Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:			Fecha:	18-05-2023						
Nombre:			Docente(2):							
División:			Nota(2):							
Legajo:			Firma(2):							
Instancia(1):	**PP**		**RPP**	X	**SP**		**RSP**		**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

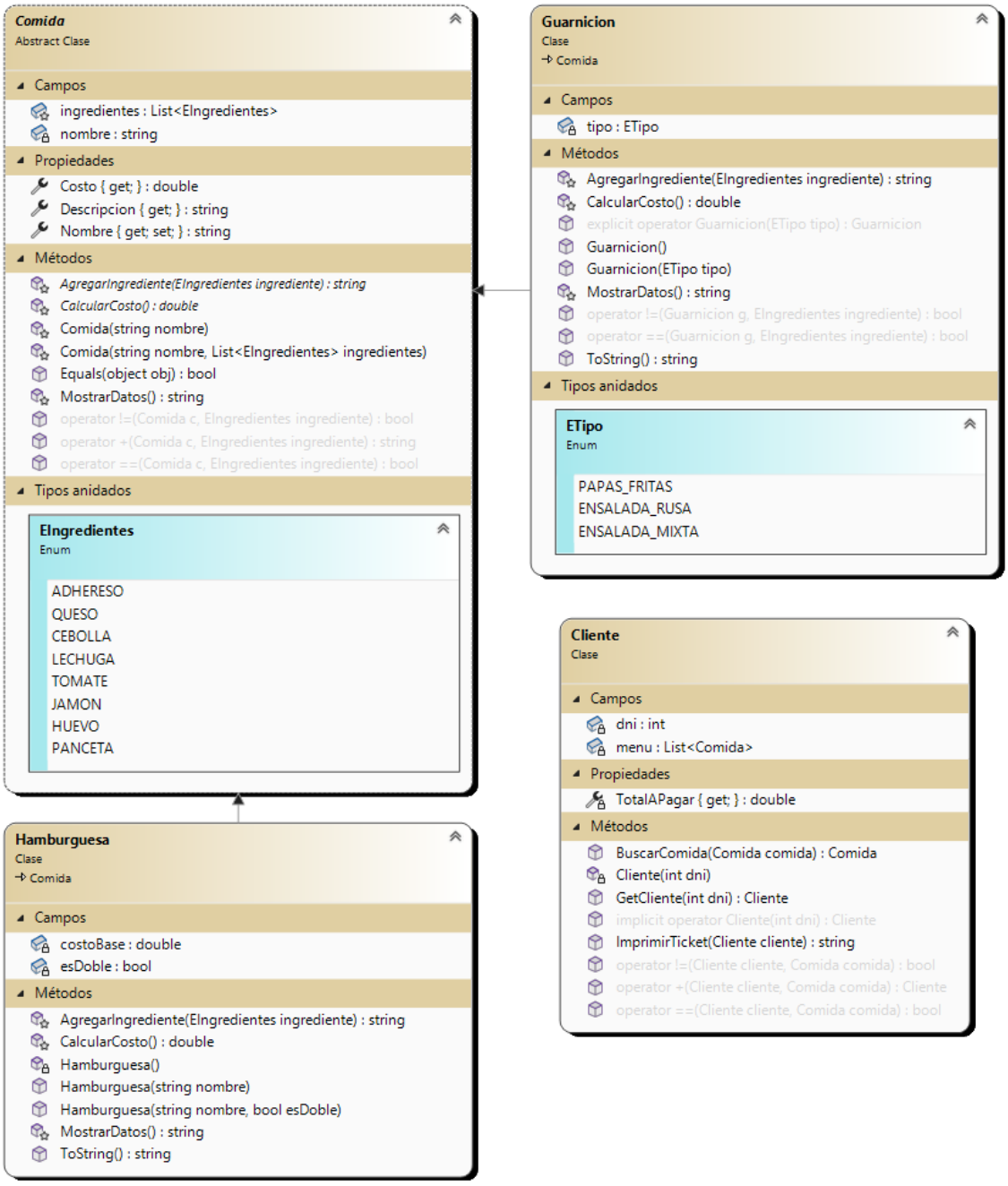
IMPORTANTE:

- **2 (dos) errores en el mismo tema anulan su puntaje.**
- La correcta documentación y reglas de estilo de la cátedra serán evaluadas.
- Colocar sus datos personales en el nombre de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2C. No sé corregirán proyectos que no sea identificable su autor.
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.
- Aplicar los principios de los 4 pilares de la POO.

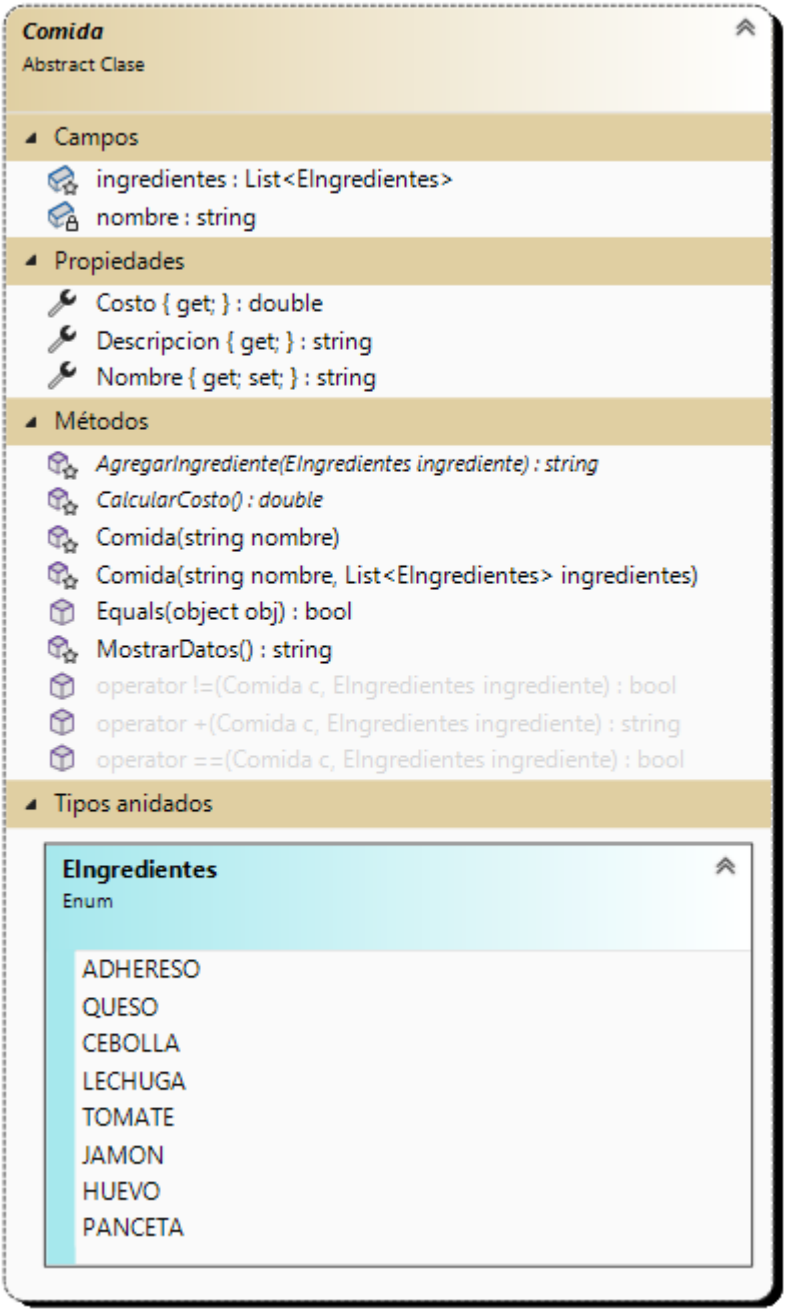
Se desea desarrollar una que gestione la nómina de un centro de atención.

Para ello se debe:

1. Crear un proyecto de tipo biblioteca de clases y con el siguiente esquema:



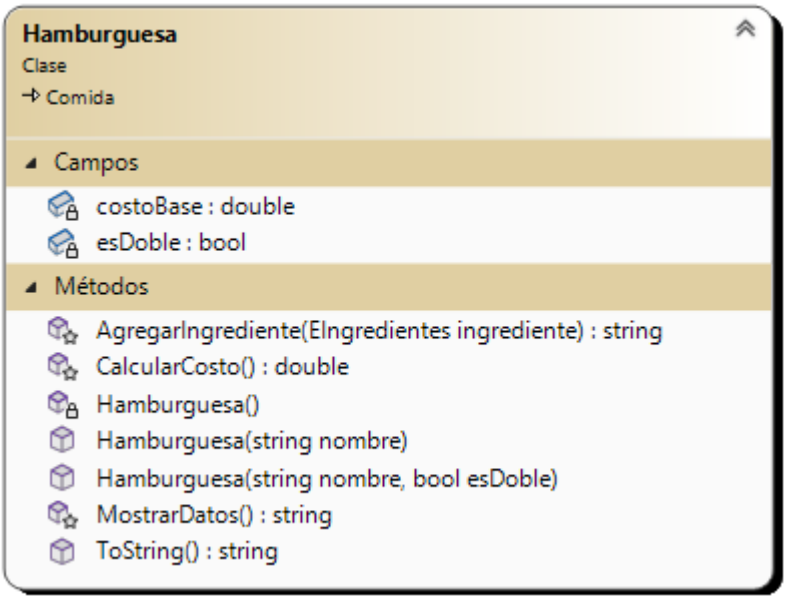
2. Clase Comida:



a. Sera abstracta.
b. Solo la colección de ingredientes será protegida.
c. La propiedad Costo, será de solo lectura y retorna el resultado de CalcularCosto.
d. La propiedad Descripción será de solo lectura y expondrá los datos de la comida.
e. Los métodos AgregarIngredientes y CalcularCosto será protegidos y deberán de ser implementados de forma obligatoria por aquellas clases que deriven de comida.
f. MostrarDatos, concatenara todos los datos de la comida (nombre, costo, etc) incluyendo los ingredientes que se encuentre agregados a esta. Usar StringBuilder.

g. Los ingredientes tendrán los siguientes valores: `{ ADHERESO, QUESO=10, CEBOLLA=8, LECHUGA=7, TOMATE=9, JAMON=12, HUEVO=13, PANCETA =15};`
h. Un ingrediente y una comida serán iguales si el ingrediente ya se encuentra dentro de la lista de ingredientes de la comida.
i. Equals, comprara las comidas a través de su nombre.
j. Sobrecargar el operador + a fin de poder agregar ingredientes a la comida.

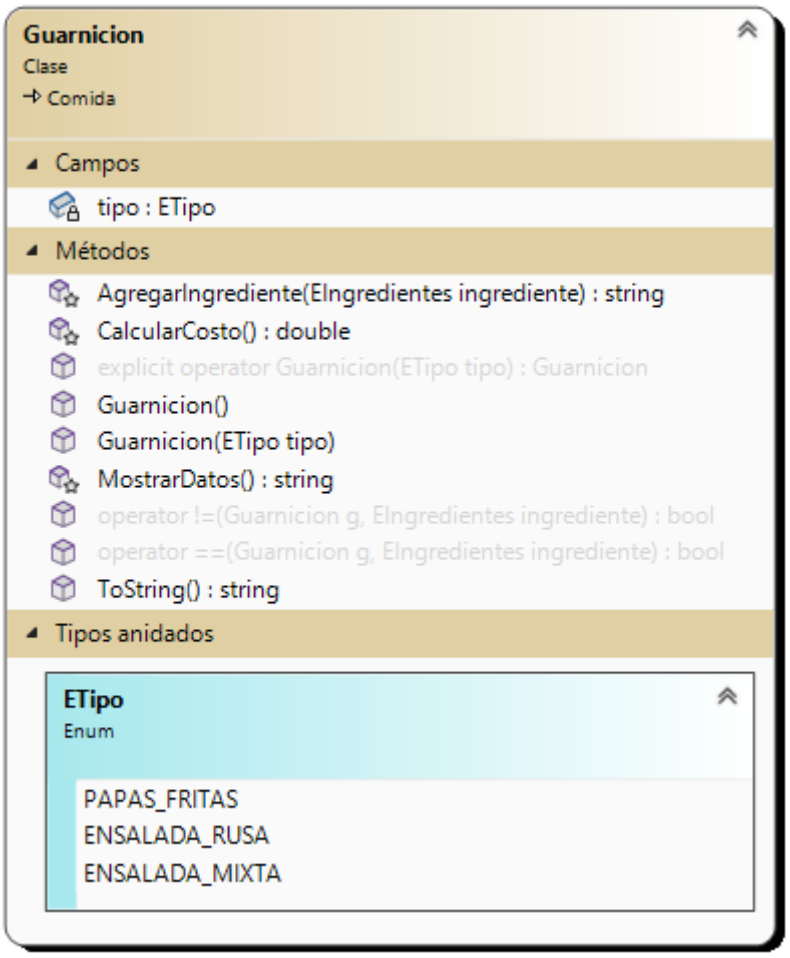
3. **Clase Hamburguesa:**



a. Heredara de Comida.
b. Todos sus atributos son privados.
c. costoBase será de estático y su valor se inicializara en el constructor de clase, siendo su valor inicial 1500.
d. AgregarIngrediente solo agregara el ingrediente si este no se encuentra dentro de la lista de ingredientes de la hamburguesa. En caso de éxito retornara `$"Se agrego {ingrediente} a su hamburguesa";` De lo contrario `$"No se pudo agregar {ingrediente} a su hamburguesa";`
e. El ToString retornara una cadena indicando si la hamburguesa es doble o simple dependiendo de su estado `$"Hamburguesa – Doble o Simple";`
f. MostrarDatos agregara el tipo de Hamburguesa (simple o doble) a los datos de la comida. (Reutilizar código).
g. CalcularCosto realizara los siguiente:
    i. Si la hamburguesa es doble le adicionara al costo $500 más del precio base.
    ii. Recorrerá la lista de ingredientes e ira incrementando el costo proporcionalmente según los valores del ingrediente:
        1. ADHERESO 0%
        2. LECHUGA 7%
        3. CEBOLLA 8%

4. TOMATE 9%
5. QUESO 10%
6. JAMON 12%
7. HUEVO 13%
8. PANCETA 15%

4. Clase Guarnición:



a. Heredará de Comida, su único atributo será privado.

b. El tipo de guarnición tendrá los siguientes valores `{ PAPAS_FRITAS = 1000, ENSALADA_RUSA = 750, ENSALADA_MIXTA = 500 }`

c. El nombre de la guarnición será igual al tipo de guarnición elegida. Por defecto una guarnición será de tipo PAPAS_FRITAS.

d. Una guarnición y un ingrediente serán iguales solo si los ingredientes son: PANCETA, ADHERESO O QUESO.

e. ToString retornara una cadena indicando `$"Guarnicion de tipo {this.tipo}";`

f. MostrarDatos agregar el tipo de guarnición a los datos de la comida. (Reutilizar código).

g. AgregarIngrediente solo podrá agregar el ingrediente a la guarnición si es que este aun no se encuentra en la lista de ingredientes de la guarnicion y si el ingrediente es igual a la guarnición. En caso de éxito retornar `$"Se`

`agrego {ingrediente} a su guarnicion"`; de lo contrario `$"No se pudo agregar {ingrediente} a su guarnicion"`;

h. CalcularCosto realizara los siguiente:

i. tendrá un costo base de acuerdo con el valor de tipo de guarnición:

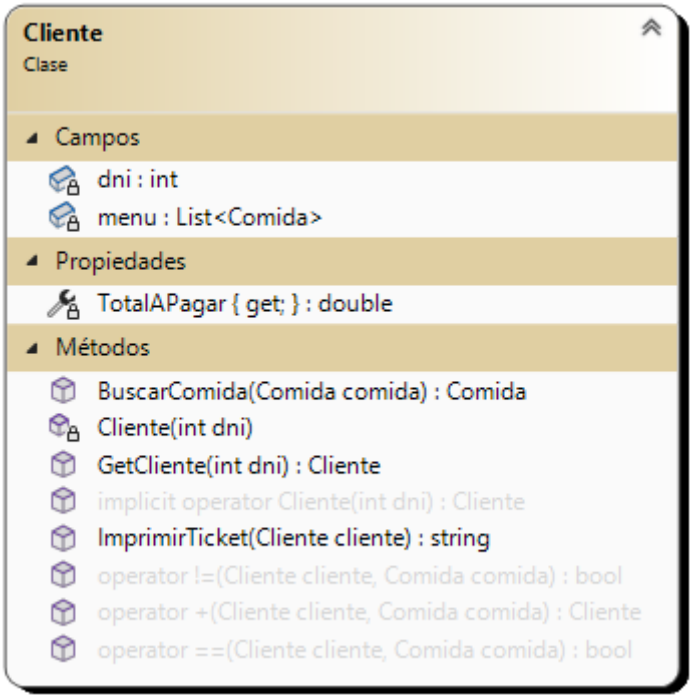
1. PAPAS_FRITAS 1000
2. ENSALADA_RUSA 750
3. ENSALDA_MIXTA 500

ii. Por último, recorrerá la lista de ingredientes permitidos para guarnición e ira incrementando el costo proporcionalmente según los valores del ingrediente:

1. ADHERESO 0%
2. QUESO 10%
3. PANCETA 15%

i. Agregar una conversión explicita a fin de que se retorne una instancia de Guarnición si se caste el tipo de guarnición.

5. Clase Cliente:



a. Todos sus atributos serán privados.

b. Su único constructor será privado.

c. Agregar una conversión implícita a fin de que se retorne una instancia de Cliente con solo asignar un numero entero (dni).

d. GetCliente será de clase y retornará una instancia de cliente.

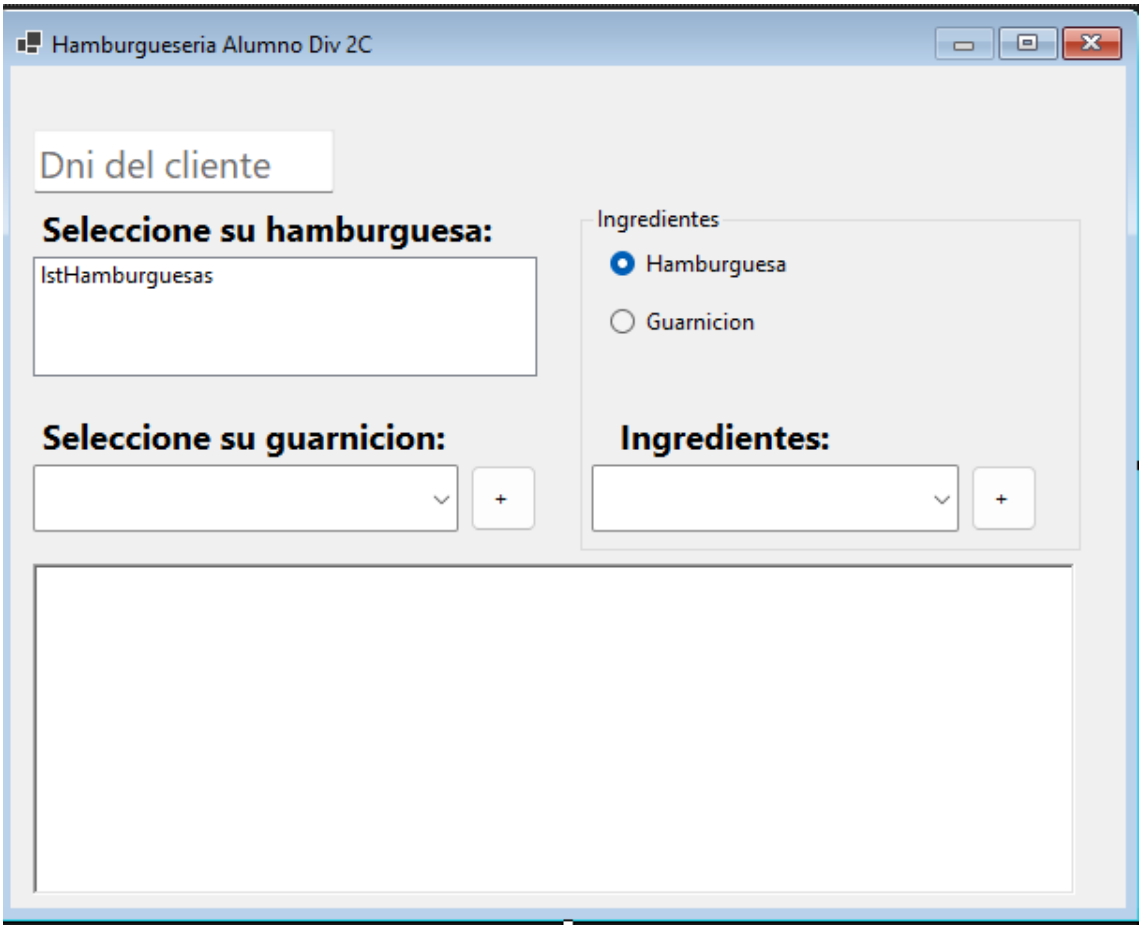
e. La propiedad TotalAPagar será privada y de solo lectura. Esta retornara la sumatoria de los costos del menú del cliente.

f. BuscarComida recibirá una de comida recorrerá el menú del cliente en busca de la comida recibida, comparará por nombre (Reutiliza código). En caso de éxito retornara la primera coincidencia. De lo contrario null.
g. Sobrecargar el operador + a fin de poder agregar una comida al menú del cliente.
h. ImprimirTicket será de clase y concatenará los datos de:
    i. El cliente (dni)
    ii. El menú del cliente. Recorrer la lista y concatenar la descripción de la comida.
    iii. Total a pagar.
    iv. Usar StringBuilder.
i. Un Cliente y una comida serán iguales si la comida se encuentra en el menú del cliente.

6. Agregar un proyecto de consola y colocar el siguiente código:

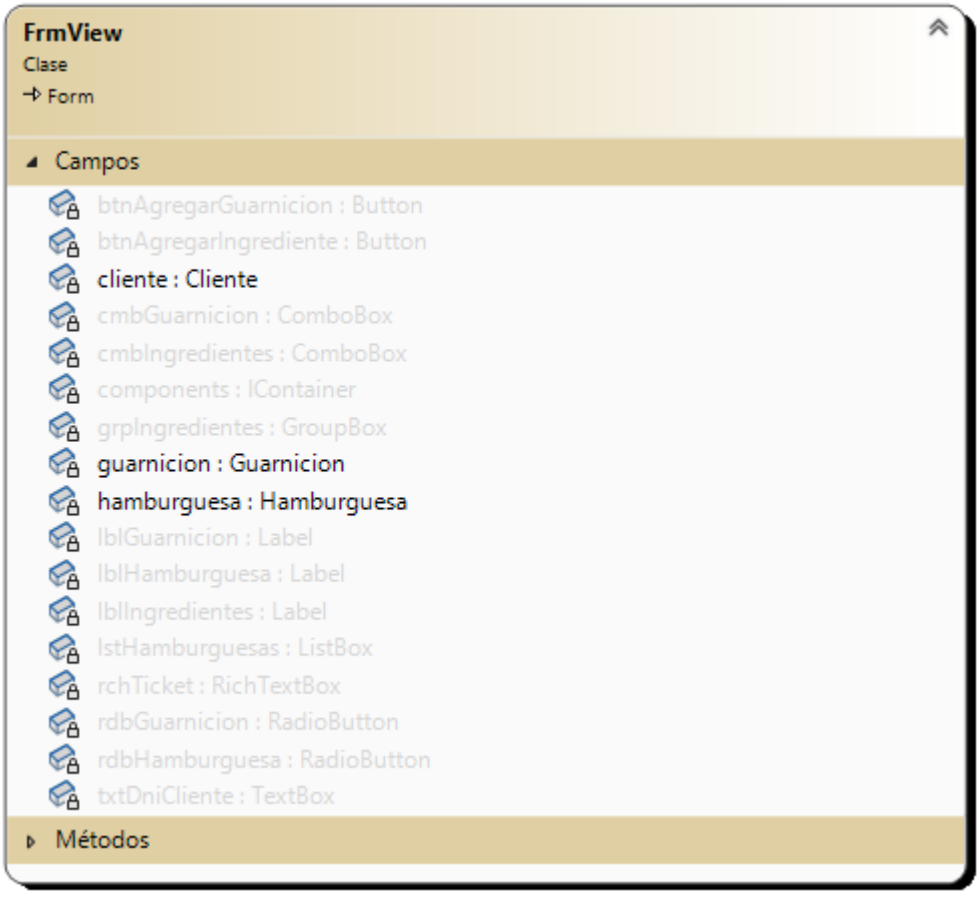
```
Cliente c = 12345678;
Comida h = new Hamburguesa("Doble cuarto de libra", true);
Console.WriteLine(h + Comida.EIngredientes.PANCETA);
Console.WriteLine(h + Comida.EIngredientes.ADHERESO);
Console.WriteLine(h + Comida.EIngredientes.TOMATE);
Console.WriteLine(h + Comida.EIngredientes.PANCETA);
Console.WriteLine("hamburguesa");
Console.WriteLine(h.Descripcion);


Comida g = new Guarnicion();

Console.WriteLine(g + Comida.EIngredientes.LECHUGA);
Console.WriteLine(g + Comida.EIngredientes.QUESO);

Console.WriteLine("Guarnicion");
Console.WriteLine(g.Descripcion);



c += h;
c += g;
Console.WriteLine("Cliente");
Console.WriteLine(Cliente.ImprimirTicket(c));
```

7. Generar un proyecto de tipo form denominado View con un diseño similar al siguiente:



a. El mismo deberá aparecer centrado en pantalla, no se podrá maximizar ni tampoco cambiar el diseño de la venta en tiempo de ejecución.

b. Se deberá respetar las reglas de estilo para los nombres de los controles:



c. Declarar los siguientes atributos como parte de la clase del form:

i. `Cliente cliente;`
ii. `Hamburguesa hamburguesa;`
iii. `Guarnicion guarnicion;`

d. ***En el evento Load del form colocar:***

```
this.cmbGuarnicion.DataSource =
Enum.GetValues(typeof(Guarnicion.ETipo));
            this.cmbIngredientes.DataSource =
Enum.GetValues(typeof(Comida.EIngredientes));
            this.cmbGuarnicion.SelectedItem =
Guarnicion.ETipo.PAPAS_FRITAS;
            this.lstHamburguesas.DataSource = new
List<Hamburguesa>() { new Hamburguesa("Simple con
queso"), new Hamburguesa("Doble con queso", true) };
```

e. ***En el evento TextChanged de txtDniCliente colocar:***

```
int dni;
int.TryParse(txtDniCliente.Text, out dni);
cliente = dni;
```

f. ***En el evento Click del botón agregar ingrediente colocar:***

```
            if (this.rdbGuarnicion.Checked &&
this.cliente == this.guarnicion)
            {
                this.guarnicion =
(Guarnicion)cliente.BuscarComida(this.guarnicion);
```

```
                this.InformarPorPantalla(guarnicion +
            (Comida.EIngredientes)this.cmbIngredientes.SelectedItem);
                }
                else if (this.rdbHamburguesa.Checked &&
            this.cliente == this.hamburguesa)
                {
                    this.hamburguesa =
            (Hamburguesa)cliente.BuscarComida(this.hamburguesa);
                    this.InformarPorPantalla(hamburguesa +
            (Comida.EIngredientes)this.cmbIngredientes.SelectedItem);
                }
                this.ActualizarTicket();
```

g. *En el evento **Mouse Double Click** de la list box de hamburguesas colocar:*

```
                this.hamburguesa =
            (Hamburguesa)this.lstHamburguesas.SelectedItem;
                this.cliente += this.hamburguesa;
                this.ActualizarTicket();
```

h. *En el evento **Click** del botón agregar guarnición colocar:*

```
                this.guarnicion =
            (Guarnicion)((Guarnicion.ETipo)this.cmbGuarnicion.Selecte
            dItem);
                this.cliente += this.guarnicion;
                this.ActualizarTicket();
```

i. *En el evento **Leave** de txt dni cliente colocar:*

```
                if
            (String.IsNullOrEmpty(this.txtDniCliente.Text))
                {
                    this.InformarPorPantalla("Debe completar
            los datos antes de seleccionar el menu");
                    this.txtDniCliente.Focus();
                }
                else
                {
                    this.ActualizarTicket();
                }
```

j. *En el evento **Select Index Changed** del combo box de guarnicion colocar:*

```
            this.guarnicion =
            (Guarnicion)((Guarnicion.ETipo)this.cmbGuarnicion.
            SelectedItem);
```

k. Insertar los siguientes 2 metodos:

```
    private void InformarPorPantalla(string mensaje)
      {
          MessageBox.Show(mensaje, "Informacion",
MessageBoxButtons.OK, MessageBoxIcon.Information);
      }



    private void ActualizarTicket()
      {
          this.rchTicket.Text =
Cliente.ImprimirTicket(this.cliente);
          }
```