Facultad: Ingeniería
Escuela: Computación
Asignatura: Programación Orientada a objetos

GUIA 5:
Clases y Objetos.

Materiales y Equipo

Nº	*Cantidad*	*Descripción*
1	1	Guía de Laboratorio #5 de Programación Orientada a Objetos
2	1	Computadora con programa: ➢ Microsoft Visual C#
3	1	Dispositivo de memoria externo

Introducción

La programación orientada a objetos es un paradigma que se enfoca en (como su nombre lo indica) la creación de objetos para la resolución de problemas.

En esta práctica se tratarán los puntos básicos para la definición de clases y creación de objetos.

Definición.

Una clase se define como la descripción de un objeto, la cual tiene un conjunto de atributos para describir las características del objeto y un conjunto de métodos que describen las operaciones o funciones del objeto.

Pero la definición de clases encierra muchas otros conceptos para cumplir sus objetivos, así en esta práctica veremos los siguientes conceptos:

- Sintaxis de definición de clases.
- Miembros.
- Acceso a miembros.
- Creación de objetos.

Sintaxis para la definición de clase:

```
class <nombre de clase>
{       private:
        <atributos y/o miembros>
        protected:
        <atributos y/o miembros>
        public:
        <atributos y/o miembros>
}
```

Para declarar una clase debe proporcionarse primero su nombre, y se definen las regiones de acceso privada (private) en la que solo la misma clase puede hacer uso de esos elementos, protegida (protected) en la que solo la misma clase y las clases heredadas podrán hacer uso de esos elementos y pública (public) en la que tanto la clase como código externo puede hacer uso de los elementos. Con public se puede acceder desde cualquier parte del proyecto.

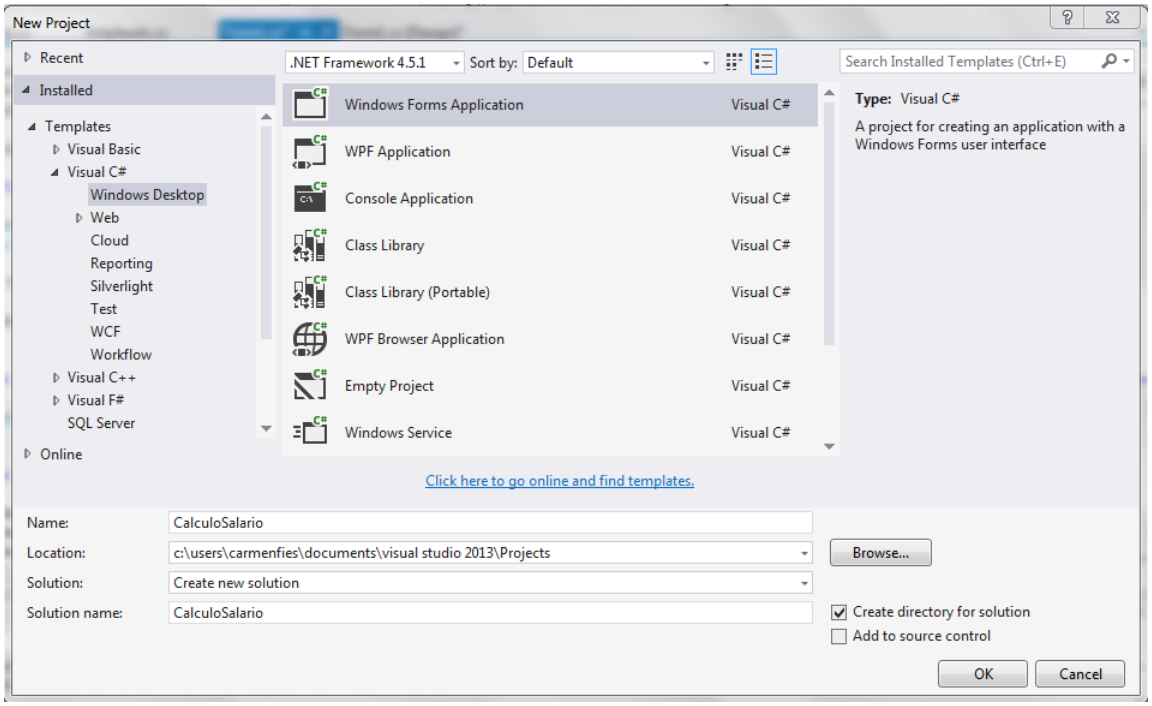
Para la declaración de un objeto la sintaxis es:

nombre_clase nombre_objeto = new nombre_clase();

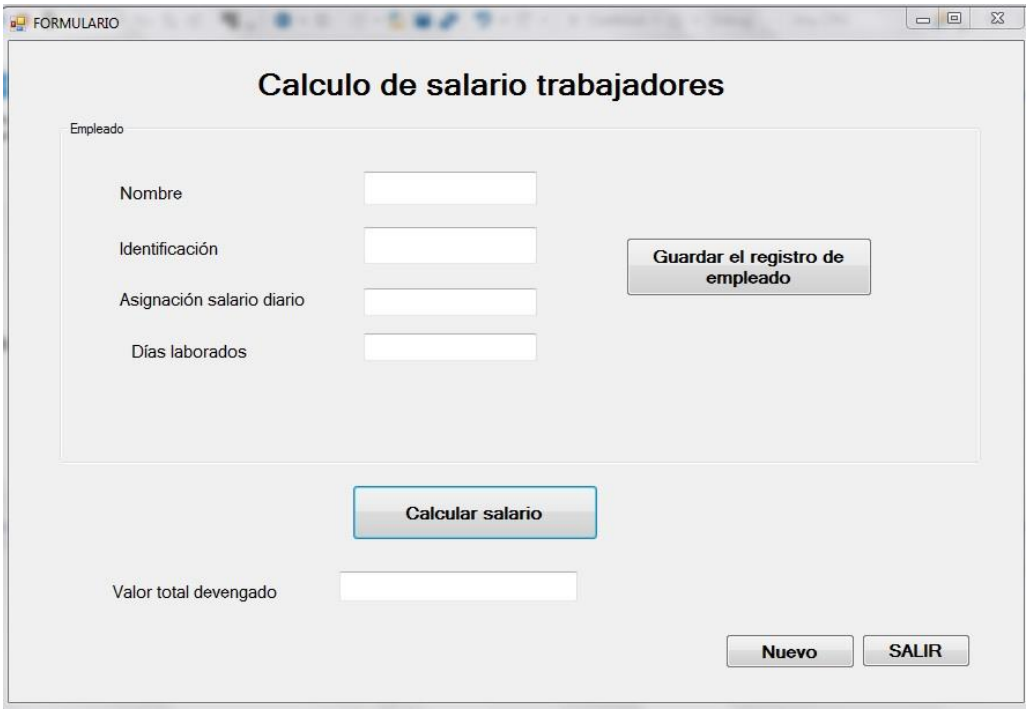
Procedimiento

EJEMPLO 1:

1. Lo primero que debemos hacer es abrir un nuevo proyecto de Windows Form Application y lo nombraremos CalculoSalario



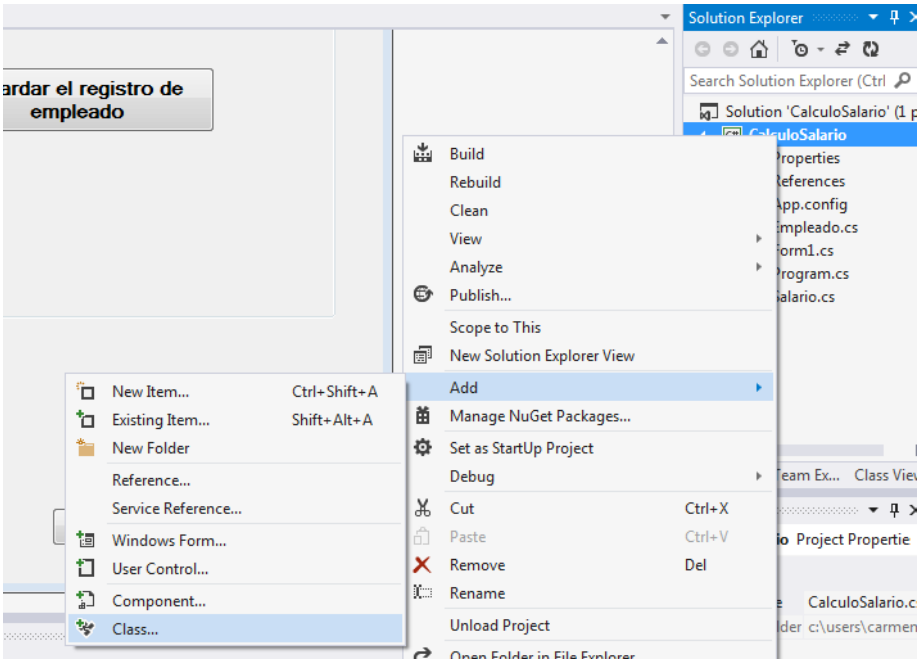
2. Nos aparecerá un entorno similar al que hemos estado trabajando en entorno gráfico. Proceda a crear un formulario que luzca similar a la siguiente imagen:

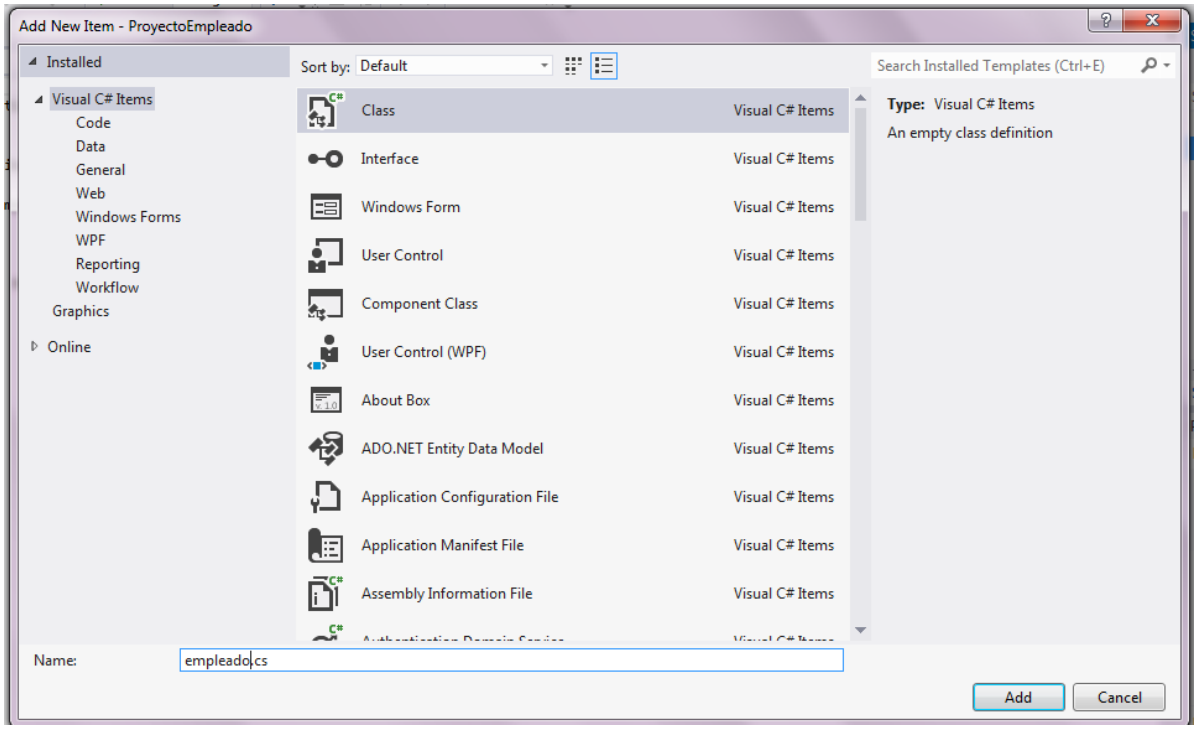


3. Para el formulario se han utilizado los siguientes elementos:

Herramienta	Propiedad	Valor
1 groupBox	text	Empleado
5 Label	text	De acuerdo a lo visto en la imagen
5 textBox	(Nombre) de la herramienta	txtnombre txtidentificacion txtdiario txtdias txtsalario
4 Button	(Nombre) de la herramienta	btnguardar btncalcular btnnuevo btnsalir

4. Ahora crearemos las clases que nos ayudarán a realizar todas las acciones para almacenar información y también para calcular el salario. Crearemos las clases **Empleado** y **Salario** de la siguiente forma: clic derecho en el nombre del proyecto→ Add(agregar) →Class(clase)





Haremos el mismo procedimiento en ambas clases y colocaremos los nombres mencionados.

5. En la clase Empleado tendremos el siguiente código

```
public class Empleado //publica para que pueda ser usada en cualquier instancia del proyecto
        {
            private string nombre; //primer atributo
            public string Nombre  //propiedad asociada al atributo nombre
            {
                get { return nombre; }
                set { nombre = value; }
            }
            private string identificacion;  //segundo atributo
            public string Identificacion   //propiedad asociada al atributo identificacion
            {
                get { return identificacion; }
                set { identificacion = value; }
            }
            private decimal salarioDiario;  //tercer atributo
            public decimal SalarioDiario   //propiedad asociada al atributo salarioDiario
            {
                get { return salarioDiario; }
                set { salarioDiario = value; }
            }
}
```

6. De igual forma incorporaremos código en la clase para Salario

```
public class Salario //publica para que pueda ser usada en cualquier instancia del proyecto
        {
            private int diasLaborados; //atributo
            public int DiasLaborados //propiedad del atributo
            { get; set;}

            //método para calcular salario
            public decimal CalcularSalario(int diaslab, decimal valordia)
            {
                decimal totalsalario = diaslab * valordia;
                return totalsalario;
            }
```

7. Una vez que hayamos definido las clases regresamos al formulario y primero debemos declarar en la clase Form1 las instancias (objetos) de las clases que creamos

```
Empleado miEmpleado = new Empleado(); //instancia de la clase Empleado
 Salario miSalario = new Salario(); //objeto de la clase Salario
```

8. Ahora codificaremos los respectivos botones, comencemos por el botón salir (no olvide activar el evento click, dando doble click sobre el botón)

```
private void btnsalir_Click(object sender, EventArgs e)
 {
     Application.Exit();
 }
```

9. El botón para guardar los datos del empleado

```
private void btnguardar_Click(object sender, EventArgs e)
        {
```

```
 /*los valores obtenidos en los textbox son pasados a los atributos por medio de sus
propiedades, note que mandamos a llamar a través de los objetos creados*/
            miEmpleado.Nombre = txtnombre.Text;
            miEmpleado.Identificacion = txtidentificacion.Text;
            miEmpleado.SalarioDiario = Convert.ToDecimal(txtdiario.Text);
            miSalario.DiasLaborados = int.Parse(txtdias.Text);
            MessageBox.Show("Datos ingresados con éxito");
        }
```

10. Botón calcular

```
private void btncalcular_Click(object sender, EventArgs e)
        {
/*Enviaremos el valor de salario calculado al textbox respectivo, como es un dato
numérico debemos convertirlo a texto, el cálculo lo hace el método de la clase salario*/
txtsalario.Text = Convert.ToString( miSalario.CalcularSalario(miSalario.DiasLaborados,
miEmpleado.SalarioDiario));
        }
```

11. Programando el botón Nuevo, que permite ingresar un nuevo dato

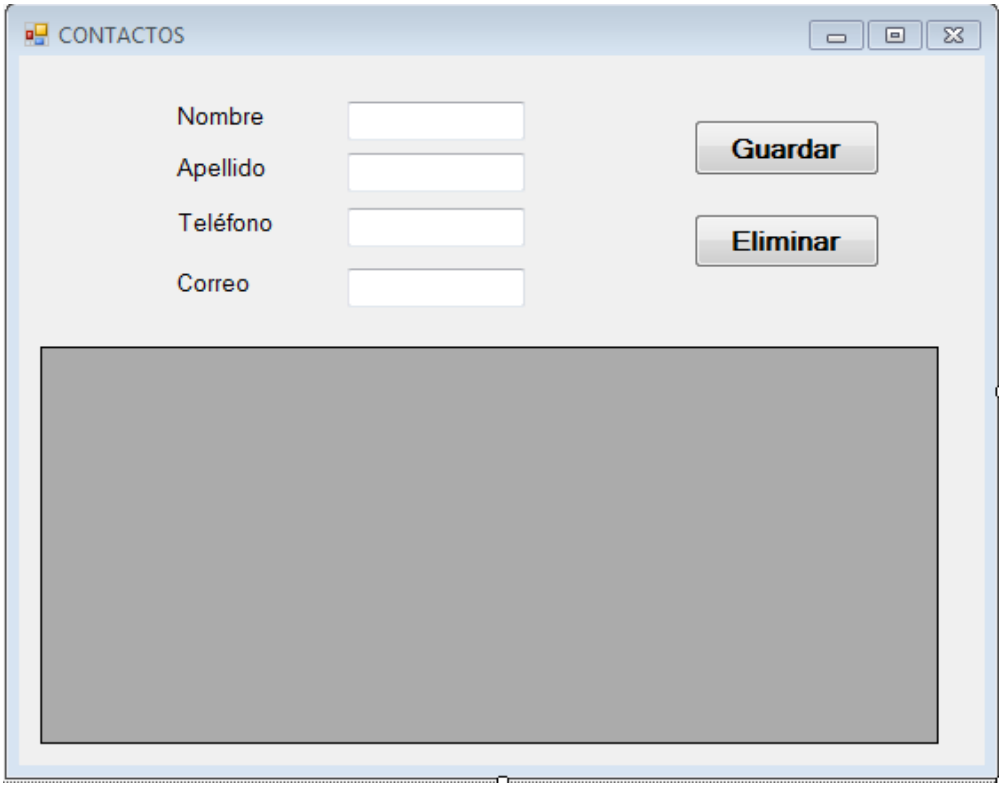
```
private void btnnuevo_Click(object sender, EventArgs e)
        {
            //limpiando todo para ingresar nuevos datos
            txtnombre.Clear();
            txtidentificacion.Clear();
            txtdias.Clear();
            txtdiario.Clear();
            txtsalario.Clear();
            txtnombre.Focus(); //regresa el cursor al textbox del nombre
        }
```

12. Cabe mencionar que si usted únicamente da clic en el botón calcular y no almacena los datos, el valor del salario nunca se actualiza, pues sigue teniendo como valores por defecto los últimos guardados.

EJEMPLO 2:

Realizaremos un ejercicio para llevar una agenda de contactos.

1. Crearemos un nuevo proyecto de Windows Form, el nombre es AgendaPersonas. El formulario lucirá como el de la imagen.



Para el formulario se han utilizado las herramientas de la tabla:

Herramienta	Propiedad	Valor
4 Label	Text	Valores según la imagen
4 TextBox	(Name) de la herramienta	txtnombre txtapellido txttelefono txtcorreo
2 Button	(Name) de la herramienta	btnguardar btneliminar
	Text	Guardar Eliminar
1 DataGridView	(Name) de la herramienta	Dgvcontactos
	SelectionMode	FullRowSelect

2. Una vez creado el formulario agregaremos una nueva clase como se hizo en el ejemplo anterior, esa clase se llamará Persona. En la clase debe ir el siguiente código:

```
class Persona
       { //atributos
           private string nombre;
           private string apellido;
           private string telefono;
           private string correo;
   //métodos
   public string Nombre
    {
        get { return nombre; }
        set { nombre = value; }
           }
   public string Apellido
    {
        get { return apellido; }
        set { apellido = value; }
           }
   public string Telefono
    {
        get { return telefono; }
        set { telefono = value; }
           }
   public string Correo
    {
        get { return correo; }
        set { correo = value; }
           }
       }
```

3. Regresando a la clase del formulario incorporaremos dos líneas de código fuera de cualquier método:

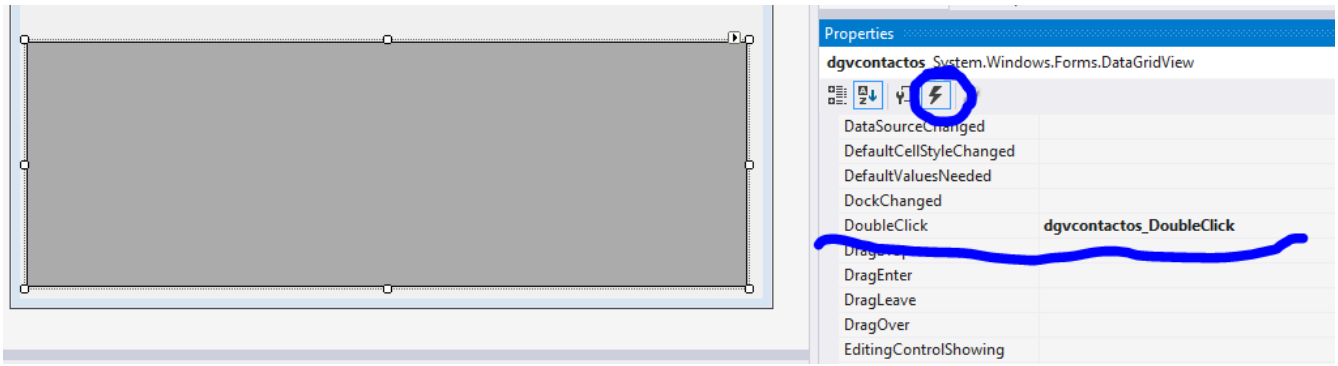
```
/*listado que permite tener varios elementos de la clase Persona*/
private List<Persona> Personas = new List<Persona>();
private int edit_indice = -1; //el índice para editar comienza en -1, esto significa que
no hay ninguno seleccionado, esto servirá para el DataGridView.
```

4. Antes de programar los botones, programaremos unos métodos que nos serán de utilidad. Estos dentro de la clase del formulario.

```
private void actualizarGrid()
        {
            dgvcontactos.DataSource = null;
            dgvcontactos.DataSource = Personas;  /*los nombres de columna que veremos
son los de las propiedades*/
        }
```

```
private void limpiar()
      {
      txtnombre.Clear();
      txtapellido.Clear();
      txttelefono.Clear();
      txtcorreo.Clear();
      }
```

5. A continuación, debemos activar el evento doble click para el DataGridView, así que nos vamos a la opción de eventos y lo seleccionamos. (Cada vez que quiero cambiar fila voy a tener que dar doble click)



Programaremos lo siguiente en el evento:

```
private void dgvcontactos_DoubleClick(object sender, EventArgs e)
 {
     DataGridViewRow seleccion = dgvcontactos.SelectedRows[0];
     int pos = dgvcontactos.Rows.IndexOf(seleccion); //almacena en cual fila estoy
     edit_indice = pos; //copio esa variable en índice editado

       Persona per = Personas[pos]; /*esta variable de tipo persona, se carga con los
       valores que le pasa el listado*/

     txtnombre.Text = per.Nombre; //lo que tiene el atributo se lo doy al textbox
     txtapellido.Text = per.Apellido;
     txttelefono.Text = per.Telefono;
     txtcorreo.Text = per.Correo;
    }
```

6. Ahora sí programaremos el botón para **guardar,** con el código:

```
private void btnguardar_Click(object sender, EventArgs e)
        {
              //creo un objeto de la clase persona y guardo a través de las propiedades
              Persona per = new Persona();
              per.Nombre = txtnombre.Text;
              per.Apellido = txtapellido.Text;
              per.Telefono = txttelefono.Text;
              per.Correo = txtcorreo.Text;
```

```
            if(edit_indice >-1) //verifica si hay un índice seleccionado
             {
                Personas[edit_indice] = per;
                    edit_indice = -1;            }
            else
            {
              Personas.Add(per); /*al arreglo de Personas le agrego el objeto creado
            con todos los datos que recolecté*/

             }

            actualizarGrid();//llamamos al procedimiento que guarda en datagrid
            limpiar();//mandamos a llamar la función que limpia
        }
```

7. Ahora es el turno del botón **Eliminar**

```
    private void btneliminar_Click(object sender, EventArgs e)
            {

                if (edit_indice > -1) //verifica si hay un índice seleccionado
                 {
                        Personas.RemoveAt(edit_indice);
                        edit_indice = -1; //resetea variable a -1
                        limpiar();
                        actualizarGrid();
                 }
                 else
                {
                        MessageBox.Show("Debe dar doble click primero sobre contacto");
                }
```

Desarrollo de habilidades

G5_Ejercicio_01:

Realizar un programa en C# (entorno gráfico), implementando clases que permita a un Banco poder inscribir clientes nuevos para cuentas.

Se solicita tomar en cuenta lo siguiente: Es necesario el DUI del cliente, su nombre, apellidos, tipo de cuenta que apertura (corriente, de ahorros o a plazos), NIT, número de cuenta con formato (CC-00001 para cuenta corriente, CA-00001 para cuenta de ahorros, CP-00001 si fuera cuenta a plazos), monto disponible en la cuenta y sucursal en la que se inscribió (esas sucursales serán dadas por el programador y solo podrán ser 5).

Cree los objetos necesarios, invoque a los métodos que sean necesarios para poder visualizar la información completa para el cliente y cuanto tiene acumulado en cada cuenta.

Investigación Complementaria

Entregar a su instructor un reporte sobre cómo resolvió el siguiente planteamiento:

1. Investigar qué es el anidamiento de clases y agregar un ejemplo de su uso en código de C#.(Debidamente comentariado)

Sitios de Consulta

- https://msdn.microsoft.com/es-es/library/bb386063(v=vs.110).aspx Sitio de Microsoft Visual Studio. Consultado Diciembre 2016.