

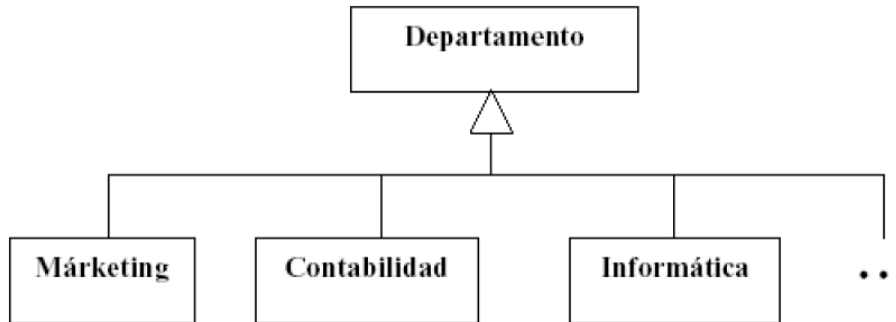


INTRODUCCIÓN

Continuando con los conceptos de la POO de la práctica anterior tenemos que dentro de las clases existe la herencia.

Herencia. Existen clases que son diseñadas desde cero. El estado y comportamiento de la clase se definen en ella misma. Sin embargo otras clases toman su definición de otra clase. En lugar de escribir otra clase de nuevo, se puede tomar el estado y comportamientos de otra clase y usarlos como punto de partida para la nueva clase. A la acción de definir una clase usando otra clase como punto de partida se le llama herencia.

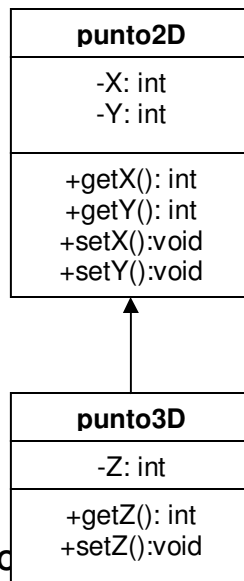
Su representación en diagramas de estructuras estáticas es de la siguiente manera:



Existen varios tipos de herencia que mencionaremos a continuación.

Herencia simple: La herencia simple se refiere a que una clase solo puede derivar de una clase base. Como ejemplo vamos a disponer de una nueva clase que defina una coordenada tridimensional X, Y y Z.

Partiremos utilizando del ejemplo anterior la clase Point que maneja una coordenada bidimensional y definiremos una clase nueva que herede de la clase Point. En diagrama UML se representa de la siguiente manera:





TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

El diagrama UML representa a una clase **punto3D** que hereda de la clase **punto2D** sus **variables Privadas X y Y** y sus **métodos Públicos getX, getY, setX y setY**, además **implementa una tercera variable Z** y sus **métodos getZ y setZ**.

Esto traducido en código C#

```
using System;
using System.Collections.Generic;
using System.Text;

namespace puntosHerencia
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 5, b = -3, c=15, d=7;
            punto2D p1 = new punto2D();
            punto2D p2 = new punto2D();
            punto3D p3 = new punto3D();
            p1.X = a;
            p1.Y = b;
            Console.WriteLine("Valores x={0} y y={1}", p1.X, p1.Y);
            p2.X = c;
            p2.Y = d;
            Console.WriteLine("Valores x={0} y y={1}", p2.X, p2.Y);
            p3.X = 8;
            p3.Y = 10;
            p3.Z = 9;
            Console.WriteLine("Valores x={0}, y={1} y z={2}", p3.X, p3.Y, p3.Z);
        }
    }
}

class punto2D
{
    private int x;
    private int y;

    public int X
    {
        get
        {
            return x;
        }
        set
        {
            if (value < 0)
                x = 0;
            else
                x = value;
        }
    }

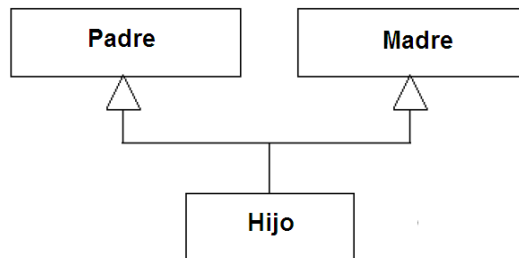
    public int Y
    {
        get
        {
            return y;
        }
        set
        {
            if (value < 0)
                y = 0;
            else
                y = value;
        }
    }
}
```



```
    }  
  }  
}  
class punto3D : punto2D  
{  
  private int z;  
  
  public int Z  
  {  
    get  
    {  
      return z;  
    }  
    set  
    {  
      if (value < 10)  
        z = 5;  
      else  
        z = value;  
    }  
  }  
}
```

Derivar una clase de otra permite automáticamente que la nueva clase tengan disponibles todos los métodos y variables públicas y protegidas de la clase Base.

Herencia múltiple. Permite que una clase derive de más de una clase Base. En este curso no vamos a utilizar este concepto debido a que C# no soporta la herencia múltiple.





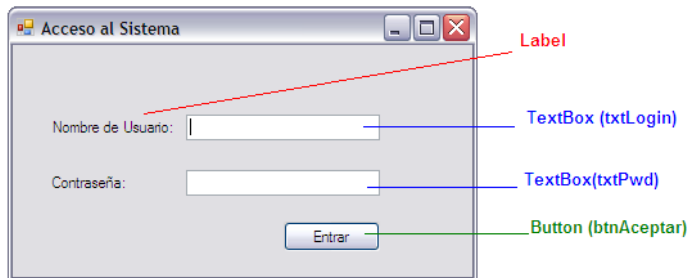
Ejercicios de la Práctica.

Parte I. Herencia.

Modifica el proyecto **puntos** de la práctica anterior y adiciona el código de la clase **punto3D**. Compila el ejemplo de Herencia mostrado previamente. Ejecútalo y anota tus comentarios y observaciones.

Parte II. Formulario Windows. Login Usuario (Continuación)

2.1 Ahora vamos a editar el proyecto **login_usuario** de la práctica anterior para que verifique si el usuario introducido es correcto.



Primero vamos a modificar la propiedad **PasswordChar** del objeto txtPwd. Para ello vamos a ir a la **ventana de Propiedades** y vamos a poner el siguiente carácter *

2.2 Ahora vamos a editar el evento clic del botón **btnAceptar** para que compare si el nombre de usuario y contraseña son iguales a determinadas variables, por ejemplo **nombre de usuario:** "tecnicas" y **contraseña:** "123". Para ello de doble clic sobre el botón, y posteriormente escriba el siguiente código:

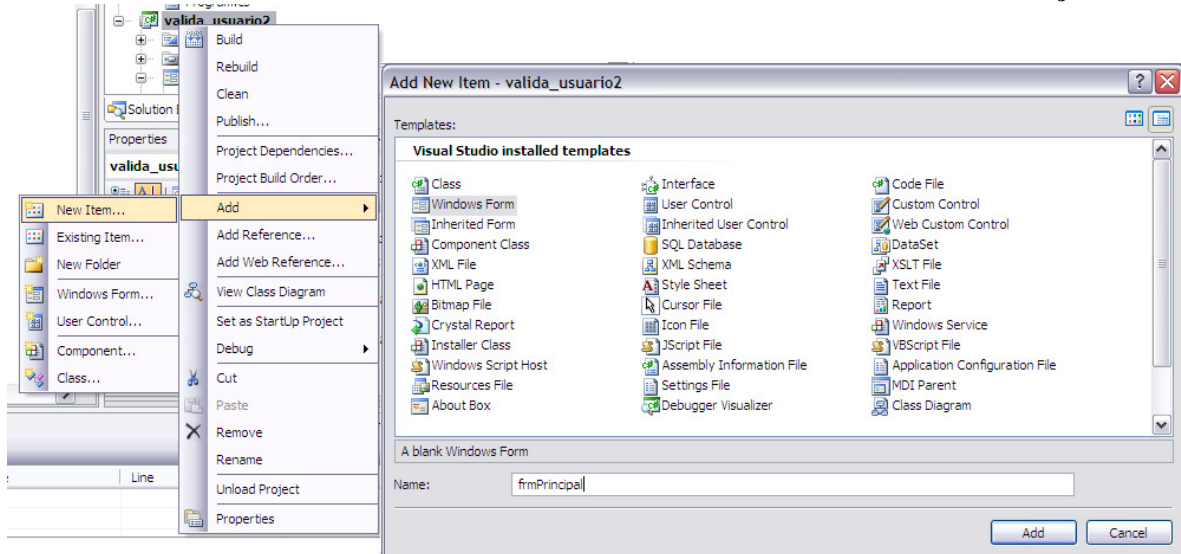
```
private void btnAceptar_Click(object sender, EventArgs e)
{
    if ((txtLogin.Text == "tecnicas") && (txtPwd.Text == "123"))
        MessageBox.Show("Bienvenido Usuario");
    else
        MessageBox.Show("Nombre de Usuario o contraseña Incorrecta");
}
```

Recuerde no copie el código, porque puede ocasionar que la aplicación no opere correctamente.

2.3 Ahora vamos a adicionar un formulario al proyecto. Para ello colóquese en el explorador de proyectos, seleccione el proyecto con el botón derecho de Mouse, del menú emergente seleccione Add> New Item...>Windows Form



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS



Llama a tu formulario **frmDatos**. Adiciona **textBox** que almacenen los siguientes datos: Nombre, número de cuenta, edad y promedio.

2.4 Posteriormente coloca un **Button** “Mostrar datos”, al dar clic deberá mostrar por medio de un **MessageBox**, esta información.

Ejemplo:

```
MessageBox.Show("Nombre " + nombre.Text + ", No. Cta. " + ncuenta.Text);
```

2.5 Adiciona un botón de nombre **btnCerrar**, da doble clic sobre él y anota el siguiente código:

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

2.6 Posteriormente vamos a editar nuevamente el **btnAceptar** del formulario **login_usuario**, de la siguiente manera:

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    if ((txtLogin.Text == "tecnicas") && (txtPwd.Text == "123"))
    {
        MessageBox.Show("Bienvenido Usuario");
        frmDatos fD = new frmDatos();
        fD.Show();
        this.visible= false;
    }
    else
        MessageBox.Show("Nombre de Usuario o contraseña Incorrecta");
}
```



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

III. Ejemplo completo:

3.1 Crea un nuevo proyecto y llámalo **banco**. Selecciona el proyecto con el botón derecho y agrega una nueva clase llamada **cuentas**. Escribe el siguiente código en ella:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace banco
{
    public class cuenta_principal
    {
        public string nombre_cliente;
        private decimal saldo;
        public long cuenta;
        public bool conectado;
        public decimal Saldo
        {
            get
            {
                if (conectado == true)
                    return saldo;
                else
                    return 0;
            }
            set
            {
                if (conectado == true)
                    saldo = value;
            }
        }

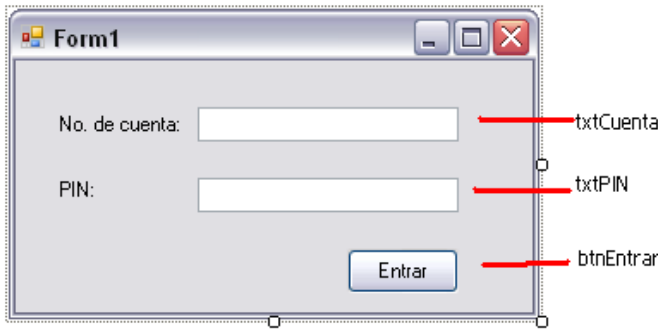
        public string retiro(decimal cantidad)
        {
            if (cantidad <= saldo)
            {
                saldo = saldo - cantidad;
                return "Entregando el dinero...";
            }
            else
            {
                return "Saldo insuficiente...";
            }
        }

        public string deposito(decimal cantidad)
        {
            if (cantidad <= 0)
            {
                return "Debe indicar la cantidad";
            }
            else
            {
                saldo = saldo + cantidad;
                return "Deposito realizado...";
            }
        }
    }
}
```



```
}  
  
class cuenta_cheques : cuenta_principal  
{  
    private decimal saldo;  
}  
class cuenta_inversion  
{  
    private decimal saldo;  
}  
}
```

3.2 Edita el formulario principal de la siguiente manera:



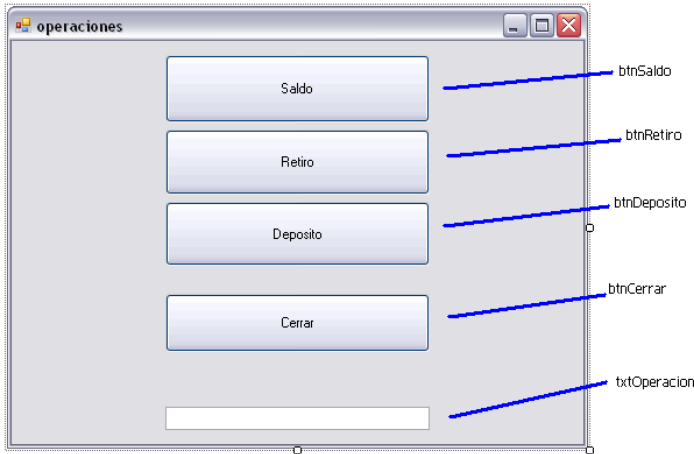
3.3 Adiciona el siguiente código:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
  
namespace banco  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void btnEntrar_Click(object sender, EventArgs e)  
        {  
            if (txtCuenta.Text.Equals("123") && txtPIN.Text.Equals("123"))  
            {  
                cuenta_principal CP = new cuenta_principal();  
                CP.nombre_cliente = "Gabriela";  
                CP.cuenta = Convert.ToInt64(txtCuenta.Text);  
                CP.conectado = true;  
                operaciones OP = new operaciones(CP);  
                OP.Show();  
            }  
        }  
    }  
}
```



```
        this.Visible = false;
    }
    else
        MessageBox.Show("usuario incorrecto");
    }
}
```

3.4 Adiciona un nuevo formulario y llámalo **operaciones**.



3.5 Cambia la propiedad **Modifiers** del **txtOperacion** a **Public**.

3.6 Adiciona el siguiente código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace banco
{
    public partial class operaciones : Form
    {
        cuenta_principal cp;
        public operaciones()
        {
            InitializeComponent();
        }

        public operaciones(cuenta_principal CP)
        {
            InitializeComponent();
            cp = CP;
        }

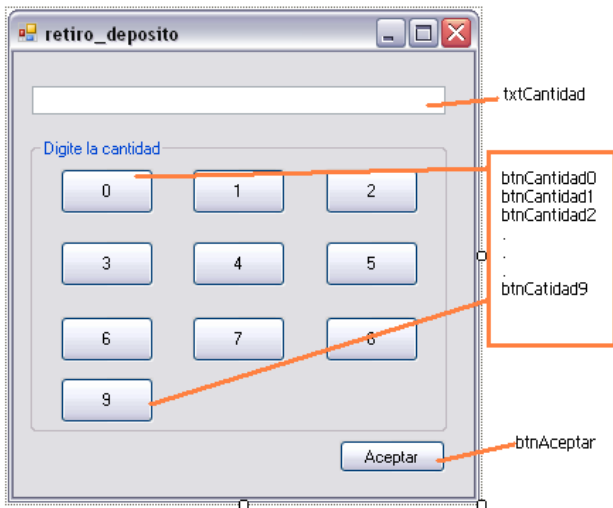
        private void btnSaldo_Click(object sender, EventArgs e)
        {
            MessageBox.Show(""+cp.Saldo);
        }
    }
}
```



```
}  
  
private void btnRetiro_Click(object sender, EventArgs e)  
{  
    retiro_deposito rd = new retiro_deposito(cp, 1, this);  
    rd.Show();  
}  
  
private void btnCerrar_Click(object sender, EventArgs e)  
{  
    Application.Exit();  
}  
  
private void btnDeposito_Click(object sender, EventArgs e)  
{  
    retiro_deposito rd = new retiro_deposito(cp, 2, this);  
    rd.Show();  
}  
}
```

Recuerda que para generar los eventos Click debes dar doble click sobre el botón o bien ir a operaciones.Designer.cs y crearlos.

3.7 Adiciona un nuevo formulario y llámalo **retiro_deposito**. Editalo de la siguiente manera:



3.8 Edita el código de la siguiente manera:



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace banco
{
    public partial class retiro_deposito : Form
    {
        cuenta_principal cp;
        operaciones op;
        byte operacion;

        public retiro_deposito()
        {
            InitializeComponent();
        }

        public retiro_deposito(cuenta_principal CP, byte Operacion, operaciones OP)
        {
            InitializeComponent();
            cp = CP;
            operacion = Operacion;
            op = OP;
        }

        private void btnAceptar_Click(object sender, EventArgs e)
        {
            if (operacion == 1) //retiro
            {
                op.txtOperacion.Text = cp.retiro(Convert.ToDecimal(txtCantidad.Text));
            }
            else //Deposito
            {
                op.txtOperacion.Text = cp.deposito(Convert.ToDecimal(txtCantidad.Text));
            }
            this.Close();
        }

        private void btnCantidad0_Click(object sender, EventArgs e)
        {
            txtCantidad.Text = txtCantidad.Text + "0";
        }

        private void btnCantidad1_Click(object sender, EventArgs e)
        {
            txtCantidad.Text = txtCantidad.Text + "1";
        }
    }
}
```

3.9 Adiciona los eventos Click de los demás botones (btnCantdad)

Compila todo el proyecto y ejecútalo. Anota tus observaciones.