

1) Definir una clase **Persona** con 3 campos públicos: **Nombre**, **Edad** y **DNI**. Escribir un algoritmo que permita al usuario ingresar en una consola una serie de datos de esta forma:

Nombre, Documento, Edad <ENTER>.

Una vez finalizada la entrada de datos, el programa debe imprimir en la consola un listado con 4 columnas de la siguiente forma:

	Nro)	Nombre	Edad	DNI.
--	-------------	---------------	-------------	-------------

Ejemplo de listado por consola:

1)	Juan Perez	40	2098745
2)	José García	41	1965412
...			

NOTA: Se puede utilizar: `Console.SetIn(new System.IO.StreamReader(nombreDeArchivo));` para cambiar la entrada estándar de la consola y poder leer directamente desde un archivo de texto.

```
using teoria4;

Persona p = new Persona ();

p.dni= 4;
p.Nombre = "a";
p.edad = 40;

Console.WriteLine(p.Nombre);
```

2) Modificar el programa anterior haciendo privados todos los campos. Definir un constructor adecuado y un método público **Imprimir()** que escriba en la consola los campos del objeto con el formato requerido para el listado.

```
public class Persona
{
    private String? Nombre;
    private int edad,dni;

    public Persona (String nombre, int edad,int dni){
        this.Nombre = nombre;
        this.edad = edad;
        this.dni = dni;
    }

    public void Imprimir (){
```

```

        Console.WriteLine("NOMBRE : " + Nombre + " EDAD : " + edad + "DNI" + dni);
    }

```

3) Agregar a la clase **Persona** un método **EsMayorQue(Persona p)** que devuelva verdadero si la persona que recibe el mensaje tiene más edad que la persona enviada como parámetro. Utilizarlo para realizar un programa que devuelva la persona más joven de la lista.

108

```

using teoria4;

Persona p = new Persona ("PERRA DE FEDE",55,445);
Persona p2=new Persona ("LEAN",18,34005);

if (p.esMayorQue(p2))
    Console.WriteLine("ES MAYOR");
else
    Console.WriteLine("NO ES MAYOR");

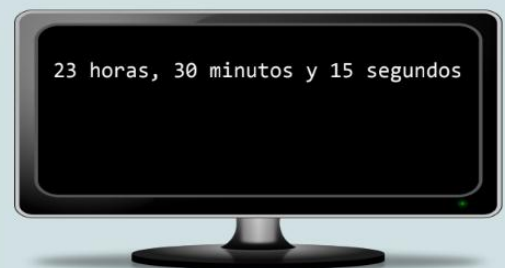
```

4) Codificar la clase **Hora** de tal forma que el siguiente código produzca la salida por consola que se observa.

```

Hora h = new Hora (23,30,15);
h.Imprimir

```



```

using System;

namespace teoria4;

public class Hora
{
    private int horas;
    private int minutos;

    private int segundos;

    public Hora (int horas,int minutos,int segundos){
        this.horas = horas;
        this.minutos = minutos;
    }
}

```

```

        this.segundos = segundos;
    }

    public void Imprimir (){
        Console.WriteLine(this.horas + " horas " + this.minutos + " minutos " +
this.segundos+"segundos");
    }
}


```

5) Agregar un segundo constructor a la clase **Hora** para que pueda especificarse la hora por medio de un único valor que admita decimales. Por ejemplo 3,5 indica la hora 3 y 30 minutos. Si se utiliza este segundo constructor, el método imprimir debe mostrar los segundos con tres dígitos decimales. Así el siguiente código debe producir la salida por consola que se observa.

```

new Hora(23, 30, 15).Imprimir();
new Hora(14.43).Imprimir();
new Hora(14.45).Imprimir();
new Hora(14.45114).Imprimir();

```



```

23 horas, 30 minutos y 15 segundos
14 horas, 25 minutos y 48,000 segundos
14 horas, 27 minutos y 0,000 segundos
14 horas, 27 minutos y 4,104 segundos

```

```

using System;

namespace teoria4;

public class Hora
{
    private int horas;
    private double minutos;

    private double segundos;

    public Hora (int horas,int minutos,int segundos){
        this.horas = horas;
        this.minutos = minutos;
        this.segundos = segundos;
    }

    public Hora (double horas){ //14,45
        this.horas = (int)horas; //14
        double parteDecimal = horas-this.horas; //14.45-14 = 0.45
        this.minutos= Math.Round(parteDecimal*60); // 26
    }
}

```

```

        this.segundos =(this.minutos-(parteDecimal*60))*60;
        this.segundos= Math.Round((this.segundos*100) /100);
    }

    public void Imprimir (){
        Console.WriteLine(this.horas + " horas " + this.minutos + " minutos " +
this.segundos+"segundos");
    }
}

```

6) Codificar una clase llamada **Ecuacion2** para representar una ecuación de 2º grado. Esta clase debe tener 3 campos privados, los coeficientes a, b y c de tipo **double**. La única forma de establecer los valores de estos campos será en el momento de la instanciación de un objeto **Ecuacion2**.

Codificar los siguientes métodos:

- **GetDiscriminante()**: devuelve el valor del discriminante (**double**), el discriminante tiene la siguiente formula, $(b^2)-4*a*c$.
- **GetCantidadDeRaices()**: devuelve 0, 1 ó 2 dependiendo de la cantidad de raíces reales que posee la ecuación. Si el discriminante es negativo no tiene raíces reales, si el discriminante es cero tiene una única raíz, si el discriminante es mayor que cero posee 2 raíces reales.
- **ImprimirRaices()**: imprime la única o las 2 posibles raíces reales de la ecuación. En caso de no poseer soluciones reales debe imprimir una leyenda que así lo especifique.

```

using System;

namespace teoria4;

public class Ecuacion2
{
    private Double _a;
    private Double _b;

    private Double _c;

    public Ecuacion2 (double a,double b,double c){
        this._a = a;
        this._b=b;
        this._c=c;
    }

    public double getDiscriminante(){
        return (Math.Pow(this._b,2))-(4*this._a*this._c);
    }

    public int getCantidadDeRaices(){
        return getDiscriminante() switch {>0 => 2, <0 => 0, _ => 1};
    }
}

```

```

    }

    public void ImprimirRaices(){
        int aux = this.getCantidadDeRaices();
        if (aux == 2){
            double x1 = ((-this._b +
Math.Sqrt(this.getDiscriminante()))/(2*this._a));
            double x2 = ((-this._b -
Math.Sqrt(this.getDiscriminante()))/(2*this._a));
            Console.WriteLine(" X1: "+x1+"    X2: "+x2);
        }
        else if (aux == 1){
            double x = (-this._b)/(2*this._a);
            Console.WriteLine("La raiz de la ecuacion es: " + x);
        }
        else
            Console.WriteLine("No posee soluciones reales");
    }
}

```

7) Implementar la clase **Matriz** que se utilizará para trabajar con matrices matemáticas. Implementar los dos constructores y todos los métodos que se detallan a continuación:

```

public Matriz(int filas, int columnas)
public Matriz(double[,] matriz)
public void SetElemento(int fila, int columna, double elemento)
public double GetElemento(int fila, int columna)
public void imprimir()
public void imprimir(string formatString)
public double[] GetFila(int fila)
public double[] GetColumna(int columna)
public double[] GetDiagonalPrincipal()
public double[] GetDiagonalSecundaria()
public double[][] getArregloDeArreglo()
public void sumarle(Matriz m)
public void restarle(Matriz m)
public void multiplicarPor(Matriz m)

```

```

using System;
using System.Dynamic;
using System.Runtime.CompilerServices;

namespace teoria4;

public class Matriz
{
    private double [,] matriz;
    public Matriz (int fila,int columnas){

```

```

        this.matriz = new double [fila,columnas];
    }
    public Matriz (double [,] matriz){
        this.matriz = matriz;
    }
    public void SetElemento(int fila,int columna,double elemento){
        this.matriz[fila,columna] = elemento;
    }
    public double getElemento (int fila,int columna){
        return this.matriz [fila,columna];
    }

    public void imprimir(){
        for (int i=0;i<this.matriz.GetLength(0)*this.matriz.GetLength(1);i++){
            Console.Write(this.matriz[i/this.matriz.GetLength(1),i%this.matriz.Ge
tLength(1)]);
            Console.Write(i%this.matriz.GetLength(1)==this.matriz.GetLength(1)-1
? "\n" : " ");
        }
    }

    public void imprimir(String stringFormat){
        for (int i=0;i<this.matriz.GetLength(0)*this.matriz.GetLength(1);i++){
            Console.Write(this.matriz[i/this.matriz.GetLength(1),i%this.matriz.Ge
tLength(1)].ToString(stringFormat));
            Console.Write(i%this.matriz.GetLength(1)==this.matriz.GetLength(1)-1
? "\n" : " ");
        }
    }

    public double [] GetFila (int fila){
        double [] f = new double [this.matriz.GetLength(1)];
        for (int i=0;i<this.matriz.GetLength(1);i++){
            f[i] = this.matriz[fila,i];
        }
        return f;
    }

    public double [] GetColumnas (int columna){
        double [] c = new double [this.matriz.GetLength(0)];
        for (int i =0;i<this.matriz.GetLength(0);i++){
            c[i]=this.matriz[i,columna];
        }
        return c;
    }

    public double [] getDiagonalPrincipal(){
        int f = this.matriz.GetLength(1);

```

```

        double [] v = new double [f];
        for (int i=0;i<this.matriz.GetLength(1);i++){
            v[i] = matriz[i,i];
        }
        return v;
    }

    public double [] getDiagonalSecundaria(){
        int f = this.matriz.GetLength(1);
        double [] v = new double [f];
        int j=this.matriz.GetLength(1)-1;
        for (int i=0;i<this.matriz.GetLength(1);i++){
            v[i] = matriz[j--,i];
        }
        return v;
    }

    public void sumarle (Matriz m){
        for (int i=0;i<this.matriz.GetLength(0);i++){
            for (int j=0;j<this.matriz.GetLength(1);j++){
                this.matriz[i,j] += m.getElemento(i,j);
            }
        }
    }

    public void restarle (Matriz m){
        for (int i=0;i<this.matriz.GetLength(0);i++){
            for (int j=0;j<this.matriz.GetLength(1);j++){
                this.matriz[i,j] -= m.getElemento(i,j);
            }
        }
    }

    public void multiplicar (Matriz m){
        if (this.matriz.GetLength(1) != this.matriz.GetLength(0))
            Console.WriteLine("LA MATRIZ NO SE PUEDE MULTIPLICAR");
        else
        {
            Matriz mat = new Matriz
(this.matriz.GetLength(0),this.matriz.GetLength(0));
            //F NO TENGO GANAS
        }
    }
}

```

8) Prestar atención a los siguientes programas (ninguno funciona correctamente):

```
Foo f = new Foo();
f.Imprimir();

class Foo
{
    string? _bar;
    public void Imprimir()
    {
        Console.WriteLine(_bar.Length);
    }
}
```

```
Foo f = new Foo();
f.Imprimir();

class Foo
{
    public void Imprimir()
    {
        string? bar;
        Console.WriteLine(bar.Length);
    }
}
```

¿Qué se puede decir acerca de la inicialización de los objetos? ¿En qué casos son inicializados automáticamente y con qué valor?

9) ¿Qué se puede decir en relación a la sobrecarga de métodos en este ejemplo?

```
class A
{
    public void Metodo(short n) {
        Console.WriteLine("short {0} - ", n);
    }
    public void Metodo(int n) {
        Console.WriteLine("int {0} - ", n);
    }
    public int Metodo(int n) {
        return n + 10;
    }
}
```

8) se debería cargar bar con null y preguntar si el valor es null y devolver una excepción

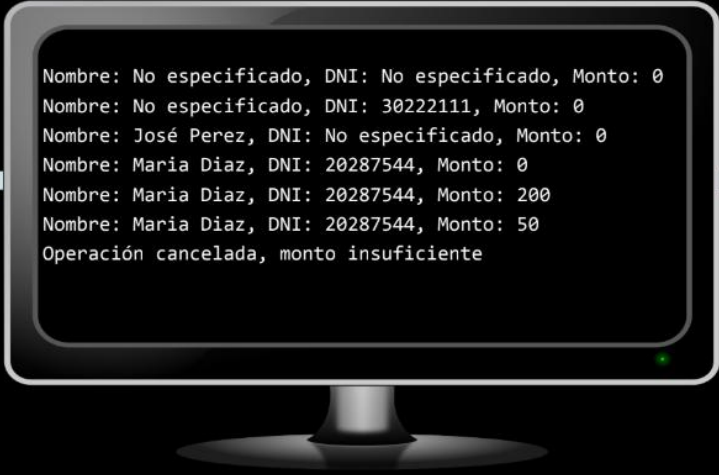
9) El método está clonado con mismo nombre y mismos parámetros por ende no funcionará.

Práctica sobre la teoría 4

10) Completar la clase **Cuenta** para que el siguiente código produzca la salida indicada:

```
Cuenta cuenta = new Cuenta();
cuenta.Imprimir();
cuenta = new Cuenta(30222111);
cuenta.Imprimir();
cuenta = new Cuenta("José Perez");
cuenta.Imprimir();
cuenta = new Cuenta("Maria Diaz", 20287544);
cuenta.Imprimir();
cuenta.Depositar(200);
cuenta.Imprimir();
cuenta.Extraer(150);
cuenta.Imprimir();
cuenta.Extraer(1500);
```

```
class Cuenta
{
    private double _monto;
    private int _titularDNI;
    private string? _titularNobre;
    . . .
}
```



```
Nombre: No especificado, DNI: No especificado, Monto: 0
Nombre: No especificado, DNI: 30222111, Monto: 0
Nombre: José Perez, DNI: No especificado, Monto: 0
Nombre: Maria Diaz, DNI: 20287544, Monto: 0
Nombre: Maria Diaz, DNI: 20287544, Monto: 200
Nombre: Maria Diaz, DNI: 20287544, Monto: 50
Operación cancelada, monto insuficiente
```

Utilizar en la medida de lo posible la sintaxis **:this** en el encabezado de los constructores para invocar a otro constructor ya definido.

112

```
using System;
using System.Runtime.InteropServices;

namespace teoria4;

public class Cuenta
{
    private double _monto;
    private int _titularDNI;
    private string? _titularNobre;

    public Cuenta (){
        this._monto = 0;
        this._titularDNI=0;
        this._titularNobre="Sin Especificar";
    }
    public Cuenta (int dni):this(){
        _titularDNI=dni;
    }
    public Cuenta(String nombre):this(){
        _titularNobre = nombre;
    }
}
```

```
}

public Cuenta (String nom , int dni):this(){
    this._titularDNI=dni;
    this._titularNobre=nom;
}

public void Imprimir (){
    Console.WriteLine ($"NOMBRE : {this._titularNobre}DNI :
{this._titularDNI}MONTO : {this._monto}");
}

public void Depositar(double p){
    this._monto += p;
    Console.WriteLine("DEPOSITO REALIZADO...");
}

public void Extraer (double p){
    if (this._monto >p) this._monto-=p;
    else
        Console.WriteLine("Fondos insuficientes...");
}
}
```

11) Reemplazar estas líneas de código por otras equivalentes que utilicen el operador null-coalescing (??) y / o la asignación null-coalescing (??=)

```
...
if (st1 == null)
{
    if (st2 == null)
    {
        st = st3;
    }
    else
    {
        st = st2;
    }
}
else
{
    st = st1;
}
if (st4 == null)
{
    st4 = "valor por defecto";
}
...
```

```
st = st1 ?? st2 ?? st3 ;    //si st1 es null va a st2 y pregunta si no es null y
                             //si no lo es va st3 y lo define
st4 ??= "ValorPorDefecto";  //pregunta si st4 no es null si lo es pone
                             //valorPorDefecto.
```

12) Crear una solución con tres proyectos: una biblioteca de clases llamada **Automotores**, una biblioteca de clases llamada **Figuras** y una aplicación de consola llamada **Aplicacion**. La biblioteca **Automotores** debe contener una clase pública **Auto** (codificada de la misma manera que la vista en la teoría). La biblioteca **Figuras** debe contener las clases públicas **Circulo** y **Rectangulo**, codificadas de tal forma que el siguiente código (escrito en **Program.cs** del proyecto **Aplicacion**) produzca la siguiente salida

```
using Figuras;
using Automotores;

//El constructor de Circulo espera recibir el radio
List<Circulo> listaCirculos = [
    new Circulo(1.1),
    new Circulo(3),
    new Circulo(3.2)
];

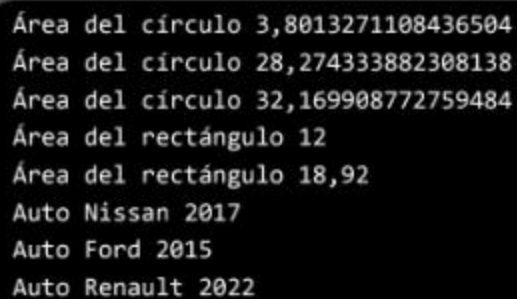
//El constructor de Rectángulo espera recibir la base y la altura
List<Rectangulo> listaRectangulos = [
    new Rectangulo(3, 4),
    new Rectangulo(4.3, 4.4)
];

//La clase Auto está codificada como la vista en la teoría
List<Auto> listaAutos = [
    new Auto("Nissan", 2017),
    new Auto("Ford", 2015),
    new Auto("Renault")
];
```

```

foreach (Circulo c in listaCirculos)
{
    Console.WriteLine($"Área del círculo {c.GetArea()}");
}
foreach (Rectangulo r in listaRectangulos)
{
    Console.WriteLine($"Área del rectángulo {r.GetArea()}");
}
foreach (Auto a in listaAutos)
{
    Console.WriteLine(a.GetDescripcion());
}

```



```

Área del círculo 3,8013271108436504
Área del círculo 28,274333882308138
Área del círculo 32,169908772759484
Área del rectángulo 12
Área del rectángulo 18,92
Auto Nissan 2017
Auto Ford 2015
Auto Renault 2022

```

```

using System;

namespace teoria4;

public class Circulo
{
    private double _radio;
    public Circulo (double r){
        this._radio = r;
    }

    public double GetArea(){
        return Math.PI*(this._radio*this._radio);
    }
}

```

```
using System;

namespace teoria4;

public class Rectangulo
{
    private double _base;
    private double _altura;

    public Rectangulo (double _base, double _altura) {
        this._base = _base;
        this._altura = _altura;
    }

    public double GetArea (){
        return this._base * this._altura;
    }
}
```

```
using System;

namespace teoria4;

public class Auto
{
    private string? _marca;
    private int _modelo;
    public Auto (String Marca){
        this._marca = Marca;
    }
    public Auto(string marca, int modelo) : this (marca)
    {
        _modelo = modelo;
    }
    public string GetDescripcion() => $"Auto {_marca} {_modelo}";
}
```