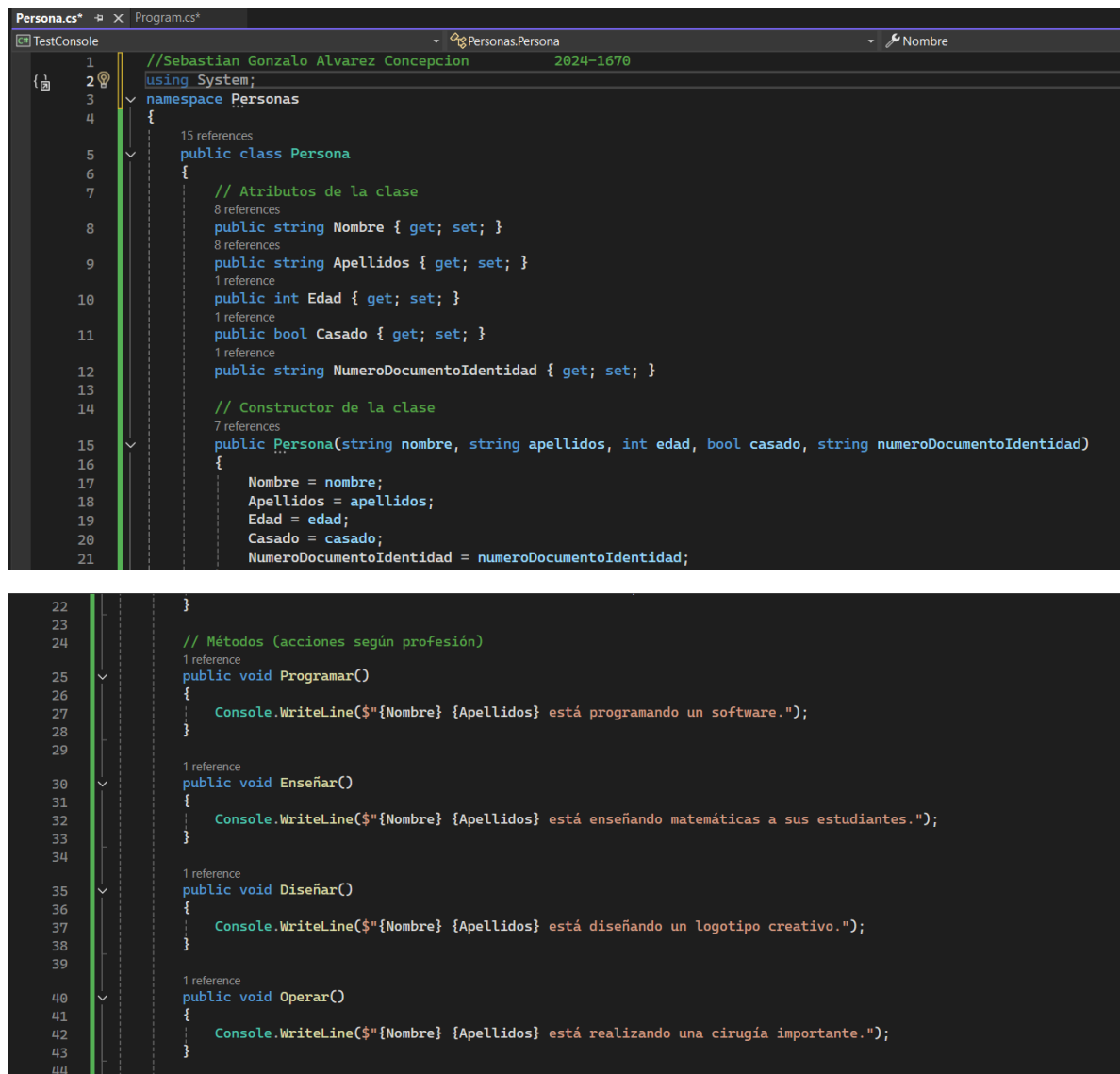


1. Considera estás desarrollando un programa donde necesitas trabajar con objetos de tipo Persona. Define una clase Persona, pero en este caso considerando los siguientes atributos de clase: nombre (String), apellidos (String), edad (int), casado (boolean), numeroDocumentoIdentidad(String) y 3 metodos como acciones diferentes por persona de acuerdo a una profesión. Define un constructor y los métodos para poder establecer y obtener los valores de los atributos. Mínimo 7 personas diferentes con acciones diferentes.



```
1 //Sebastian Gonzalo Alvarez Concepcion 2024-1670
2 using System;
3 namespace Personas
4 {
5     15 references
6     public class Persona
7     {
8         // Atributos de la clase
9         8 references
10        public string Nombre { get; set; }
11        8 references
12        public string Apellidos { get; set; }
13        1 reference
14        public int Edad { get; set; }
15        1 reference
16        public bool Casado { get; set; }
17        1 reference
18        public string NumeroDocumentoIdentidad { get; set; }
19
20        // Constructor de la clase
21        7 references
22        public Persona(string nombre, string apellidos, int edad, bool casado, string numeroDocumentoIdentidad)
23        {
24            Nombre = nombre;
25            Apellidos = apellidos;
26            Edad = edad;
27            Casado = casado;
28            NumeroDocumentoIdentidad = numeroDocumentoIdentidad;
29
30            // Métodos (acciones según profesión)
31            1 reference
32            public void Programar()
33            {
34                Console.WriteLine($"{Nombre} {Apellidos} está programando un software.");
35            }
36
37            1 reference
38            public void Enseñar()
39            {
40                Console.WriteLine($"{Nombre} {Apellidos} está enseñando matemáticas a sus estudiantes.");
41            }
42
43            1 reference
44            public void Diseñar()
45            {
46                Console.WriteLine($"{Nombre} {Apellidos} está diseñando un logotipo creativo.");
47            }
48
49            1 reference
50            public void Operar()
51            {
52                Console.WriteLine($"{Nombre} {Apellidos} está realizando una cirugía importante.");
53            }
54        }
55    }
56 }
```

```
45 1 reference
46 public void Cocinar()
47 {
48     Console.WriteLine($"{Nombre} {Apellidos} está preparando una comida deliciosa.");
49 }
50 1 reference
51 public void Construir()
52 {
53     Console.WriteLine($"{Nombre} {Apellidos} está construyendo una casa.");
54 }
55 1 reference
56 public void Pintar()
57 {
58     Console.WriteLine($"{Nombre} {Apellidos} está pintando un paisaje hermoso.");
59 }
60 }

61 0 references
62 class Program
63 {
64     0 references
65     static void Main(string[] args)
66     {
67         // Creación de 7 objetos Persona con profesiones diferentes
68         Persona programador = new Persona("Luis", "Gómez", 30, false, "12345678");
69         Persona maestro = new Persona("Maria", "Pérez", 40, true, "87654321");
70         Persona diseñador = new Persona("Carlos", "Ramírez", 28, false, "11223344");
71         Persona doctor = new Persona("Ana", "Torres", 35, true, "22334455");
72         Persona chef = new Persona("Jorge", "Hernández", 45, true, "33445566");
73         Persona albañil = new Persona("Pedro", "Martínez", 50, false, "44556677");
74         Persona pintor = new Persona("Sofia", "López", 25, false, "55667788");
75
76         // Llamando a las acciones de cada persona
77         programador.Programar();
78         maestro.Enseñar();
79         diseñador.Diseñar();
80         doctor.Operar();
81         chef.Cocinar();
82         albañil.Construir();
83         pintor.Pintar();
84     }
85 }
```

2. Crea una clase Cuenta con los métodos ingreso, reintegro y transferencia. La clase contendrá un constructor por defecto, un constructor con parámetros y los métodos getters y setters para mostrar e ingresar.

```
Cuenta.cs  Program.cs
TestConsole  BancoApp.Cuenta  NumeroCuenta

1 //Sebastian Gonzalo Alvarez Concepcion 2024-1670
2 using System;
3 namespace BancoApp
4 {
5     7 references
6     public class Cuenta
7     {
8         // Atributos de la clase
9         2 references
10        public string NumeroCuenta { get; set; }
11        7 references
12        public string Titular { get; set; }
13        14 references
14        public decimal Saldo { get; private set; }
15
16        // Constructor por defecto
17        0 references
18        public Cuenta()
19        {
20            NumeroCuenta = "Sin asignar";
21            Titular = "Sin asignar";
22            Saldo = 0.0m;
23        }
24
25        // Constructor con parámetros
26        2 references
27        public Cuenta(string numeroCuenta, string titular, decimal saldoInicial)
28        {
29        }
```

```
23        NumeroCuenta = numeroCuenta;
24        Titular = titular;
25        Saldo = saldoInicial >= 0 ? saldoInicial : throw new ArgumentException("El saldo inicial no puede ser negativo.");
26    }
27
28    // Método para realizar un ingreso
29    2 references
30    public void Ingreso(decimal monto)
31    {
32        if (monto <= 0)
33        {
34            Console.WriteLine("El monto a ingresar debe ser positivo.");
35            return;
36        }
37
38        Saldo += monto;
39        Console.WriteLine($"Se han ingresado {monto:C}. Nuevo saldo: {Saldo:C}");
40    }
41
42    // Método para realizar un reintegro (retiro)
43    1 reference
44    public void Reintegro(decimal monto)
45    {
46        if (monto <= 0)
47        {
48            Console.WriteLine("El monto a retirar debe ser positivo.");
49            return;
50        }
```

```
48        }
49
50        if (monto > Saldo)
51        {
52            Console.WriteLine("Fondos insuficientes para realizar el retiro.");
53            return;
54        }
55
56        Saldo -= monto;
57        Console.WriteLine($"Se han retirado {monto:C}. Nuevo saldo: {Saldo:C}");
58    }
59
60    // Método para realizar una transferencia
61    1 reference
62    public void Transferencia(Cuenta destino, decimal monto)
63    {
64        if (monto <= 0)
65        {
66            Console.WriteLine("El monto a transferir debe ser positivo.");
67            return;
68        }
69
70        if (monto > Saldo)
71        {
72            Console.WriteLine("Fondos insuficientes para realizar la transferencia.");
73            return;
74        }
```

```

74         Saldo -= monto;
75         destino.Ingreso(monto);
76         Console.WriteLine($"Se han transferido {monto:C} a la cuenta de {destino.Titular}. Nuevo saldo: {Saldo:C}");
77     }
78 }
79
80
81 0 references
82 class Program
83 {
84     0 references
85     static void Main(string[] args)
86     {
87         // Crear cuentas
88         Cuenta cuenta1 = new Cuenta("12345678", "Juan Pérez", 1000.00m);
89         Cuenta cuenta2 = new Cuenta("87654321", "Ana López", 500.00m);
90
91         // Mostrar saldos iniciales
92         Console.WriteLine($"Saldo inicial de {cuenta1.Titular}: {cuenta1.Saldo:C}");
93         Console.WriteLine($"Saldo inicial de {cuenta2.Titular}: {cuenta2.Saldo:C}");
94
95         // Realizar operaciones
96         cuenta1.Ingreso(200.00m); // Ingresar dinero en cuenta1
97         cuenta1.Reintegro(150.00m); // Retirar dinero de cuenta1
98         cuenta1.Transferencia(cuenta2, 300.00m); // Transferir dinero de cuenta1 a cuenta2
99
100        // Mostrar saldos finales
101        Console.WriteLine($"Saldo final de {cuenta1.Titular}: {cuenta1.Saldo:C}");
102
103        // Mostrar saldos finales
104        Console.WriteLine($"Saldo final de {cuenta1.Titular}: {cuenta1.Saldo:C}");
105        Console.WriteLine($"Saldo final de {cuenta2.Titular}: {cuenta2.Saldo:C}");
106    }
107 }

```

3. Crea una clase Contador con los métodos para incrementar y decrementar el contador. La clase contendrá un constructor por defecto, un constructor con parámetros, y los métodos getters y setters.

```

1 //Sebastian Gonzalo Alvarez Concepcion      2024-1670
2 using System;
3
4 namespace ContadorApp
5 {
6     6 references
7     public class Contador
8     {
9         // Atributo de la clase
10        private int _valor;
11
12        // Propiedad para acceder al atributo (getter y setter)
13        2 references
14        public int Valor
15        {
16            get { return _valor; }
17            set
18            {
19                _valor = value >= 0 ? value : 0; // Evitar valores negativos
20            }
21        }
22
23        // Constructor por defecto
24        1 reference
25        public Contador()
26        {
27            _valor = 0; // Inicializa el contador en 0
28        }
29
30        // Constructor con parámetros
31        public Contador(int valor)
32        {
33            _valor = valor;
34        }
35
36        // Método para incrementar el contador
37        public void Incrementar()
38        {
39            _valor++;
40        }
41
42        // Método para decrementar el contador
43        public void Decrementar()
44        {
45            _valor--;
46        }
47    }
48 }

```

```

26
27 // Constructor con parámetros
28 1 reference
29 public Contador(int valorInicial)
30 {
31     _valor = valorInicial >= 0 ? valorInicial : 0; // Evitar valores negativos
32 }
33
34 // Método para incrementar el contador
35 2 references
36 public void Incrementar()
37 {
38     _valor++;
39     Console.WriteLine($"El contador se incrementó a: {_valor}");
40 }
41
42 // Método para decrementar el contador
43 3 references
44 public void Decrementar()
45 {
46     if (_valor > 0)
47     {
48         _valor--;
49         Console.WriteLine($"El contador se decrementó a: {_valor}");
50     }
51     else
52     {
53         Console.WriteLine("El contador ya está en 0 y no puede ser decrecido más.");
54     }
55 }
56
57 0 references
58 class Program
59 {
60     0 references
61     static void Main(string[] args)
62     {
63         // Crear un contador con el constructor por defecto
64         Contador contador1 = new Contador();
65         Console.WriteLine($"Valor inicial del contador1: {contador1.Valor}");
66
67         // Incrementar y decrementar el contador
68         contador1.Incrementar(); // Incrementa a 1
69         contador1.Decrementar(); // Decrementa a 0
70         contador1.Decrementar(); // Intenta decrementar más allá de 0
71
72         // Crear un contador con el constructor con parámetros
73         Contador contador2 = new Contador(5);
74         Console.WriteLine($"Valor inicial del contador2: {contador2.Valor}");
75
76         // Incrementar y decrementar el contador
77         contador2.Incrementar(); // Incrementa a 6
78         contador2.Decrementar(); // Decrementa a 5
79     }
80 }

```

4. Crea una clase Libro con los métodos préstamo, devolución y ToString. La clase contendrá un constructor por defecto, un constructor con parámetros y los métodos getters y setters.

Cuenta.cs* Libro.cs* Contador.cs Persona.cs* Program.cs*

C# TestConsole BibliotecaApp.Program

```
1 //Sebastian Gonzalo Alvarez Concepcion 2024-1670
2 using System;
3
4 namespace BibliotecaApp
5 {
6     6 references
7     public class Libro
8     {
9         // Atributos de la clase
10        7 references
11        public string Titulo { get; set; }
12        3 references
13        public string Autor { get; set; }
14        3 references
15        public int AñoPublicacion { get; set; }
16        7 references
17        public bool Prestado { get; private set; }
18
19        // Constructor por defecto
20        1 reference
21        public Libro()
22        {
23            Titulo = "Sin titulo";
24            Autor = "Anónimo";
25            AñoPublicacion = 0;
26            Prestado = false;
27        }
28
29        // Constructor con parámetros
30        1 reference
31        public Libro(string titulo, string autor, int añoPublicacion)
32        {
33            Titulo = titulo;
34            Autor = autor;
35            AñoPublicacion = añoPublicacion;
36            Prestado = false; // Por defecto el libro no está prestado
37        }
38
39        // Método para prestar un libro
40        2 references
41        public void Prestamo()
42        {
43            if (Prestado)
44            {
45                Console.WriteLine($"El libro \"{Titulo}\" ya está prestado.");
46            }
47            else
48            {
49                Prestado = true;
50                Console.WriteLine($"El libro \"{Titulo}\" ha sido prestado.");
51            }
52        }
53    }
54 }
```

```

43     }
44 }
45
46 // Método para devolver un libro
47 2 references
48 public void Devolucion()
49 {
50     if (!Prestado)
51     {
52         Console.WriteLine($"El libro \"{Titulo}\" no estaba prestado.");
53     }
54     else
55     {
56         Prestado = false;
57         Console.WriteLine($"El libro \"{Titulo}\" ha sido devuelto.");
58     }
59 }
60
61 // Método ToString para mostrar información del libro
62 4 references
63 public override string ToString()
64 {
65     return $"Título: {Titulo}, Autor: {Autor}, Año de Publicación: {AñoPublicacion}, Prestado: {(Prestado ? "Sí" : "No")}";
66 }
67
68 0 references
69 class Program
70 {
71     0 references
72     static void Main(string[] args)
73     {
74         // Crear un libro con el constructor por defecto
75         Libro libro1 = new Libro();
76         Console.WriteLine(libro1.ToString());
77
78         // Crear un libro con el constructor con parámetros
79         Libro libro2 = new Libro("1984", "George Orwell", 1949);
80         Console.WriteLine(libro2.ToString());
81
82         // Realizar préstamo y devolución
83         libro2.Prestamo(); // Prestar el libro
84         Console.WriteLine(libro2.ToString());
85         libro2.Prestamo(); // Intentar prestarlo de nuevo
86         libro2.Devolucion(); // Devolver el libro
87         Console.WriteLine(libro2.ToString());
88         libro2.Devolucion(); // Intentar devolverlo de nuevo
89     }
90 }

```

5. Crea una clase Fracción con métodos para sumar, restar, multiplicar y dividir fracciones.

```
Cuenta.cs*  Fraccion.cs*  Libro.cs*  Contador.cs  Persona.cs*  Program.cs*
TestConsole  FraccionesApp.Fraccion

1 //Sebastian Gonzalo Alvarez Concepcion 2024-1670
2 using System;
3
4 namespace FraccionesApp
5 {
6     public class Fraccion
7     {
8         // Atributos: numerador y denominador
9         public int Numerador { get; set; }
10        public int Denominador { get; set; }
11
12        // Constructor por defecto
13        public Fraccion()
14        {
15            Numerador = 0;
16            Denominador = 1; // El denominador no puede ser 0
17        }
18
19        // Constructor con parámetros
20        public Fraccion(int numerador, int denominador)
21        {
22            if (denominador == 0)
23                throw new ArgumentException("El denominador no puede ser 0.");
24            Numerador = numerador;
25            Denominador = denominador;
26            Simplificar(); // Simplifica la fracción automáticamente
27        }
28
29        // Método para sumar dos fracciones
30        public Fraccion Sumar(Fraccion otra)
31        {
32            int nuevoNumerador = (Numerador * otra.Denominador) + (otra.Numerador * Denominador);
33            int nuevoDenominador = Denominador * otra.Denominador;
34            return new Fraccion(nuevoNumerador, nuevoDenominador);
35        }
36    }
37}
```

```
36
37        // Método para restar dos fracciones
38        public Fraccion Restar(Fraccion otra)
39        {
40            int nuevoNumerador = (Numerador * otra.Denominador) - (otra.Numerador * Denominador);
41            int nuevoDenominador = Denominador * otra.Denominador;
42            return new Fraccion(nuevoNumerador, nuevoDenominador);
43        }
44
45        // Método para multiplicar dos fracciones
46        public Fraccion Multiplicar(Fraccion otra)
47        {
48            int nuevoNumerador = Numerador * otra.Numerador;
49            int nuevoDenominador = Denominador * otra.Denominador;
50            return new Fraccion(nuevoNumerador, nuevoDenominador);
51        }
52
53        // Método para dividir dos fracciones
54        public Fraccion Dividir(Fraccion otra)
55        {
56            if (otra.Numerador == 0)
57                throw new DivideByZeroException("No se puede dividir por una fracción con numerador 0.");
58            int nuevoNumerador = Numerador * otra.Denominador;
59            int nuevoDenominador = Denominador * otra.Numerador;
60            return new Fraccion(nuevoNumerador, nuevoDenominador);
61        }
62
63        // Método para simplificar la fracción
64        private void Simplificar()
65        {
66            int mcd = MCD(Numerador, Denominador);
67            Numerador /= mcd;
68            Denominador /= mcd;
69
70            // Asegurarse de que el denominador siempre sea positivo
71            if (Denominador < 0)
72            {
73            }
```



```

73         Numerador = -Numerador;
74         Denominador = -Denominador;
75     }
76 }
77
78 // Método para calcular el Máximo Común Divisor (MCD)
79 1 reference
80 private int MCD(int a, int b)
81 {
82     a = Math.Abs(a);
83     b = Math.Abs(b);
84     while (b != 0)
85     {
86         int temp = b;
87         b = a % b;
88         a = temp;
89     }
90     return a;
91 }
92
93 // Método ToString para mostrar la fracción
94 0 references
95 public override string ToString()
96 {
97     return $"{Numerador}/{Denominador}";
98 }
99
100 0 references
101 class Program
102 {
103     0 references
104     static void Main(string[] args)
105     {
106         // Crear dos fracciones
107         Fraccion fraccion1 = new Fraccion(3, 4); // 3/4
108         Fraccion fraccion2 = new Fraccion(2, 5); // 2/5
109
110         // Mostrar las fracciones iniciales
111         Console.WriteLine($"Fracción 1: {fraccion1}");
112         Console.WriteLine($"Fracción 2: {fraccion2}");

```

```

113
114         // Operaciones
115         Fraccion suma = fraccion1.Sumar(fraccion2);
116         Console.WriteLine($"Suma: {suma}");
117
118         Fraccion resta = fraccion1.Restar(fraccion2);
119         Console.WriteLine($"Resta: {resta}");
120
121         Fraccion multiplicacion = fraccion1.Multiplicar(fraccion2);
122         Console.WriteLine($"Multiplicación: {multiplicacion}");
123
124         Fraccion division = fraccion1.Dividir(fraccion2);
125         Console.WriteLine($"División: {division}");
126     }
127 }

```