (usuario == "admin" && contrasena == "admin")

            {

                // Mostrar historial de ventas

                Console.WriteLine("Historial de Ventas:");

                foreach (Venta venta in Ventas)

                {

    Console.WriteLine("- " + venta.Funcion.Nombre + " (" + venta.Funcion.Horario + ") - " + venta.NumeroEntradas + " entradas - Total: " + venta.CostoTotal + " pesos");

                }

            }

            else

            {

                Console.WriteLine("Nombre de usuario o contraseña incorrectos");

            }

        }

        public void VenderBoletos(Funcion funcion, Cliente cliente, int numeroEntradas)

        {

            // Verificar que haya suficientes boletos disponibles

            if (funcion.CantidadBoletosDisponibles() < numeroEntradas)

            {

                Console.WriteLine("Lo sentimos, no hay suficientes boletos disponibles");

                return;

            }

            // Calcular costo total

            int costoTotal = numeroEntradas \* funcion.Costo;

            // Registrar venta

            Ventas.Add(new Venta(funcion, cliente, numeroEntradas, costoTotal));

            // Actualizar cantidad de boletos disponibles

            funcion.VenderBoletos(numeroEntradas);

            // Mostrar confirmación

            Console.WriteLine("¡Compra realizada exitosamente! " + numeroEntradas + " boletos para " + funcion.Nombre + " el " + funcion.Horario + " - Total: " + costoTotal + " pesos");

        }

    }

}

**Archivo Cliente.cs:**

namespace CineTeatroHermanosRodriguez

{

    class Cliente

    {

        public string Nombre { get; set; }

        public string Apellido { get; set; }

        public string Direccion { get; set; }

        public string NumeroCuentaPayPal { get; set; }

        public Cliente(string nombre, string apellido, string direccion, string numeroCuentaPayPal)

        {

            Nombre = nombre;

            Apellido = apellido;

            Direccion = direccion;

            NumeroCuentaPayPal = numeroCuentaPayPal;

        }

    }

}

**Archivo Funcion.cs:**

using System;

namespace CineTeatroHermanosRodriguez

{

    abstract class Funcion

{

    public string Nombre { get; set; }

    public string Horario { get; set; }

    public int Costo { get; set; }

    private int boletosDisponibles;

    public Funcion(string nombre, string horario, int costo, int boletosDisponibles)

    {

        Nombre = nombre;

        Horario = horario;

        Costo = costo;

        this.boletosDisponibles = boletosDisponibles;

    }

    public int CantidadBoletosDisponibles()

    {

        return boletosDisponibles;

    }

    public void VenderBoletos(int numeroBoletos)

    {

        boletosDisponibles -= numeroBoletos;

    }

    public abstract void MostrarInformacion();

}

    class Pelicula : Funcion

{

    public string Sala { get; set; }

    public Pelicula(string nombre, string sala, string horario, int costo) : base(nombre, horario, costo, 100)

    {

        Sala = sala;

    }

    public override void MostrarInformacion()

    {

        Console.WriteLine("- " + Nombre + " (" + Sala + ", " + Horario + ") - Costo: " + Costo + " pesos por persona");

    }

}

class ObraTeatro : Funcion

{

    public string Escenario { get; set; }

    public string Elenco { get; set; }

    public string Director { get; set; }

    public ObraTeatro(string nombre, string elenco, string director, string escenario, string horario, int costo) : base(nombre, horario, costo, 80)

    {

        Escenario = escenario;

        Elenco = elenco;

        Director = director;

    }

    public override void MostrarInformacion()

    {

        Console.WriteLine("- " + Nombre + " (" + Escenario + ", " + Horario + ") - Costo: " + Costo + " pesos por persona");

    }

}

    }

**Archivo Venta.cs:**

namespace CineTeatroHermanosRodriguez

{

    class Venta

    {

        public Funcion Funcion { get; set; }

        public Cliente Cliente { get; set; }

        public int NumeroEntradas { get; set; }

        public int CostoTotal { get; set; }

        public Venta(Funcion funcion, Cliente cliente, int numeroEntradas, int costoTotal)

        {

            Funcion = funcion;

            Cliente = cliente;

            NumeroEntradas = numeroEntradas;

            CostoTotal = costoTotal;

        }

    }

}

**Archivo Program.cs:**

using System;

namespace CineTeatroHermanosRodriguez

{

    class Program

    {

        static void Main(string[] args)

        {

            // Cargar las películas y obras de teatro por defecto

            Pelicula[] peliculas = {

                new Pelicula("1 Avengers: Endgame", "Sala 1", "12:00pm", 150),

                new Pelicula("2 Star Wars: El Ascenso de Skywalker", "Sala 2", "3:00pm", 120),

                new Pelicula("3 Joker", "Sala 3", "6:00pm", 100)

            };

            ObraTeatro[] obrasTeatro = {

                new ObraTeatro("1 Romeo y Julieta", "Leonardo DiCaprio, Claire Danes", "Baz Luhrmann", "Escenario 1", "8:00pm", 200),

                new ObraTeatro("2 Hamlet", "Kenneth Branagh", "Kenneth Branagh", "Escenario 2", "6:00pm", 180),

                new ObraTeatro("3 Los Miserables", "Hugh Jackman, Russell Crowe", "Tom Hooper", "Escenario 3", "9:00pm", 220)

            };

            // Crear el cine-teatro y agregar las películas y obras de teatro

            CineTeatro cineTeatro = new CineTeatro("Cine-Teatro Hermanos Rodríguez");

            foreach (Pelicula pelicula in peliculas)

            {

                cineTeatro.AgregarPelicula(pelicula);

            }

            foreach (ObraTeatro obraTeatro in obrasTeatro)

            {

                cineTeatro.AgregarObraTeatro(obraTeatro);

            }

            // Mostrar el menú principal

            bool salir = false;

            while (!salir)

            {

                 Console.WriteLine("\*\*\*\*\*\* para iniciar sesión como dueño es usuario: admin  contraseña: admin  \*\*\*\*\*\*\*");

                Console.WriteLine("Bienvenido al Cine-Teatro Hermanos Rodríguez");

                Console.WriteLine("1. Películas");

                Console.WriteLine("2. Obras de Teatro");

                Console.WriteLine("3. Iniciar sesión como dueño");

                Console.WriteLine("4. Registrarse como cliente");

                Console.WriteLine("5. Salir");

                Console.Write("Ingrese una opción: ");

                string opcion = Console.ReadLine();

                Console.WriteLine();

                switch (opcion)

                {

                    case "1":

                        // Mostrar películas

                        Console.WriteLine("Películas:");

                        foreach (Pelicula pelicula in cineTeatro.Peliculas)

                        {

                            Console.WriteLine("- " + pelicula.Nombre + " (" + pelicula.Sala + ", " + pelicula.Horario + ") - Costo: " + pelicula.Costo + " pesos por persona");

                        }

                        Console.WriteLine();

                        break;

                    case "2":

                        // Mostrar obras de teatro

                        Console.WriteLine("Obras de Teatro:");

                        foreach (ObraTeatro obraTeatro in cineTeatro.ObrasTeatro)

                        {

                            Console.WriteLine("- " + obraTeatro.Nombre + " (" + obraTeatro.Escenario + ", " + obraTeatro.Horario + ") - Costo: " + obraTeatro.Costo + " pesos por persona");

                        }

                        Console.WriteLine();

                        break;

                    case "3":

                        // Iniciar sesión como dueño

                        Console.Write("Ingrese el nombre de usuario: ");

                        string usuario = Console.ReadLine();

                        Console.Write("Ingrese la contraseña: ");

                        string contrasena = Console.ReadLine();

                        // Mostrar historial de ventas

                        cineTeatro.IniciarSesionDueño(usuario, contrasena);

                        Console.WriteLine();

                        break;

                    case "4":

                        // Registrarse como cliente

                        Console.Write("Ingrese su nombre: ");

                        string nombre = Console.ReadLine();

                        Console.Write("Ingrese su apellido: ");

                        string apellido = Console.ReadLine();

                        Console.Write("Ingrese su dirección: ");

                        string direccion = Console.ReadLine();

                        Console.Write("Ingrese su número de cuenta PayPal: ");

                        string numeroCuentaPayPal = Console.ReadLine();

                        // Crear el cliente y mostrar cartelera

                        Cliente cliente = new Cliente(nombre, apellido, direccion, numeroCuentaPayPal);

                        Console.WriteLine();

                        Console.WriteLine("Cartelera:");

                        Console.WriteLine("1. Películas");

                        Console.WriteLine("2. Obras de Teatro");

                        Console.Write("Ingrese una opción: ");

                        string opcionCartelera = Console.ReadLine();

                        Console.WriteLine();

                        switch (opcionCartelera)

                        {

                            case "1":

                                // Mostrar películas y vender boletos

                                Console.WriteLine("Películas:");

                                foreach (Pelicula pelicula in cineTeatro.Peliculas)

                                {

                                    pelicula.MostrarInformacion();

                                }

                                Console.WriteLine();

                                Console.Write("Ingrese el número de la película que desea ver: ");

                                int numeroPelicula = int.Parse(Console.ReadLine());

                                Pelicula peliculaSeleccionada = cineTeatro.Peliculas[numeroPelicula - 1];

                                Console.Write("Ingrese el número de boletos que desea comprar: ");

                                int numeroBoletos = int.Parse(Console.ReadLine());

                                cineTeatro.VenderBoletos(peliculaSeleccionada, cliente, numeroBoletos);

                                break;

                            case "2":

                                // Mostrar obras de teatro y vender boletos

                                Console.WriteLine("Obras de Teatro:");

                                foreach (ObraTeatro obraTeatro in cineTeatro.ObrasTeatro)

                                {

                                    obraTeatro.MostrarInformacion();

                                }

                                Console.WriteLine();

                                Console.Write("Ingrese el número de la obra de teatro que desea ver: ");

                                int numeroObraTeatro = int.Parse(Console.ReadLine());

                                ObraTeatro obraTeatroSeleccionada = cineTeatro.ObrasTeatro[numeroObraTeatro - 1];

                                Console.Write("Ingrese el número de boletos que desea comprar: ");

                                int numeroBoletosObraTeatro = int.Parse(Console.ReadLine());

                                cineTeatro.VenderBoletos(obraTeatroSeleccionada, cliente, numeroBoletosObraTeatro);

                                break;

                            default:

                                Console.WriteLine("Opción inválida");

                                break;

                        }

                        Console.WriteLine();

                        break;

                    case "5":

                        // Salir

                        salir = true;

                        break;

                    default:

                        Console.WriteLine("Opción inválida");

                        Console.WriteLine();

                        break;

                }

            }

        }

    }

}

**1. ¿Qué beneficio encuentras al implementar el concepto de Polimorfismo al problema**

**presentado?**

La implementación del concepto de Polimorfismo en el problema presentado permite que las clases Pelicula y ObraTeatro compartan una misma interfaz pública (el método MostrarInformacion() de la clase Funcion). Esto simplifica el código al permitir que ambas clases se traten de manera polimórfica y se reduzca la cantidad de código específico que se tiene que escribir para cada una de las clases.

Además, el polimorfismo permite escribir un código más flexible y escalable, ya que se pueden agregar nuevas clases que hereden de la clase Funcion sin tener que cambiar el código existente que las utiliza.

**2. ¿Qué problemas tuviste al actualizar tus clases conociendo y aplicando el concepto**

**de Polimorfismo?**

Al actualizar las clases para incluir el concepto de Polimorfismo, el principal problema que se puede presentar es la necesidad de modificar el código existente para adaptarlo a la nueva estructura de clases. En este caso, se tuvo que agregar un método abstracto a la clase Funcion y sobrescribirlo en las clases Pelicula y ObraTeatro.

Otro posible problema es que la implementación del Polimorfismo puede hacer que el código sea más complejo y difícil de entender para los desarrolladores que no están familiarizados con el concepto.

**3. ¿En qué momento del desarrollo se debe de aplicar el concepto de Polimorfismo?**

El concepto de Polimorfismo se debe aplicar en el momento en que se necesite trabajar con objetos de diferentes clases de manera uniforme, es decir, cuando se necesita que las clases compartan una misma interfaz pública. Esto puede ocurrir en diferentes momentos del desarrollo, dependiendo de los requisitos del proyecto.

En general, se recomienda utilizar el polimorfismo desde el principio del diseño del proyecto, ya que esto puede simplificar el código y hacerlo más escalable a medida que se agregan nuevas clases. Sin embargo, si el polimorfismo no se consideró desde el principio, aún es posible implementarlo en una etapa posterior del desarrollo.

**Diagrama de clases ejercicio 4:**

**Diagrama, Texto

Descripción generada automáticamente**

**CONCLUSIONES INDIVIDUALES:**

Antonio Martínez Rodrigo.

El uso del polimorfismo es importante en la programación orientada a objetos debido a los siguientes motivos:

1. Flexibilidad: El polimorfismo permite que un objeto de una clase se comporte como otro objeto de una clase relacionada. Esto proporciona un alto grado de flexibilidad en el diseño y la implementación de una aplicación.

2. Reutilización de código: Al utilizar la herencia y el polimorfismo, se puede reutilizar el código de la clase base en las subclases, lo que reduce la duplicación de código y aumenta la eficiencia del desarrollo.

3. Mantenibilidad del código: Al utilizar el polimorfismo, se puede actualizar la implementación de una clase base y todas sus subclases se actualizarán automáticamente. Esto facilita el mantenimiento del código y reduce la posibilidad de errores.

4. Abstracción: El polimorfismo permite que las clases trabajen con objetos de otras clases sin conocer los detalles de implementación de esas clases. Esto proporciona una mayor abstracción y ayuda a reducir la complejidad del código.

Briseño Vázquez Angel Geovany.

En conclusión, el concepto de Polimorfismo es muy importante en la programación orientada a objetos, ya que permite trabajar con objetos de diferentes clases de manera uniforme y simplifica el código al permitir que varias clases compartan una misma interfaz pública. Pudimos ver cómo la implementación del Polimorfismo en las clases permitió tratar a las clases de manera polimórfica y reducir la cantidad de código específico que se tiene que escribir para cada una de las clases. Además, pudimos ver cómo el Polimorfismo hace que el código sea más flexible y escalable, ya que se pueden agregar nuevas clases que hereden de la clase sin tener que cambiar el código existente que las utiliza.

En general, el Polimorfismo es una herramienta poderosa en la programación orientada a objetos y su uso adecuado puede simplificar el código, hacerlo más mantenible y permitir una mayor flexibilidad y escalabilidad del software. Es importante entender y aplicar correctamente el concepto de Polimorfismo en nuestros programas para aprovechar todos sus beneficios y mejorar la calidad del software que desarrollamos.

Nishimura Guerrero Christian Jesús.

El polimorfismo es un concepto fundamental en la programación orientada a objetos que permite crear código más flexible y escalable. En el caso de mi ejercicio Implementamos el polimorfismo utilizando una clase abstracta Producto y sus clases hijas Lavadora y Televisor, las cuales implementaron el método abstracto Informacion de manera diferente.

Al utilizar el polimorfismo, pudimos crear una lista de productos que contenía tanto lavadoras como televisores, y pudimos llamar al método Informacion en cada objeto de la lista sin saber de antemano qué tipo de objeto era. Esto nos permitió escribir código más genérico y reutilizable, ya que no tuvimos que escribir lógica específica para cada tipo de objeto.

Además, al utilizar el polimorfismo, pudimos agregar fácilmente nuevos tipos de productos en el futuro simplemente creando una nueva clase que implementara el método Informacion. Esto hace que el código sea más modular y fácil de mantener.

En resumen, el polimorfismo es una herramienta poderosa en la programación orientada a objetos que permite crear código más flexible, escalable y modular. Pudimos ver cómo el polimorfismo nos permitió escribir código más genérico y reutilizable, y cómo facilitó la adición de nuevos tipos de productos en el futuro.

**CONCLUSIONES POR EQUIPO:**

La implementación del Polimorfismo en nuestros códigos es importante porque nos permite tratar a objetos de diferentes clases de manera uniforme y simplifica la estructura de nuestro código. Al utilizar el Polimorfismo, podemos definir una interfaz común para un conjunto de clases relacionadas y hacer que estas clases compartan esa interfaz pública.

Esto nos permite escribir código más genérico y reutilizable, ya que podemos trabajar con diferentes objetos de manera polimórfica, sin importar si son de diferentes clases. Por ejemplo, si tenemos una clase "Animal" y varias clases derivadas como "Perro", "Gato", "Conejo", etc., podemos trabajar con todos los objetos de estas clases de manera uniforme utilizando la interfaz pública de la clase base "Animal".

Además, el Polimorfismo hace que nuestro código sea más flexible y escalable, ya que podemos agregar nuevas clases que hereden de la clase base sin tener que cambiar el código existente que las utiliza. Esto es especialmente útil en proyectos grandes y complejos, donde se pueden agregar nuevas clases con frecuencia.

Por último, la implementación del Polimorfismo en nuestros códigos mejora la calidad del software que desarrollamos, ya que permite escribir código más genérico, reutilizable, flexible y escalable. Es una herramienta poderosa en la programación orientada a objetos y su uso adecuado puede simplificar el código y hacer que sea más mantenible y fácil de entender.