Programación Orientada a Objetos 1

Contendido

Objeto - 1

- Un **objeto** es una agrupación de código, compuesta de propiedades y métodos, que pueden ser manipulados como una entidad independiente
- Las *variables o propiedades* definen los datos o información del objeto, permitiendo consultar o modificar su estado;
- Los *métodos* son las rutinas de código que definen su comportamiento.

Objeto - 2

- Un objeto es una pieza que se ocupa de desempeñar un trabajo concreto dentro de una estructura organizativa de nivel superior, formada por múltiples objetos, cada uno de los cuales ejerce la tarea particular para la que ha sido diseñado.

Clase

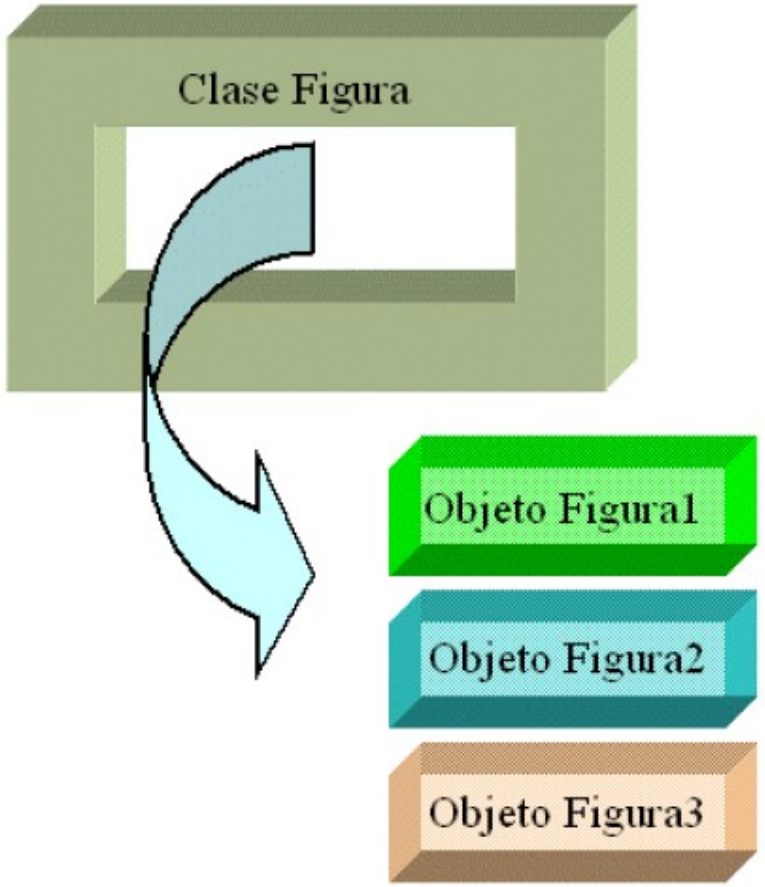
- Una **clase** es el conjunto de especificaciones o normas que definen cómo va a ser creado un objeto de un tipo determinado;
  - algo parecido a un manual de instrucciones conteniendo las indicaciones para crear el objeto.

Clase vs Objeto

- Los términos objeto y clase son utilizados en POO con gran profusión y en contextos muy similares,
  - Una clase constituye la representación abstracta de algo,
  - Un objeto constituye la representación concreta de lo que una clase define.
- La clase determina el conjunto de puntos clave que ha de cumplir un objeto para ser considerado perteneciente a dicha clase o categoría
  - no es obligatorio que dos objetos creados a partir de la misma clase sean exactamente iguales, basta con que cumplan las especificaciones clave de la clase.

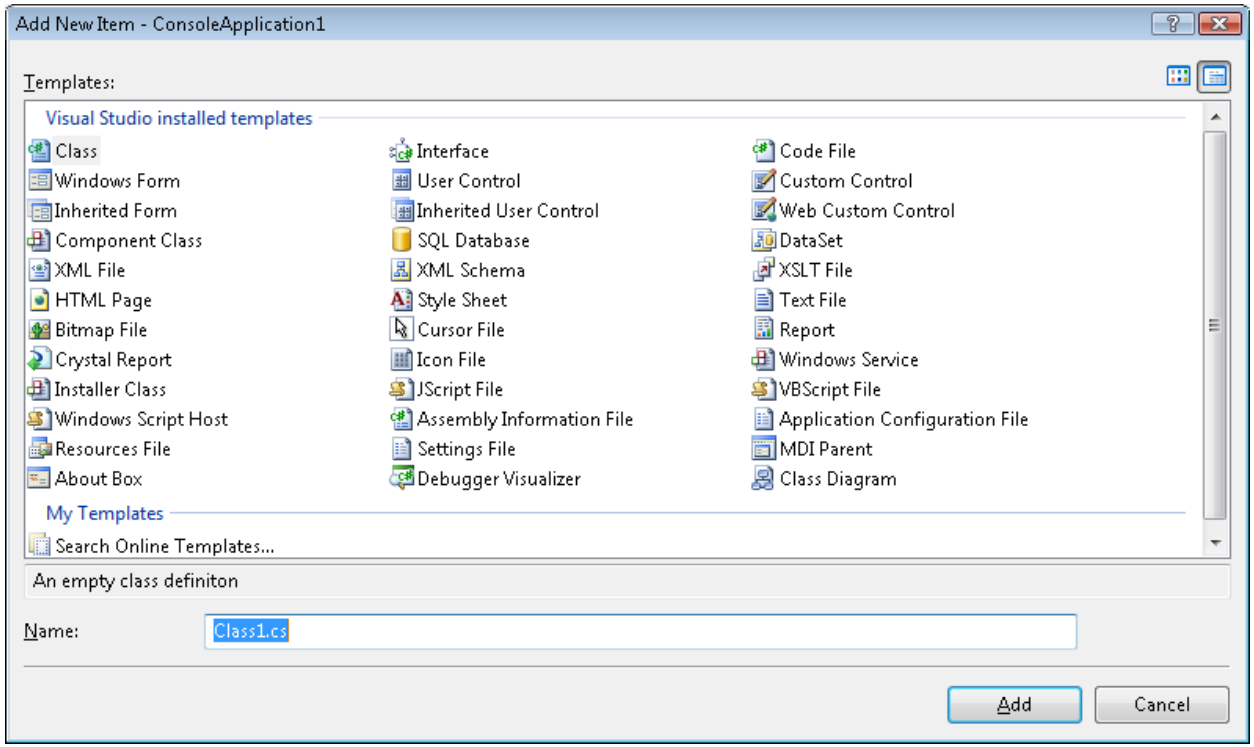
Instancias de una clase

- El proceso por el cuál se obtiene un objeto a partir de las especificaciones de una clase se conoce como instanciación de objetos.



Creación de Clases - 1

- Para crear una clase en C#
  - Project – Add Class



Creación de Clases - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class empleado
    {

    }
}
```


Instanciación de objetos - 1

- El modo más común de trabajar con una instancia de una clase, o lo que es lo mismo, con un objeto, pasa por asignar dicho objeto a una variable.
- Instanciaremos un objeto en el código utilizando la sintaxis de declaración de variables junto a la palabra clave **new**
- Existen dos formas de instanciar una clase
  - Sintaxis de una línea
    - Crear variable y reservar espacio en memoria
  - Sintaxis de dos partes
    - Crear la variable
    - Reservar espacio en memoria en otra parte del código

Instanciación de objetos - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Instanciación de una clase
            // y reservar su espacio en memoria
            empleado empleado01 = new empleado();

            // Instanciación de una clase y
            // posteriormente resarvar su espacio en memoria
            empleado empleado02;
            empleado02 = new empleado();
        }
    }
}
```


Código de clase y código cliente

- En función del lugar desde el que sea manipulado, debemos distinguir entre código de clase y código cliente.
- Código de clase.
  - Se trata del código que escribimos para crear nuestra clase, y que se encuentra en el bloque delimitado por las llaves { }.
- Código cliente.
  - Se trata del código que hace uso de la clase mediante la acción de crear o instanciar objetos a partir de la misma.
  - Aquí se englobaría todo el código que se encuentra fuera de la clase.

Miembros de la clase

- Los elementos de una clase que contienen sus datos y definen su comportamiento, es decir, las propiedades y métodos, reciben además nombre de *miembros de la clase*.
- Existen dos formas de almacenar los datos o información en una clase: a través de
  - Campos de clase
  - Propiedades de clase
- El uso de campos o propiedades para una clase es una cuestión de diseño, no pudiendo afirmar categóricamente que un tipo de almacenamiento de datos sea mejor que otro.

Creación de miembros de clase - 1

- Un campo de una clase no es otra cosa que una variable, generalmente con ámbito público, accesible desde el exterior de la clase.
- Para manipular un campo desde código cliente, debemos instanciar un objeto, a continuación de la variable que lo contiene situar un punto (.), y finalmente el nombre del campo a manipular.
- Este modo de operación es común para todos los miembros de clases, tanto creadas por el programador, como pertenecientes a la propia plataforma .NET Framework.

Creación de miembros de clase - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class empleado
    {
        public string nombre;
        public int edad;
    }
}
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            empleado empleado01 = new empleado();
            empleado01.nombre = "Juan Perez";
            empleado01.edad = 12;
            Console.WriteLine("El nombre es {0} y la edad {1}", empleado01.nombre,
empleado01.edad);
            Console.ReadLine();
        }
    }
}
```


Controlando el acceso a los miembros

- Los métodos públicos representan la vista del cliente es decir *los servicios* que la clase proporciona
- Los métodos deberían realizar solo una tarea
  - Si un método necesita realizar otra tarea para calcular su resultado, este debería usar un método auxiliar
  - El código cliente no debería tener acceso a los métodos auxiliares y estos deberían ser declarados **private**
- Se deben usar las propiedades para proporcionar acceso seguro a los datos
  - Los miembros de datos deben ser declarados como **private**, con propiedades públicas que permiten un acceso seguro a ellos
- Propiedades
  - **get** : habilita al cliente para leer los datos
  - **set** : habilita al cliente para modificar los datos

Creación de propiedades - 1

- Una propiedad en la clase se define, por norma general, mediante dos elementos: una variable de propiedad y un procedimiento de propiedad.
- La variable de propiedad, es una variable con ámbito privado a nivel de la clase, que se encarga de guardar el valor de la propiedad.
- Por su parte el procedimiento de propiedad o es el encargado de actuar de puente entre el código cliente y la variable de propiedad, realizando las operaciones de acceso y asignación de valores a dicha variable.

Creación de propiedades - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class empleado
    {
        private string nombre;
        private int edad;
        public string Nombre
        {
            get { return nombre; }
            set { nombre = value; }
        }
        public int Edad
        {
            get { return edad; }
            set { edad = value; }
        }

    }
}
```


Creación de propiedades - 3

- A la hora de manipular una propiedad desde el código cliente no es diferente a manipular un campo, la ventaja al usar propiedades es que el código queda perfectamente encapsulado.

Propiedades solo lectura / solo escritura - 1

- Una propiedad de solo lectura solo tiene el bloque **get** mientras que una propiedad de solo escritura solo tiene el bloque **set**.
- Igualmente obtendremos un error del compilador, si en el código cliente intentamos asignar un valor a una propiedad de solo lectura, u obtener un valor de una propiedad de solo escritura.

Propiedades solo lectura / solo escritura - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Empleado
    {
        private string nombre;
        private int edad;
        private string ctaBancaria;
        private string entidad;
        public string Nombre
        {
            get { return nombre; }
            set { nombre = value; }
        }
        public int Edad
        {
            get { return edad; }
            set { edad = value; }
        }
        public string CuentaBancaria
        {
            set {
                switch(value.Substring(0,4))
                {
                    case "1111":
                        entidad = "Banco Universal";
                        break ;
                    case "2222":
                        entidad = "Banco General";
                        break;
                    case "3333":
                        entidad = "Caja Metropolitana";
                        break;
                    default:
                        entidad = "entidad sin catalogar";
                        break;
                }
            }

        public string EntidadBancaria
        {
            get { return entidad; }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Empleado empleado01 = new Empleado();
            empleado01.Nombre = "Juan Perez";
            empleado01.Edad = 35;

            // Asignando valor propiedad de solo escritura
            empleado01.CuentaBancaria = "2222-56-7779995555";

            // Obteniendo valor de propiedad solo lectura
            Console.WriteLine("La entidad del empleado es {0}
es {1}", empleado01.Nombre, empleado01.EntidadBancaria);
            Console.ReadLine();
        }
    }
}
```


Métodos - 1

- Un método es una entidad ejecutable de código
- Un método incluye
  - Un modificador de acceso opcional, se asume **private**
  - El tipo de retorno para el método,
    - un método puede regresar cualquier tipo de dato, incluyendo los definidos por el usuario
    - Los métodos que no regresan ningún valor deben ser declarados como **void**
  - El nombre del método
  - Entre parentesis, la lista de parámetros aceptado por el método
    - Si hay más de un parámetro van separados por coma

Métodos - 2

- Entre llaves {}, el bloque de código que define la lógica del método
  - Si el método tiene un valor de retorno, todas las rutas dentro del bloque de código deben regresar un valor
  - Para regresar un valor, usa la palabra clave ***return*** seguida de la expresión que resulta en el mismo tipo de dato del tipo de dato de retorno del método
  - Si el método no regresa valores (tipo de retorno void) se puede usar la sentencia return para salir del método
  - Cualquier variable definida dentro del método es local al método
- Todos los métodos tienen que ser declarados dentro de una clase

Métodos - 3

- Para usarlo, llamarlo desde cualquier parte del programa
  - Pasar parámetros si es necesario
- Los parámetros pueden ser pasados a los métodos en dos formas
- Por valor
  - Se pasa una copia del contenido de la variable al método
  - Los cambios al valor pasado no afectan la variable original
  - Es el método de paso de parámetros por defecto en C#
- Por referencia
  - Permite que los cambios a la variable sean persistentes fuera del método
  - El método llamado modifica la variable original así como regresa el nuevo valor de la variable
  - Para pasar por referencia, utiliza la palabra clave ***ref*** cuando se declare los parámetros del método.

Métodos - 4

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Empleado
    {
        private string nombre;
        private int edad;
        private string ctaBancaria;
        private string entidad;
     ...
          ...
        public void CalcularVacaciones(DateTime inicio , double  dias) {
           // en este método calculamos el periodo de vacaciones del empleado
            //mostrando los resultados en consola
            DateTime final;

            final = inicio.AddDays(dias);

            Console.WriteLine("Empleado {0} ", nombre);

            Console.WriteLine("Tiene vacaciones desde {0} hasta {1}",
            inicio.ToString("dd/MMM/yy"),  final.ToString("d/MMMM/yyyy"));
        }
    }
}
```


Constructores - 1

- Las instancias de una clase son inicializadas por los constructores
- Los constructores inicializan las variables de instancia de los objetos
- Se utiliza la sobrecarga de constructores para proporcionar diferentes formas de inicializar objetos de clase
- Incluso si el constructor no hace nada explicitamente, todos los miembros de datos son inicializados
  - Tipos numéricos primitivos son puestos 0
  - Los tipos booleanos son puestos a false
  - Los tipos de referencia son puestos a null
- Si la clase no tiene constructor, se proporciona un constructor por default
  - No tiene código y no toma parametros

Constructores - 1

- Uso de la palabra reservada **this**
  - Cada objeto puede referenciarse así mismo usando la palabra reservada **this**
  - Frecuentemente usada para distinguir entre variables de método y las variables de instancia de un objeto.

Constructores - 2

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Empleado
    {
        private string nombre;
        private int edad;
        private string ctaBancaria;
        private string entidad;
        public Empleado()
        {
            estableceValores("",0);
        }
        public Empleado(string valnombre)
        {
            estableceValores(valnombre, 0);
        }
        public Empleado(string valnombre, int valedad)
        {
            estableceValores(valnombre, valedad);
        }
        private void estableceValores(string valnombre, int valedad)
        {
            this.nombre = valnombre;
            this.edad = (valedad >= 0 && valedad <= 80) ? valedad : 0;
        }
... ... ...
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Empleado empleado01 = new Empleado();
            Empleado empleado02 = new Empleado("Juan");
            Empleado empleado03 = new Empleado("Maria", 25);
            Console.ReadLine();
        }
    }
}
```