Nombre: Rubby Keyther Martinez Zorrilla

Matricula: 2024-2036

Dia: viernes.

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.

```
using System.Dynamic;

Console.WriteLine("INFORMACIONES PERSONALES");

Persona persona1 = new Persona("David", "Espinoso", 30, true, "001-345664-1", "doctor");
Persona persona2 = new Persona("Rosa", "Melano", 25, false, "401-172372-1", "bombero");
Persona persona3 = new Persona("Henri", "Rodríguez", 40, true, "001-882898-1", "arquitecto");
Persona persona4 = new Persona("Laraveling", "Martínez", 28, false, "001-293383-1", "maestro");
Persona persona5 = new Persona("Juanito", "Alimaña", 35, true, "001-2346332-1", "electricista");
Persona persona6 = new Persona("Pedro", "Navaja", 22, false, "401-1234632-4", "actor");
Persona persona7 = new Persona("Luis", "Sánchez", 50, true, "001-247672-7", "cantante");


persona1.RealizarAcciones();
persona2.RealizarAcciones();
persona3.RealizarAcciones();
persona4.RealizarAcciones();
persona5.RealizarAcciones();
persona6.RealizarAcciones();
persona7.RealizarAcciones();

Console.ReadLine();
```

```
class Persona
{
    private string Nombre;
    private string Apellido;
    private int Edad;
    private bool Casado;
    private string NumeroDocumentoIdentidad;
    private string Profesion;

    7 references
    public Persona(string nombre, string apellido, int edad, bool casado, string numeroDocumentoIdentidad, string profesion)
    {
        this.Nombre = nombre;
        this.Apellido = apellido;
        this.Edad = edad;
        this.Casado = casado;
        this.NumeroDocumentoIdentidad = numeroDocumentoIdentidad;
        this.Profesion = profesion;
    }
```

```
1 reference
public string GetNombre() { return Nombre; }
1 reference
public string GetProfesion() { return Profesion; }

7 references
public void RealizarAcciones()
{
    Console.WriteLine($"\n{GetNombre()} ({GetProfesion()}) está realizando sus actividades:");

    Accion1();
    Accion2();
    Accion3();
}
```

```
private void Accion1()
{
    switch (Profesion.ToLower())
    {
        case "doctor": Console.WriteLine(" Realiza una cirugía."); break;
        case "bombero": Console.WriteLine(" Apaga un incendio en una casa."); break;
        case "arquitecto": Console.WriteLine(" Diseña un nuevo puente."); break;
        case "maestro": Console.WriteLine(" Explica matemáticas a sus estudiantes."); break;
        case "electricista": Console.WriteLine(" Instala un nuevo sistema eléctrico."); break;
        case "actor": Console.WriteLine(" Interpreta un papel en una película."); break;
        case "cantante": Console.WriteLine(" Canta en un concierto en vivo."); break;
        default: Console.WriteLine(" No tiene una acción específica."); break;
    }
}

1 reference
private void Accion2()
{
    switch (Profesion.ToLower())
    {
        case "doctor": Console.WriteLine(" Diagnostica a un paciente."); break;
        case "bombero": Console.WriteLine(" Rescata a una persona atrapada."); break;
        case "arquitecto": Console.WriteLine(" Supervisa la construcción de un edificio."); break;
        case "maestro": Console.WriteLine(" Revisa exámenes de sus alumnos."); break;
        case "electricista": Console.WriteLine(" Repara un cortocircuito en una casa."); break;
        case "actor": Console.WriteLine(" Ensaya una escena con otros actores."); break;
        case "cantante": Console.WriteLine(" Graba una nueva canción en el estudio."); break;
        default: Console.WriteLine(" No tiene una acción específica."); break;
    }
}
```

```
1 reference
private void Accion3()
{
    switch (Profesion.ToLower())
    {
        case "doctor": Console.WriteLine(" Da una charla sobre salud pública."); break;
        case "bombero": Console.WriteLine(" Enseña prevención de incendios en una escuela."); break;
        case "arquitecto": Console.WriteLine(" Presenta su diseño a un cliente."); break;
        case "maestro": Console.WriteLine(" Organiza un evento escolar."); break;
        case "electricista": Console.WriteLine(" Capacita a nuevos aprendices."); break;
        case "actor": Console.WriteLine(" Atiende una rueda de prensa sobre su película."); break;
        case "cantante": Console.WriteLine(" Participa en una entrevista en televisión."); break;
        default: Console.WriteLine(" No tiene una acción específica."); break;
    }
}
```

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.

```
using System;

7 references
class Cuenta
{
    private string titular;
    private double saldo;

    // Constructor por defecto
    0 references
    public Cuenta()
    {
        titular = "Desconocido";
        saldo = 0.0;
    }

    // Constructor con parámetros
    2 references
    public Cuenta(string titular, double saldoInicial)
    {
        this.titular = titular;
        this.saldo = saldoInicial;
    }
```

```
// Método para ingresar dinero
2 references
public void Ingreso(double cantidad)
{
    if (cantidad > 0)
    {
        saldo += cantidad;
        Console.WriteLine($"{titular} ha ingresado {cantidad:C}. Nuevo saldo: {saldo:C}");
    }
    else
    {
        Console.WriteLine("La cantidad a ingresar debe ser positiva.");
    }
}
```

```
// Método para reintegro dinero
1 reference
public void Reintegro(double cantidad)
{
    if (cantidad > 0 && cantidad <= saldo)
    {
        saldo -= cantidad;
        Console.WriteLine($"{titular} ha retirado {cantidad:C}. Nuevo saldo: {saldo:C}");
    }
    else
    {
        Console.WriteLine("Saldo insuficiente o cantidad no válida.");
    }
}
```

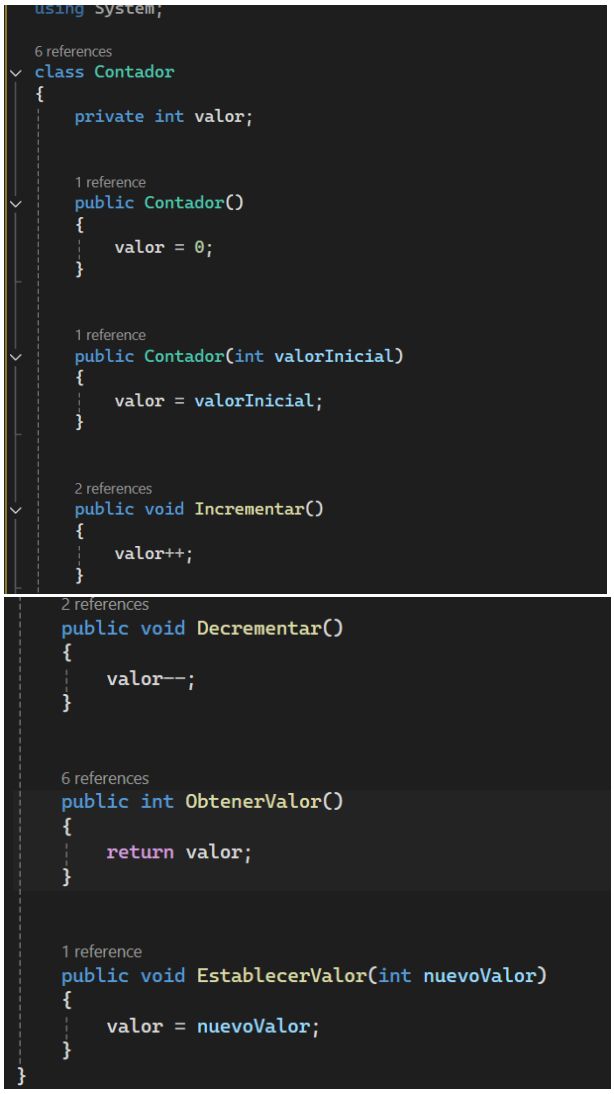
```
// Método para transferir dinero
1 reference
public void Transferencia(Cuenta destino, double cantidad)
{
    if (cantidad > 0 && cantidad <= saldo)
    {
        saldo -= cantidad;
        destino.Ingreso(cantidad);
        Console.WriteLine($"{titular} ha transferido {cantidad:C} a {destino.GetTitular()}.");
    }
    else
    {
        Console.WriteLine("No se pudo realizar la transferencia. Verifique el saldo.");
    }
}
```

```
// Métodos Get y Set
3 references
public string GetTitular()
{
    return titular;
}

2 references
public double GetSaldo()
{
    return saldo;
}

0 references
public void SetTitular(string nuevoTitular)
{
    titular = nuevoTitular;
}
```

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.



```
using System;

6 references
class Contador
{
    private int valor;

    1 reference
    public Contador()
    {
        valor = 0;
    }

    1 reference
    public Contador(int valorInicial)
    {
        valor = valorInicial;
    }

    2 references
    public void Incrementar()
    {
        valor++;
    }

    2 references
    public void Decrementar()
    {
        valor--;
    }

    6 references
    public int ObtenerValor()
    {
        return valor;
    }

    1 reference
    public void EstablecerValor(int nuevoValor)
    {
        valor = nuevoValor;
    }
}
```

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.

```
class Libro
{
    private string Nombre;
    private string Autor;
    private string Propietario;
    private int Copias;

    0 references
    public Libro()
    {
        Nombre = " ";
        Autor = " ";
        Propietario = " ";
        Copias = 0;
    }

    0 references
    public Libro(string nombre, string autor, string propietario, int copias)
    {
        Nombre = nombre;
        Autor = autor;
        Propietario = propietario;
        Copias = copias;
    }
```

```
    //Getter y Setter para Nombre
    0 references
    public string GetNombre(){return Nombre;}

    0 references
    public void SetNombre(string nombre){Nombre = nombre;}

    // Getter y Setter para Autor
    0 references
    public string GetAutor(){return Autor;}

    0 references
    public void SetAutor(string autor){Autor = autor;}

    // Getter y Setter para Propietario
    0 references
    public string GetPropietario(){return Propietario;}

    0 references
    public void SetPropietario(string propietario){Propietario = propietario;}

    // Getter y Setter para Copias
    0 references
    public int GetCopias(){return Copias;}

    0 references
    public void SetCopias(int copias){Copias = copias;}
```

```
0 references
public void Prestamo(string Destino, int cantidad)
{
    if (cantidad <= Copias)
    {
        Copias -= cantidad;

        Console.WriteLine($"{Propietario} a realizado un prestamo de: {cantidad} copias del libro {Nombre} a {Destino}");

    }
    else
    {
        Console.WriteLine($"No hay suficientes copias del libro {Nombre} para realizar prestamo");
    }

}
```

```
public void Devolucion(string Usuario, int cantidad)
{

    if (cantidad <= Copias) { Copias += cantidad;

        Console.WriteLine($"{Usuario} a realizado una devolucion de: {cantidad}, del libro {Nombre} a {Propietario}");

    }
}
0 references
public override string ToString()
{
    return $"Libro: {Nombre}, Autor: {Autor}, Propietario: {Propietario}, Copias disponibles: {Copias}";
}
}
```

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

```
public class Fraccion
{
    12 references
    public int Numerador { get; set; }
    15 references
    public int Denominador { get; set; }


    0 references
    public Fraccion()
    {
        Numerador = 0;
        Denominador = 1;
    }


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

        Numerador = numerador;
        Denominador = denominador;
    }
```

```
    // Método para sumar fracciones
    0 references
    public Fraccion Sumar(Fraccion otraFraccion)
    {
        int nuevoNumerador = (this.Numerador * otraFraccion.Denominador) + (otraFraccion.Numerador * this.Denominador);
        int nuevoDenominador = this.Denominador * otraFraccion.Denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    // Método para restar fracciones
    0 references
    public Fraccion Restar(Fraccion otraFraccion)
    {
        int nuevoNumerador = (this.Numerador * otraFraccion.Denominador) - (otraFraccion.Numerador * this.Denominador);
        int nuevoDenominador = this.Denominador * otraFraccion.Denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    // Método para multiplicar fracciones
    0 references
    public Fraccion Multiplicar(Fraccion otraFraccion)
    {
        int nuevoNumerador = this.Numerador * otraFraccion.Numerador;
        int nuevoDenominador = this.Denominador * otraFraccion.Denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }
```

```
    // Método para dividir fracciones
    0 references
    public Fraccion Dividir(Fraccion otraFraccion)
    {
        if (otraFraccion.Numerador == 0)
        {
            throw new ArgumentException("No se puede dividir entre una fracción con numerador cero.");
        }

        int nuevoNumerador = this.Numerador * otraFraccion.Denominador;
        int nuevoDenominador = this.Denominador * otraFraccion.Numerador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }


    0 references
    public override string ToString()
    {
        return $"{Numerador}/{Denominador}";
    }
}
```