FUNDAMENTOS DE PROGRAMACIÓN

Moisés Valenzuela Gutiérrez

publicatuslibros.com
cinco años en la red

FUNDAMENTOS DE PROGRAMACIÓN

Moisés Valenzuela Gutiérrez

INTRODUCCION

Este libro está dirigido a estudiantes que cursen Ciclos Formativos de Grado Superior en Administración de Sistemas Informáticos y Desarrollo de Aplicaciones Informáticas. Concretamente está orientado para la asignatura de primer curso Fundamentos de Programación , habiendo parte de la asignatura que se orienta hacia el desarrollo de la programación orientada a objetos.

En este libro se detallan las características básicas de la programación orientada a objetos utilizando para ello el lenguaje de programación y orientado a objetos Visual C# que se encuentra dentro de paquete .NET perteneciente a la compañía Microsoft .

Se detalla la definición de clase y objeto detallado mediante ejemplos , así como todas las sentencias iterativas y simples necesarias para la implementación de programas. También se detalla la creación de constructores y por ultimo una propiedad propia de los lenguajes orientados a objetos como es la herencia.

INDICE.

TEMA 1. LENGUAJE C#.

1.1.- PROGRAMACION ORIENTADA A OBJETOS.

Clases y objetos.

Vamos a entrar de lleno para comprender las clases, los objetos y en que se diferencian.

No son lo mismo las clases y los objetos. Para que pueda haber un objeto debe existir previamente una clase, pero no al revés.

Ejemplo:

Tenemos un coche, todos ellos tienen una serie de características comunes: tienen un motor, ruedas, un volante, chasis.....; todos funcionan de un modo parecido para acelerar, frenar, meter las marchas...; sin embargo, cada uno de ellos es diferente de los demás, puesto que cada uno es de una marca, modelo, color, numero de bastidor, propiedades que lo diferencian de los demás, aunque una o varias de ellas puedan coincidir en varios coches. Diríamos entonces que todos los coches están basados en una plantilla, o un tipo de objeto, es decir, pertenecen todos a la misma clase: la **clase coche.** Sin embargo, cada uno de los coches es un objeto de esa clase; todos comparten la interfaz, pero no tienen porque compartir los datos (marca, modelo, color, etc). Se dice entonces que cada uno de los objetos es una instancia de la clase a la que pertenece, es decir, un objeto. Veamos como seria esto en C#. El diseño de la clase Coche sería algo parecido a esto (aunque más ampliado):

```
class  Coche
{
  public  Coche(string marca, string modelo, string color, string numbastidor)
  {
    this.Marca=marca;
```

```
 this.Modelo=modelo;
 this.Color=color;
 this.NumBastidor=numbastidor;
}
public double Velocidad
{
 get
    {
      return this.velocidad;
     }
 }
 public double velocidad=0;
 public string Marca;
 public string Modelo;
 public string Color;

public void Acelerar (double cantidad)
{
  // Aquí se le dice al motor que aumente las revoluciones pertinentes, y .....
  Console.WriteLine ("Incrementando la velocidad en {0} km/h", cantidad);
  this.velocidad = cantidad;
}

public void Girar (double cantidad)
{
  //Aquí iría el código para girar
  Console.WriteLine("Girando el coche {0} grados ", cantidad);
```

```
  }


}
```

Veamos una clase con un método Main para ver como se comportaría esta clase:

```
class EjemploCocheApp
{
  static void Main()
    {
        Coche MiCoche=new Coche("Peugeot", "306", "Azul","15343534");


        Console.WriteLine("Los datos de mi coche son:");
        Console.WriteLine("Marca: {0}", MiCoche.Marca);
        Console.WriteLine("Modelo: {0}", MiCoche.Modelo);
        Console.WriteLine("Color: {0}", MiCoche.Color);
        Console.WriteLine("Numero de bastidor: {0}", MiCoche.Numbastidor);
        MiCoche.Acelerar(100);
        Console.WriteLine("La velocidad actual es de {0} km/h",
MiCoche.Velocidad);
        MiCoche.Girar(45);
  }
}
```

El resultado que aparecería en la consola al ejecutar este programa seria esta:

Los datos de mi coche son:

Marca: Peugeot

Modelo: 306

Color: Azul

Numero de bastidor: 15343534

Incrementado la velocidad en 100 km/h

La velocidad actual es de 100 km/h

Girando el coche 45 grados.

No te preocupes por no entender el código, ya lo iremos viendo largo y tendido. Nos fijamos que en la clase es donde se definen los datos y se programan todas las acciones que han de manejar los objetos de esta clase. Los datos son Velocidad, Marca, Modelo, Color y NumBastidor, y los métodos son Acelerar y Girar. Sin embargo, el objeto, MiCoche (creado en la primera línea del método Main) no define absolutamente nada. Simplemente usa la interfaz diseñada en la clase (la interfaz de una clase es el conjunto de métodos y propiedades que esta ofrece para su manejo). Por lo tanto, Coche es la clase y MiCoche un objeto de esta clase.

1.2.- BASES SINTACTICAS DE C#.

Aplicación básica ¡Hola a todos!

Como primer contacto con el lenguaje, nada mejor que el típico programa de iniciación "¡Hola a todos!" que lo único que hace al ejecutarse es mostrar por pantalla el mensaje ¡Hola a todos! Su código es:

```
1:    class HolaMundo
2:    {
3:        static void Main()
4:        {
5:            System.Console.WriteLine("¡Hola a todos!");
6:        }
7:    }
```

Todo el código escrito en C# se ha de escribir dentro de una definición de clase, y lo que en la línea **1:** se dice es que se va a definir una clase (**class**) de nombre HolaMundo cuya definición estará comprendida entre la llave de apertura de la línea **2:** y su correspondiente llave de cierre en la línea línea **7:**

Dentro de la definición de la clase (línea **3:**) se define un método de nombre Main cuyo código es el indicado entre la llave de apertura de la línea **4:** y su respectiva llave de cierre (línea **6:**) Un método no es más que un conjunto de instrucciones a las que se les asocia un nombre, de modo que para posteriormente ejecutarlas baste referenciarlas por su nombre en vez de tener que rescribirlas.

La partícula que antecede al nombre del método indica cuál es el tipo de valor que se devuelve tras la ejecución del método, y en este caso es **void** que significa que no se devuelve nada. Por su parte, los paréntesis colocados tras el nombre del método indican cuáles son los parámetros que éste toma, y el que estén vacíos significa que el método no toma ninguno. Los parámetros de un método permiten modificar el resultado de su ejecución en función de los valores que se les dé en cada llamada.

La palabra **static** que antecede a la declaración del tipo de valor devuelto es un **modificador** del significado de la declaración de método que indica que el método está asociado a la clase dentro de la que se define y no a los objetos que se creen a partir de ella. Main() es lo que se denomina el **punto de entrada** de la aplicación, que no es más que el método por el que comenzará su ejecución. Necesita del modificador **static** para evitar que para llamarlo haya que crear algún objeto de la clase donde se haya definido.

Finalmente, la línea **5:** contiene la instrucción con el código a ejecutar, que lo que se hace es solicitar la ejecución del método **WriteLine()** de la clase **Console** definida en el espacio de nombres **System** pasándole como parámetro la cadena de texto con el contenido ¡Hola Mundo! Nótese que las cadenas de textos son secuencias de caracteres delimitadas por comillas dobles aunque dichas comillas no forman parte de la cadena. Por su parte, un espacio de nombres puede considerarse que es para las clases algo similar a lo que un directorio es para los ficheros: una forma de agruparlas.

1.3.- TIPOS.

Los tipos en C# se clasifican en: tipos valor y tipo referencia. Una variable de un tipo valor almacena directamente un valor (datos en general), mientras que una variable de un referencia lo que permite almacenar es una referencia a un objeto (posición de memoria donde está el objeto). Por ejemplo:

```
int suma = 0;            //Suma almacena un entero.
string cadena =españa;   //Cadena permitirá almacenar una referencia
                         //  a un objeto string
```

.

1.4.- DECLARACION DE VARIABLES.

Una variable representa un espacio de memoria para almacenar un valor de un determinado tipo. La sintaxis para declarar una variable es la siguiente:

```
tipo identificador[ ,identificador]…
```

En el ejemplo se declaran dos variables de tipo int, una variable de tipo string y dos variables de tipo string:

```
void variables()
{
   int contador = 0;
   string Nombre =" ";
   int dia = 20;
   Apellidos = "Ceballos";
 }
```

Ejemplo1:

Veamos el siguiente ejercicio. Guarda el proyecto como **OperacionesAritmeticas**.

El programa declara tres variables y a dos de ellas le asigna un valor. A continuación se realiza la suma de esos valores y se escriben los datos y el resultado. Copia el programa y lo compilas, ejecútalo a continuación y comprueba los resultados.

```
class Program
    {
        /* Operaciones Aritmeticas*/
        static void Main(string[] args)
        {
            int dato1, dato2, resultado;
            dato1 = 20;
            dato2 = 10;
            //Suma
            resultado = dato1 + dato2;
            System.Console.WriteLine("{0}+{1}={2}", dato2, dato2,
resultado);
            //Resta
            resultado = dato1 - dato2;
            System.Console.WriteLine("{0}-{1}={2}", dato2, dato2,
resultado);
            //Producto
            resultado = dato1 * dato2;
            System.Console.WriteLine("{0}*{1}={2}", dato2, dato2,
resultado);
            //Cociente
            resultado = dato1 / dato2;
            System.Console.WriteLine("{0}/{1}={2}", dato2, dato2,
resultado);


        }
    }
```

1.5.- SENTENCIA IF....ELSE.

La sentencia **if ...else** ejecuta una sentencia, simple una o ninguna vez dependiendo del valor de un expresión. Su sintaxis es la siguiente:

If (condición)

Sentencia1

else

sentencia2;

Ejemplo 1:

Crea un programa que introduzca tres índices de polución de una ciudad en distintos lugares de dicha ciudad y que el programa averigüé la media de dichos índices tomados y que si supera el índice máxima de 50, nos muestre por pantalla que nos encontramos ante una situación de riesgo y si no que nos encontramos en situación segura.

```
class Polucion
    {
        static void Main(string[] args)
        {
            const int corte = 50;

            int indice1, indice2, indice3, media;

            Console.WriteLine("Introduce el indice de polucion 1:");
            indice1 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Introduce el indice de polucion 2:");
            indice2 = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Introduce el indice de polucion 3:");
            indice3 = Int32.Parse(Console.ReadLine());
            media = (indice1 + indice2 + indice3) / 3;
            Console.WriteLine("El indice de polucion medio es:",
+media);
            if (media > corte)
                Console.WriteLine("Situacion de alerta");
            else
```

```
                Console.WriteLine("Situacion segura");
            }
        }
```

1.6.- SENTENCIA SWITCH.

Permite ejecutar una de varias acciones, en función del valor de una expresión. Es una sentencia especial para decisiones múltiples. La sintaxis para utilizar esta sentencia es.

switch (expresión)

{

case expresión-constante 1:

[sentencia 1:]

[case expresión-constante 2:]

[sentencia 2:]

[case expresión-constante 3:]

[sentencia 3:]

[default:]

[sentencia n:]

}

Donde expresión es una expresión de tipo entero, enumerado o string y expresión-constante es una constante del mismo tipo que expresión o de un tipo que se pueda convertir implícitamente al tipo de expresión; y sentencia es una sentencia simple o compuesta. En el caso de que se trate de una sentencia compuesta, no hace falta incluir las sentencias simples que la forman entre {}

El siguiente ejemplo da como resultado los días correspondientes al mes:

```
switch (mes)


case 1: case 2: case 5: case 7: case 8: case 10: case 12:
 dias = 31;
 break;
case 4: case 6: case 8: case 11:
 dias =30;
case 2:
 // ¿Es el año bisiesto?
 if ((año%4 ==0) && (año %100 i=0) || (año%400==0))
    días = 29;
 else
    dias = 28;
    break;
default:
 System.Console.WriteLine( "\n El mes no es valido");
 break;
}
```

ACTIVIDADES.

Ejercicio1:

Modicfica el ejercicio anterior modificandolo para que ahora realice las operaciones de sumar, restar y multiplicar con tres datos: dato1, dato2, dato3. Guardalo como **OperacionesAritmeticas2**.

Ejercicio 2:

Programa el ejercicio anterior, introduciendo un mes y un año. Apareciendo por pantalla el número de días q tiene dicho mes.

Ejercicio 3:

Realiza el ejercicio 7 (temperaturas y deporte) con la sentencia de control ***Switch case***.

Nota: Utiliza temperaturas fijas para la realización del ejercicio.

1.7.- SENTENCIA WHILE.

La sentencia While ejecuta una sentencia, simple o compuesta, cero o mas veces, dependiendo del valor de una expresión bolean. Su expresión es:

while (condición)

sentencia;

donde **condición** es cualquier expresión booleana y **sentencia** es una sentencia simple o compuesta.

La ejecución de la sentencia while sucede así:

1.- Se evalúa la condición y se obtiene un resultado verdadero o falso.

2.- Si el resultado es falso (false), la sentencia no se ejecuta y se pasa el control a la siguiente en el programa.

3.- Si el resultado de la evaluación es verdadero (true) , se ejecuta la sentencia y el proceso descrito se repite desde el punto 1.

Ejemplo 1:

Realizar un programa que a través de un menú permita realizar las operaciones de sumar, restar, multiplicar, dividir y salir. Las operaciones constaran solamente de dos operandos.

```
class sumarmultiplicar
    {
        static void Main(string[] args)
        {
            int numero1 = 0;
            int numero2 = 0;
            int resultado = 0;
```

```
            int opcion = 0;

            while (opcion != 5)
            {

                Console.WriteLine("Introduce primer numero: ");
                numero1 = Int32.Parse(Console.ReadLine());
                Console.WriteLine("Introduce segundo numero: ");
                numero2 = Int32.Parse(Console.ReadLine());
                Console.Clear();
                Console.WriteLine("==========================");
                Console.WriteLine("1.-Sumar");
                Console.WriteLine("2.-Restar");
                Console.WriteLine("3.-Multiplicar");
                Console.WriteLine("4.-Dividir");
                Console.WriteLine("5.-Salir");
                Console.WriteLine("Elige opcion");
                opcion = Int32.Parse(Console.ReadLine());

                if (opcion == 1)
                {
                    resultado = numero1 + numero2;
                    Console.WriteLine("El resultado de la operacion
es: " + resultado);
                }
                else if (opcion == 2)
                {
                    resultado = numero1 - numero2;
                    Console.WriteLine("El resultado de la operacion
es: " + resultado);
                }
                else if (opcion == 3)
                {
```

```
                    resultado = numero1 * numero2;

                    Console.WriteLine("El resultado de la operacion
es: " + resultado);

                }

                else if (opcion == 4)

                {

                    resultado = numero1 / numero2;

                    Console.WriteLine("El resultado de la operacion
es: " + resultado);

                }

                else if (opcion == 5)

                    Console.WriteLine("Usted ha elegido la opcion
salir. !Hasta otra!");

                else

                    Console.WriteLine("Ha escogido una opcion
equivocada");




            }

        }

    }
```

1.6.- SENTENCIA DO….WHILE.

La sentencia **do...while** ejecuta una sentencia, simple o compuesta, una o mas veces dependiendo del valor de un expresión. Su sintaxis es la siguiente:

do

  sentencia;

while (condición);

donde condición es cualquier expresión booleana y sentencia es una sentencia simple o compuesta. Observe que la sentencia **do...while** finaliza con un punto y coma.

Ejemplo 1:

Si quiere averiguar tu numero de Tarot, suma los números de tu fecha de nacimiento y a continuación redúcelos a un único digito; por ejemplo si tu fecha de nacimiento es 17 de octubre de 1980 los cálculos a realizar serian:

17+10+1970=1+9+9+7=26 -> 2+6=8

lo que quiere decir que tu numero de tarot es el 8.

Realiza un programa que pida una fecha día, mes y año (son enteros) y de cómo resultado el numero de tarot. El programa verificara si la fecha es correcta, esto es, los valores están dentro de los rangos permitidos.

```
class tarot
{
    static void Main(string[] args)
    {
        int dia = 0;
        int mes = 0;
        int año = 0;
        int suma = 0;
        int div = 0;
        do
        {
            Console.WriteLine("Introduce dia: ");
            dia = Int32.Parse(Console.ReadLine());
        }
        while (dia < 1 || dia > 31);
```

```
            do
            {
                Console.WriteLine("Introduce mes: ");
                mes = Int32.Parse(Console.ReadLine());
            }
            while (mes < 1 || mes > 12);
            do
            {
                Console.WriteLine("Introduce año: ");
                año = Int32.Parse(Console.ReadLine());
            }
            while (año < 1900 || año > 2007);

            suma = dia + mes + año;

            do
            {
                for (int i = 3; i >= 0; i--)
                {
                    div += suma / (int)Math.Pow(10, i);
                    suma = suma % (int)Math.Pow(10, i);
                }
                suma = div;
                div = 0;
            }
            while (suma > 9);

            Console.WriteLine("Su numero de Tarot es: " + suma);

        }
    }
```

1.7.- SENTENCIA FOR.

Sentencia for

La sentencia **for** permite ejecutar una sentencia simple o compuesta, repetidamente un numero de veces conocido.

Por ejemplo la sentencia **for** siguiente imprime los números del 1 al 50.

```
int i;
for (i=1; i<=50; i++)
 System.Console.Write (i+ " ");
```

1.8.- MATRICES.

Una matriz es un conjunto de elementos seguidos, todos del mismo tipo que comparten un nombre común.

La declaración de una matriz de una dimensión se hace de la forma siguiente:

```
tipo[] nombre;
```

Ejemplos:

```
int[] m;
float[] temperatura;
```

Para crearla seria, mediante el ejemplo siguiente:

```
nombre =new tipo[tamaño];
```

donde nombre es el nombre de la matriz previamente declarada; tipo es el tipo de los elementos de la matriz, y tamaño es una expresión entera positiva .

Para iniciarla seria de la siguiente forma:

```
float[] temperatura=new float[5]  (10.2,12.3,3.4,14.5,16.7);
```

Para accede a los elementos de una matriz:

int[] m=new int[100]

int k=0,a=0;

a=m[1]+m[99];

Matrices multidimensionales

Pueden ser de varias dimensiones.

Ejemplo 1:

int[,] m=new int[2,3];

EJEMPLOS RESUELTOS.

Ejemplo 1:

Realiza ahora un programa que lea la nota media obtenida por cada alumno de un determinado curso, las almacene en una matriz y de cómo resultado la nota media del curso.

```
class Program
    {
        static void Main(string[] args)
        {
            int nAlumnos;

            do
            {
                Console.WriteLine("Numero de alumnos: ");
                nAlumnos = Int32.Parse(Console.ReadLine());

            }
            while (nAlumnos < 1);

            float[] nota = new float[nAlumnos]; //creo la matriz nota.
            int i = 0; //subindice
            float suma = 0;  //Suma total de las notas medias

            Console.WriteLine("Introducir las notas medias del curso:
");
            for (i=0;i<nota.Length;i++)
            {
                Console.WriteLine("Nota media del alumno:" +
(i+1)+":");
                nota[i]=Int32.Parse(Console.ReadLine());
            }

            //Sumar las notas medias
            for (i=0;i<nota.Length;i++)
```

```
                suma+=nota[i];

            //Visualizar la nota media del curso.

            Console.WriteLine("\n\nNota media del curso: "+
suma/nAlumnos);
```

Ejemplo 2:

Un estudio demográfico del área metropolitana de una determinada ciudad la divide en tres regiones (urbana, semiurbana y rural), y han publicado la tabla siguiente, mostrando la migración anual de una región a otra (los numeros representan porcentajes):

	Urbana	Semiurbana	Rural
Urbana	1,1	0,3	0,7
Semiurbana	0,1	1,2	0,3
Rural	0,2	0,6	1,3

Por ejemplo el 0,3 por ciento de los urbanos emigran cada año a las zonas residenciales del exterior de la ciudad (es decir, a la zona semiurbana). Los elementos de la diagonal principal representan el crecimiento interno de cada zona. Escribe un programa que, utilizando un array bidimensional cuyos índices pertenezcan a un tipo enumerado, determine la población de cada región después de 10,20,30,40 y 50 años. Supón que las poblaciones actuales de las áreas urbanas, semiurbanas y rurales son 2,1 millones, 1,4 millones y 0,9 millones respectivamente.

```
namespace demografia

{

    class Program

    {

        static void Main(string[] args)

        {

            //ESTUDIO DEMOGRAFICO SOBRE LOS MOVIMIENTOS DE POBLACION
EN UN AREA
```

```
            //METROPOLITANA Y DURANTE UNA DETERMINADA EPOCA.

            int opcion = 0; //Se utiliza para el menu principal.
            int opcion1=0; //Para los submenus(años).
            double poblacionurbana=2100000;
            double poblacionsemiurbana=1400000;
            double poblacionrural=900000;
            double poblacion=0;
            double porcentaje=0;

            //Array bidimensional para introducir los porcentajes.

            double[,] porciento = new double[3, 3];

            //Obviamos los movimientos internos que no suponen
cambios.

            porciento[0, 1] = 0.3;
            porciento[0, 2] = 0.7;
            porciento[1, 0] = 0.1;
            porciento[1, 2] = 0.3;
            porciento[2, 0] = 0.2;
            porciento[2, 1] = 0.6;


            //Menu para escoger el area donde queremos hacer el
estudio.
            do
            {
                Console.WriteLine("======================");
                Console.WriteLine("1.- Area Urbana.");
                Console.WriteLine("2.- Area SemiUrbana.");
                Console.WriteLine("3.- Area Rural.");
```

```
                Console.WriteLine("4.- Salir del programa.");
                Console.WriteLine("Elige la opcion para realizar el
estudio: ");
                Console.WriteLine("======================");
                opcion = Int32.Parse(Console.ReadLine());
                Console.Clear();
                switch (opcion)
                {
                    case 1:
                        {
                            do
                            {
                                Console.WriteLine("Elige año para
realizar estudio.");
                                Console.WriteLine("1.-Poblacion en año
2017.");
                                Console.WriteLine("2.-Poblacion en año
2027.");
                                Console.WriteLine("3.-Poblacion en año
2037.");
                                Console.WriteLine("4.-Poblacion en año
2047.");
                                Console.WriteLine("5.-Poblacion en año
2057.");
                                Console.WriteLine("6.-Terminar.");
                                Console.WriteLine("Elige opcion:");
                                opcion1 =
Int32.Parse(Console.ReadLine());

                                switch (opcion1)
                                {
                                    case 1:
                                        {
                                            porcentaje = porciento[0,
1] + porciento[0, 2] - porciento[1, 0] - porciento[2, 0];
```

```
                                        poblacion =
(poblacionurbana / 100) * porcentaje;

                                        poblacionurbana =
poblacionurbana - poblacion;

                                        Console.Clear();

Console.WriteLine("Poblacion en 2017:" + poblacionurbana + "
habitantes.");


                                        break;

                                    }

                                case 2:

                                    {

                                        poblacion = poblacion * 2;

                                        poblacionurbana =
poblacionurbana - poblacion;

                                        Console.Clear();

Console.WriteLine("Poblacion en 2027: " + poblacionurbana + "
habitantes.");


                                        break;

                                    }

                                case 3:

                                    {

                                        poblacion = poblacion * 3;

                                        poblacionurbana =
poblacionurbana - poblacion;

                                        Console.Clear();

Console.WriteLine("Poblacion en 2037: " + poblacionurbana + "
habitantes.");


                                        break;

                                    }

                                case 4:
```

```
                                        {
                                            poblacion = poblacion * 4;
                                            poblacionurbana =
poblacionurbana - poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2047: " + poblacionurbana + "
habitantes.");
                                            break;
                                        }
                                    case 5:
                                        {
                                            poblacion = poblacion * 5;
                                            poblacionurbana =
poblacionurbana - poblacion;
                                            Console.WriteLine();

Console.WriteLine("Poblacion en 2037: " + poblacionurbana + "
habitantes.");
                                            Console.Clear();
                                            break;
                                        }
                                    case 6:
                                        {
                                            Console.Clear();
                                            Console.WriteLine("Termina
el estudio zona urbana");

                                            break;
                                        }
                                    default:
                                        {
                                            Console.Clear();
                                            Console.WriteLine("Error,
opcion introducida incorrecta.");
```

```
                                    break;
                                }
                            }
                        } while (opcion1 != 6);
                        Console.Clear();
                        break;

                    }

                case 2:
                    {
                        do
                        {
                            Console.WriteLine("Elige año para
realizar estudio.");
                            Console.WriteLine("1.-Poblacion en año
2017.");
                            Console.WriteLine("2.-Poblacion en año
2027.");
                            Console.WriteLine("3.-Poblacion en año
2037.");
                            Console.WriteLine("4.-Poblacion en año
2047.");
                            Console.WriteLine("5.-Poblacion en año
2057.");
                            Console.WriteLine("6.-Terminar.");
                            Console.WriteLine("Elige opcion:");
                            opcion1 =
Int32.Parse(Console.ReadLine());

                            switch (opcion1)
                            {
                                case 1:
                                    {
```

```
                                        porcentaje = porciento[0,
1] + porciento[2, 1] - porciento[1, 0] - porciento[1, 2];

                                        poblacion =
(poblacionsemiurbana / 100) * porcentaje;

                                        poblacionsemiurbana =
poblacionsemiurbana + poblacion;

                                        Console.Clear();


Console.WriteLine("Poblacion en 2017:" + poblacionsemiurbana + "
habitantes.");

                                        break;

                                    }

                                case 2:

                                    {

                                        poblacion = poblacion * 2;

                                        poblacionsemiurbana =
poblacionsemiurbana + poblacion;

                                        Console.Clear();


Console.WriteLine("Poblacion en 2027: " + poblacionsemiurbana + "
habitantes.");


                                        break;

                                    }

                                case 3:

                                    {

                                        poblacion = poblacion * 3;

                                        poblacionsemiurbana =
poblacionsemiurbana + poblacion;

                                        Console.Clear();


Console.WriteLine("Poblacion en 2037: " + poblacionsemiurbana + "
habitantes.");


                                        break;

                                    }

                                case 4:
```

```
                                        {
                                            poblacion = poblacion * 4;
                                            poblacionsemiurbana =
poblacionsemiurbana + poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2047: " + poblacionsemiurbana + "
habitantes.");


                                            break;
                                        }
                                    case 5:
                                        {
                                            poblacion = poblacion * 5;
                                            poblacionsemiurbana =
poblacionsemiurbana + poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2037: " + poblacionsemiurbana + "
habitantes.");


                                            break;
                                        }
                                    case 6:
                                        {
                                            Console.Clear();
                                            Console.WriteLine("Termina
el estudio zona SemiUrbana");


                                            break;
                                        }
                                    default:
                                        {
                                            Console.Clear();
```

```
                                        Console.WriteLine("Error,
opcion introducida incorrecta.");

                                        break;
                                    }
                                }
                            } while (opcion1 != 6);
                            Console.Clear();
                            break;
                        }


                    case 3:
                        {
                            do
                            {
                                Console.WriteLine("Elige año para
realizar estudio.");
                                Console.WriteLine("1.-Poblacion en año
2017.");
                                Console.WriteLine("2.-Poblacion en año
2027.");
                                Console.WriteLine("3.-Poblacion en año
2037.");
                                Console.WriteLine("4.-Poblacion en año
2047.");
                                Console.WriteLine("5.-Poblacion en año
2057.");
                                Console.WriteLine("6.-Terminar.");
                                Console.WriteLine("Elige opcion:");
                                opcion1 =
Int32.Parse(Console.ReadLine());

                                switch (opcion1)
                                {
```

```
                                    case 1:
                                        {
                                            porcentaje = porciento[0,
2] + porciento[1, 2] - porciento[2, 0] - porciento[2, 1];
                                            poblacion =
(poblacionrural / 100) * porcentaje;
                                            poblacionrural =
poblacionrural + poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2017:" + poblacionrural + "
habitantes.");


                                            break;
                                        }
                                    case 2:
                                        {
                                            poblacion = poblacion * 2;
                                            poblacionrural =
poblacionrural + poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2027: " + poblacionrural + "
habitantes.");


                                            break;
                                        }
                                    case 3:
                                        {
                                            poblacion = poblacion * 3;
                                            poblacionrural =
poblacionrural + poblacion;
                                            Console.Clear();

Console.WriteLine("Poblacion en 2037: " + poblacionrural + "
habitantes.");
```

```
                    break;
                }
            case 4:
                {
                    poblacion = poblacion * 4;
                    poblacionrural =
poblacionrural + poblacion;
                    Console.Clear();

Console.WriteLine("Poblacion en 2047: " + poblacionrural + "
habitantes.");

                    break;
                }
            case 5:
                {
                    poblacion = poblacion * 5;
                    poblacionrural =
poblacionrural + poblacion;
                    Console.Clear();

Console.WriteLine("Poblacion en 2037: " + poblacionrural + "
habitantes.");

                    break;
                }
            case 6:
                {
                    Console.Clear();
                    Console.WriteLine("Termina
el estudio zona Rural");

                    break;
                }
            default:
```

```
                                        {
                                            Console.Clear();
                                            Console.WriteLine("Error,
opcion introducida incorrecta.");

                                            break;
                                        }
                                    }
                                } while (opcion1 != 6);
                                Console.Clear();
                                break;
                            }
                        default:
                            {
                                Console.Clear();
                                Console.WriteLine("Final");

                                break;
                            }

                    }

                } while (opcion != 4);

        }
    }
}
```

Ejemplo 3:

Programa que introduciendo un numero de empleado, las horas trabajadas y la tarifa horaria, calcule el salario como horas* tarifa para un número de horas trabajadas y una tarifa horaria de algunos empleados. Tiene que mostrar por

pantalla el número de empleado, número de horas trabajadas, tipo de tarifa y salario total.

Trabajar de 8 de la mañana a 8 de la tarde = 7 euros la hora. Tarifa1

Trabajar de 8 de la noche a 8 de la mañana = 12 euros la hora. Tarifa2.

```
class Trabajador
    {
        static void Main(string[] args)
        {
            int Numempleado, horastrabajadas, salario, tarifahoraria =
0;

            Console.WriteLine("Introduce numero de empleado: ");
            Numempleado = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Introduce numero de horas trabajadas:
");
            horastrabajadas = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Introduce tarifa horaria(1-dia..2-
noche): ");
            tarifahoraria = Int32.Parse(Console.ReadLine());
            if (tarifahoraria==1)
                salario= horastrabajadas*7;
            else
                salario= horastrabajadas*12;

            Console.WriteLine("================================");
            Console.WriteLine("Numero de empleado:" +Numempleado);
            Console.WriteLine("Numero de horas trabajadas:"
+horastrabajadas+ " horas");
            if (tarifahoraria==1)
                Console.WriteLine("Tarifa Diurna.");
            else
                Console.WriteLine("Tarifa Nocturna.");
```

```
            Console.WriteLine("Salario total del dia:"+salario+ "
euros.");

            Console.WriteLine("================================");

        }

    }
```

Ejemplo 4:

Escribir un programa que pida una temperatura para, seguidamente, visualizar/escribir el deporte apropiado para dicha temperatura según el siguiente criterio:

- Natación: mas de 30 grados.
- Tenis: mas de 25 grados.
- Golf: mas de 20 grados.
- Esquí: mas de 5 grados.
- Ajedrez: hasta 5 grados.

Nota: Los topes de temperaturas para cada deporte son los mínimos para el deporte anterior.

```
class temperatura

    {

        static void Main(string[] args)

        {

            int temperatura = 0;


            Console.WriteLine("Introduzca temperatura: ");
```

```
            temperatura = Int32.Parse(Console.ReadLine());


            if (temperatura > 30)
                Console.WriteLine(" El deporte a realizar es
Natacion.");
            else if (temperatura > 25)
                Console.WriteLine(" El deporte a realizar es Tenis.");
            else if (temperatura>20)
                Console.WriteLine("El deporte a realizar es Golf.");
            else if (temperatura>5)
                Console.WriteLine("El deporte a realizar es Esqui");
            else
                Console.WriteLine("¡¡¡¡¡Quedate en casa y juega al
Ajedrez!!!!!!!");


        }
    }
```

Ejemplo 5:

La tarifa de un TAXI es la siguiente:

- Una cantidad fija de 20 euros, sino se sobrepasan los 30 km.
- Para mas de 30 km, se consideraran los siguientes supuestos:
  - Si no se sobrepasan los 100 km, 1 euro por km, que exceda de los 30, además de los 20 euros.
  - Si sobrepasa los 100 km, 0,50 céntimos por km que exceda de los 100, 1 euro por km desde los 30 a los 100 y los 20 euros.

Diseñar un programa que pida los kilómetros recorridos y calcule el total a pagar según la tarifa anterior.

```
class taxi
    {
        static void Main(string[] args)
        {
            const int tarifa30= 20;
            double km, precio = 0;
```

```
            Console.WriteLine("================================");
            Console.WriteLine("Introduce kilometros recorridos:");
            km = Int32.Parse(Console.ReadLine());
            if (km <= tarifa30)
                Console.WriteLine("El costo del taxi es 20 euros");
            else if (km > 30 && km <= 100)
            {
                km = km - 30;
                precio = tarifa30 + (1 * km);
                Console.WriteLine("El costo del taxi es:" + precio +
"euros");
            }
            else
            {
                km = km - 100;
                precio = tarifa30 + 70 + (0.50 * km);
                Console.WriteLine("El costo del taxi es:" + precio +
"euros");
            }


        }
    }
```

ACTIVIDADES.

Ejercicio 1:

Introducir salarios en euros de tal manera que nos visualice al final (después de haber introducido **X** salarios) por pantalla el salario máximo y el salario mínimo y que al introducir un salario igual a 0, dejemos de introducir salarios.

Ejercicio 2:

Realizar un programa que calcule y visualice la suma de los múltiplos de 5 comprendidos entre dos valores a y b. El programa no permitirá introducir valores negativos para a y b, y verificara que a es menos que b. Si a es mayor que b, intercambiara esos valores.

Ejercicio 3:

Realizar un programa que lea una lista de valores introducida por el teclado. A continuación, y sobre la lista, buscar los valores máximo y mínimo, y escribirlos.

Ejercicio 4:

El precio de unas camisas es de 10 euros cada una si la compra es superior a 3 unidades y de 12 euros en cualquier otro caso. Hacer un programa que pida el número de camisas a comprar y visualice el coste total

Ejercicio 5:

Escribe un programa como el del ejercicio 5 ,pero con la modificación de que si el numero de empleado es mayor a 1000, el programa leerá un salario anual y calculara la paga semanal del empleado dividiendo dicho salario entre 52. Si el numero de empleado es menor que 1000, el salario se calculara en base a la tarifa horaria, como en el ejercicio anterior. Si es anual se aplicara tarifa diaria.

Ejercicio 6:

Modifica el ejercicio 8 de manera que se apliquen dos tarifas si es jubilado tendrá un 20% de descuento sobre la tarifa base y por otro lado a partir del kilómetro 100 y siendo menor a 200 para los jubilados la tarifa será gratis para ese intervalo. Realiza el programa teniendo en cuenta esta salvedad.

Ejercicio 7:

Tenemos una empresa que necesita incorporar a su plantilla varios empleados en diversos departamentos. Se reciben multitud de Currículo Vitae y se intenta meter en una pequeña aplicación para realizar una primera selección y en base a su resultado, comprobaremos si es apto o no apto para optar al cargo.

- Necesita la empresa:
  - Un administrativo.
  - Un transportista.
  - Dos operarios.
  - Tres guardias de seguridad.
- Para todos los puestos tienen que tener 18 años
- Para administrativo y transportista pueden tener hasta 55 años.
- Para operarios no pueden superar los 50 años.
- Para guardia de seguridad no pueden superar los 45 años.
- Para administrativo se requiere el Ciclo superior en Administración y Finanzas.
- Para los demás puestos el titulo de la ESO.

Una vez haya superado los requerimientos anteriores, introduciremos el nombre, apellidos, dirección y número de DNI.

Ejercicio 8:

Escribe un programa para predecir el tiempo que va a hacer mañana a partir

de varios datos atmosféricos suministrados por el usuario. Estos datos son:

a. La presión atmosférica: puede ser alta, media o baja.

b. La humedad relativa: también puede ser alta, media o baja

Tienes que calcular la probabilidad de lluvia, la probabilidad de sol y la probabilidad de que haga frío, mediante estos cálculos:

Para calcular la probabilidad de lluvia:

Presión	**Humedad**	**Probabilidad de lluvia**
Baja	Alta	Muy alta
Baja	Media	alta
Baja	Baja	alta
Media	Media	media
En cualquier otro caso		baja

Para calcular la probabilidad de que haga sol:

Presión	**Humedad**	**Probabilidad de sol**
Baja	Alta	Baja
Baja	Media	Media
Baja	Alta	Media
Media	Media	Media
En cualquier otro caso		Alta

Para calcular la probabilidad de que haga frió:

Presión	Humedad	Probabilidad de frió
Baja	Alta	Alta
Baja	Media	Alta
Media	Alta	Alta
Media	Media	Media
En cualquier otro caso		Baja

Presenta por pantalla los resultados del tiempo que va hacer.

TEMA 2. CLASES Y OBJETOS.

2.1. INTRODUCCION A CLASES.

C# es un lenguaje orientado a objetos, por tanto sus programas se componen solo de objetos.

Todo programa C# está formado por al menos una clase que define un metodo nombrado **Main**:

Una clase que contiene un método **Main** es una plantilla para crear lo que vamos a denominar objeto principal, objeto que tiene como misión iniciar y finalizar la ejecución del programa. Precisamente el método Main es el punto de entrada y de salida del programa.

Para definir una clase se utiliza la sintaxis siguiente:

```
[public | private | protected | internal ]

class nombre-clase

{

 //atributos y métodos de la clase

}
```

- Una clase con nivel de protección **internal** solo puede ser utilizada por las clases de su mismo proyecto.
- Una clase con nivel de protección **public** puede ser utilizada por cualquier otra clase de otro proyecto.
- Una clase con nivel de protección **private** es accesible solamente desde dentro de su contexto de declaración.
- Una clase con nivel de protección **protected** es accesible solamente desde dentro de su propia declaración o desde una clase derivada.

2.2. METODOS.

Un programa orientado a objetos se compone solamente de objetos. Cada uno de ellos es una entidad que tiene unas propiedades particulares, los atributos, y unas formas de operar sobre ellos, los métodos. , son borrados.

2.3. DISEÑO DE UNA CLASE DE OBJETOS.

Vamos a ir explicando el significado de métodos, atributos, objetos mediante un ejemplo:

La clase se inicia con:

```
class CCoche

{
```

```
// Cuerpo de la clase: atributos y métodos.
}
```

Ejemplo 1:

Diseña una clase CCoche que represente coches. Incluye los atributos marca, modelo y color; y los métodos que simulen, enviando mensajes, las acciones de arrancar el motor, cambiar de velocidad, acelerar, frenar y para el motor.

```
public class Coche
        {
            //Variables globales que se utilizan para la clase.

            private string m_color;
            private string m_marca;
            private string m_tipo;
            private int marcha;

            //Ejecucion de todos los metodos de la clase.

            public string Color
            {
                get
                {
                    return m_color;
                }
                set
                {
                    if (value == null)
                    {
                        m_color = "color desconocido";
                    }
```

```
            else
            {
                m_color = value;
            }
        }
    }

    public string Marca
    {
        get
        {
            return m_marca;
        }
        set
        {
            if (value == null)
            {
                m_marca = "marca desconocida";
            }
            else
            {
                m_marca = value;
            }
        }
    }

    public string Tipo
    {
        get
        {
            return m_tipo;
        }
```

```
        set
        {
            if (value == null)
            {
                m_tipo = "marca desconocida";
            }
            else
            {
                m_tipo = value;
            }
        }
    }

    public void ArrancarMotor()
    {
        System.Console.WriteLine("BRrrrrrrrrummmmm...");
    }

    public void Acelerar()
    {
        System.Console.WriteLine("Acelerando...");
    }

    public void SubirMarcha()
    {
        if (marcha < 5)
        {
            marcha = marcha + 1;
        }
        System.Console.WriteLine("Marcha: " + marcha);
    }
```

```
            public void BajarMarcha()
            {
                if (marcha > -1)
                {
                    marcha = marcha - 1;
                }
                // -1 = marcha atras
                System.Console.WriteLine("Marcha: " + marcha);
            }

            public void Frenar()
            {
                System.Console.WriteLine("Frenando...");
            }

            public void PararMotor()
            {
                System.Console.WriteLine("Motor parado.");
            }

            public void DescribirCoche()
            {
                System.Console.WriteLine(" -- Mi coche es un " + Marca
+ " " + Color + " " + Tipo);
            }

        static void Main(string[] args)
        {

        //Creacion del objeto micoche a traves de la clase Coche, que
se encuentra
        //misma plantilla.
```

```
        Coche micoche=new Coche();

      //LLamada a los metodos por asignacion.

           micoche.Marca = "BMW";
           micoche.Color = "Negro metalizado";
           micoche.Tipo = "descapotable";

       //LLamada a los metodos con variablres de entrada y salida.

           micoche.ArrancarMotor();
           micoche.Acelerar();
           micoche.SubirMarcha();
           micoche.Acelerar();
           micoche.SubirMarcha();
           micoche.Acelerar();
           micoche.SubirMarcha();
           micoche.Frenar();
           micoche.BajarMarcha();
           micoche.Frenar();
           micoche.BajarMarcha();
           micoche.Frenar();
           micoche.BajarMarcha();
           micoche.PararMotor();
           micoche.DescribirCoche();
    }
}
```

ACTIVIDADES.

Ejercicio 1:

Diseña una clase Persona. Incluirá al menos 6 atributos y en cuanto a métodos (métodos de acciones, sin incluir los métodos de propiedades) incluirá varios (mínimo de 6 métodos) y que se engloben en un proceso lineal y continuo para el objeto creado a partir de la clase Persona.

Ejercicio 2:

Vamos a diseñar una clase Barco. Incluirá una serie de atributos introducidos por teclado (no serán por asignación), los atributos podrán ser:

- Nombre barco.
- Tipo de barco (barca, zodiac, yate,….).
- Numero de tripulantes que puede albergar.
- Longitud del barco
- Y otros tres atributos que creas conveniente.

A continuación se crearan los métodos por asignación correspondientes a los atributos detallados anteriormente, después crearas otros 6 métodos de acciones referentes a la clase.

TEMA 3. CONSTRUCTORES Y HERENCIA .

3.1.- CONSTRUCTOR.

Un constructor es un método especial de una clase que es llamado automáticamente siempre que se crea un objeto de esa clase. Su función es iniciar el objeto.

Siempre que en una clase se define explícitamente un constructor, el constructor implícito (constructor por omisión) es reemplazado por este.

Una línea como la siguiente invocara al constructor sin parámetros:

CPersona persona1= new CPersona(); //Invoca al constructor CPersona

El operador new crea un nuevo objeto, en este caso de la clase CPersona, y a continuación se invoca al constructor de su clase para realizar las operaciones de iniciación que estén programadas. Y una línea como la siguiente invocara al constructor con cuatro parámetros de la misma clase:

CPersona persona2= new CPersona("Juan Pérez", "30 años", Madrid);

3.2.- HERENCIA.

La herencia permite que una clase herede los atributos y métodos de otra clase (los constructores no se heredan). Esto garantiza la reutilización del código.

Con la herencia todas las clases están clasificadas en una jerarquía estricta. Cada clase tiene su superclase (la clase superior en la jerarquía, también llamada clase base), y cada clase puede tener una o mas subclases (las clases inferiores en la jerarquía; también llamadas clases derivadas).

Las clases que están en la parte inferior en la jerarquía se dice que heredan de las clases que están en la parte superior.

Como ejemplo vamos a añadir al programa anterior una nueva clase CPersona2 que sea subclase de CCPersona. Para ello, editamos un nuevo fichero CPersona2.cs y escribimos:

```
Class CPersona2 : CPersona
{


}
```

3.2.1.- Jerarquía de clases.

Una subclase puede asimismo ser un superclase de otra clase, y así sucesivamente. En la siguiente figura se puede ver esto con claridad:

Podemos aplicar para manipular la subclase *CPersona1* de la superclase *CPersona* o la subclase *CPersona3* de la superclase *CPersona1* son las mismas que hemos aplicado anteriormente para la subclase *CPersona1* de la superclase *CPersona*, y lo mismo diremos para cualquier otra subclase que deseemos añadir.

```
class CPersona1 :CPersona
{
  //Atributos

  //Métodos.
}
class CPersona2 :CPersona1
{
  //Atributos.

  //Métodos.
```

}

ACTIVIDADES.

Ejercicio 1:

Crea un nuevo proyecto con una clase principal llamada CAlumno con los siguientes atributos:

- Nombre
- Edad
- DNI
- Ciudad de residencia

A continuación crea los métodos correspondientes para estos atributos y además añade los métodos de acciones:

- Alumno enciende el equipo.
- Alumno ejecuta aplicación C#.
- Alumno ejecuta ejercicio1.
- Alumno ejecuta ejercicio2.
- Alumno ejecuta Navegador Explorer.
- Alumno entra en pagina www.youtube.com
- Alumno cierra todas las aplicaciones.
- Alumno apaga equipo.

Crea una clase derivada llamada CAlumno2 y que se derive de la clase CAlumno con los siguientes atributos:

- Altura.
- Color de pelo.
- Color de ojos.

Crea los métodos correspondientes para estos atributos y además añade los métodos de acciones:

- Alumno sale de clase.
- Alumno sube autobús.
- Alumno llega a parada uno.
- Alumno llega a parada dos.
- Alumno llega a parada tres.
- Alumno llega a casa.

Nota: los métodos repetitivos por lógica deben ser implementados en un solo método.

Ejercicio 2:

Crea un nuevo proyecto llamado Coche incluyendo una clase principal llamada CCoche1 y que incluya los siguientes atributos:

- Utilitario, Todoterreno, Monovolumen (añade los que creas oportuno).
- Marca
- Potencia en CV.
- Cilindrada en centímetros cúbicos.
- Tipo de tracción. (simple o 4 ruedas), deberías utilizar booleano.

A continuación añade los siguientes métodos.

- Arrancar vehículo.
- Vehículo sale de garaje.
- Vehículo primera velocidad.
- Incremento de velocidad (hasta máxima velocidad).
- Decrementar velocidad hasta stop.
- Incorporación autopista (incremento velocidades).
- Pasar peaje y pagar peaje (habrás tenido que decrementar velocidades).
- Seguir autopista (incremento velocidades).
- Salir autopista.

- Aparcar en zona descanso
- Parar vehículo.

A continuación crea una clase derivada llamada CCoche2 , basada en CCoche1 que incorpore lo siguiente:

- Navegador a bordo (si o no).
- Color del coche.

Finalmente incorpora una tercera clase llamada CCoche3 basada en CCoche2 que salga de zona de descanso (arrancar coche), entre en autopista incremente velocidades y finalmente salga de autopista y aparque en hotel (aparcando y parando vehículo).

Nota: En este caso la introducción de datos (atributos) se hará por teclado.

Ejercicio 6:

Necesitamos realizar sobre una población un nuevo empadronamiento, ya que aún no lo tenemos informatizado. Para ello necesitamos recabar diversa información sobre cada uno de los habitantes de la población (no menos de 5 atributos asignados directamente). A continuación se realizaran diversas acciones sobre esta superclase creada (implementa las que consideres oportunas).

Posteriormente entendemos que hay carencias en el estudio y posterior implementación de lo realizado anteriormente ,añadiendo nuevos atributos (si es inmigrante, si esta empleado/desempleado,etc..) comprendiendo por tanto que se necesitaran crear nuevos métodos y será implementado en una nueva clase.