

## **GUÍA PRACTICA Nº 5**

### **INTRODUCCIÓN.**

La abstracción, es un acto mental en el que se aísla conceptualmente una propiedad o función concreta de un objeto, y se piensa qué es, ignorando otras propiedades del objeto en cuestión. El encapsulamiento es el ocultamiento de los datos miembro de un objeto de manera que solo se pueda cambiar mediante las operaciones definidas para ese objeto. La sobrecarga se refiere a la posibilidad de tener dos o más funciones con el mismo nombre, pero con funcionalidad diferente. El polimorfismo es la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.

### **OBJETIVOS DE APRENDIZAJE.**

Al finalizar esta práctica, es estudiante será capaz de:



- Identificar conceptos de abstracción, encapsulamiento, sobrecarga y polimorfismo.
- Aplicar abstracción a objetos del mundo real.
- Aplicar métodos de encapsulamiento, sobrecarga y polimorfismos al diseño de sistemas.

### **MATERIAL Y EQUIPO.**



- Guía de Laboratorio.
- Computadora con sistema operativo Windows 8.
- Páginas de papel bond o cuaderno
- Internet.

## TIPOS DE RELACIONES.

Existen cuatro tipos de relaciones entre los elementos de un modelo UML: *Asociación*, *dependencia*, *generalización* y *realización*.

En las relaciones se habla de una clase destino y de una clase origen. La clase origen es desde la que se realiza la acción de relacionar. Es decir, desde la que parte la flecha. La clase destino es la que recibe la flecha. Las relaciones se pueden modificar con estereotipos o con restricciones.

Multiplicidad	Significado
1	Uno y sólo uno
0..1	Cero o uno
N..M	Desde N hasta M
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios (al menos uno)

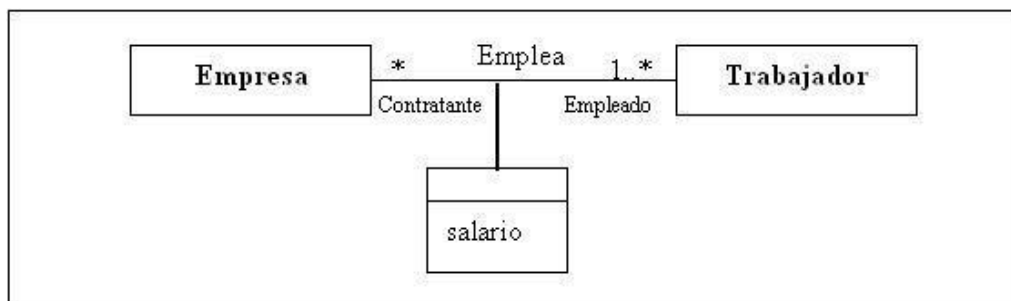
### Valores de multiplicidad

### EJEMPLO.

En el ejemplo de la relación "Un trabajador labora en una empresa", aplicando la multiplicidad sería "Uno o varios empleados trabajan en una empresa". Y el diagrama se vería así:



Ejemplo de Clase Asociación

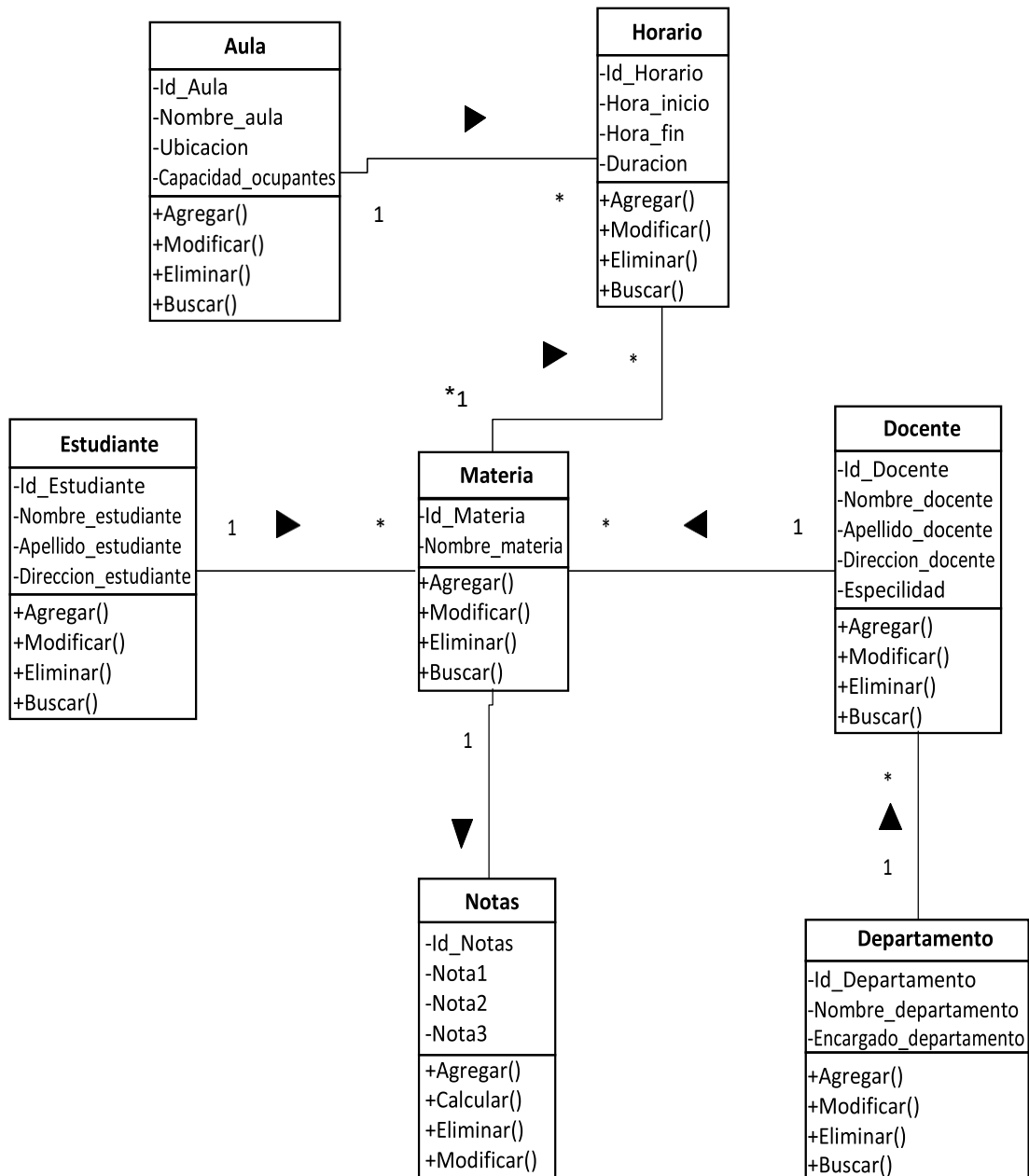


## EJERCICIO PRÁCTICO

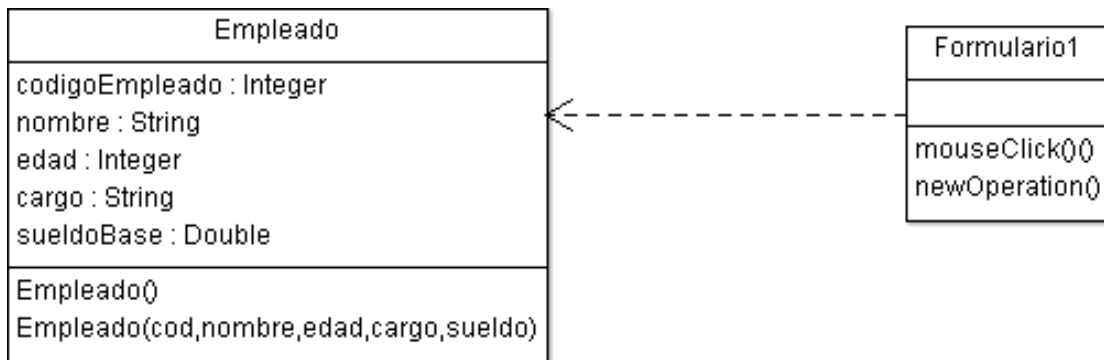


1. En el siguiente diagrama tomar cada una de las relaciones y describir su asociatividad tomando en cuenta su multiplicidad.

### SISTEMA DE CONTROL DE NOTAS Y HORARIOS.

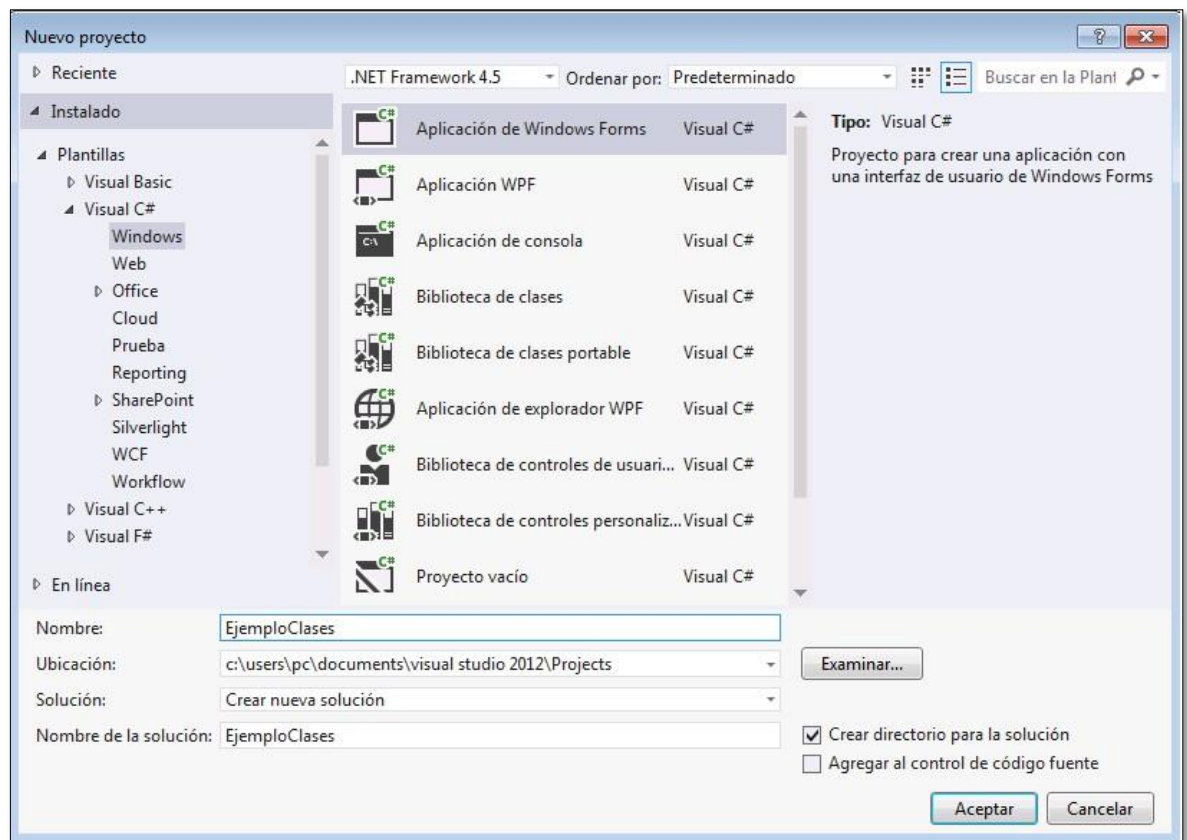


2. En el IDE de Visual Studio, deberá de codificar el siguiente diagrama de clases:

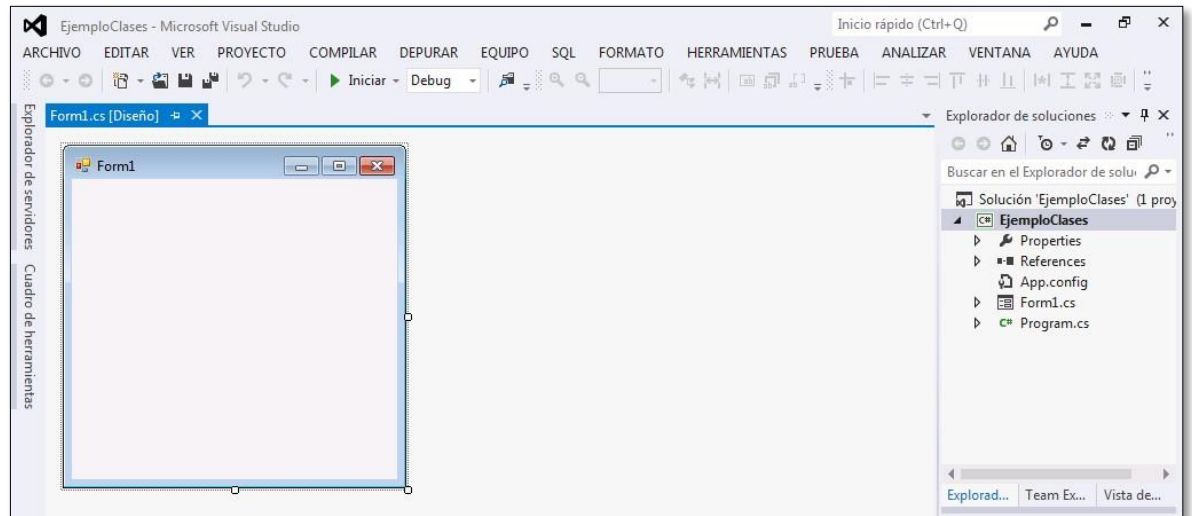


**Desarrollar cada uno de los siguientes pasos:**

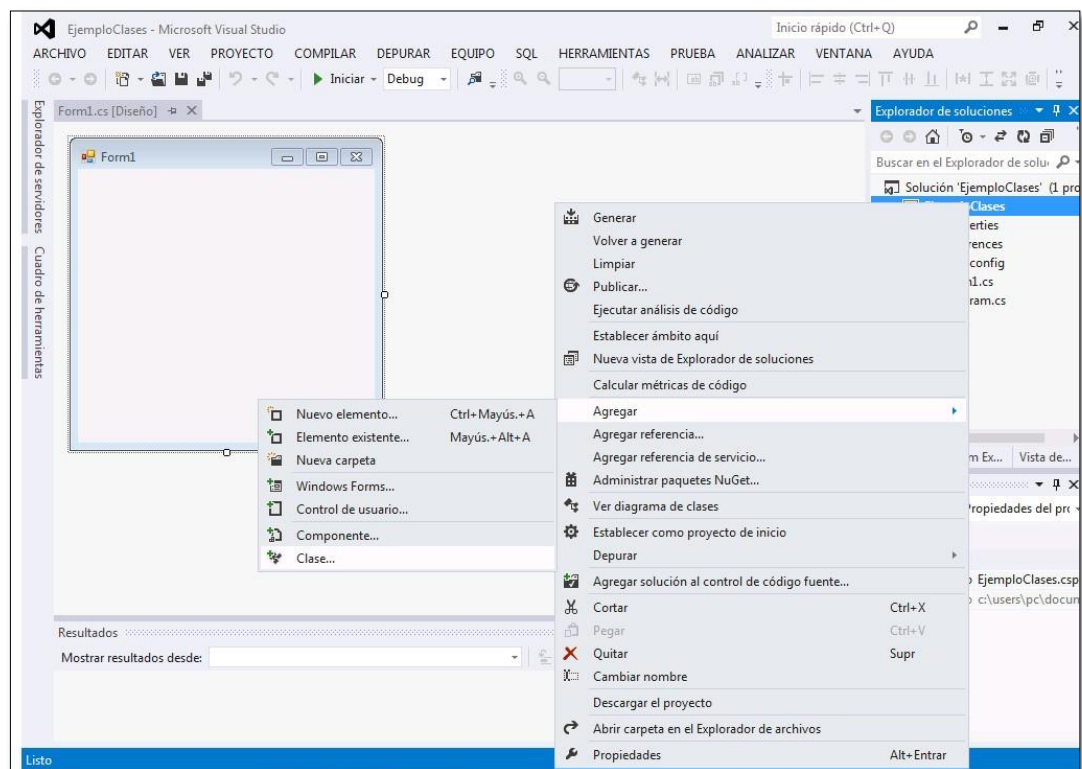
**Paso 1.** Crear un nuevo proyecto de tipo Windows form con C#, con el nombre de EjemploClases.



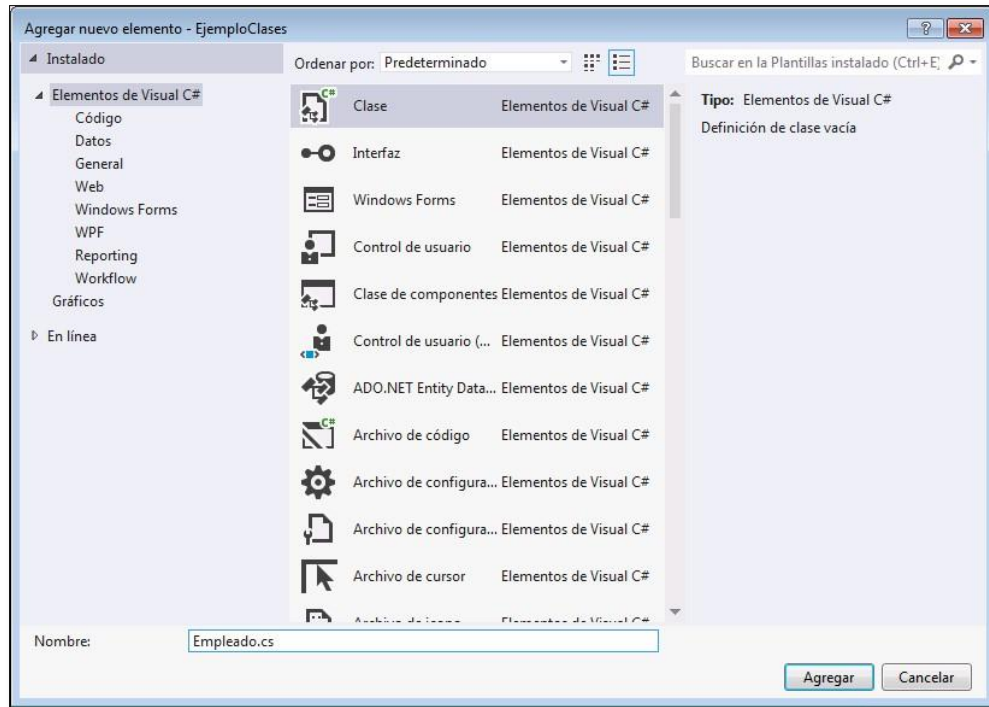
**Paso 2.** Se mostrará la siguiente imagen.



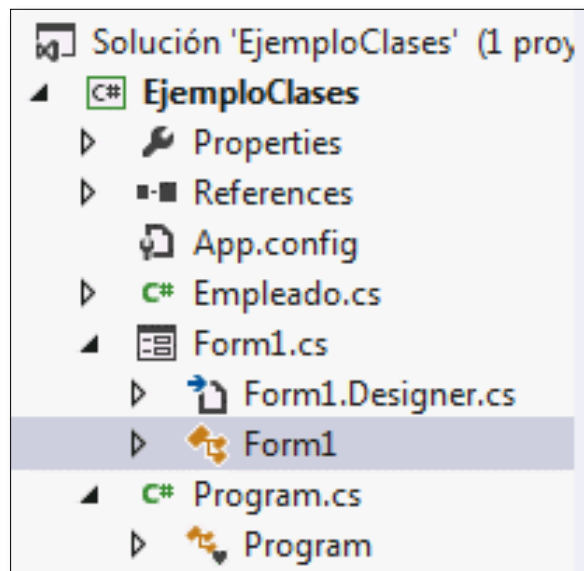
**Paso 3.** Se agregará una nueva clase, clic derecho sobre el nombre del proyecto, Agregar/Clase.



**Paso 4.** Se mostrará la siguiente pantalla, en la que se le colocará el nombre de Empleado a la clase.



**Paso 5.** El proyecto mostrará la siguiente estructura, en el explorador de soluciones.



**Paso 6.** Dar doble clic a la clase empleado, codificar lo siguiente:

```
EjemploClases.Empleado Empleado()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//Importaciones de bibliotecas utilizadas en la clase, que vienen por default al crear la clase

namespace EjemploClases
{
    //Autores: Claudia Rodríguez, Giovanni Tzec
    {
        //Nombre de la clase
        public class Empleado //Agregando la visibilidad publica, se hace visible en todo el proyecto
        {
            //Atributos de la clase
            private int codigoEmpleado;
            private String nombre; //Se declaran los atributos privados, para hacer posible el
            private int edad; // encapsulamiento en la POO
            private String cargo;
            private Double sueldoBase;

            //Constructor vacio, permite crear objetos sin inicializar sus atributos
            //Al tener 2 metodos con el mismo nombre se abre paso a la sobrecarga de métodos en una clase
            public Empleado()
            {
            }
        }
    }
}
```

```
EjemploClases.Empleado Empleado()
//Constructor con parametros, permite crear objetos inicializando parámetros
public Empleado(int codigo, String nom, int eda, String car, Double sueldo)
{
    codigoEmpleado = codigo;
    nombre = nom;
    edad = eda;
    cargo = car;
    sueldoBase = sueldo;
}
//Métodos de acceso (Get and Set)
//Método Set, permite asignar un valor a mi atributo privado
public void setCodigoEmpleado(int codigoEmpleado)
{
    this.codigoEmpleado = codigoEmpleado;
}
//Función Get, permite recuperar un valor de mi atributo privado
public int getCodigoEmpleado()
{
    return this.codigoEmpleado;
}
//Método Set, permite asignar un valor a mi atributo privado
public void setNombre(String nombre)
{
    this.nombre = nombre;
}
}
```

EjemploClases.Empleado Empleado()

```
public String getNombre()
{
    return this.nombre;
}

//Método Set, permite asignar un valor a mi atributo privado
public void setEdad(int edad)
{
    this.edad = edad;
}

//Función Get, permite recuperar un valor de mi atributo privado
public int getEdad()
{
    return this.edad;
}

//Método Set, permite asignar un valor a mi atributo privado
public void setCargo(String cargo)
{
    this.cargo = cargo;
}
```

Form1.cs Empleado.cs Form1.cs [Diseño]

EjemploClases.Empleado Empleado()

```
//Función Get, permite recuperar un valor de mi atributo privado
public String getCargo()
{
    return this.cargo;
}

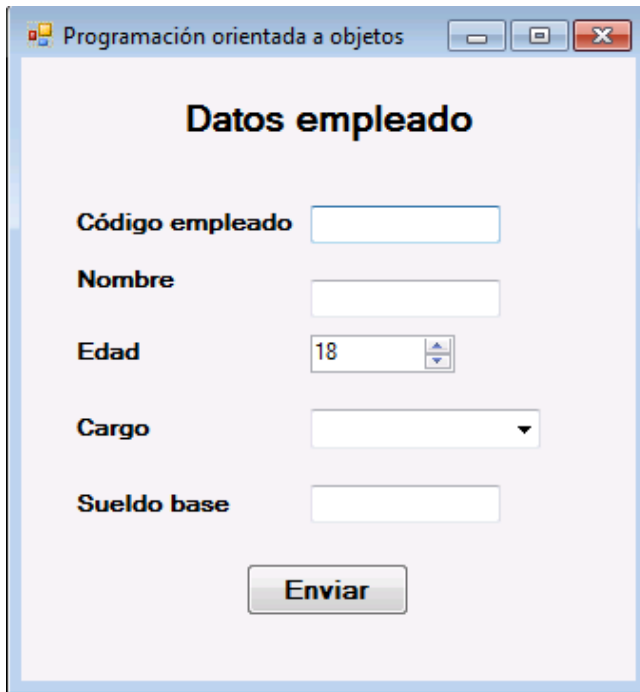
//Método Set, permite asignar un valor a mi atributo privado
public void setSueldoBase(Double sueldoBase)
{
    this.sueldoBase = sueldoBase;
}

//Función Get, permite recuperar un valor de mi atributo privado
public Double getSueldoBase()
{
    return this.sueldoBase;
}

}
```



**Paso 7.** Crear el siguiente diseño en el formulario:



Programación orientada a objetos

### Datos empleado

**Código empleado**

**Nombre**

**Edad**

**Cargo**

**Sueldo base**

**Enviar**

Nombre de los controles:

**txtCodigo** (Código de empleado)

**txtNombre** (Caja de texto)

**numEdad** (NumericUpDown)

**comboCargo** (ComboBox)

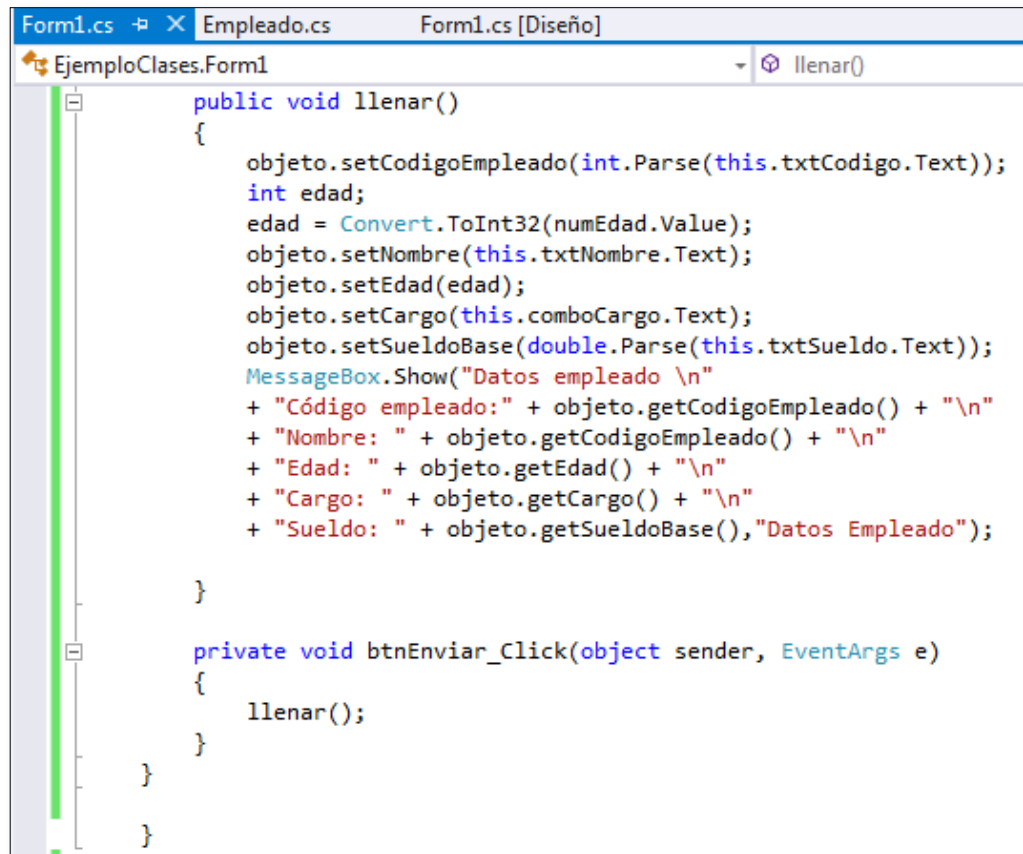
**txtSueldo** (Caja de texto)

**btnEnviar** (Button)

**Paso 8.** Dar doble clic en el botón del formulario, codificar lo siguiente:

```
Form1.cs  x Empleado.cs  Form1.cs [Diseño]
EjemploClases.Form1  llenar()
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//Autor: Claudia Rodríguez, Giovanni Tzec
//Importaciones de las bibliotecas que son usadas actualmente en el proyecto.
namespace EjemploClases
{
    //Espacio de nombre de la clase, adopta el nombre inicial del proyecto.
    public partial class Form1 : Form
    {
        //Constructor de la clase. A la vez permite inicializar todos los componentes
        public Form1()
        {
            InitializeComponent();
        }
        //Instanciando un objeto de la clase empleado
        Empleado objeto = new Empleado();

        //Al crear el método llenar se pretende darle un nivel alto de abstracción a la aplicación.
        //La finalidad es invocar el método llenar en el evento mouseClicked del botón.
    }
}
```



```
Form1.cs  X Empleado.cs  Form1.cs [Diseño]
EjemploClases.Form1  llenar()

public void llenar()
{
    objeto.setCodigoEmpleado(int.Parse(this.txtCodigo.Text));
    int edad;
    edad = Convert.ToInt32(numEdad.Value);
    objeto.setNombre(this.txtNombre.Text);
    objeto.setEdad(edad);
    objeto.setCargo(this.comboCargo.Text);
    objeto.setSueldoBase(double.Parse(this.txtSueldo.Text));
    MessageBox.Show("Datos empleado \n"
        + "Código empleado:" + objeto.getCodigoEmpleado() + "\n"
        + "Nombre: " + objeto.getNombre() + "\n"
        + "Edad: " + objeto.getEdad() + "\n"
        + "Cargo: " + objeto.getCargo() + "\n"
        + "Sueldo: " + objeto.getSueldoBase(), "Datos Empleado");
}

private void btnEnviar_Click(object sender, EventArgs e)
{
    llenar();
}
}
```

**Paso 9.** Luego de codificar todo el ejercicio, realizar las pruebas respectivas.

## TAREA

Desarrollar los siguientes ejercicios.

- 1- Desarrollar una representación gráfica de la asociación entre un cliente y un vendedor si se sabe que en esta asociación está implícita la clase venta y que a su vez esta se asocia con productos.
- 2- Determine las asociaciones que se pueden presentar cuando usted realiza un viaje en autobús, no olvide establecer roles y multiplicidad.
- 3- Establecer roles, multiplicidad en la asociación de una familia pequeña que cuenta con un padre, madre e hijo.
- 4- Represente en un diagrama el elemento derivado para el atributo IVA