Nombre: Wandrys Ferrand

Matricula: 20242038

Dia de clase: Viernes en la Noche

---

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.**

```
namespace PeopleApp
{
   public class Person
   {
      public string FirstName { get; set; }
      public string LastName { get; set; }
      public int Age { get; set; }
      public bool IsMarried { get; set; }
      public string IdentityNumber { get; set; }

      public Person(string firstName, string lastName, int age, bool isMarried, string identityNumber)
      {
         FirstName = firstName;
         LastName = lastName;
```

```
        Age = age;
        IsMarried = isMarried;
        IdentityNumber = identityNumber;
    }

    public void PerformAction1()
    {
        Console.WriteLine($"{FirstName} {LastName} is working in an office.");
    }

    public void PerformAction2()
    {
        Console.WriteLine($"{FirstName} {LastName} is teaching students.");
    }

    public void PerformAction3()
    {
        Console.WriteLine($"{FirstName} {LastName} is working on the construction of a building.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Person person1 = new Person("John", "Doe", 30, true, "123456789");
        Person person2 = new Person("Mary", "Smith", 25, false, "987654321");
```

```
    Person person3 = new Person("Carlos", "Johnson", 40, true, "112233445");
    Person person4 = new Person("Ana", "Davis", 35, true, "556677889");
    Person person5 = new Person("Luis", "Martinez", 28, false, "998877665");
    Person person6 = new Person("Elena", "Lopez", 50, true, "667788990");
    Person person7 = new Person("Peter", "Mora", 60, false, "443322110");

    person1.PerformAction1();
    person2.PerformAction2();
    person3.PerformAction3();
    person4.PerformAction1();
    person5.PerformAction2();
    person6.PerformAction3();
    person7.PerformAction1();

    ShowDetails(person1);
    ShowDetails(person2);
    ShowDetails(person3);
    ShowDetails(person4);
    ShowDetails(person5);
    ShowDetails(person6);
    ShowDetails(person7);
}

static void ShowDetails(Person person)
{
    Console.WriteLine($"Full Name: {person.FirstName} {person.LastName}");
    Console.WriteLine($"Age: {person.Age}");
```

```
            Console.WriteLine($"Married: {(person.IsMarried ? "Yes" : "No")}");
            Console.WriteLine($"Identity Number: {person.IdentityNumber}\n");
        }
    }
}
```

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.**

```
namespace BankingApp
{
    public class Cuenta
    {
        private string Titular { get; set; }
        private decimal Saldo { get; set; }

        public Cuenta()
        {
            Titular = "Desconocido";
            Saldo = 0.0m;
        }

        public Cuenta(string titular, decimal saldoInicial)
        {
            Titular = titular;
```

```
    Saldo = saldoInicial;
}

public string ObtenerTitular()
{
    return Titular;
}

public void EstablecerTitular(string titular)
{
    Titular = titular;
}

public decimal ObtenerSaldo()
{
    return Saldo;
}

public void EstablecerSaldo(decimal saldo)
{
    Saldo = saldo;
}

public void Ingreso(decimal cantidad)
{
    if (cantidad > 0)
    {
```

```
        Saldo += cantidad;
        Console.WriteLine($"Ingreso de {cantidad:C} realizado con éxito.");
    }
    else
    {
        Console.WriteLine("La cantidad a ingresar debe ser mayor que cero.");
    }
}

public void Reintegro(decimal cantidad)
{
    if (cantidad > 0 && cantidad <= Saldo)
    {
        Saldo -= cantidad;
        Console.WriteLine($"Reintegro de {cantidad:C} realizado con éxito.");
    }
    else
    {
        Console.WriteLine("Fondos insuficientes o cantidad no válida.");
    }
}

public void Transferencia(Cuenta cuentaDestino, decimal cantidad)
{
    if (cantidad > 0 && cantidad <= Saldo)
    {
        Reintegro(cantidad);
```

```
            cuentaDestino.Ingreso(cantidad);
            Console.WriteLine($"Transferencia de {cantidad:C} realizada con éxito a {cuentaDestino.ObtenerTitular()}.");
        }
        else
        {
            Console.WriteLine("Fondos insuficientes o cantidad no válida para la transferencia.");
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Cuenta cuenta1 = new Cuenta("Juan Pérez", 1000.0m);
        Cuenta cuenta2 = new Cuenta("María González", 500.0m);

        Console.WriteLine($"{cuenta1.ObtenerTitular()} tiene un saldo de {cuenta1.ObtenerSaldo():C}");
        Console.WriteLine($"{cuenta2.ObtenerTitular()} tiene un saldo de {cuenta2.ObtenerSaldo():C}");

        cuenta1.Ingreso(300.0m);
        Console.WriteLine($"{cuenta1.ObtenerTitular()} tiene un saldo de {cuenta1.ObtenerSaldo():C}");

        cuenta2.Reintegro(100.0m);
```

```
        Console.WriteLine($"{cuenta2.ObtenerTitular()} tiene un saldo de
{cuenta2.ObtenerSaldo():C}");


        cuenta1.Transferencia(cuenta2, 200.0m);


        Console.WriteLine($"{cuenta1.ObtenerTitular()} tiene un saldo de
{cuenta1.ObtenerSaldo():C}");
        Console.WriteLine($"{cuenta2.ObtenerTitular()} tiene un saldo de
{cuenta2.ObtenerSaldo():C}");
    }
  }
}
```

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.**

```
namespace ContadorApp
{
  public class Contador
  {
    private int _valor;


    public Contador()
    {
      _valor = 0;
    }
```

```
    public Contador(int valorInicial)
    {
        _valor = valorInicial;
    }

    public int ObtenerValor()
    {
        return _valor;
    }

    public void EstablecerValor(int valor)
    {
        _valor = valor;
    }

    public void Incrementar()
    {
        _valor++;
        Console.WriteLine($"Contador incrementado. Valor actual: {_valor}");
    }

    public void Decrementar()
    {
        _valor--;
        Console.WriteLine($"Contador decrementado. Valor actual: {_valor}");
    }
}
```

```
    class Program
    {
        static void Main(string[] args)
        {
            Contador contador1 = new Contador();
            Console.WriteLine($"Valor inicial del contador1: {contador1.ObtenerValor()}");

            Contador contador2 = new Contador(10);
            Console.WriteLine($"Valor inicial del contador2: {contador2.ObtenerValor()}");

            contador1.Incrementar();

            contador2.Decrementar();

            contador1.EstablecerValor(5);
            Console.WriteLine($"Nuevo valor de contador1: {contador1.ObtenerValor()}");

            contador1.Incrementar();
            contador2.Decrementar();
        }
    }
}
```

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.**

```
namespace LibroApp
{
    public class Libro
    {
        private string _titulo;
        private string _autor;
        private string _isbn;
        private bool _prestado;

        public Libro()
        {
            _titulo = "Desconocido";
            _autor = "Desconocido";
            _isbn = "0000000000000";
            _prestado = false;
        }

        public Libro(string titulo, string autor, string isbn)
        {
            _titulo = titulo;
            _autor = autor;
            _isbn = isbn;
            _prestado = false;
        }

        public string ObtenerTitulo()
        {
```

```
    return _titulo;
}

public void EstablecerTitulo(string titulo)
{
    _titulo = titulo;
}

public string ObtenerAutor()
{
    return _autor;
}

public void EstablecerAutor(string autor)
{
    _autor = autor;
}

public string ObtenerIsbn()
{
    return _isbn;
}

public void EstablecerIsbn(string isbn)
{
    _isbn = isbn;
}
```

```
public bool ObtenerPrestado()
{
   return _prestado;
}

public void EstablecerPrestado(bool prestado)
{
   _prestado = prestado;
}

public void Prestamo()
{
   if (_prestado)
   {
      Console.WriteLine($"El libro '{_titulo}' ya está prestado.");
   }
   else
   {
      _prestado = true;
      Console.WriteLine($"El libro '{_titulo}' ha sido prestado.");
   }
}

public void Devolucion()
{
   if (!_prestado)
```

```
        {
            Console.WriteLine($"El libro '{_titulo}' no está prestado, no se puede devolver.");
        }
        else
        {
            _prestado = false;
            Console.WriteLine($"El libro '{_titulo}' ha sido devuelto.");
        }
    }

    public override string ToString()
    {
        return $"Título: {_titulo}\nAutor: {_autor}\nISBN: {_isbn}\nPrestado: {(_prestado ? "Sí" : "No")}";
    }
}

class Program
{
    static void Main(string[] args)
    {
        Libro libro1 = new Libro();
        Console.WriteLine("Libro 1:\n" + libro1.ToString() + "\n");

        Libro libro2 = new Libro("1984", "George Orwell", "9780451524935");
        Console.WriteLine("Libro 2:\n" + libro2.ToString() + "\n");

        libro2.Prestamo();
```

```
            libro2.Devolucion();
            libro2.Devolucion();
            libro2.Prestamo();
            libro2.Prestamo();

            Console.WriteLine("\nEstado final de los libros:");
            Console.WriteLine(libro1.ToString());
            Console.WriteLine(libro2.ToString());
        }
    }
}
```

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

```
namespace FraccionApp
{
    public class Fraccion
    {
        private int _numerador;
        private int _denominador;

        public Fraccion()
        {
            _numerador = 0;
            _denominador = 1;
        }
```

```
public Fraccion(int numerador, int denominador)
{
   if (denominador == 0)
      throw new ArgumentException("El denominador no puede ser cero.");

   _numerador = numerador;
   _denominador = denominador;
}

public int ObtenerNumerador()
{
   return _numerador;
}

public int ObtenerDenominador()
{
   return _denominador;
}

public void EstablecerNumerador(int numerador)
{
   _numerador = numerador;
}

public void EstablecerDenominador(int denominador)
{
```

```
        if (denominador == 0)
            throw new ArgumentException("El denominador no puede ser cero.");

        _denominador = denominador;
    }

    public Fraccion Sumar(Fraccion otraFraccion)
    {
        int numeradorResultante = _numerador * otraFraccion.ObtenerDenominador() +
_denominador * otraFraccion.ObtenerNumerador();
        int denominadorResultante = _denominador *
otraFraccion.ObtenerDenominador();
        return new Fraccion(numeradorResultante, denominadorResultante);
    }

    public Fraccion Restar(Fraccion otraFraccion)
    {
        int numeradorResultante = _numerador * otraFraccion.ObtenerDenominador() -
_denominador * otraFraccion.ObtenerNumerador();
        int denominadorResultante = _denominador *
otraFraccion.ObtenerDenominador();
        return new Fraccion(numeradorResultante, denominadorResultante);
    }

    public Fraccion Multiplicar(Fraccion otraFraccion)
    {
        int numeradorResultante = _numerador * otraFraccion.ObtenerNumerador();
        int denominadorResultante = _denominador *
otraFraccion.ObtenerDenominador();
```

```
        return new Fraccion(numeradorResultante, denominadorResultante);
    }

    public Fraccion Dividir(Fraccion otraFraccion)
    {
        if (otraFraccion.ObtenerNumerador() == 0)
            throw new ArgumentException("No se puede dividir por una fracción cuyo numerador sea cero.");

        int numeradorResultante = _numerador * otraFraccion.ObtenerDenominador();
        int denominadorResultante = _denominador * otraFraccion.ObtenerNumerador();
        return new Fraccion(numeradorResultante, denominadorResultante);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Fraccion fraccion1 = new Fraccion(3, 4);
        Fraccion fraccion2 = new Fraccion(5, 6);

        Fraccion suma = fraccion1.Sumar(fraccion2);
        Console.WriteLine($"La suma de {fraccion1.ObtenerNumerador()}/{fraccion1.ObtenerDenominador()} y {fraccion2.ObtenerNumerador()}/{fraccion2.ObtenerDenominador()} es: {suma.ObtenerNumerador()}/{suma.ObtenerDenominador()}");

        Fraccion resta = fraccion1.Restar(fraccion2);
```

```
        Console.WriteLine($"La resta de
{fraccion1.ObtenerNumerador()}/{fraccion1.ObtenerDenominador()} y
{fraccion2.ObtenerNumerador()}/{fraccion2.ObtenerDenominador()} es:
{resta.ObtenerNumerador()}/{resta.ObtenerDenominador()}");


        Fraccion multiplicacion = fraccion1.Multiplicar(fraccion2);

        Console.WriteLine($"La multiplicación de
{fraccion1.ObtenerNumerador()}/{fraccion1.ObtenerDenominador()} y
{fraccion2.ObtenerNumerador()}/{fraccion2.ObtenerDenominador()} es:
{multiplicacion.ObtenerNumerador()}/{multiplicacion.ObtenerDenominador()}");


        try

        {

            Fraccion division = fraccion1.Dividir(fraccion2);

            Console.WriteLine($"La división de
{fraccion1.ObtenerNumerador()}/{fraccion1.ObtenerDenominador()} y
{fraccion2.ObtenerNumerador()}/{fraccion2.ObtenerDenominador()} es:
{division.ObtenerNumerador()}/{division.ObtenerDenominador()}");

        }

        catch (ArgumentException e)

        {

            Console.WriteLine($"Error: {e.Message}");

        }

    }

  }

}
```