**Laboratorio de Programación Orientada a Objetos**

Práctica 7:

Herencia.



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

#### Integrantes:

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

#### Fecha de realización:

13/04/2023

Práctica 7. Herencia.

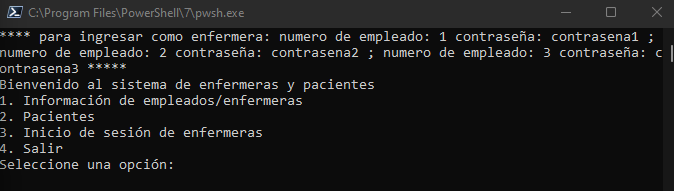
**PROBLEMAS PARA PRESENTAR:**

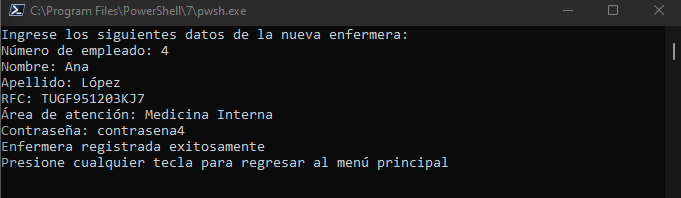
Realizar los siguientes programas en c#.

Texto

Descripción generada automáticamente con confianza media

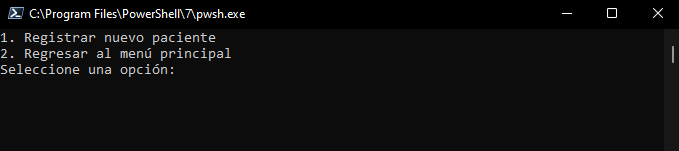
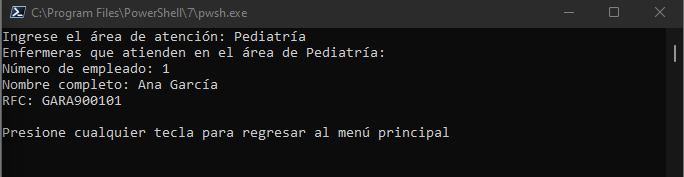
**Captura:**

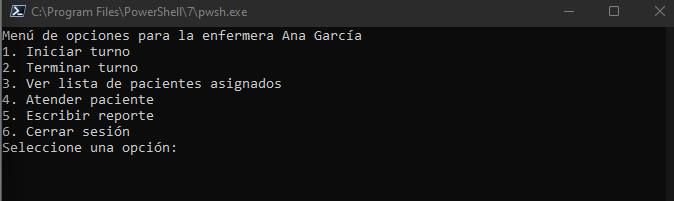
**Interfaz de usuario gráfica, Texto

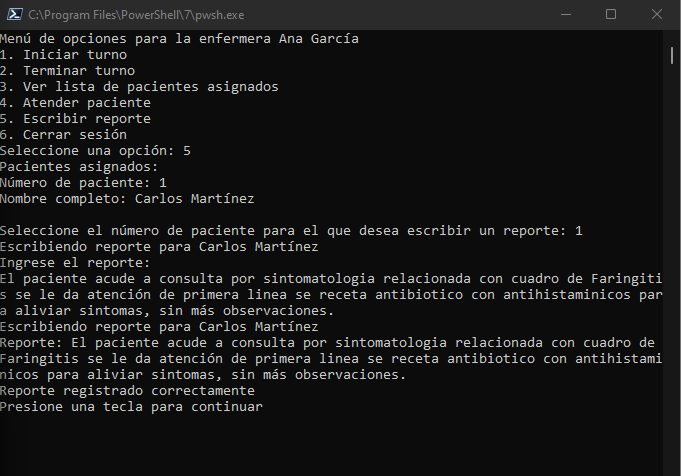
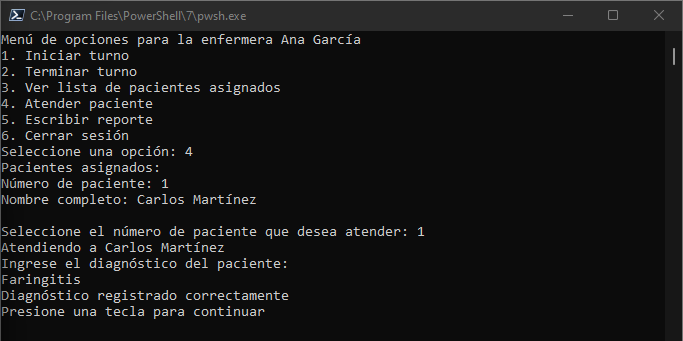
Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente**

**Código:**

**Archivo Empleado.cs:**

using System;

namespace ProyectoEnfermeras

{

    public class Empleado

    {

        public int NumeroEmpleado { get; set; }

        public string Nombre { get; set; }

        public string Apellido { get; set; }

        public string RFC { get; set; }

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

        {

            NumeroEmpleado = numeroEmpleado;

            Nombre = nombre;

            Apellido = apellido;

            RFC = rfc;

        }

    }

  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 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

    {

        public int NumeroPaciente { get; set; }

        public string Nombre { get; set; }

        public string Apellido { 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)

        {

            NumeroPaciente = numeroPaciente;

            Nombre = nombre;

            Apellido = apellido;

            CURP = curp;

            Direccion = direccion;

            EnfermeraAsignada = enfermeraAsignada;

        }

        public void AsignarDiagnostico(string diagnostico)

        {

            Diagnostico = 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();

    }

}

    }

}

**Diagrama de clases:**

**Texto

Descripción generada automáticamente**

Texto

Descripción generada automáticamente

**Captura:**

**Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

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<Lavadora> listaLavadoras;

    private List<Televisor> listaTelevisores;

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

        NumeroTrabajador = numTrabajador;

        Nombre = nombre;

        Apellido = apellido;

        Area = area;

        listaLavadoras = new List<Lavadora>();

        listaTelevisores = new List<Televisor>();

    }

    public void IniciarSesion() {

        Console.WriteLine("Iniciando sesión de empleado...");

    }

    public void AgregarLavadora(string marca, string modelo, string numSerie, double kgMax, double costo, string dimensiones, string color) {

        Lavadora lavadora = new Lavadora(marca, modelo, numSerie, kgMax, costo, dimensiones, color);

        listaLavadoras.Add(lavadora);

    }

    public void AgregarTelevisor(string marca, string modelo, string numSerie, double pulgadas, string resolucion, double costo) {

        Televisor televisor = new Televisor(marca, modelo, numSerie, pulgadas, resolucion, costo);

        listaTelevisores.Add(televisor);

    }

    public void MostrarListaLavadoras() {

        foreach (Lavadora lavadora in listaLavadoras) {

            Console.WriteLine(lavadora);

        }

    }

    public void MostrarListaTelevisores() {

        foreach (Televisor televisor in listaTelevisores) {

            Console.WriteLine(televisor);

        }

    }

    public void VenderLavadora(string numSerie) {

        Lavadora lavadora = listaLavadoras.Find (e => e.NumSerie == numSerie);

     if (lavadora != null) {

        listaLavadoras.Remove(lavadora);

        Console.WriteLine("Se vendió la lavadora con número de serie " + numSerie + ".");

    } else {

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

    }

}

public void VenderTelevisor(string numSerie) {

    Televisor televisor = listaTelevisores.Find(e => e.NumSerie == numSerie);

    if (televisor != null) {

        listaTelevisores.Remove(televisor);

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

    } else {

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

    }

}

}

**Archivo Lavadora.cs:**

using System;

public class Lavadora {

public string Marca { get; set; }

public string Modelo { get; set; }

public string NumSerie { get; set; }

public double KgMax { get; set; }

public double Costo { 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) {

    Marca = marca;

    Modelo = modelo;

    NumSerie = numSerie;

    KgMax = kgMax;

    Costo = costo;

    Dimensiones = dimensiones;

    Color = color;

}

public override string ToString() {

    return "Marca: " + Marca + ", Modelo: " + Modelo + ", Número de serie: " + NumSerie + ", Kilogramos máximos: " + KgMax + ", Costo: " + Costo + ", Dimensiones: " + Dimensiones + ", 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 {

public string Marca { get; set; }

public string Modelo { get; set; }

public string NumSerie { get; set; }

public double Pulgadas { get; set; }

public string Resolucion { get; set; }

public double Costo { get; set; }

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

    Marca = marca;

    Modelo = modelo;

    NumSerie = numSerie;

    Pulgadas = pulgadas;

    Resolucion = resolucion;

    Costo = costo;

}

public override string ToString() {

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

}

}

**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.AgregarLavadora(lavadora1.Marca, lavadora1.Modelo, lavadora1.NumSerie, lavadora1.KgMax, lavadora1.Costo, lavadora1.Dimensiones, lavadora1.Color);

        empleado.AgregarLavadora(lavadora2.Marca, lavadora2.Modelo, lavadora2.NumSerie, lavadora2.KgMax, lavadora2.Costo, lavadora2.Dimensiones, lavadora2.Color);

        empleado.AgregarLavadora(lavadora3.Marca, lavadora3.Modelo, lavadora3.NumSerie, lavadora3.KgMax, lavadora3.Costo, lavadora3.Dimensiones, lavadora3.Color);

        empleado.AgregarTelevisor(televisor1.Marca, televisor1.Modelo, televisor1.NumSerie, televisor1.Pulgadas, televisor1.Resolucion, televisor1.Costo);

        empleado.AgregarTelevisor(televisor2.Marca, televisor2.Modelo, televisor2.NumSerie, televisor2.Pulgadas, televisor2.Resolucion, televisor2.Costo);

        empleado.AgregarTelevisor(televisor3.Marca, televisor3.Modelo, televisor3.NumSerie, televisor3.Pulgadas, televisor3.Resolucion, televisor3.Costo);

        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\*\*\*\*\*");

        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("\n-----MENU-----");

            Console.WriteLine("1. Agregar lavadora");

            Console.WriteLine("2. Agregar televisor");

            Console.WriteLine("3. Mostrar lista de lavadoras");

            Console.WriteLine("4. Mostrar lista de televisores");

            Console.WriteLine("5. Vender lavadora");

            Console.WriteLine("6. Vender televisor");

            Console.WriteLine("7. Salir");

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

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

            switch (opcion) {

                case 1:

                    Console.WriteLine("Ingrese la marca de la lavadora:");

                    string marcaLavadora = Console.ReadLine();

                    Console.WriteLine("Ingrese el modelo de la lavadora:");

                    string modeloLavadora = Console.ReadLine();

                    Console.WriteLine("Ingrese el número de serie de la lavadora:");

                    string numSerieLavadora = Console.ReadLine();

                    Console.WriteLine("Ingrese los kilogramos máximos de la lavadora:");

                    double kgMaxLavadora = Convert.ToDouble(Console.ReadLine());

                    Console.WriteLine("Ingrese el costo de la lavadora:");

                    double costoLavadora = Convert.ToDouble(Console.ReadLine());

                    Console.WriteLine("Ingrese las dimensiones de la lavadora:");

                    string dimensionesLavadora = Console.ReadLine();

                    Console.WriteLine("Ingrese el color de la lavadora:");

                    string colorLavadora = Console.ReadLine();

                    empleado.AgregarLavadora(marcaLavadora, modeloLavadora, numSerieLavadora, kgMaxLavadora, costoLavadora, dimensionesLavadora, colorLavadora);

                    Console.WriteLine("Se agregó la lavadora correctamente.");

                    break;

                case 2:

                    Console.WriteLine("Ingrese la marca del televisor:");

                    string marcaTelevisor = Console.ReadLine();

                    Console.WriteLine("Ingrese el modelo del televisor:");

                    string modeloTelevisor = Console.ReadLine();

                    Console.WriteLine("Ingrese el número de serie del televisor:");

                    string numSerieTelevisor = Console.ReadLine();

                    Console.WriteLine("Ingrese las pulgadas del televisor:");

                    double pulgadasTelevisor = Convert.ToDouble(Console.ReadLine());

                    Console.WriteLine("Ingrese la resolución del televisor:");

                    string resolucionTelevisor = Console.ReadLine();

                    Console.WriteLine("Ingrese el costo del televisor:");

                    double costoTelevisor = Convert.ToDouble(Console.ReadLine());

                    empleado.AgregarTelevisor(marcaTelevisor, modeloTelevisor, numSerieTelevisor, pulgadasTelevisor, resolucionTelevisor, costoTelevisor);

                    Console.WriteLine("Se agregó el televisor correctamente.");

                    break;

                case 3:

                    Console.WriteLine("Lista de lavadoras:");

                    empleado.MostrarListaLavadoras();

                    break;

                case 4:

                    Console.WriteLine("Lista de televisores:");

                    empleado.MostrarListaTelevisores();

                    break;

                case 5:

                    Console.WriteLine("Ingrese el número de serie de la lavadora a vender:");

                    string numSerieLavadoraVender = Console.ReadLine();

                    empleado.VenderLavadora(numSerieLavadoraVender);

                    break;

                case 6:

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

                    string numSerieTelevisorVender = Console.ReadLine();

                    empleado.VenderTelevisor(numSerieTelevisorVender);

                    break;

                case 7:

                    Console.WriteLine("Sesión finalizada.");

                    return;

                default:

                    Console.WriteLine("Opción inválida.");

                    break;

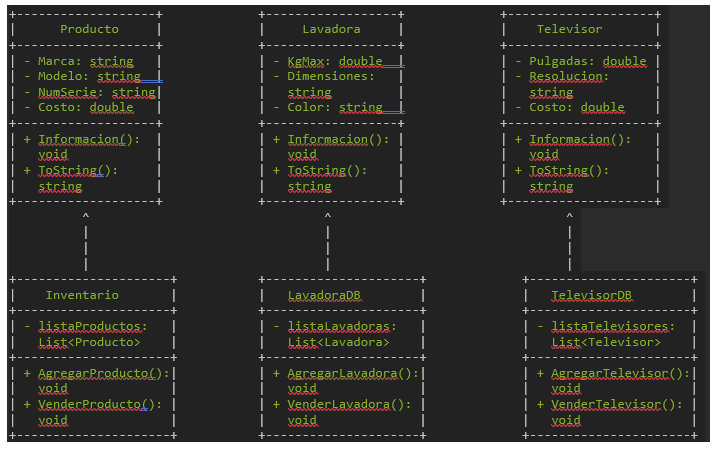
            }

        }

    }

}

**Diagrama de clases ejercicio 2:**



*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;

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 virtual void CalcularComisiones(float monto)

    {

        // Este método se implementará en las clases hijas

    }

}

**Archivo Program.cs:**

using System;

using System.Collections.Generic;

class Program

{

    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));

                            break;

                        case 3:

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

                            break;

                        case 2:

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

                            break;

                        case 3:

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

                            break;

                        default:

                            Console.WriteLine("Opción inválida");

                            break;

                    }

                    break;

                case 3:

                    Console.WriteLine("Lista de empleados:");

                    foreach (Empleado empleado in listaEmpleados)

                    {

                        Console.WriteLine("Nombre: " + empleado.nombre + ", Apellido: " + empleado.apellido + ", Número de trabajador: " + empleado.numTrabajador);

                    }

                    break;

                case 4:

                    float comisionesCajeros = 0;

                    float comisionesEjecutivos = 0;

                    foreach (Empleado empleado in listaEmpleados)

                    {

                        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;

            }

        }

    }

}

**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;

        }

    }

    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;

        }

    }

    class ObraTeatro : Funcion

    {

        public string Actores { get; set; }

        public string Director { get; set; }

        public string Escenario { get; set; }

        public ObraTeatro(string nombre, string actores, string director, string escenario, string horario, int costo) : base(nombre, horario, costo, 50)

        {

            Actores = actores;

            Director = director;

            Escenario = escenario;

        }

    }

}

**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)

                                {

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

                                }

                                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)

                                {

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

                                }

                                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;

                }

            }

        }

    }

}

**Diagrama de clases ejercicio 4:**

**Texto

Descripción generada automáticamente**

**CONCLUSIONES INDIVIDUALES:**

Antonio Martínez Rodrigo.

En mi opinión, este programa es un ejemplo de un sistema bancario que utiliza herencia en su implementación. En la parte superior del código, se define una lista de objetos Empleado, que es la clase base para los tipos de empleados Cajero y Ejecutivo.

En el método Main, se agregan tres objetos a la lista: dos objetos del tipo Cajero y un objeto del tipo Ejecutivo. Luego, se presenta un menú de opciones que permite al usuario seleccionar diferentes acciones, como atender a un cliente o realizar una operación bancaria.

La herencia se utiliza en los métodos switch de cada opción para llamar a los métodos específicos de cada tipo de empleado. Por ejemplo, en el caso 1 del switch, se llama a los métodos AtenderCliente, RealizarDeposito y RealizarRetiro, que son específicos para la clase Cajero.

En el caso 2 del switch, se llaman a los métodos AbrirCuentaDebito, AbrirCuentaCredito e IniciarFondoInversion, que son específicos para la clase Ejecutivo.

En el caso 4 del switch, se utilizan las propiedades específicas de cada clase para calcular las comisiones totales de los cajeros y los ejecutivos.

Briseño Vázquez Angel Geovany.

Realicé el primer y cuarto programa de la práctica, para el cuarto programa en particular, se tienen dos clases hijas (Pelicula y ObraTeatro) que heredan de una clase padre, y una clase que utiliza ambas clases hijas y se llama CineTeatro.

La herencia permite que las clases hijas hereden los atributos y métodos de la clase padre, y también les permite agregar nuevos atributos y métodos propios. En este caso, la clase Pelicula y la clase ObraTeatro heredan los atributos y métodos de la clase padre Venta, y agregan nuevos atributos específicos de cada uno. Por ejemplo, la clase Pelicula tiene un atributo Sala, que indica en qué sala se proyectará la película.

Además, en el programa se utiliza la herencia para crear un objeto CineTeatro que puede contener tanto películas como obras de teatro. La clase CineTeatro tiene métodos para agregar películas y obras de teatro, y también tiene métodos para vender boletos y mostrar la cartelera.

En resumen, el uso de la herencia en este programa permite crear clases hijas que heredan atributos y métodos de una clase padre, y también permite crear objetos que pueden contener objetos de diferentes clases hijas. En este caso, se utiliza la herencia para modelar un cine-teatro que puede mostrar películas y obras de teatro, y que tiene métodos para vender boletos y mostrar la cartelera.

Para el primer programa es un sistema para el registro y manejo de enfermeras y pacientes en una clínica. La herencia en este programa se refiere a la relación entre las clases Enfermera y Paciente, donde Paciente hereda atributos y métodos de la clase Persona, que es la clase base. Esto permite una estructura jerárquica en la que Paciente es una subclase de Persona y Enfermera es una subclase de Persona y, a su vez, una subclase de Empleado.

El uso de la herencia en este programa permite una mejor organización del código y una mayor modularidad, ya que se pueden definir atributos y métodos comunes en la clase base y luego heredarlos en las subclases. Además, la herencia permite una mayor flexibilidad en el diseño del programa, ya que se pueden agregar nuevas subclases en el futuro sin tener que reescribir todo el código.

En resumen, la herencia es importante en estos programas porque permite una mejor organización y modularidad del código, y también proporciona flexibilidad en el diseño de los programas.

Nishimura Guerrero Christian Jesús.

Se realizó el ejercicio 2 que es un ejemplo de una aplicación de gestión de inventario para una tienda de electrodomésticos que utiliza clases y herencia en su implementación. En el método Main, se crean dos objetos Empleado y se agregan a la lista de empleados. A continuación, se agregan algunos objetos Lavadora y Televisor para simular el inventario de la tienda.

En el método IniciarSesion, se solicita al usuario que ingrese su número de trabajador y, si se encuentra un empleado con ese número, se inicia sesión y se muestra su información. Luego, se llama al método MostrarMenu para mostrar un menú de opciones que permiten al empleado agregar y vender lavadoras y televisores, y mostrar la lista de productos.

La herencia se utiliza en las clases Lavadora y Televisor, que heredan de la clase base Producto. Además, la clase Empleado tiene métodos para agregar y vender lavadoras y televisores, que utilizan métodos de las clases Lavadora y Televisor.

En resumen, este programa muestra cómo se pueden utilizar clases y herencia para implementar una aplicación de gestión de inventario de una tienda de electrodomésticos. La implementación es simple y fácil de entender, y muestra cómo las clases y la herencia pueden mejorar la organización y la reutilización de código.

**CONCLUSIONES POR EQUIPO:**

En primer lugar, la herencia es una de las características fundamentales de la programación orientada a objetos (POO) y nos permite crear relaciones entre clases, donde una clase puede heredar los atributos y métodos de otra clase, y agregar o modificar características específicas. Esto nos permite reutilizar código y mejorar la organización y mantenibilidad de nuestro programa.

La implementación de la herencia en nuestros programas nos ayuda a crear una jerarquía de clases que refleja la relación entre los objetos del mundo real que estamos modelando en nuestro programa. Por ejemplo, en los programas que vimos anteriormente, las clases nos permiten crear una estructura jerárquica que refleja la relación entre los objetos.

Además, la herencia nos permite crear clases más específicas y especializadas a partir de clases más generales y abstractas. Por ejemplo, en los programas anteriores, la clases pueden contener métodos, pero cada tipo de método tiene sus propias características específicas que son manejadas por las clases.

En resumen, la implementación de la herencia en nuestros programas nos permite mejorar la organización y mantenibilidad del código, y crear una jerarquía de clases que refleja las relaciones entre los objetos del mundo real que estamos modelando. Además, nos permite crear clases más específicas y especializadas a partir de clases más generales y abstractas. Por estas razones, la herencia es una herramienta importante en la programación orientada a objetos y debe ser utilizada de manera adecuada y responsable.