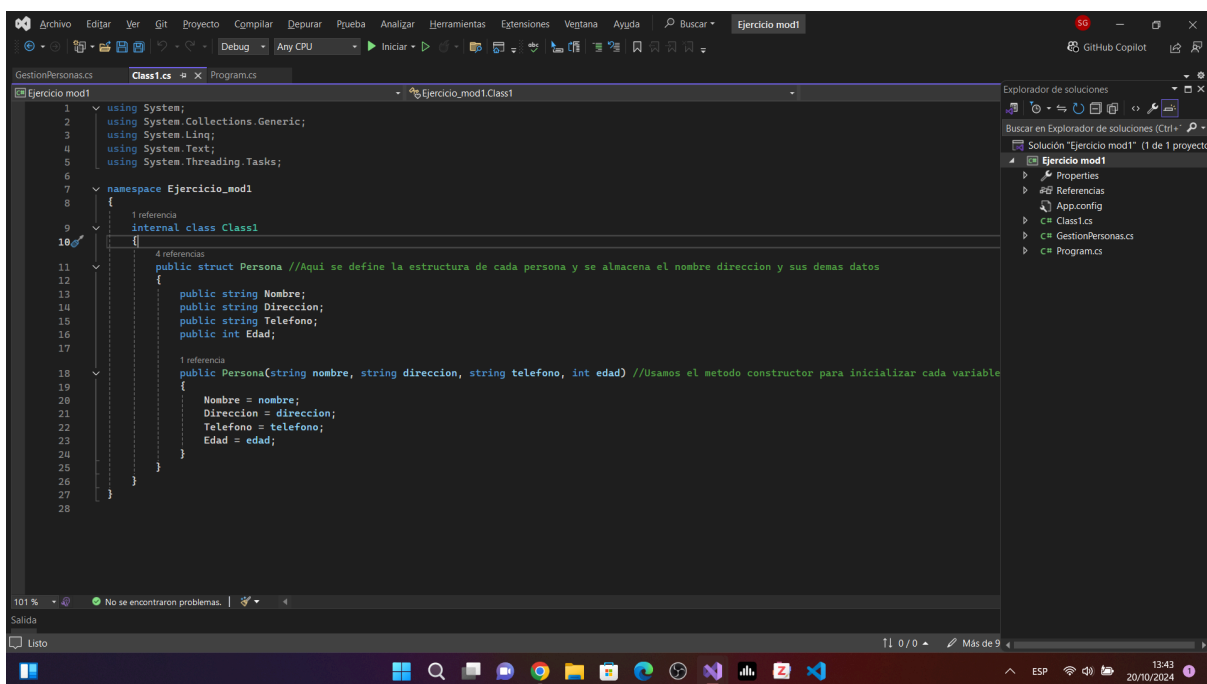


## Ejercicio número 1. Silvio Mejia

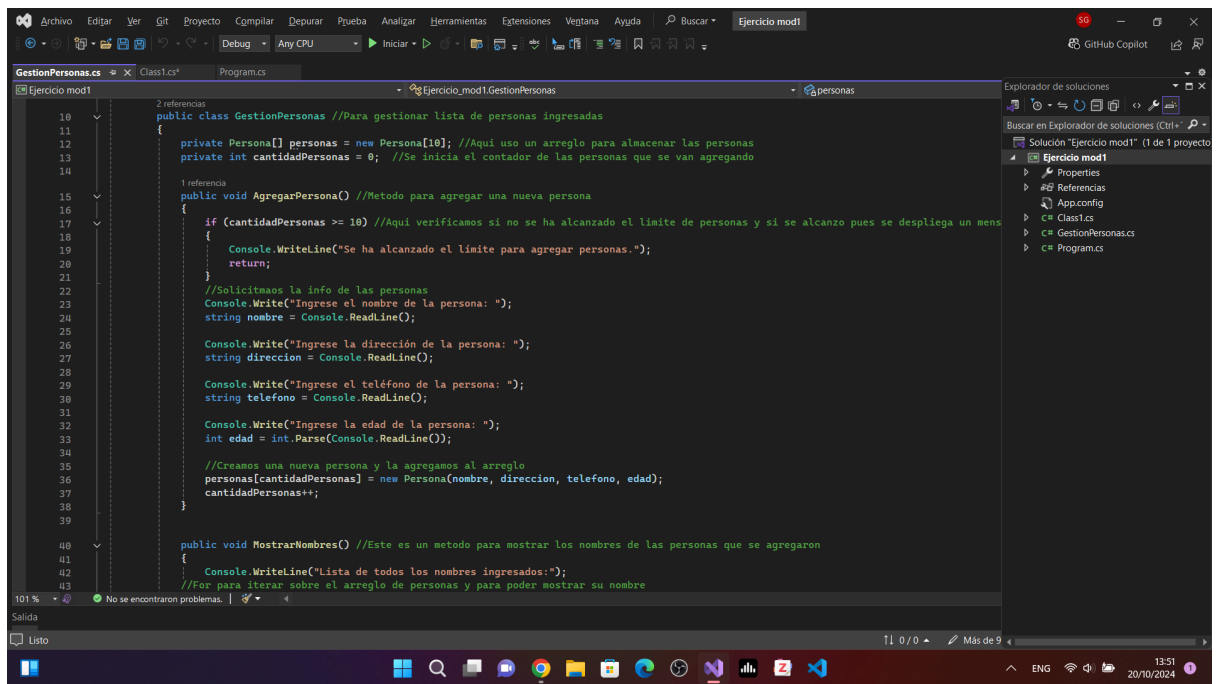
Programa que sea capaz de almacenar los datos de 10 personas: nombre, dirección, teléfono, edad (usando structs). Deberá ir pidiendo los datos uno por uno, hasta que el usuario lo decida. Entonces deberá aparecer un menú que permita:

- Mostrar la lista de todos los nombres.
- Mostrar las personas de una cierta edad.
- Mostrar las personas que coincidan con un nombre.
- Salir del programa.

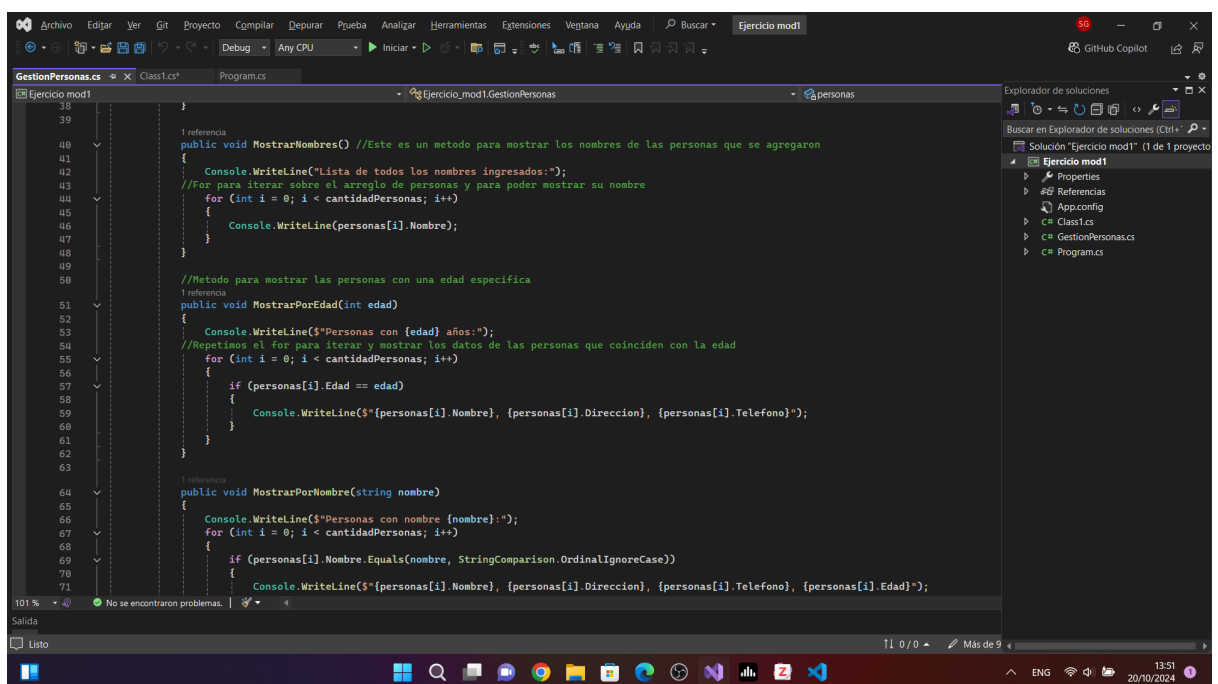
- 1) En esta clase se define el struct llamado “Persona” aquí se almacenarán cuatro variables, el nombre, dirección, teléfono y la edad de cada persona que se ingrese. Incluimos el constructor para inicializar cada variable cuando.



- 2) La clase “GestionPersonas” es la encargada de gestionar las operaciones relacionadas con cada persona, aquí se crea un arreglo para 10 personas y utilizamos un contador para controlar cuántas personas se agregan. Utilizamos algunas funciones como `AgregarPersona()` que permite que el usuario agregue una nueva persona, esto gracias a que solicitamos al usuario que proporcione la info de cada persona. En esta misma función se verifica el número de personas agregadas para que no se supere el límite y si se superó pues imprimimos un mensaje para que no se agreguen más.
- 3) Tenemos otra función llamada `MostrarNombres()` `MostrarPorEdad()` `MostrarPorNombre()` que recorren el arreglo de las personas para mostrar la información que el usuario desee.



```
10 2 referencias
11 public class GestionPersonas //Para gestionar lista de personas ingresadas
12 {
13     private Persona[] personas = new Persona[10]; //Aquí uso un arreglo para almacenar las personas
14     private int cantidadPersonas = 0; //Se inicia el contador de las personas que se van agregando
15
16     1 referencia
17     public void AgregarPersona() //Metodo para agregar una nueva persona
18     {
19         if (cantidadPersonas >= 10) //Aquí verificamos si no se ha alcanzado el límite de personas y si se alcanza pues se despliega un mensaje
20         {
21             Console.WriteLine("Se ha alcanzado el límite para agregar personas.");
22             return;
23         }
24         //Solicitamos la info de las personas
25         Console.WriteLine("Ingrese el nombre de la persona: ");
26         string nombre = Console.ReadLine();
27
28         Console.WriteLine("Ingrese la dirección de la persona: ");
29         string direccion = Console.ReadLine();
30
31         Console.WriteLine("Ingrese el teléfono de la persona: ");
32         string telefono = Console.ReadLine();
33
34         Console.WriteLine("Ingrese la edad de la persona: ");
35         int edad = int.Parse(Console.ReadLine());
36
37         //Creamos una nueva persona y la agregamos al arreglo
38         personas[cantidadPersonas] = new Persona(nombre, direccion, telefono, edad);
39         cantidadPersonas++;
40
41     }
42
43     public void MostrarNombres() //Este es un metodo para mostrar los nombres de las personas que se agregaron
44     {
45         Console.WriteLine("Lista de todos los nombres ingresados:");
46         //For para iterar sobre el arreglo de personas y para poder mostrar su nombre
47     }
48 }
49
50 Salida
51 Listo
```



```
38 }
39
40 1 referencia
41 public void MostrarNombres() //Este es un metodo para mostrar los nombres de las personas que se agregaron
42 {
43     Console.WriteLine("Lista de todos los nombres ingresados:");
44     //For para iterar sobre el arreglo de personas y para poder mostrar su nombre
45     for (int i = 0; i < cantidadPersonas; i++)
46     {
47         Console.WriteLine(personas[i].Nombre);
48     }
49 }
50
51 //Metodo para mostrar las personas con una edad especifica
52 1 referencia
53 public void MostrarPorEdad(int edad)
54 {
55     Console.WriteLine($"Personas con {edad} años:");
56     //Repetimos el for para iterar y mostrar los datos de las personas que coinciden con la edad
57     for (int i = 0; i < cantidadPersonas; i++)
58     {
59         if (personas[i].Edad == edad)
60         {
61             Console.WriteLine($"{personas[i].Nombre}, {personas[i].Direccion}, {personas[i].Telefono}");
62         }
63     }
64 }
65
66 1 referencia
67 public void MostrarPorNombre(string nombre)
68 {
69     Console.WriteLine($"Personas con nombre {nombre}:");
70     for (int i = 0; i < cantidadPersonas; i++)
71     {
72         if (personas[i].Nombre.Equals(nombre, StringComparison.OrdinalIgnoreCase))
73         {
74             Console.WriteLine($"{personas[i].Nombre}, {personas[i].Direccion}, {personas[i].Telefono}, {personas[i].Edad}");
75         }
76     }
77 }
78
79 Salida
80 Listo
```

3) Finalmente tenemos la clase main aquí es donde se ejecutará el ciclo del programa, primero mostraremos el menú al usuario donde este decida que acción tomar esto lo realizamos mediante un ciclo do - while que mantiene el programa hasta que el usuario digite la opción 5 que es salir, creamos una instancia de la clase GestionPersonas, porque esta es la encargada de manejar el almacenamiento y registro de cada persona, dependiendo de lo que el usuario desee el programa llamara a las funciones anteriormente mencionadas.

```
9 internal class Program
10 {
11     0 referencias
12     static void Main(string[] args)
13     {
14
15         GestionPersonas gestion = new GestionPersonas();
16         int opcion;
17
18         do
19         {
20             // =====
21             Console.WriteLine("
22             Console.WriteLine("
23             Console.WriteLine("
24             Console.WriteLine("
25             Console.WriteLine("
26             Console.WriteLine("
27             Console.WriteLine("
28             Console.WriteLine("
29             Console.WriteLine("Selecciona la opción a seguir: ");
30             opcion = int.Parse(Console.ReadLine());
31
32             switch (opcion)
33             {
34                 case 1:
35                     gestion.AgregarPersona();
36                     break;
37                 case 2:
38                     gestion.MostrarNombres();
39                     break;
40                 case 3:
41                     Console.WriteLine("Ingrese la edad que desea buscar: ");
42                     int edad = int.Parse(Console.ReadLine());
43                     gestion.MostrarPorEdad(edad);
44                     break;
45             }
46         } while (opcion != 5);
47     }
48 }
```

```
26 Console.WriteLine("
27 Console.WriteLine("
28 Console.WriteLine("
29 Console.WriteLine("Selecciona la opción a seguir: ");
30 opcion = int.Parse(Console.ReadLine());
31
32 switch (opcion)
33 {
34     case 1:
35         gestion.AgregarPersona();
36         break;
37     case 2:
38         gestion.MostrarNombres();
39         break;
40     case 3:
41         Console.WriteLine("Ingrese la edad que desea buscar: ");
42         int edad = int.Parse(Console.ReadLine());
43         gestion.MostrarPorEdad(edad);
44         break;
45     case 4:
46         Console.WriteLine("Ingrese el nombre que desea buscar: ");
47         string nombre = Console.ReadLine();
48         gestion.MostrarPorNombre(nombre);
49         break;
50     case 5:
51         Console.WriteLine("Saliendo del programa...");
52         break;
53     default:
54         Console.WriteLine("Opción no válida.");
55         break;
56 }
57 } while (opcion != 5);
58 }
59 }
60 }
```

### Programa número 3. Silvio Mejia

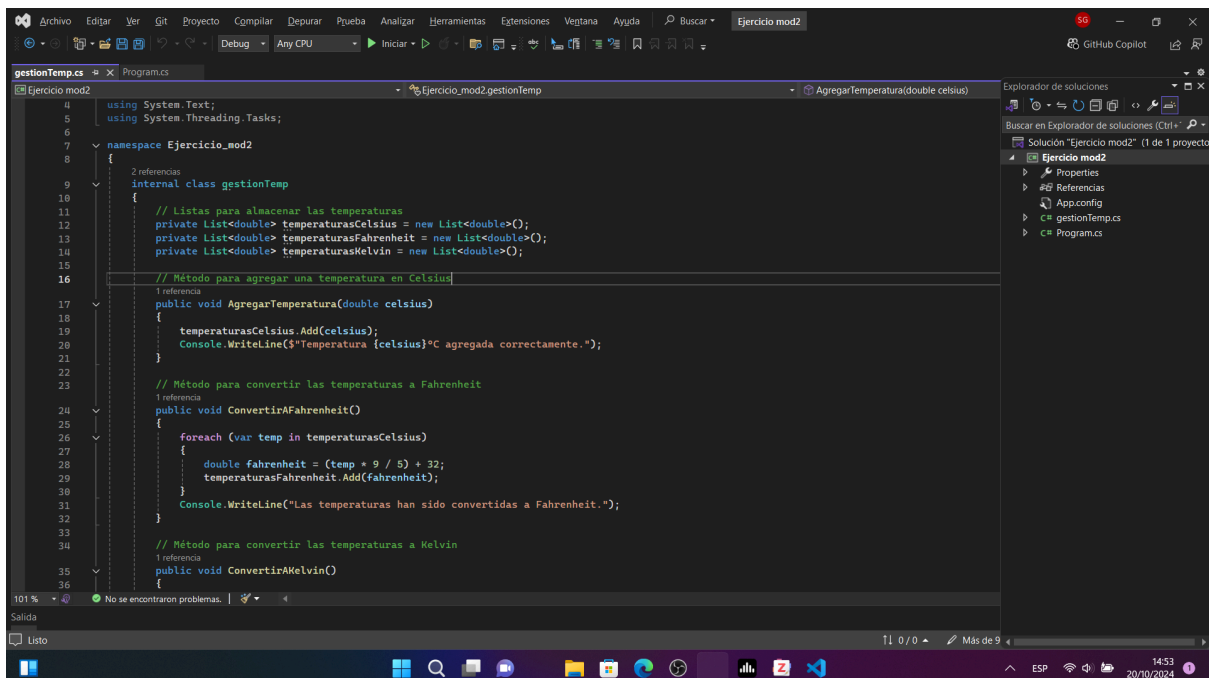
Crea un programa que use una lista genérica para almacenar temperaturas en grados Celsius. Implementa funciones para convertirlas a Fahrenheit y Kelvin, y mostrar las temperaturas convertidas, las cuales se almacenan en nuevas listas respectivamente. - - - El programa se repetirá las veces que el usuario lo decida. Utilizar funciones con parámetros. Agregar una función que permita eliminar de la lista que contiene las temperaturas convertidas. La que el usuario elija

- 1) Clase gestionTemp, esta clase tiene la tarea de gestionar las temperaturas ingresadas en grados celsius y convertirlas a otras unidades. Tambien permite eliminar las temperaturas convertidas

A esta clase se le gregaron listas que almacenan las temperaturas ingresadas por el usuario en grados celsius, las temperaturas convertidas a fahrenheit y kelvin

La clase tiene funciones como agregarTemperatura que permite agregar temp en grados celsius, tambien tiene la funcion convertir a fahrenheit y convertir a kelvin.

Tiene otras funciones como mostrar temperaturas convertidas y eliminar temperaturas convertidas.



```
using System.Text;
using System.Threading.Tasks;

namespace Ejercicio_mod2
{
    internal class gestionTemp
    {
        // Listas para almacenar las temperaturas
        private List<double> temperaturasCelsius = new List<double>();
        private List<double> temperaturasFahrenheit = new List<double>();
        private List<double> temperaturasKelvin = new List<double>();

        // Método para agregar una temperatura en Celsius
        public void AgregarTemperatura(double celsius)
        {
            temperaturasCelsius.Add(celsius);
            Console.WriteLine($"Temperatura {celsius}°C agregada correctamente.");
        }

        // Método para convertir las temperaturas a Fahrenheit
        public void ConvertirAFahrenheit()
        {
            foreach (var temp in temperaturasCelsius)
            {
                double fahrenheit = (temp * 9 / 5) + 32;
                temperaturasFahrenheit.Add(fahrenheit);
            }
            Console.WriteLine("Las temperaturas han sido convertidas a Fahrenheit.");
        }

        // Método para convertir las temperaturas a Kelvin
        public void ConvertirAKelvin()
        {
        }
    }
}
```

This screenshot shows the Visual Studio IDE with the file 'gestionTemp.cs' open. The code defines a class 'Ejercicio\_mod2.gestionTemp' with three methods: 'Convertir a Kelvin', 'Mostrar Temperaturas Convertidas', and 'Eliminar Temperatura Convertida'. The 'Convertir a Kelvin' method iterates over 'temperaturasCelsius' and adds 273.15 to each value to convert to Kelvin. The 'Mostrar Temperaturas Convertidas' method prints the converted values in both Fahrenheit and Kelvin. The 'Eliminar Temperatura Convertida' method is partially visible at the bottom.

```
34 // metodo para convertir las temperaturas a Kelvin
35 1 referencia
36 public void ConvertirAKelvin()
37 {
38     foreach (var temp in temperaturasCelsius)
39     {
40         double kelvin = temp + 273.15;
41         temperaturasKelvin.Add(kelvin);
42     }
43     Console.WriteLine("Las temperaturas han sido convertidas a Kelvin.");
44 }
45
46 // Método para mostrar las temperaturas convertidas
47 public void MostrarTemperaturasConvertidas()
48 {
49     Console.WriteLine("Temperaturas en Fahrenheit:");
50     foreach (var temp in temperaturasFahrenheit)
51     {
52         Console.WriteLine($"{temp}°F");
53     }
54     Console.WriteLine("Temperaturas en Kelvin:");
55     foreach (var temp in temperaturasKelvin)
56     {
57         Console.WriteLine($"{temp}°K");
58     }
59 }
60
61 // Método para eliminar una temperatura de la lista de Fahrenheit o Kelvin
62 1 referencia
63 public void EliminarTemperaturaConvertida(string tipo, double valor)
64 {
65     if (tipo.ToLower() == "fahrenheit")
66     {
67         if (temperaturasFahrenheit.Remove(valor))
68         {
69             Console.WriteLine($"{temperatura {valor}°F eliminada de la lista.");
70         }
71         else
72         {
73             Console.WriteLine($"{temperatura {valor}°F no encontrada.");
74         }
75     }
76     else if (tipo.ToLower() == "kelvin")
77     {
78         if (temperaturasKelvin.Remove(valor))
79         {
80             Console.WriteLine($"{temperatura {valor}°K eliminada de la lista.");
81         }
82         else
83         {
84             Console.WriteLine($"{temperatura {valor}°K no encontrada.");
85         }
86     }
87     else
88     {
89         Console.WriteLine("Tipo de temperatura no válido. Use 'fahrenheit' o 'kelvin'.");
90     }
91 }
92
93 }
94
```

This screenshot shows the same Visual Studio IDE, but with the 'Eliminar Temperatura Convertida' method fully visible. The method takes a string 'tipo' and a double 'valor' as input. It checks if the temperature is in Fahrenheit or Kelvin and attempts to remove it from the respective list. If successful, it prints a confirmation message; otherwise, it prints a message indicating the temperature was not found. If the type is invalid, it prompts the user to use 'fahrenheit' or 'kelvin'.

```
62 public void EliminarTemperaturaConvertida(string tipo, double valor)
63 {
64     if (tipo.ToLower() == "fahrenheit")
65     {
66         if (temperaturasFahrenheit.Remove(valor))
67         {
68             Console.WriteLine($"{temperatura {valor}°F eliminada de la lista.");
69         }
70         else
71         {
72             Console.WriteLine($"{temperatura {valor}°F no encontrada.");
73         }
74     }
75     else if (tipo.ToLower() == "kelvin")
76     {
77         if (temperaturasKelvin.Remove(valor))
78         {
79             Console.WriteLine($"{temperatura {valor}°K eliminada de la lista.");
80         }
81         else
82         {
83             Console.WriteLine($"{temperatura {valor}°K no encontrada.");
84         }
85     }
86     else
87     {
88         Console.WriteLine("Tipo de temperatura no válido. Use 'fahrenheit' o 'kelvin'.");
89     }
90 }
91
92 }
93
94
```

- 2) En la clase main se crea la instancia de la clase gestionTemp y se crea el objeto gestión para poder acceder a las funciones de la clase. Repetimos el ciclo do - while para poder mostrar el menú del programa y las opciones del menú llaman a sus respectivas funciones ya sean AgregarTemperatura(), etc.

