

Animal.cs

```
using System;
namespace ClaseAbstracta.Cobaya {
```

```
public abstract class Animal {
    public string Nombre { get; set; }
    public string Especie { get; set; }
```

```
// Constructor sin parámetros
```

```
protected Animal() {
    Nombre = "Sin nombre";
    Especie = "Desconocido";
```

```
}
```

```
// Métodos abstractos
```

```
public abstract string HacerSonido();
public abstract string Movise();
```

```
public virtual void Mostrar() {
```

```
    console.WriteLine($"{{Nombre}} - {{Especie}}");
```

```
}
```

```
public class Perro : Animal {
```

```
public Perro() : base() {
```

```
    Nombre = "Rex";
```

```
    Especie = "Perro";
```

```
}
```

```
public override string HacerSonido() {
```

```
    return "Guau guau!";
```

```
}
```

```
public override string Movise() {
```

```
    return "Corriendo en 4 patas";
```

```
}
```

```
public override void Mostrar() {
```

```
    base.Mostrar();
```

```
    console.WriteLine($"Sonido: {{Hacer sonido ()}}");
```

```
    console.WriteLine($"Movimiento: {{Movise()}}");
```

```
}
```

```
class Program {
```

```
static void Main(string [] args) {
```

```
    Animal perro = new Perro();
```

```
    perro.Mostrar();
```

```
}
```

```
}
```

```
}
```

```

Documento.cs
using System;
namespace ClaseAbstracta.CSharp {
    public abstract class Documento {
        public string Titulo { get; set; }
        public string Autor { get; set; }
        public abstract void Imprimir();
        public abstract void Guardar();
        public abstract void Mostrar();
    }

    public class Reporte : Documento {
        public string Contenido { get; set; }
        public DateTime Fecha { get; set; }
        public bool EsConfidencial { get; set; }

        // Constructor con parámetros
        public Reporte(string Titulo, string Autor, string contenido, DateTime Fecha, bool esConfidencial) {
            Titulo = Titulo;
            Autor = Autor;
            Contenido = contenido;
            Fecha = Fecha;
            EsConfidencial = esConfidencial;
        }

        public override void Imprimir() {
            string confidencial = EsConfidencial ? "Confidencial" : "Público";
            Console.WriteLine($"Imprimiendo reporte: [{Titulo}] ({Confidencial})");
        }

        public override void Guardar() {
            Console.WriteLine($"Guardando reporte [{Titulo}]");
        }

        public override void Mostrar() {
            string confidencial = EsConfidencial ? "Sí" : "No";
            Console.WriteLine($"Reporte: [{Titulo}]");
            Console.WriteLine($"Autor: [{Autor}]");
            Console.WriteLine($"Fecha: [{Fecha : dd/MM/yyyy}]");
            Console.WriteLine($"Confidencial: [{confidencial}]");
            Console.WriteLine($"Contenido: [{Contenido}]. Substring(0, Math.Min(50, Contenido.Length)}");
        }
    }

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

'Documento reporte = new Reporte C
 "Reporte Trimestral",
 "Agosto 2024";
 "Este es el contenido completo del reporte trimestral que detalla...",
 new DateTime(2024, 1, 15),
 true
)';

reporte.Mostrar();
reporte.Titular();
reporte.Imprimir();

----- O ----- D ----- O ----- O ----- O -----
Empleado.cs

using System;

```
namespace ClaseAbstracta.Csharp {  
    public abstract class Empleado {  
        public string Nombre { get; set; }  
        public abstract string Trabajar();  
        public abstract double CalcularSalario();  
        public abstract void Mostrar();  
    }  
}
```

```
public class Empleado : Empleado {  
    public string Departamento { get; set; }  
    public double Salario { get; set; }  
}
```

```
// Constructor con parámetros  
public Empleado(string nombre, string departamento, double salario) {  
    Nombre = nombre;  
    Departamento = departamento;  
    Salario = salario;  
}  
}
```

```
// Constructor de copia  
public Empleado(Empleado other) {  
    Nombre = other.Nombre;  
    Departamento = other.Departamento;  
    Salario = other.Salario;  
}  
}
```

```
public override string Trabajar() {  
    return $"Estoyiendo el departamento {Departamento}.";  
}  
}
```

```
public override double CalcularSalario() {
```

3) rotar Salario:

```
public override void Mostral () {
    console.WriteLine ($"Empleado : {Nombre} ");
    console.WriteLine ($"Departamento : {Departamento}, Salario : ${Salario:F2} ");
    console.WriteLine ($"Trabajo : {Trabajo} ");
}
```

3)

```
class Program {
```

```
    static void Main (string [] args) {
        Empleado original = new Empleado ("Carlos", "Ventas", 75000.00);
        Empleado copia = new Empleado (original);

        original.Mostral ();
        copia.Mostral ();
    }
}
```

3)

Figura.cs

using System;

```
namespace ClaseAbstracta.Csharp {
    public abstract class Figura {
        public string color { get; set; }

        protected Figura () {
            color = "negro";
        }
    }
}
```

```
    public abstract double CalcularArea ();
    public abstract double CalcularPerimetro ();
}
```

```
    public virtual void Mostral () {
        console.WriteLine ($"color : {color} ");
    }
}
```

3)

```
public class Cuadrado : Figura {
    public double Lado { get; set; }

    public Cuadrado () : base () {
        Lado = 1.0;
    }
}
```

```
    public override double CalcularArea () {
        return Lado * Lado;
    }
}
```

```
public override double CalcularPerimetro(){
```

```
    base.Mostrar();
```

```
    Console.WriteLine($"Cuadrado - Lado: {Lado}");
```

```
    Console.WriteLine($"Área: {CalcularArea()}");
```

```
    Console.WriteLine($"Perímetro: {CalcularPerimetro()}");
```

```
}
```

```
class Program{
```

```
    static void Main(string[] args){
```

```
        Figura cuadrado = new Cuadrado();
```

```
        cuadrado.Mostrar();
```

```
}
```

```
0 -
```

```
0 -
```

```
0 -
```

```
Vehiculo.cs
```

```
using System;
```

```
namespace ClaseAbstracta.Csharp{
```

```
    public abstract class Vehiculo{
```

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

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

```
        public abstract void Encender();
```

```
        public abstract void Apagar();
```

```
        public abstract void Mostrar();
```

```
3
```

```
public class Automovil : Vehiculo{
```

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

```
    public double Precio {get; set;}
```

```
    public bool Encendido {get; set;}
```

```
// Constructor 1: solo marca, modelo
```

```
public Automovil(string marca, string modelo){
```

```
    Marca = marca;
```

```
    Modelo = modelo;
```

```
    Año = DateTime.Now.Year;
```

```
    Precio = 0.0;
```

```
    Encendido = false;
```

```
3
```

```
// Constructor 2: Todos los parámetros
```

```
public Automovil(string marca, string modelo, int año, double precio, bool encendido){
```

```
    Marca = marca;
```

```
    Modelo = modelo;
```

```
    Año = año;
```

Precio = precio;
Encendido = encendido;

}

```
public override void Encender () {  
    Encendido = true;  
    Console.WriteLine ("Automóvil encendido");  
}
```

```
public override void Apagar () {  
    Encendido = false;  
    Console.WriteLine ("Automóvil apagado");  
}
```

```
public override void Mostrar () {  
    string estado = Encendido ? "Encendido" : "Apagado";  
    Console.WriteLine ($"{{Modelo}} {{Año}} - $ {{Precio: F2.3}}");  
    Console.WriteLine ($"Estado: {{Estado}}");  
}
```

}

```
class Program {  
    static void Main (string [] args) {  
        Vehiculo auto 1 = new Automóvil ("Toyota", "Corolla");  
        Vehiculo auto 2 = new Automóvil ("Honda", "Civic", 2023, 25000.00, false);  
  
        auto 1. Mostrar ();  
        auto 2. Mostrar ();  
  
        auto 2. Encender ();  
        auto 2. Mostrar ();  
    }  
}
```

3