

Ejercicio: Clase padre: "Venta", dos clases heredadas: "Contado" y "Credito"

Clase Venta:

```
2 references
class Venta
{
    3 references
    public string cliente { get; set; }

    3 references
    public string ruc { get; set; }

    3 references
    public DateTime fecha { get; set; }

    3 references
    public DateTime hora { get; set; }

    5 references
    public string producto { get; set; }

    5 references
    public int cantidad { get; set; }
}
```

```

// Método para asignar precios según producto
3 references
public double asignaPrecio()
{
    switch (producto)
    {
        case "Lavadora": return 1500;
        case "Refrigeradora": return 3500;
        case "Licuadora": return 500;
        case "Extractoradora": return 150;
        case "Radiograbadora": return 750;
        case "DVD": return 100;
        case "Blue Ray": return 250;
    }

    return 0; // Si no está
}

// Método para calcular el subtotal
9 references
public double calculaSubTotal()
{
    return asignaPrecio() * cantidad;
}

```

Clase Contado:

```

class Contado : Venta // : Venta => Hereda de la clase Venta
{
    public static int n; // Cantidad de productos que lleva el cliente

    // Constructor para el conteo (cada vez que se crea un objeto, aumenta)
    1 reference
    public Contado()
    {
        n += 1;
    }

    // Método GET para n
    1 reference
    public int getN()
    {
        return n;
    }
}

```

```
// Método que calcula descuento
1 reference
public double calculaDescuento(double subTotal)
{
    if (subTotal < 1000)
    {
        return 0.02 * subTotal;
    }
    else
    {
        if (subTotal >= 1000 && subTotal <= 3000)
        {
            return 0.05 * subTotal;
        }
        else
        {
            return 0.12 * subTotal;
        }
    }
}
```

```
// Método para el calculo neto
1 reference
public double calculaNeto(double subTotal, double descuento)
{
    return subTotal - descuento;
}
```

Clase Credito:

```

class Credito : Venta // Clase heredada de Venta
{
    // Variable "x" para saber la cantidad de productos comprados
    public int x;

    // COnstructor para el conteo
    1 reference
    public Credito()
    {
        x += 1;
    }

    // Método GET
    1 reference
    public int getX()
    {
        return x;
    }

    // Cantidad de letras (por ser a crédito)
    2 references
    public int letras { get; set; }
}

```

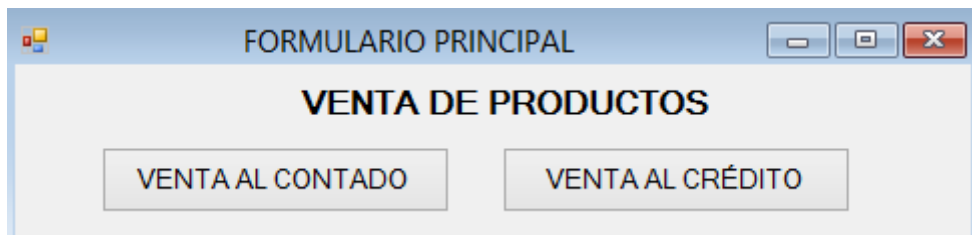
```

// Calcular el monto del interés
1 reference
public double calculaMontoInteres()
{
    switch (letras)
    {
        case 3: return 0.05 * calculaSubTotal();
        case 6: return 0.1 * calculaSubTotal();
        case 9: return 0.15 * calculaSubTotal();
        case 12: return 0.25 * calculaSubTotal();
    }
    return 0;
}

// Calcular monto mensual
0 references
public double calculaMontoMensual()
{
    return (calculaSubTotal() + calculaMontoInteres()) / letras;
}

```

Formulario principal



FORMULARIO PRINCIPAL

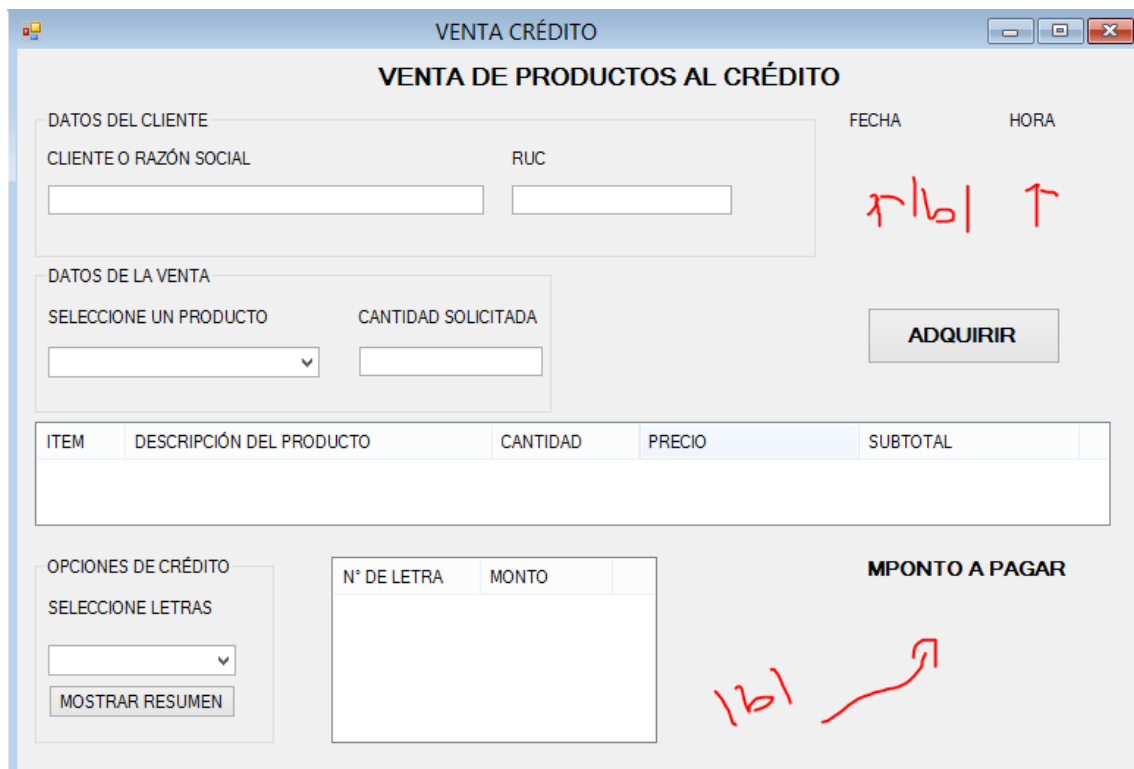
VENTA DE PRODUCTOS

VENTA AL CONTADO VENTA AL CRÉDITO

```
1 reference
private void btnContado_Click(object sender, EventArgs e)
{
    // Abrir el formulario de venta al contado
    frmContado formContado = new frmContado();
    formContado.Show();
}

1 reference
private void btnCredito_Click(object sender, EventArgs e)
{
    // Abrir el formulario de venta al crédito
    frmCredito formCredito = new frmCredito();
    formCredito.Show();
}
```

Formulario para venta al crédito



VENTA CRÉDITO

VENTA DE PRODUCTOS AL CRÉDITO

DATOS DEL CLIENTE

CLIENTE O RAZÓN SOCIAL RUC

FECHA HORA

DATOS DE LA VENTA

SELECCIONE UN PRODUCTO CANTIDAD SOLICITADA

ADQUIRIR

ITEM	DESCRIPCIÓN DEL PRODUCTO	CANTIDAD	PRECIO	SUBTOTAL
------	--------------------------	----------	--------	----------

OPCIONES DE CRÉDITO

SELECCIONE LETRAS

MOSTRAR RESUMEN

N° DE LETRA MONTO

MPONTO A PAGAR

```

public partial class frmCredito : Form
{
    // Arreglos de productos y letras
    private static string[] productos = { "Lavadora", "Refrigeradora", "Licuadora",
                                           "Extractor", "Radiograbadora", "DVD", "Blue Ray"};
    private static int[] letras = { 3, 6, 9, 12};

    // Declarando los arreglos para productos y letras
    private ArrayList aProductos = new ArrayList(productos);

    private ArrayList aLetras = new ArrayList(letras);

    // Subtotal
    double tSubTotal = 0;
    public frmCredito()
    {
        InitializeComponent();
    }
}

```

```

// Método para parsear la fecha
private void mostrarFecha()
{
    lblFecha.Text = DateTime.Now.ToShortDateString();
}

// Método para mostrar la hora
private void mostrarHora()
{
    lblHora.Text = DateTime.Now.ToShortTimeString();
}

private void frmCredito_Load(object sender, EventArgs e)
{
    cboProducto.DataSource = aProductos; // Llenar el combo don DataSource
    cboLetras.DataSource = aLetras; // Llenar el combo don DataSource
    mostrarFecha(); // Cargar al inicio la fecha
    mostrarHora(); // Cargar al inicio la hora
}

```

```

private void btnAdquirir_Click(object sender, EventArgs e)
{
    // Objeto de la clase Credito
    Credito objCredito = new Credito();

    // Datos del cliente
    objCredito.cliente = txtCliente.Text;
    objCredito.ruc = txtRuc.Text;
    objCredito.fecha = DateTime.Parse(lblFecha.Text);
    objCredito.hora = DateTime.Parse(lblHora.Text);

    // Datos del producto => Viene de la clase padre Venta
    objCredito.producto = cboProducto.Text;
    objCredito.cantidad = int.Parse(txtCantidad.Text);

    // Cantidad de letras
    objCredito.letras = int.Parse(cboLetras.Text);

    // Calculando interés
    double interes = objCredito.calculaMontoInteres();

```

```

    // Imprimiendo
    ListViewItem fila = new ListViewItem(objCredito.getX().ToString());
    fila.SubItems.Add(objCredito.producto);
    fila.SubItems.Add(objCredito.cantidad.ToString());
    fila.SubItems.Add(objCredito.asignaPrecio().ToString("C"));
    fila.SubItems.Add(objCredito.calculaSubTotal().ToString(""));
    lvDetalle.Items.Add(fila);

    // Subtotal + interés
    tSubTotal += objCredito.calculaSubTotal() + interes;
    lblMonto.Text = tSubTotal.ToString("0.00");
}

```

```

// Método para calcular el monto de cada letra
4 references
private void montoLetras(int letras)
{
    double montoMensual = double.Parse(lblMonto.Text) / letras;
    lvResumen.Items.Clear();
    for (int i = 1; i <= letras; i++)
    {
        ListViewItem fila = new ListViewItem(i.ToString());
        fila.SubItems.Add(montoMensual.ToString("C"));
        lvResumen.Items.Add(fila);
    }
}

```

Mostrar resumen

```
private void button1_Click(object sender, EventArgs e)
{
    int letras = int.Parse(cboLetras.Text);

    switch (letras)
    {
        case 3: montoLetras(3); break;
        case 6: montoLetras(6); break;
        case 9: montoLetras(9); break;
        case 12: montoLetras(12); break;
        default:
            break;
    }
}
```

Formulario para venta el contado

VENTA AL CONTADO

VENTA DE PRODUCTOS AL CONTADO

DATOS DEL CLIENTE

CLIENTE O RAZÓN SOCIAL RUC

DATOS DE LA VENTA

SELECCIONE UN PRODUCTO CANTIDAD SOLICITADA

ADQUIRIR

ITEM	DESCRIPCIÓN DEL PRODUCTO	CANTIDAD	PRECIO	SUBTOTAL

RESUMEN

lstResumen

NETO A PAGAR


```

public partial class frmContado : Form
{
    // Arreglo de productos
    private static string[] productos = { "Lavadora", "Refrigeradora", "Licuadora",
                                           "Extractor", "Radiograbadora", "DVD", "Blue Ray"};

    // Colocando el arreglo a un objeto de tipo ArrayList
    private ArrayList aProductos = new ArrayList(productos);

    // Acumulador de sutotoal, inicializado en cero
    private double tSubTotales = 0;

    1 reference
    public frmContado()
    {
        InitializeComponent();
    }
}

```

```

// Método para parsear la fecha
1 reference
private void mostrarFecha()
{
    lblFecha.Text = DateTime.Now.ToShortDateString();
}

// Método para mostrar la hora
1 reference
private void mostrarHora()
{
    lblHora.Text = DateTime.Now.ToShortTimeString();
}

```

```

private void frmContado_Load(object sender, EventArgs e)
{
    cboProducto.DataSource = aProductos; // Llenar el combo con DataSource
    mostrarFecha(); // Cargar al inicio la fecha
    mostrarHora(); // Cargar al inicio la hora
}

```

```
private void btnAdquirir_Click(object sender, EventArgs e)
{
    // Creando el objeto de la clase Contado
    Contado objConstado = new Contado();

    // Datos del cliente
    objConstado.cliente = txtCliente.Text; // Nombre de cliente
    objConstado.ruc = txtRuc.Text; // RUC
    objConstado.fecha = DateTime.Parse(lblFecha.Text); // Fecha
    objConstado.hora = DateTime.Parse(lblHora.Text); // Fecha

    // Datos del producto
    objConstado.producto = cboProducto.Text;
    objConstado.cantidad = int.Parse(txtCantidad.Text);

    // Agregar al listado
    ListViewItem fila = new ListViewItem(objConstado.getN().ToString());
    fila.SubItems.Add(objConstado.producto);
    fila.SubItems.Add(objConstado.cantidad.ToString());
    fila.SubItems.Add(objConstado.asignaPrecio().ToString("C"));
    fila.SubItems.Add(objConstado.calculaSubTotal().ToString(""));
    lvDetalle.Items.Add(fila);
}
```

```
    // Listar => Método
    listado(objConstado);
}
```

```
// Método para listar el resultado
private void listado(Contado objC)
{
    tSubTotales += objC.calculaSubTotal();
    lstResumen.Items.Clear();
    lstResumen.Items.Add("*** RESUMEN DE VENTA ***");
    lstResumen.Items.Add("-----");
    lstResumen.Items.Add("Cliente: " + objC.cliente);
    lstResumen.Items.Add("RUC: " + objC.ruc);
    lstResumen.Items.Add("Fecha: " + objC.fecha);
    lstResumen.Items.Add("Hora: " + objC.hora);
    lstResumen.Items.Add("-----");
    lstResumen.Items.Add("SubTotal: " + tSubTotales.ToString("C"));
    // Calculando descuento
    double descuento = objC.calculaDescuento(tSubTotales);
    // Calculando Neto
    double neto = objC.calculaNeto(tSubTotales, descuento);
    // Añadir a la lista
    lstResumen.Items.Add("Descuento: " + descuento.ToString("C"));
    lstResumen.Items.Add("Neto: " + neto.ToString("C"));
}
```

```
    lblNeto.Text = neto.ToString("C");
}
```