EJERCICIOS RESUELTOS TEMA 7

LA PALABRA “STATIC”

(7.1.1) Amplía el ejemplo 07_01a con una función "static" llamada "EscribirCentrado", que escriba centrado horizontalmente el texto que se le indique como parámetro.

```
using System;

namespace EjerciciosTema7
{
    public class Hardware
    {
        public static void BorrarPantalla()
        {
            for (byte i = 0; i < 25; i++) Console.WriteLine();
        }
    }

    public class Ejemplo_07_01a
    {
        public static void Main()
        {
            Console.WriteLine("Pulsa Intro para borrar");
            Console.ReadLine();

            Hardware.BorrarPantalla();
            Console.WriteLine("Borrado!");

            Console.Write("Introduce una cadena de texto: ");
            string cadena = Console.ReadLine();
            EscribirCentrado(cadena);

            Console.ReadLine();
        }

        public static void EscribirCentrado(string cadena)
        {
            Console.Clear();
            Console.SetCursorPosition(40, 12);
            Console.Write(cadena);
        }
    }
}
```

(7.1.2) Amplía el ejemplo 07_01b con una función llamada "EscribirCentrado", que escriba centrado horizontalmente el texto que se le indique como parámetro. Al contrario que en el ejercicio 7.1.1, esta versión no será "static".

```
using System;

namespace EjerciciosTema7
{
    public class Hardware
    {
        public void BorrarPantalla()
        {
            for (byte i = 0; i < 25; i++) Console.WriteLine();
```

```
        }

        public void EscribirCentrado(string cadena)
        {
            Console.Clear();
            Console.SetCursorPosition(40, 12);
            Console.Write(cadena);
        }
    }

    public class Ejemplo_07_01b
    {
        public static void Main()
        {
            Console.WriteLine("Pulsa Intro para borrar");
            Console.ReadLine();
            Hardware miPantalla = new Hardware();
            miPantalla.BorrarPantalla();
            Console.WriteLine("Borrado!");

            Hardware hardware = new Hardware();
            Console.Write("Introduce una cadena de texto: ");
            string cadena = Console.ReadLine();
            hardware.EscribirCentrado(cadena);

            Console.ReadLine();
        }
    }
}
```

(7.1.3) Crea una nueva versión del ejercicio 5.2.3 (base de datos de ficheros, descompuesta en funciones), en la que los métodos y variables no sean "static".

```
using System;

namespace _7._1._3
{
    public class Ejemplo_04_06a
    {
        public static void Main()
        {
            int opcion;
            Casos caso1 = new Casos();
            Casos caso2 = new Casos();
            Casos caso3 = new Casos();
            Casos caso4 = new Casos();
            Menu menu = new Menu();

            do
            {
                switch (menu.EstablecerMenu(out opcion))
                {
                    case 1:
                        caso1.PrimerCaso();
                        break;
                    case 2:
                        caso2.SegundoCaso();
                        break;
                    case 3:
                        caso3.TercerCaso();
```

```
                    break;
                case 4:
                    caso4.CuartoCaso();
                    break;
                case 5:
                    Console.WriteLine("Fin del programa");
                    break;
                default:
                    Console.WriteLine("Opción desconocida!");
                    break;
            }
        } while (opcion != 5);
    }
}
```

```
using System;

namespace _7._1._3
{
    class Menu
    {
        public int EstablecerMenu(out int opcion)
        {
            Console.WriteLine();
            Console.WriteLine("Escoja una opción:");
            Console.WriteLine("1.- Añadir datos de un nuevo fichero");
            Console.WriteLine("2.- Mostrar los nombres de todos los ficheros");
            Console.WriteLine("3.- Mostrar ficheros por encima de un cierto
tamaño");
            Console.WriteLine("4.- Ver datos de un fichero");
            Console.WriteLine("5.- Salir");
            return opcion = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

```
using System;

namespace _7._1._3
{
    class Casos
    {
        struct TipoFicha
        {
            public string nombreFich;
            public long tamanyo;
        }

        static TipoFicha[] fichas = new TipoFicha[1000];
        static int numeroFichas = 0;

        public void PrimerCaso()
        {
            if (numeroFichas < 1000)
            {
                Console.WriteLine("Introduce el nombre del fichero: ");
                fichas[numeroFichas].nombreFich = Console.ReadLine();
                Console.WriteLine("Introduce el tamaño en KB: ");
                fichas[numeroFichas].tamanyo =
Convert.ToInt32(Console.ReadLine());
                numeroFichas++;
            }
            else
```

```
                Console.WriteLine("Máximo de fichas alcanzado (1000)!");
        }

        public void SegundoCaso()
        {
            for (int i = 0; i < numeroFichas; i++)
                Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
fichas[i].nombreFich, fichas[i].tamanyo);
        }

        public void TercerCaso()
        {
            long tamanyoBuscar;

            Console.WriteLine("¿A partir de que tamaño quieres ver?");
            tamanyoBuscar = Convert.ToInt64(Console.ReadLine());
            for (int i = 0; i < numeroFichas; i++)
                if (fichas[i].tamanyo >= tamanyoBuscar)
                    Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
fichas[i].nombreFich, fichas[i].tamanyo);
        }

        public void CuartoCaso()
        {
            string textoBuscar;

            Console.WriteLine("¿De qué fichero quieres ver todos los datos?");
            textoBuscar = Console.ReadLine();
            for (int i = 0; i < numeroFichas; i++)
                if (fichas[i].nombreFich == textoBuscar)
                    Console.WriteLine("Nombre: {0}; Tamaño: {1} KB",
fichas[i].nombreFich, fichas[i].tamanyo);
        }
    }
}
```

Con array de objetos:

```
using System;

namespace _7._1._3
{
    public class Ejemplo_04_06a
    {
        public static void Main()
        {
            int opcion;
            /*Casos caso1 = new Casos();
            Casos caso2 = new Casos();
            Casos caso3 = new Casos();
            Casos caso4 = new Casos();*/
            Casos[] casos = new Casos[4];
            for (byte i=0; i<4; i++)
            {
                casos[i] = new Casos();
            }
            Menu menu = new Menu();

            do
            {
                switch (menu.EstablecerMenu(out opcion))
```

```
                {
                    case 1:
                        casos[0].PrimerCaso();
                        break;
                    case 2:
                        casos[1].SegundoCaso();
                        break;
                    case 3:
                        casos[2].TercerCaso();
                        break;
                    case 4:
                        casos[3].CuartoCaso();
                        break;
                    case 5:
                        Console.WriteLine("Fin del programa");
                        break;
                    default:
                        Console.WriteLine("Opción desconocida!");
                        break;
                }
            } while (opcion != 5);
        }
    }
}
```

ARRAYS DE OBJETOS

(7.2.1) Crea una versión ampliada del ejercicio 6.8.1 (clase Trabajador y relacionadas), en la que no se cree un único objeto de cada clase, sino un array de tres objetos.

```
using System;

namespace _7._2._1
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[3];
            for (byte i=0; i<3; i++)
            {
                trabajadores[i] = new Trabajador();
            }
            Console.WriteLine();

            Programador[] programadores = new Programador[3];
            for (byte i = 0; i < 3; i++)
            {
                programadores[i] = new Programador();
            }
            Console.WriteLine();

            Analista[] analistas = new Analista[3];
            for (byte i = 0; i < 3; i++)
            {
                analistas[i] = new Analista();
            }
            Console.WriteLine();

            Ingeniero[] ingenieros = new Ingeniero[3];
```

```
            for (byte i = 0; i < 3; i++)
            {
                ingenieros[i] = new Ingeniero();
            }
            Console.WriteLine();

            IngenieroInformatico[] ingenierosInformaticos = new
IngenieroInformatico[3];
            for (byte i = 0; i < 3; i++)
            {
                ingenierosInformaticos[i] = new IngenieroInformatico();
            }

            Console.ReadLine();
        }
    }
}
```

El resto de las clases permanecen igual.

(7.2.2) Amplía el proyecto Libro (ejercicio 6.7.2), de modo que permita guardar hasta 1.000 libros. Main mostrará un menú que permita añadir un nuevo libro o ver los datos de los ya existentes.

```
using System;

namespace Libreria
{
    class PruebaLibro
    {
        public static void Main()
        {
            int opcion, numeroPaginas, contadorLibros = 0;
            string titulo, autor, ubicacion;

            Libro[] libros = new Libro[1000];
            for (int i=0; i<1000; i++)
            {
                libros[i] = new Libro();
            }

            do
            {
                Console.Write("\n1) Añadir un nuevo libro\n2) Mostrar base de
datos\n3) Salir\nElige una opción: ");
                opcion = Convert.ToInt32(Console.ReadLine());

                switch (opcion)
                {
                    case 1:
                        if (contadorLibros < 1000)
                        {
                            Console.Write("Introduce el título del libro: ");
                            titulo = Console.ReadLine();
                            libros[contadorLibros].SetTitulo(titulo);
                            Console.Write("\nIntroduce el autor del libro: ");
                            autor = Console.ReadLine();
                            libros[contadorLibros].SetAutor(autor);
                            Console.Write("\nIntroduce la ubicación del libro:
");
```

```
                            ubicacion = Console.ReadLine();
                            libros[contadorLibros].SetUbicacion(ubicacion);
                            Console.Write("\nIntroduce el número de páginas del
libro: ");
                            numeroPaginas = Convert.ToInt32(Console.ReadLine());
                            libros[contadorLibros].SetPaginas(numeroPaginas);

                            contadorLibros++;
                        }
                        else Console.Write("El array está lleno.");
                        break;
                    case 2:
                        for (int i = 0; i < contadorLibros; i++)
                        {
                            Console.Write("\nEl libro \"{0}\" del autor {1}.
Ubicación: {2}. Tiene {3} páginas.\n",
                                libros[i].GetTitulo(), libros[i].GetAutor(),
libros[i].GetUbicacion(), libros[i].GetPaginas());
                        }
                        break;
                    case 3:
                        Console.Write("\nSaliendo de la base de datos. Pulsa
Intro para terminar.");
                        break;
                    default:
                        Console.Write("\nLa opción introducida es
incorrecta!\n");
                        break;
                }
            } while (opcion != 3);
            Console.ReadLine();
        }
    }
}
using System;

namespace Libreria
{
    class Documento
    {
        protected string autor;
        protected string titulo;
        protected string ubicacion;

        public string GetAutor()
        {
            return autor;
        }

        public void SetAutor(string nuevoAutor)
        {
            autor = nuevoAutor;
        }

        public string GetTitulo()
        {
            return titulo;
        }

        public void SetTitulo(string nuevoTitulo)
        {
            titulo = nuevoTitulo;
```

```
        }

        public string GetUbicacion()
        {
            return ubicacion;
        }

        public void SetUbicacion(string nuevaUbicacion)
        {
            ubicacion = nuevaUbicacion;
        }
    }
}
```

```
using System;

namespace Libreria
{
    class Libro : Documento
    {
        protected int paginas;

        public Libro()
        {

        }

        public Libro(string autor, string titulo, string ubicacion)
        {
            this.autor = autor;
            this.titulo = titulo;
            this.ubicacion = ubicacion;
        }

        public Libro(string autor, string titulo)
        {
            this.autor = autor;
            this.titulo = titulo;
            ubicacion = "no detallada";
        }

        public int GetPaginas()
        {
            return paginas;
        }

        public void SetPaginas(int nuevoValor)
        {
            paginas = nuevoValor;
        }
    }
}
```

(7.2.4) A partir del ejemplo 07.02b y del ejercicio 6.8.1 (clase Trabajador y relacionadas), crea un array de trabajadores en el que no sean todos de la misma clase.

```
using System;

namespace _7._2._1
{
    class Program
```

```
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[10];
            for (byte i=0; i<3; i++)
            {
                trabajadores[i] = new Trabajador();
            }
            Console.WriteLine();

            for (byte i = 3; i < 6; i++)
            {
                trabajadores[i] = new Ingeniero();
            }
            Console.WriteLine();

            for (byte i = 6; i < 9; i++)
            {
                trabajadores[i] = new Analista();
            }
            Console.WriteLine();

            trabajadores[9] = new IngenieroInformatico();



            Console.ReadLine();
        }
    }
}
```

El resto de las clases permanecen igual.

(7.2.5) Amplía el proyecto Libro (ejercicio 7.2.2), de modo que permita guardar 1000 documentos de cualquier tipo. A la hora de añadir un documento, se preguntará al usuario si desea guardar un documento genérico o un libro, para usar el constructor adecuado.

```
using System;

namespace Libreria
{
    class PruebaLibro
    {
        public static void Main()
        {
            int opcion, contadorLibros = 0;
            string titulo, autor, ubicacion, tipoDocumento;

            Documento[] documentos = new Documento[1000];

            do
            {
                Console.Write("\n1) Añadir un nuevo documento\n2) Mostrar base de
datos\n3) Salir\nElige una opción: ");
                opcion = Convert.ToInt32(Console.ReadLine());

                switch (opcion)
                {
                    case 1:
                        if (contadorLibros < 1000)
```

```
                        {
                            Console.Write("¿Quieres guardar un documento genérico
o un libro?: ");
                            tipoDocumento = Console.ReadLine();
                            if (tipoDocumento == "libro")
documentos[contadorLibros] = new Libro();
                            else documentos[contadorLibros] = new Documento();
                            Console.Write("Introduce el título del libro: ");
                            titulo = Console.ReadLine();
                            documentos[contadorLibros].SetTitulo(titulo);
                            Console.Write("\nIntroduce el autor del libro: ");
                            autor = Console.ReadLine();
                            documentos[contadorLibros].SetAutor(autor);
                            Console.Write("\nIntroduce la ubicación del libro:
");
                            ubicacion = Console.ReadLine();
                            documentos[contadorLibros].SetUbicacion(ubicacion);

                            contadorLibros++;
                        }
                        else Console.Write("El array está lleno.");
                        break;
                    case 2:
                        for (int i = 0; i < contadorLibros; i++)
                        {
                            Console.Write("\nEl libro \"{0}\" del autor {1}.
Ubicación: {2}.\n",
                                documentos[i].GetTitulo(),
documentos[i].GetAutor(), documentos[i].GetUbicacion());
                        }
                        break;
                    case 3:
                        Console.Write("\nSaliendo de la base de datos. Pulsa
Intro para terminar.");
                        break;
                    default:
                        Console.Write("\nLa opción introducida es
incorrecta!\n");
                        break;
                }
            } while (opcion != 3);
            Console.ReadLine();
        }
    }
}
using System;

namespace Libreria
{
    class Documento
    {
        protected string autor;
        protected string titulo;
        protected string ubicacion;

        public Documento()
        {
            Console.WriteLine("Llamada al constructor de documento.");
        }

        public string GetAutor()
        {
```

```
            return autor;
        }

        public void SetAutor(string nuevoAutor)
        {
            autor = nuevoAutor;
        }

        public string GetTitulo()
        {
            return titulo;
        }

        public void SetTitulo(string nuevoTitulo)
        {
            titulo = nuevoTitulo;
        }

        public string GetUbicacion()
        {
            return ubicacion;
        }

        public void SetUbicacion(string nuevaUbicacion)
        {
            ubicacion = nuevaUbicacion;
        }
    }
}

using System;

namespace Libreria
{
    class Libro : Documento
    {
        private int paginas;

        public Libro()
        {
            Console.WriteLine("Llamada al constructor de Libro.");
        }

        public Libro(string autor, string titulo, string ubicacion)
        {
            this.autor = autor;
            this.titulo = titulo;
            this.ubicacion = ubicacion;
        }

        public Libro(string autor, string titulo)
        {
            this.autor = autor;
            this.titulo = titulo;
            ubicacion = "no detallada";
        }

        public int GetPaginas()
        {
            return paginas;
        }

        public void SetPaginas(int nuevoValor)
```

```
        {
            paginas = nuevoValor;
        }
    }
}
```

FUNCIONES VIRTUALES. LA PALABRA “OVERRIDE”

(7.3.1) Crea una versión ampliada del ejercicio 6.5.1 (Persona, PersonaInglesa, etc), en la que se cree un único array que contenga personas de varios tipos.

```
using System;

namespace Personas
{
    class PruebaPersona
    {
        static void Main()
        {
            Persona[] personas = new Persona[4];
            personas[0] = new PersonaInglesa();
            personas[1] = new PersonaInglesa();
            personas[2]  = new PersonaItaliana();
            personas[3] = new Persona();

            personas[3].SetNombre("Pepe");
            personas[0].SetNombre("Jack");
            personas[1].SetNombre("Kevin");
            personas[2].SetNombre("Marco");

            for (int i=0; i<4; i++)
            {
                personas[i].Saludar();
            }
            personas[1].TomarTe();


            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace Personas
{
    class Persona
    {
        protected string nombre;

        public void SetNombre(string nuevoNombre)
        {
            nombre = nuevoNombre;
        }

        public virtual void Saludar()
        {
            Console.WriteLine("Hola soy {0}", nombre);
        }
```

```
        public virtual void TomarTe()
        {

        }
    }
}
using System;

namespace Personas
{
    class PersonaInglesa : Persona
    {
        public override void TomarTe()
        {
            Console.WriteLine("Estoy tomando té");
        }

        public override void Saludar()
        {
            Console.WriteLine("Hi, I am {0}", nombre);
        }
    }
}

using System;

namespace Personas
{
    class PersonaItaliana : Persona
    {
        public override void Saludar()
        {
            Console.WriteLine("Ciao, io sono {0}", nombre);
        }
    }
}
```

(7.3.2) Crea una variante del ejercicio 7.2.2 (array de Trabajador y derivadas), en la que se cree un único array "de trabajadores", que contenga un objeto de cada clase, y exista un método "Saludar" que se redefina en todas las clases hijas, usando "new" y probándolo desde "Main".

```
using System;

namespace _7._2._1
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[5];
            trabajadores[0] = new Trabajador();
            trabajadores[1] = new Programador();
            trabajadores[2] = new Analista();
            trabajadores[3] = new Ingeniero();
            trabajadores[4] = new IngenieroInformatico();

            for (int i=0; i<5; i++)
            {
                trabajadores[i].Saludar();
            }
```

```
            Console.ReadLine();
        }
    }
}

using System;

namespace _7._2._1
{
    class Trabajador
    {
        public Trabajador()
        {
        }

        public void Saludar()
        {
            Console.WriteLine("Hola, soy un trabajador.");
        }
    }
}

using System;

namespace _7._2._1
{
    class Programador : Trabajador
    {
        public Programador()
        {
        }

        public new void Saludar()
        {
            Console.WriteLine("Hola, soy un programador.");
        }
    }
}

using System;

namespace _7._2._1
{
    class Analista : Trabajador
    {
        public Analista()
        {
        }

        public new void Saludar()
        {
            Console.WriteLine("Hola, soy un analista.");
        }
    }
}
using System;

namespace _7._2._1
{
    class Ingeniero : Trabajador
    {
        public Ingeniero()
```

```
        {
        }

        public new void Saludar()
        {
            Console.WriteLine("Hola, soy un ingeniero.");
        }
    }
}

using System;

namespace _7._2._1
{
    class IngenieroInformatico : Ingeniero
    {
        public IngenieroInformatico()
        {
        }

        public new void Saludar()
        {
            Console.WriteLine("Hola, soy un ingeniero informático.");
        }
    }
}
```

Podemos comprobar que no funciona con “new”.

(7.3.3) Crea una variante del ejercicio anterior (7.3.2), que use "override" en vez de "new".

```
using System;

namespace _7._2._1
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[5];
            trabajadores[0] = new Trabajador();
            trabajadores[1] = new Programador();
            trabajadores[2] = new Analista();
            trabajadores[3] = new Ingeniero();
            trabajadores[4] = new IngenieroInformatico();

            for (int i=0; i<5; i++)
            {
                trabajadores[i].Saludar();
            }

            Console.ReadLine();
        }
    }
}

using System;

namespace _7._2._1
{
    class Trabajador
    {
```

```
        public Trabajador()
        {
        }

        public virtual void Saludar()
        {
            Console.WriteLine("Hola, soy un trabajador.");
        }
    }
}
```

```
using System;

namespace _7._2._1
{
    class Programador : Trabajador
    {
        public Programador()
        {
        }

        public override void Saludar()
        {
            Console.WriteLine("Hola, soy un programador.");
        }
    }
}
```

```
using System;

namespace _7._2._1
{
    class Analista : Trabajador
    {
        public Analista()
        {
        }

        public override void Saludar()
        {
            Console.WriteLine("Hola, soy un analista.");
        }
    }
}
```

```
using System;

namespace _7._2._1
{
    class Ingeniero : Trabajador
    {
        public Ingeniero()
        {
        }

        public override void Saludar()
        {
            Console.WriteLine("Hola, soy un ingeniero.");
        }
    }
}
```

```
using System;
```

```
namespace _7._2._1
{
    class IngenieroInformatico : Ingeniero
    {
        public IngenieroInformatico()
        {
        }

        public override void Saludar()
        {
            Console.WriteLine("Hola, soy un ingeniero informático.");
        }
    }
}
```

(7.3.4) Amplía el proyecto Libro (ejercicio 7.2.5): tanto la clase Documento como la clase Libro, tendrán un método ToString, que devuelva una cadena de texto formada por título, autor y ubicación, separados por guiones. Crea una clase Articulo, que añada el campo "procedencia". El cuerpo del programa permitirá añadir Artículos o Libros, no documentos genéricos. El método ToString deberá mostrar también el número de páginas de un libro y la procedencia de un artículo. La opción de mostrar datos llamará a los correspondientes métodos ToString. Recuerda usar "virtual" y "override" si en un primer momento no se comporta como debe.

```
using System;

namespace Libreria
{
    class PruebaLibro
    {
        public static void Main()
        {
            int opcion, numPag, contadorLibros = 0;
            string titulo, autor, ubicacion, tipoDocumento, procedencia;

            Documento[] documentos = new Documento[1000];

            do
            {
                Console.Write("\n1) Añadir un nuevo artículo o libro\n2) Mostrar
base de datos\n3) Salir\nElige una opción: ");
                opcion = Convert.ToInt32(Console.ReadLine());

                switch (opcion)
                {
                    case 1:
                        if (contadorLibros < 1000)
                        {
                            Console.Write("¿Quieres guardar un artículo o un
libro?: ");
                            tipoDocumento = Console.ReadLine();
                            if (tipoDocumento == "libro")
documentos[contadorLibros] = new Libro();
                            else documentos[contadorLibros] = new Articulo();
                            Console.Write("Introduce el título: ");
                            titulo = Console.ReadLine();
                            documentos[contadorLibros].SetTitulo(titulo);
                            Console.Write("\nIntroduce el autor: ");
                            autor = Console.ReadLine();
                            documentos[contadorLibros].SetAutor(autor);
                            Console.Write("\nIntroduce la ubicación: ");
```

```
                            ubicacion = Console.ReadLine();
                            documentos[contadorLibros].SetUbicacion(ubicacion);

                            if (tipoDocumento == "libro")
                            {
                                Console.Write("Introduce un número de páginas:
");
                                numPag = Convert.ToInt32(Console.ReadLine());
                                ((Libro)documentos[contadorLibros]).Paginas =
numPag;
                            } else
                            {
                                Console.Write("Introduce la procedencia: ");
                                procedencia = Console.ReadLine();

((Articulo)documentos[contadorLibros]).Procedencia = procedencia;
                            }

                            contadorLibros++;
                        }
                        else Console.Write("El array está lleno.");
                        break;
                    case 2:
                        for (int i = 0; i < contadorLibros; i++)
                        {
                            Console.Write(documentos[i].ToString());
                        }
                        break;
                    case 3:
                        Console.Write("\nSaliendo de la base de datos. Pulsa
Intro para terminar.");
                        break;
                    default:
                        Console.Write("\nLa opción introducida es
incorrecta!\n");
                        break;
                }
            } while (opcion != 3);
            Console.ReadLine();
        }
    }
}


using System;

namespace Libreria
{
    class Documento
    {
        protected string autor;
        protected string titulo;
        protected string ubicacion;

        public Documento()
        {
            Console.WriteLine("Llamada al constructor de documento.");
        }

        public string GetAutor()
        {
            return autor;
```

```
        }

        public void SetAutor(string nuevoAutor)
        {
            autor = nuevoAutor;
        }

        public string GetTitulo()
        {
            return titulo;
        }

        public void SetTitulo(string nuevoTitulo)
        {
            titulo = nuevoTitulo;
        }

        public string GetUbicacion()
        {
            return ubicacion;
        }

        public void SetUbicacion(string nuevaUbicacion)
        {
            ubicacion = nuevaUbicacion;
        }

        public override string ToString()
        {
            return "\nEl libro \"" + titulo + "\" del autor " + autor + "se
encuentra en: " + ubicacion;
        }
    }
}



using System;

namespace Libreria
{
    class Libro : Documento
    {
        public Libro()
        {
            Console.WriteLine("Llamada al constructor de Libro.");
        }

        public Libro(string autor, string titulo, string ubicacion)
        {
            this.autor = autor;
            this.titulo = titulo;
            this.ubicacion = ubicacion;
        }

        public Libro(string autor, string titulo)
        {
            this.autor = autor;
            this.titulo = titulo;
            ubicacion = "no detallada";
        }

        public int Paginas { get; set; }
```

```
        public override string ToString()
        {
            return base.ToString() + "y tiene " + Paginas + " páginas.";
        }
    }
}
```

```
using System;

namespace Libreria
{
    class Articulo : Documento
    {
        public string Procedencia { get; set; }

        public override string ToString()
        {
            return base.ToString() + "y viene de " + Procedencia;
        }
    }
}
```

LLAMANDO A UN MÉTODO DE LA CLASE “PADRE”

(7.4.1) Crea una versión ampliada del ejercicio 7.3.3, en la que el método "Hablar" de todas las clases hijas se apoye en el de la clase "Trabajador".

```
using System;

namespace Trabajadores
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[5];
            trabajadores[0] = new Trabajador();
            trabajadores[1] = new Programador();
            trabajadores[2] = new Analista();
            trabajadores[3] = new Ingeniero();
            trabajadores[4] = new IngenieroInformatico();

            for (int i = 0; i < 5; i++)
            {
                trabajadores[i].Saludar();
            }

            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Trabajador
    {
```

```
        public Trabajador()
        {
        }

        public virtual void Saludar()
        {
            Console.Write("Hola, soy un trabajador");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Analista : Trabajador
    {
        public Analista()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como analista.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Programador : Trabajador
    {
        public Programador()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como programador.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Ingeniero : Trabajador
    {
        public Ingeniero()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como ingeniero.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class IngenieroInformatico : Ingeniero
    {
        public IngenieroInformatico()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine("Mi especialidad es la informática.");
        }
    }
}
```

(7.4.2) Crea una versión ampliada del ejercicio 7.4.1, en la que el constructor de todas las clases hijas se apoye en el de la clase "Trabajador".

```
using System;

namespace Trabajadores
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[5];
            trabajadores[0] = new Trabajador();
            trabajadores[1] = new Programador();
            trabajadores[2] = new Analista();
            trabajadores[3] = new Ingeniero();
            trabajadores[4] = new IngenieroInformatico();

            for (int i = 0; i < 5; i++)
            {
                trabajadores[i].Saludar();
            }

            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Trabajador
    {
        public Trabajador()
        {
        }

        public virtual void Saludar()
        {
            Console.Write("Hola, soy un trabajador");
        }
    }
```

```
}
```

```
using System;

namespace Trabajadores
{
    class Programador : Trabajador
    {
        public Programador() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como programador.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Analista : Trabajador
    {
        public Analista() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como analista.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Ingeniero : Trabajador
    {
        public Ingeniero() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.Write(", trabajo como ingeniero.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class IngenieroInformatico : Ingeniero
    {
        public IngenieroInformatico() : base()
```

```
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine("Mi especialidad es la informática.");
        }
    }
}
```

(7.4.3) Refina el proyecto Libro (ejercicio 7.3.4), para que el método ToString de la clase Libro se apoye en el de la clase Documento, y también lo haga el de la clase Articulo.

```
using System;

namespace Libreria
{
    class PruebaLibro
    {
        public static void Main()
        {
            int opcion, numPag, contadorLibros = 0;
            string titulo, autor, ubicacion, tipoDocumento, procedencia;

            Documento[] documentos = new Documento[1000];

            do
            {
                Console.Write("\n1) Añadir un nuevo artículo o libro\n2) Mostrar
base de datos\n3) Salir\nElige una opción: ");
                opcion = Convert.ToInt32(Console.ReadLine());

                switch (opcion)
                {
                    case 1:
                        if (contadorLibros < 1000)
                        {
                            Console.Write("¿Quieres guardar un artículo o un
libro?: ");
                            tipoDocumento = Console.ReadLine();
                            if (tipoDocumento == "libro")
documentos[contadorLibros] = new Libro();
                            else documentos[contadorLibros] = new Articulo();
                            Console.Write("Introduce el título: ");
                            titulo = Console.ReadLine();
                            documentos[contadorLibros].SetTitulo(titulo);
                            Console.Write("\nIntroduce el autor: ");
                            autor = Console.ReadLine();
                            documentos[contadorLibros].SetAutor(autor);
                            Console.Write("\nIntroduce la ubicación: ");
                            ubicacion = Console.ReadLine();
                            documentos[contadorLibros].SetUbicacion(ubicacion);

                            if (tipoDocumento == "libro")
                            {
                                Console.Write("Introduce un número de páginas:
");
                                numPag = Convert.ToInt32(Console.ReadLine());
                                ((Libro)documentos[contadorLibros]).Paginas =
numPag;
```

```
                            } else
                            {
                                Console.Write("Introduce la procedencia: ");
                                procedencia = Console.ReadLine();

((Articulo)documentos[contadorLibros]).Procedencia = procedencia;
                            }

                            contadorLibros++;
                        }
                        else Console.Write("El array está lleno.");
                        break;
                    case 2:
                        for (int i = 0; i < contadorLibros; i++)
                        {
                            Console.Write(documentos[i].ToString());
                        }
                        break;
                    case 3:
                        Console.Write("\nSaliendo de la base de datos. Pulsa
Intro para terminar.");
                        break;
                    default:
                        Console.Write("\nLa opción introducida es
incorrecta!\n");
                        break;
                }
            } while (opcion != 3);
            Console.ReadLine();
        }
    }
}
using System;

namespace Libreria
{
    class Documento
    {
        protected string autor;
        protected string titulo;
        protected string ubicacion;

        public Documento()
        {
            Console.WriteLine("Llamada al constructor de documento.");
        }

        public string GetAutor()
        {
            return autor;
        }

        public void SetAutor(string nuevoAutor)
        {
            autor = nuevoAutor;
        }

        public string GetTitulo()
        {
            return titulo;
        }
```

```
        public void SetTitulo(string nuevoTitulo)
        {
            titulo = nuevoTitulo;
        }

        public string GetUbicacion()
        {
            return ubicacion;
        }

        public void SetUbicacion(string nuevaUbicacion)
        {
            ubicacion = nuevaUbicacion;
        }

        public override string ToString()
        {
            return "\nEl libro \"" + titulo + "\" del autor " + autor + "se
encuentra en: " + ubicacion;
        }
    }
}
```

```
using System;

namespace Libreria
{
    class Libro : Documento
    {
        public Libro()
        {
            Console.WriteLine("Llamada al constructor de Libro.");
        }

        public Libro(string autor, string titulo, string ubicacion)
        {
            this.autor = autor;
            this.titulo = titulo;
            this.ubicacion = ubicacion;
        }

        public Libro(string autor, string titulo)
        {
            this.autor = autor;
            this.titulo = titulo;
            ubicacion = "no detallada";
        }

        public int Paginas { get; set; }

        public override string ToString()
        {
            return base.ToString() + "y tiene " + Paginas + " páginas.";
        }
    }
}
```

```
using System;

namespace Libreria
{
    class Articulo : Documento
    {
```

```
        public string Procedencia { get; set; }

        public override string ToString()
        {
            return base.ToString() + "y viene de " + Procedencia;
        }
    }
}
```

LA PALABRA “THIS”: EL OBJETO ACTUAL

(7.5.1) Crea una versión ampliada del ejercicio 7.4.2, en la que el constructor sin parámetros de la clase "Trabajador" se apoye en otro constructor que reciba como parámetro el nombre de esa persona. La versión sin parámetros asignará el valor "Nombre no detallado" al nombre de esa persona.

```
using System;

namespace Trabajadores
{
    class Program
    {
        static void Main()
        {
            Trabajador[] trabajadores = new Trabajador[5];
            trabajadores[0] = new Trabajador();
            trabajadores[1] = new Programador();
            trabajadores[2] = new Analista();
            trabajadores[3] = new Ingeniero();
            trabajadores[4] = new IngenieroInformatico();

            for (int i = 0; i < 5; i++)
            {
                trabajadores[i].Saludar();
            }

            Console.ReadLine();
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Trabajador
    {
        private string nombre;

        public Trabajador(string nombre)
        {
            this.nombre = nombre;
        }

        public Trabajador() : this("Nombre no detallado")
        {
        }

        public virtual void Saludar()
        {
            Console.Write("Hola, soy un trabajador");
```

```
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Analista : Trabajador
    {
        public Analista() : base ()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como analista.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Programador : Trabajador
    {
        public Programador() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine(", trabajo como programador.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class Ingeniero : Trabajador
    {
        public Ingeniero() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.Write(", trabajo como ingeniero.");
        }
    }
}
```

```
using System;

namespace Trabajadores
{
    class IngenieroInformatico : Ingeniero
    {
```

```
        public IngenieroInformatico() : base()
        {
        }

        public override void Saludar()
        {
            base.Saludar();
            Console.WriteLine("Mi especialidad es la informática.");
        }
    }
}
```

(7.5.2) Crea una clase Puerta con un ancho, un alto y un método "MostrarEstado" que muestre su ancho y su alto. Crea una clase Casa, que contenga 3 puertas y otro método "MostrarEstado" que escriba "Casa" y luego muestre el estado de sus tres puertas.

```
using System;

namespace _7._5._2
{
    class Program
    {
        static void Main()
        {
            Casa casa = new Casa();
            casa.MostrarEstado();
        }
    }
}
using System;

namespace _7._5._2
{
    class Puerta
    {
        private int ancho, alto;

        public Puerta(int ancho, int alto)
        {
            this.ancho = ancho;
            this.alto = alto;
        }

        public void MostrarEstado()
        {
            Console.Write("El ancho de la puerta es: {0} y el alto: {1}", ancho,
alto);
        }
    }
}
using System;

namespace _7._5._2
{
    class Casa
    {
        Puerta[] puertas = new Puerta[3];

        public Casa()
        {
            for (int i=0; i<3; i++)
```

```
            {
                puertas[i] = new Puerta(i*2, i*3);
            }
        }

        public void MostrarEstado()
        {
            Console.Write("Casa");
            for (int i = 0; i < 3; i++)
            {
                puertas[i].MostrarEstado();
            }
        }
    }
}
//No se utiliza la palabra this. Es un preámbulo del siguiente ejercicio.
```

(7.5.3) Crea una clase Casa, con una superficie (por ejemplo, 90 m2) y un método "MostrarEstado" que escriba su superficie. Cada casa debe contener 3 puertas. Las puertas tendrán un ancho, un alto y un método "MostrarEstado" que muestre su ancho y su alto y la superficie de la casa en la que se encuentran. Crea un programa de prueba que cree una casa y muestre sus datos y los de sus tres puertas.

```
using System;

namespace _7._5._2
{
    class Program
    {
        static void Main()
        {
            Casa casa = new Casa(90);
            casa.MostrarEstado();
            Console.ReadLine();
        }
    }
}
using System;

namespace _7._5._2
{
    class Casa
    {
        public int Superficie { get; set; }

        Puerta[] puertas = new Puerta[3];

        public Casa(int superficie)
        {
            for (int i=0; i<3; i++)
            {
                puertas[i] = new Puerta(i*2, i*3, this);
            }

            Superficie = superficie;
        }

        public void MostrarEstado()
        {
```

```
            Console.Write("La superficie de la casa es " + Superficie);
            for (int i = 0; i < 3; i++)
            {
                puertas[i].MostrarEstado();
            }
        }
    }
}
using System;

namespace _7._5._2
{
    class Puerta
    {
        private int ancho, alto;
        private Casa casa;

        public Puerta(int ancho, int alto, Casa casa)
        {
            this.ancho = ancho;
            this.alto = alto;
            this.casa = casa;
        }

        public void MostrarEstado()
        {
            Console.Write("El ancho de la puerta es: {0} y el alto: {1}. La
superficie de la casa contenedora es {2}", ancho, alto, casa.Superficie);
        }
    }
}
```

SOBRECARGA DE OPERADORES

(7.6.1) Desarrolla una clase "Matriz", que represente a una matriz de 3x3, con métodos para indicar el valor que hay en una posición, leer el valor de una posición, escribir la matriz en pantalla y sumar dos matrices. (Nota: en C# puedes sobrecargar el operador "+", pero no el operador "[]", de modo que tendrás que crear métodos "get" y "set" para leer los valores de posiciones de la matriz y para cambiar su valor).

```
using System;

namespace _7._6._1
{
    class Program
    {
        static void Main(string[] args)
        {
            Matriz matriz1 = new Matriz();
            Matriz matriz2 = new Matriz();

            matriz1.Set(0, 0, 1);
            matriz1.Set(0, 1, 1);
            matriz1.Set(0, 2, 1);
            matriz1.Set(1, 0, 1);
            matriz1.Set(1, 1, 1);
            matriz1.Set(1, 2, 1);
            matriz1.Set(2, 0, 1);
            matriz1.Set(2, 1, 1);
            matriz1.Set(2, 2, 1);
```

```
            matriz2.Set(0, 0, 2);
            matriz2.Set(0, 1, 2);
            matriz2.Set(0, 2, 2);
            matriz2.Set(1, 0, 2);
            matriz2.Set(1, 1, 2);
            matriz2.Set(1, 2, 2);
            matriz2.Set(2, 0, 2);
            matriz2.Set(2, 1, 2);
            matriz2.Set(2, 2, 2);

            Console.WriteLine("La matriz 1 es \n" + matriz1);
            Console.WriteLine("La matriz 2 es \n " + matriz2);
            Console.WriteLine("La suma de ambas es \n" + (matriz1 + matriz2));
            Console.ReadKey();
        }
    }
}
using System;

namespace _7._6._1
{
    class Matriz
    {
        private int[][] matriz;

        public Matriz()
        {
            matriz = new int[3][];
            for (int i = 0; i < 3; i++)
                matriz[i] = new int[3];
        }

        public int Get(int x, int y)
        {
            return matriz[x][y];
        }

        public void Set(int x, int y, int valor)
        {
            matriz[x][y] = valor;
        }

        public override string ToString()
        {
            string resultado = "";
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                    resultado += " " + matriz[i][j];
                resultado += "\n";
            }
            return resultado;
        }

        public static Matriz operator + (Matriz m1, Matriz m2)
        {
            Matriz resultado = new Matriz();
            for (int i=0; i<3; i++)
            {
                for (int j=0; j<3; j++)
                {
```

```
                    resultado.Set(i, j, m1.Get(i, j) + m2.Get(i, j));
                }
            }
            return resultado;
        }
    }
}
```

(7.6.4) Crea una clase "Vector3", que represente a un vector en el espacio de 3 dimensiones. Redefine los operadores "+" y "-" para sumar y restar dos vectores, "*" para hallar el producto escalar de dos vectores y "%" para calcular su producto vectorial.

```
using System;

namespace _7._6._4
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, z;

            Console.Write("Introduce la coordenada x del primer vector: ");
            x = Convert.ToInt32(Console.ReadLine());
            Console.Write("Introduce la coordenada y del primer vector: ");
            y = Convert.ToInt32(Console.ReadLine());
            Console.Write("Introduce la coordenada z del primer vector: ");
            z = Convert.ToInt32(Console.ReadLine());

            Vector3 v1 = new Vector3(x, y, z);

            Console.WriteLine("Vector 1 = ({0}, {1}, {2})", x, y, z);

            Console.Write("Introduce la coordenada x del segundo vector: ");
            x = Convert.ToInt32(Console.ReadLine());
            Console.Write("Introduce la coordenada y del segundo vector: ");
            y = Convert.ToInt32(Console.ReadLine());
            Console.Write("Introduce la coordenada z del segundo vector: ");
            z = Convert.ToInt32(Console.ReadLine());

            Vector3 v2 = new Vector3(x, y, z);

            Console.WriteLine("Vector 2 = ({0}, {1}, {2})", x, y, z);

            Console.WriteLine("El resultado de sumar el vector 1 y el vector 2
es: {0}", v1 + v2);
            Console.WriteLine("El resultado de restar el vector 1 y el vector 2
es: {0}", v1 - v2);
            Console.WriteLine("El resultado del producto escalar del vector 1 y
el vector 2 es: {0}", v1 * v2);
            Console.WriteLine("El resultado del producto vectorial del vector 1 y
el vector 2 es: {0}", v1 % v2);
            Console.ReadLine();
        }
    }
}


using System;

namespace _7._6._4
```

```
{
    class Vector3
    {
        int x, y, z;

        public Vector3 (int x, int y, int z)
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }

        public static Vector3 operator + (Vector3 v1, Vector3 v2)
        {
            return new Vector3(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);
        }

        public static Vector3 operator - (Vector3 v1, Vector3 v2)
        {
            return new Vector3(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);
        }

        public static Vector3 operator * (Vector3 v1, Vector3 v2)
        {
            return new Vector3(v1.x * v2.x, v1.y * v2.y, v1.z * v2.z);
        }

        public static Vector3 operator % (Vector3 v1, Vector3 v2)
        {
            return new Vector3(v1.y * v2.z - v1.z * v2.y,
                v1.x * v2.z - v1.z * v2.x,
                v1.x * v2.y - v1.y * v2.x);
        }

        public override string ToString()
        {
            return string.Format("({0}, {1}, {2})", x, y, z);
        }
    }
}
```