2021-1 Programación Orientada a Objetos I

Examen 3 parcial

LIC. EDUARDO CORRALES ITURBE

Descripción breve

Reporte de programa para la exposición del tercer parcial abarcando todas las asignaturas durante el semestre ocupando todos los temas relacionados

Wieneber macias ansurez

111214@alumnouninter.mx

Cuernavaca Morelos 12 November 2020

\indice

RESUMEN

Crear un programa en visual en lenguaje c# para consola tomando en cuenta todos los temas relacionados con lo que se vio en los temas durante todo el parcial y realizar una exposición no mas de 10 minutos con todos los temas vistos durante el semestre, a continuación, mostraremos en procedimiento del programa y sus funciones correspondientes.

MARCO TEORICO

C# definición

Es pronunciado como “C sharp” es un lenguaje de programación diseñado por Microsoft fue estandarizado en hace un tiempo por ECM e ISO.

Programa orientado a objetos es una rama de la informática que usa como su propio nombre indica los objetos y las interacciones de estos para diseñar aplicaciones y programas informáticos sobre su antecesor el c++.

DESARROLLO DEL TRABAJO

Dentro del main no tenemos mas que un objeto llamando a la clase principal

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;

namespace herencia29102020
{
    0 referencias
    class principal
    {
        0 referencias
        static void Main(string[] args)
        {
            menu acceso = new menu();

            acceso.validacion();
        }
    }
}
```

Clases y herencias

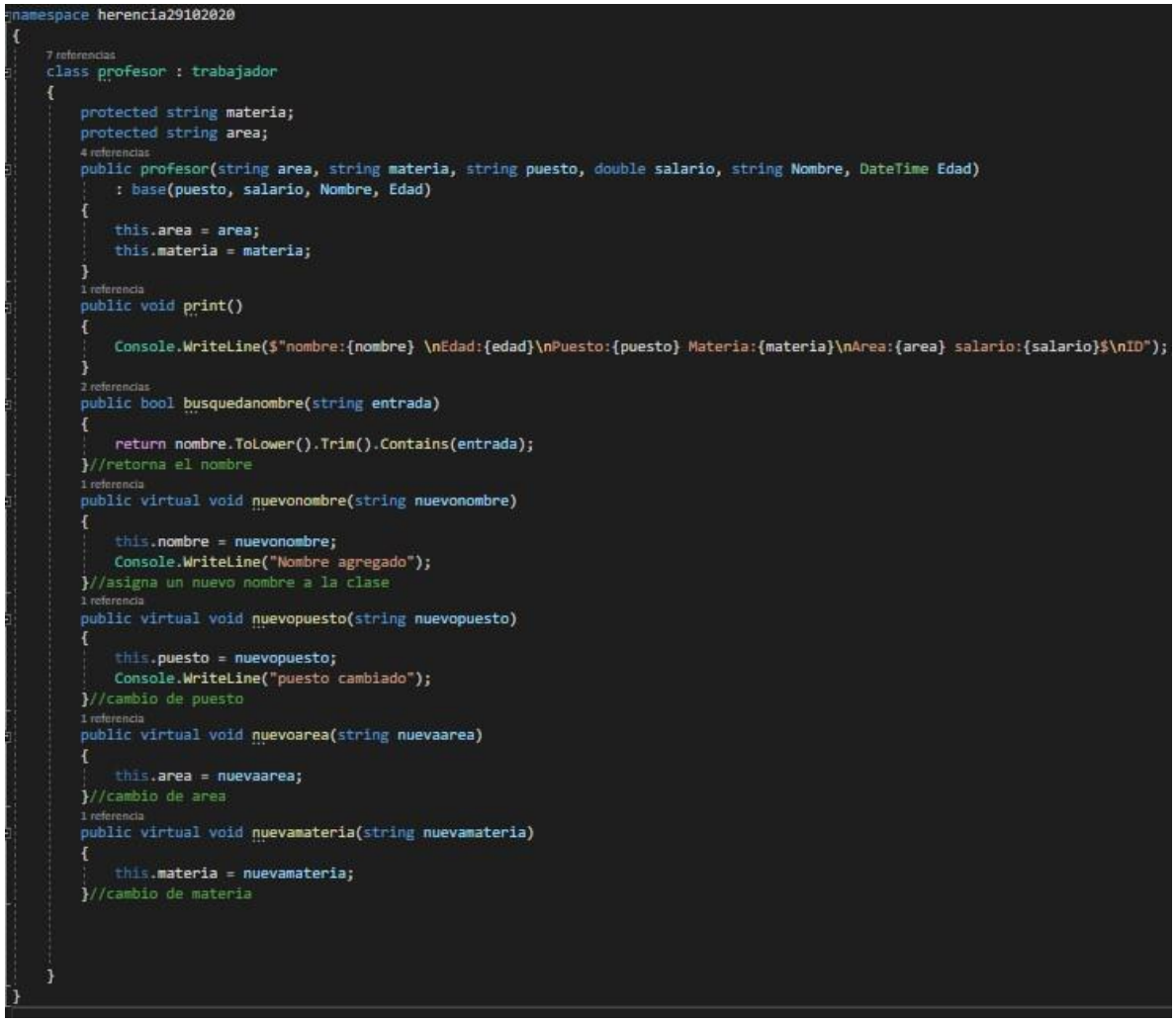
Clase padre

```
namespace herencia29102020
{
    3 referencias
    class persona
    {
        protected string nombre;
        protected int edad;
        protected DateTime fecha;
        2 referencias
        public persona(string Nombre, DateTime Edad)
        {
            nombre = Nombre;
            edad = salida(Edad);
            this.fecha = Edad;
        }//constructor
        1 referencia
        int salida(DateTime x)
        {
            int edadanios = ((TimeSpan)(DateTime.Now - x)).Days;
            return edadanios / 365;
        }//insertar edad en anios
        0 referencias
        public void salidapersona()
        {
            Console.WriteLine($"\nNombre:[{nombre}]  Edad:[{edad}]");
        }//imprimir nombre
    }

}
```

Clase hijos y nietos

```
namespace herencia29102020
{

    2 referencias
    class trabajador : persona
    {
        protected DateTime fechainicio = DateTime.Now;
        protected double salario;
        protected string puesto;
        protected string id;
        1 referencia
        public trabajador(string puesto, double salario, string Nombre, DateTime Edad) : base(Nombre, Edad)
        {
            this.salario = salario;
            this.puesto = puesto;
            this.id = idjex();

        }//constructor
        1 referencia
        public void imprimirid()
        {
            Console.WriteLine($"[{id}]");
        }//imprime credencial de trabajador
        1 referencia
        private static string idjex()
        {
            Random rdm = new Random();
            string hexValue = string.Empty;
            int num;

            for (int i = 0; i < new Random().Next(1, 4); i++)
            {
                num = rdm.Next(0, int.MaxValue);
                hexValue += num.ToString("X8");
            }

            return hexValue;
        }//general el id de la credencial del trabajador
    }
}
```

```
namespace herencia29102020
{
    7 referencias
    class estudiate : persona
    {
        3 referencias
        protected int id { get; set; }
        3 referencias
        protected string carrera { get; set; }
        3 referencias
        protected int semestre { get; set; }
        4 referencias
        public estudiate(string Nombre, DateTime Edad, string carrera, int semestre) : base(Nombre, Edad)
        {
            this.id = generarid(id);
            this.carrera = carrera;
            this.semestre = semestre;
        }
        1 referencia
        private int generarid(int id)
        {
            var seed = Environment.TickCount;
            var Random = new Random(seed);
            return id = Random.Next(0, 500);
        }//generacion de credencial
        1 referencia
        public void print()
        {
            Console.WriteLine($"nombre:[{nombre}] \nEdad:[{edad}] carrera:[{carrera}]\nsemestre:[{semestre}]\n ID:[{id + 1}]");
        }//imprimir estudiante
        2 referencias
        public bool busquedanombre(string entrada)
        {
            return nombre.ToLower().Trim().Contains(entrada);
        }//retorna el nombre
        1 referencia
        public virtual void nuevonombre(string nuevonombre)
        {
            this.nombre = nuevonombre;
            Console.WriteLine("Nombre agregado");
        }//asigna un nuevo nombre a la clase
        1 referencia
        public virtual void nuevosemestre(int nuevosemestre)
        {
            this.semestre = nuevosemestre;
            Console.WriteLine("Semestre agregado");
        }//asigna un nuevo semestre a la clase
        1 referencia
        public virtual void nuevacarrera(string nuevacarrera)
        {
            this.carrera = nuevacarrera; Console.WriteLine("carrera agregada");
        }//asinga una nueva carrera a la clase

    }
}
```



```
namespace herencia29102020
{
    7 referencias
    class profesor : trabajador
    {
        protected string materia;
        protected string area;
        4 referencias
        public profesor(string area, string materia, string puesto, double salario, string Nombre, DateTime Edad)
            : base(puesto, salario, Nombre, Edad)
        {
            this.area = area;
            this.materia = materia;
        }
        1 referencia
        public void print()
        {
            Console.WriteLine($"nombre:{nombre} \nEdad:{edad}\nPuesto:{puesto} Materia:{materia}\nArea:{area} salario:{salario}$\nID");
        }
        2 referencias
        public bool busquedanombre(string entrada)
        {
            return nombre.ToLower().Trim().Contains(entrada);
        }//retorna el nombre
        1 referencia
        public virtual void nuevonombre(string nuevonombre)
        {
            this.nombre = nuevonombre;
            Console.WriteLine("Nombre agregado");
        }//asigna un nuevo nombre a la clase
        1 referencia
        public virtual void nuevopuesto(string nuevopuesto)
        {
            this.puesto = nuevopuesto;
            Console.WriteLine("puesto cambiado");
        }//cambio de puesto
        1 referencia
        public virtual void nuevoarea(string nuevaarea)
        {
            this.area = nuevaarea;
        }//cambio de area
        1 referencia
        public virtual void nuevamateria(string nuevamateria)
        {
            this.materia = nuevamateria;
        }//cambio de materia



    }
}
```

Clase principal donde se hacen todas las condiciones

```
namespace herencia29102020
{
    8 referencias
    public enum edicion { nombre =1, puesto, area, materia, salir }
    2 referencias
    class menu
    {
        private List<estudiate> alumno = new List<estudiate>();
        private List<profesor> _profesor = new List<profesor>();
        private string _nombre;
        private DateTime _edad;
        private string _carrera;
        private int _semestre;
        private string[] administrador = {
            "wieneber"
        };
        private string[] pass = {
            "123"
        };
        1 referencia
        public void validacion()
        {
            int intentos = 3;
            do
            {
                Console.WriteLine("////REGISTRO////\n");
                Console.WriteLine("USER");
                string user1 = Console.ReadLine();
                Console.WriteLine("password");
                string pass1 = Console.ReadLine();
                if (user1.ToLower().Trim() == administrador[0] && pass1.ToLower().Trim() == pass[0])
                {
                    Console.WriteLine($"hola [{user1}]");
                    registro();
                    Console.ReadLine();
                    principal();
                }
                else { intentos--; Console.WriteLine($"Error De Acceso Intentos[{intentos}]"); if (intentos == 0) { Environment.Exit(1); } }

            } while (true);
        }//acceso al sistema por administrador
        14 referencias
        void principal()//menu principal
        {
            Console.Clear();
            Console.WriteLine("1.-NUEVO REGISTRO\n2.-VER REGISTRO\n3.-BUSCAR REGISTRO\n4.-SALIR");
            int op = Convert.ToInt32(Console.ReadLine());
            switch (op)
            {
                case 1:
                    newregistro(true);
                    break;
                case 2:
                    newregistro(false);
                    break;
                case 3:
                    modificacion();
                    break;
                case 4:
                    Environment.Exit(1);
                    break;
                default:
                    Console.WriteLine("NO VALIDADO"); Console.ReadKey();
                    principal();
                    break;
            }
        }
        4 referencias
```

```
void newregistro(bool p)// menu de regritro de usuario dependiendo la persona
{
    Console.Clear();
    Console.WriteLine("REGISTRO?\n");
    Console.WriteLine("1.-ESTUDIANTE\n2.-PROFESOR");
    int op = Convert.ToInt32(Console.ReadLine());
    switch (op)
    {
        case 1:
            estudiante(p);
            break;
        case 2:
            profesor(p);
            break;
        default:
            Console.WriteLine("NO VALIDADO"); Console.ReadKey();
            newregistro(p);
            break;
    }
}
2 referencias
void modificacion()//modificacion de los usuarios ya sea editar o eliminar usuarios
{
    Console.Clear();
    Console.WriteLine("1.-EDITAR\n2.-ELIMINIAR");
    int op = Convert.ToInt32(Console.ReadLine());
    switch (op)
    {
        case 1:
            modificacionregistro(true);
            break;
        case 2:
            modificacionregistro(false);
            break;
        default:
            Console.WriteLine("NO VALIDADO"); Console.ReadKey();
            modificacion();
            break;
    }
}
2 referencias
void modificacionregistro(bool p)//seleccion de resgistro a modificar ya sea estudiante o profesor
{
    Console.Clear();
    Console.WriteLine("REGISTRO?\n");
    Console.WriteLine("1.-ESTUDIANTE\n2.-PROFESOR");
    int op = Convert.ToInt32(Console.ReadLine());
    switch (op)
    {
        case 1:
            buscarestudiante(p);
            break;
        case 2:
            buscarprofesor(p);
            break;
        default:
            Console.WriteLine("NO VALIDADO"); Console.ReadKey();
            newregistro(p);
            break;
    }
}
2 referencias
```

```
void buscarestudiante(bool p)//modificacion y eliminacion de estudiante
{
    if (p == true)
    {
        Console.WriteLine("Nombre del estudiante");
        string comparacion = Console.ReadLine();
        foreach (var item in alumno)
        {
            if (item.busquedanombre(comparacion.ToLower().Trim()))
            {
                int op = condicionestudiante();
                switch (op)
                {
                    case 1:
                        Console.WriteLine("Nuevo Nombre");
                        item.nuevonombre(Console.ReadLine());
                        principal();
                        break;
                    case 2:
                        Console.WriteLine("Nueva Semestre");
                        item.nuevosemestre(Convert.ToInt32(Console.ReadLine()));
                        break;
                    case 3:
                        Console.WriteLine("Nueva Carrera");
                        item.nuevacarrera(Console.ReadLine());
                        break;
                    case 4:
                        principal();
                        break;
                    default:
                        Console.WriteLine("NO VALIDADO"); Console.ReadKey();
                        buscarestudiante(p);
                        break;
                }
            }
        }
    }
    else
    {
        Console.WriteLine("Nombre del estudiante");
        string comparacion = Console.ReadLine();
        for (int i = 0; i < alumno.Count; i++)
        {
            if (alumno[i].busquedanombre(comparacion.ToLower().Trim()))
            {
                alumno.RemoveAt(i);
                i--;
                Console.WriteLine("Estudiante Eliminado"); Console.ReadKey();
            }
        }
        principal();

    }
}
```

```
void buscarprofesor(bool p)//modificacion y eliminacion de profesor
{
    if (p == true)
    {
        Console.WriteLine("Nombre del Profesor");
        string comparacion = Console.ReadLine();
        foreach (var item in _profesor)
        {
            if (item.busquedanombre(comparacion.ToLower().Trim()) == true)
            {
                int op = condicionprofesor();
                switch (op)
                {
                    case 1:
                        Console.WriteLine("Nuevo Nombre");
                        item.nuevonombre(Console.ReadLine());
                        principal();
                        break;
                    case 2:
                        Console.WriteLine("Nuevo Puesto");
                        item.nuevopuesto(Convert.ToString(Console.ReadLine()));
                        break;
                    case 3:
                        Console.WriteLine("Nueva Area");
                        item.nuevoarea(Console.ReadLine());
                        break;
                    case 4:
                        Console.WriteLine("Nueva Materia");
                        item.nuevamateria(Console.ReadLine());
                        break;
                    case 5:
                        principal();
                        break;
                    default:
                        Console.WriteLine("NO VALIDADO"); Console.ReadKey();
                        principal();
                        break;
                }
            }
        }
    }
    else
    {
        Console.WriteLine("Nombre del Profesor");
        string comparacion = Console.ReadLine();
        for (int i = 0; i < _profesor.Count; i++)
        {
            if (_profesor[i].busquedanombre(comparacion.ToLower().Trim()))
            {
                _profesor.RemoveAt(i);
                i--;
                Console.WriteLine("Pofesor Eliminado"); Console.ReadKey();
            }
        }
        principal();

    }
}
1 referencia
```

```
int condicionestudiante()
{
    Console.Clear();
    Console.WriteLine("\nEDICION A?");
    Console.WriteLine("1.-Nombre\n2.-Carrera\n3.-Semestre\n4.-Salir");
    return Convert.ToInt32(Console.ReadLine());
}//condicion de edicion por metodo de enum retornando un numero entero
1 referencia
int condicionprofesor()//metodo de edicion de la lista
{
    int i = 1;
    Console.Clear();
    Console.WriteLine("\nEDICION A?");
    foreach (var item in Enum.GetValues(typeof(edicion)))
    {
        Console.WriteLine($"[{i++}]{item}");
    }
    edicion op = (edicion)(Convert.ToInt32(Console.ReadLine()));
    switch (op)
    {
        case edicion.nombre:
            return 1;
            break;
        case edicion.puesto:
            return 2;
            break;
        case edicion.area:
            return 3;
            break;
        case edicion.materia:
            return 4;
            break;
        case edicion.salir:
            return 5;
            break;
        default:
            principal();
            break;
    }
    return 0;

}
1 referencia
void estudiante(bool p)//insercion y visualizacion de lista metodo .add de estudiante
{
    if (p == true)
    {

        Console.WriteLine("Cantidad de Alumnos");
        int integrantes = Convert.ToInt32(Console.ReadLine());
        for (int i = 1; i < integrantes + 1; i++)
        {
            Console.WriteLine("Nombre");
            _nombre = Console.ReadLine();
            Console.WriteLine("Fecha de Nacimiento");
            _edad = Convert.ToDateTime(Console.ReadLine());
            Console.WriteLine("Carrera");
            _carrera = Console.ReadLine();
            Console.WriteLine("Semestre actual");
            _semestre = Convert.ToInt32(Console.ReadLine());
            alumno.Add(new estudiate(Nombre: _nombre, Edad: _edad, carrera: _carrera, semestre: _semestre));
        }
        Console.WriteLine("MENU PRINCIPAL?\n1.-Si\n2.-No");
        int salir = Convert.ToInt32(Console.ReadLine());
        if (salir == 1) { principal(); } else { Environment.Exit(1); }
        Console.ReadKey();
    }
    else
    {
        foreach (var item in alumno)
        {
            Console.WriteLine("-----------");
            item.print();
        }
        Console.ReadKey();
        principal();
    }
}
1 referencia
```

```
        void profesor(bool p)//insercion y visualizacion de lista por metodo .add de profesor
        {
            if (p == true)
            {
                Console.WriteLine("Cantidad de profesores");
                int integrantes = Convert.ToInt32(Console.ReadLine());
                for (int i = 1; i < integrantes + 1; i++)
                {
                    Console.WriteLine("Nombre");
                    _nombre = Console.ReadLine();
                    Console.WriteLine("Fecha de Nacimiento");
                    _edad = Convert.ToDateTime(Console.ReadLine());
                    Console.WriteLine("Materia");
                    string Materia = Console.ReadLine();
                    Console.WriteLine("Area actual");
                    string Area = Console.ReadLine();
                    Console.WriteLine("Salario Actual");
                    double Salario = Convert.ToDouble(Console.ReadLine());
                    _profesor.Add(new profesor(area: Area, materia: Materia, puesto: "Profesor", salario: Salario, Nombre: _nombre, Edad: _edad));
                }
                Console.WriteLine("MENU PRINCIPAL?\n1.-SI\n2.-NO");
                int salir = Convert.ToInt32(Console.ReadLine());
                if (salir == 1) { principal(); } else { Environment.Exit(1); }
                Console.ReadKey();
            }
            else
            {
                foreach (var item in _profesor)
                {
                    Console.WriteLine("-----------");
                    item.print();
                    item.imprimirid();
                }
                Console.ReadKey();
                principal();
            }
        }
        1 referencia
        void registro()
        {
            _profesor.Add(new profesor(area: "ESCAT", materia: "MATEMATICAS", puesto: "Profesor", salario: 8000, Nombre: "Alan Brito Delgado", Edad: new DateTime(1990, 5, 1)));
            _profesor.Add(new profesor(area: "ESCAT", materia: "FISICA", puesto: "Profesor", salario: 8500.15, Nombre: "Jose Boquita de La Corona", Edad: new DateTime(1995, 9, 1)));
            _profesor.Add(new profesor(area: "ESCAT", materia: "PROGRAMACION", puesto: "Profesor", salario: 5200.10, Nombre: "Dolorez De Cueva", Edad: new DateTime(1985, 4, 1)));
            alumno.Add(new estudiate(Nombre: "ZoYla sorda", Edad: new DateTime(1995, 4, 1), carrera: "SISTEMAS", semestre: 5));
            alumno.Add(new estudiate(Nombre: "Alex Plosivo", Edad: new DateTime(199, 8, 2), carrera: "ADMINISTRACION", semestre: 8));
            alumno.Add(new estudiate(Nombre: "Zoyla Vaca", Edad: new DateTime(1995, 5, 9), carrera: "MECATRONICA", semestre: 7));
        }//registro de ususarios predeterminados
    }
}
```

Orden de las clases

```
namespace herencia29102020
{
    8 referencias
    public enum edicion { nombre =1, puesto, area, materia, salir }
    2 referencias
    class menu
    {
        private List<estudiate> alumno = new List<estudiate>();
        private List<profesor> _profesor = new List<profesor>();
        private string _nombre;
        private DateTime _edad;
        private string _carrera;
        private int _semestre;
        private string[] administrador = {
            "wieneber"
        };
        private string[] pass = {
            "123"
        };
        1 referencia
        public void validacion()...//acceso al sistema por administrador
        14 referencias
        void principal()//menu principal ...
        4 referencias
        void newregistro(bool p)// menu de regritro de usuario dependiendo la persona...
        2 referencias
        void modificacion()//modificacion de los usuarios ya sea editar o eliminar usuarios...
        2 referencias
        void modificacionregistro(bool p)//seleccion de resgistro a modificar ya sea estudiante o profesor...
        2 referencias
        void buscarestudiante(bool p)//modificacion y eliminacion de estudiante...
        1 referencia
        void buscarprofesor(bool p)//modificacion y eliminacion de profesor...
        1 referencia
        int condicionestudiante()...//condicion de edicion por metodo de enum retornando un numero entero
        1 referencia
        int condicionprofesor()//metodo de edicion de la lista...
        1 referencia
        void estudiante(bool p)//insercion y visualizacion de lista metodo .add de estudiante...
        1 referencia
        void profesor(bool p)//insercion y visualizacion de lista por metodo .add de profesor...
        1 referencia
        void registro()...//registro de ususarios predetermindados
    }
}
```

Clases `abstract`

```
1 referencia
abstract class hola
{

    2 referencias
    public abstract string saludo();

}
3 referencias
class accion:hola
{
    protected string hola;
    1 referencia
    public accion(string hola)
    {
        this.hola = hola;
    }
    2 referencias
    public override string saludo()
    {
        return $"hola {hola} desde abstract";
    }
}
```

```
public void saludo()
{
    accion hola = new accion(nombre);
    Console.WriteLine($"{ hola.saludo()}");
}
```

```
foreach (var item in _profesor)
{
    item.saludo();
    Console.WriteLine("------------");
    item.print();
    item.imprimirid();
}
Console.ReadKey();
principal();
```