Homework May 7

Structs:

1. **Value Types**: Los structs son tipos de valor. Esto significa que cuando se asigna un struct a una variable o se pasa como argumento a un método, se copia el valor completo. Cada variable que contiene un struct tiene su propia copia independiente de los datos.
2. **Almacenamiento en Stack**: Por lo general, los structs se almacenan en la pila de memoria, lo que los hace más eficientes en términos de acceso y manipulación rápida de datos.
3. **No admiten Herencia**: Los structs no pueden heredar de otros tipos ni pueden ser heredados. Tienen un comportamiento de sellado.
4. **No permiten destrucción**: No se puede definir un destructor para un struct. La memoria asociada con un struct se gestiona automáticamente por el recolector de basura.
5. **Uso para tipos de datos pequeños y de valor simple**: Los structs son útiles para representar tipos de datos simples y pequeños, como coordenadas, números complejos, etc.

Clases:

1. **Reference Types**: Las clases son tipos de referencia. Cuando se asigna una clase a una variable o se pasa como argumento a un método, se pasa una referencia al objeto en lugar de una copia del objeto en sí mismo. Varias variables pueden hacer referencia al mismo objeto en memoria.
2. **Almacenamiento en Heap**: Las instancias de clases se almacenan en el montón (heap) de memoria, lo que significa que son más adecuadas para objetos grandes y que requieren un tiempo de vida dinámico.
3. **Admite Herencia**: Las clases admiten herencia, lo que significa que una clase puede heredar los campos y métodos de otra clase.
4. **Permite destrucción**: Las clases pueden tener destructores definidos, que se utilizan para liberar recursos no administrados.
5. **Uso para modelar objetos complejos y con comportamiento**: Las clases son ideales para modelar objetos complejos con comportamientos y propiedades complejas, como usuarios, productos, vehículos, etc.

Struct Coordenada

```
using System;



public struct Coordenada
{

        public double X { get; }
        public double Y { get; }

        public Coordenada(double x, double y)
        {
                X = x;
                Y = y;
        }

        public double CalcularDistancia(Coordenada otraCoordenada)
        {
                double distanciaX = otraCoordenada.X - X;
                double distanciaY = otraCoordenada.Y - Y;
                return Math.Sqrt(distanciaX * distanciaX + distanciaY *
distanciaY);
        }

        public double CalcularAngulo(Coordenada otraCoordenada)
        {
                double deltaX = otraCoordenada.X - X;
                double deltaY = otraCoordenada.Y - Y;
                return Math.Atan2(deltaY, deltaX);
        }




        public override string ToString()
        {
                return $"({X}, {Y})";
        }

}
```

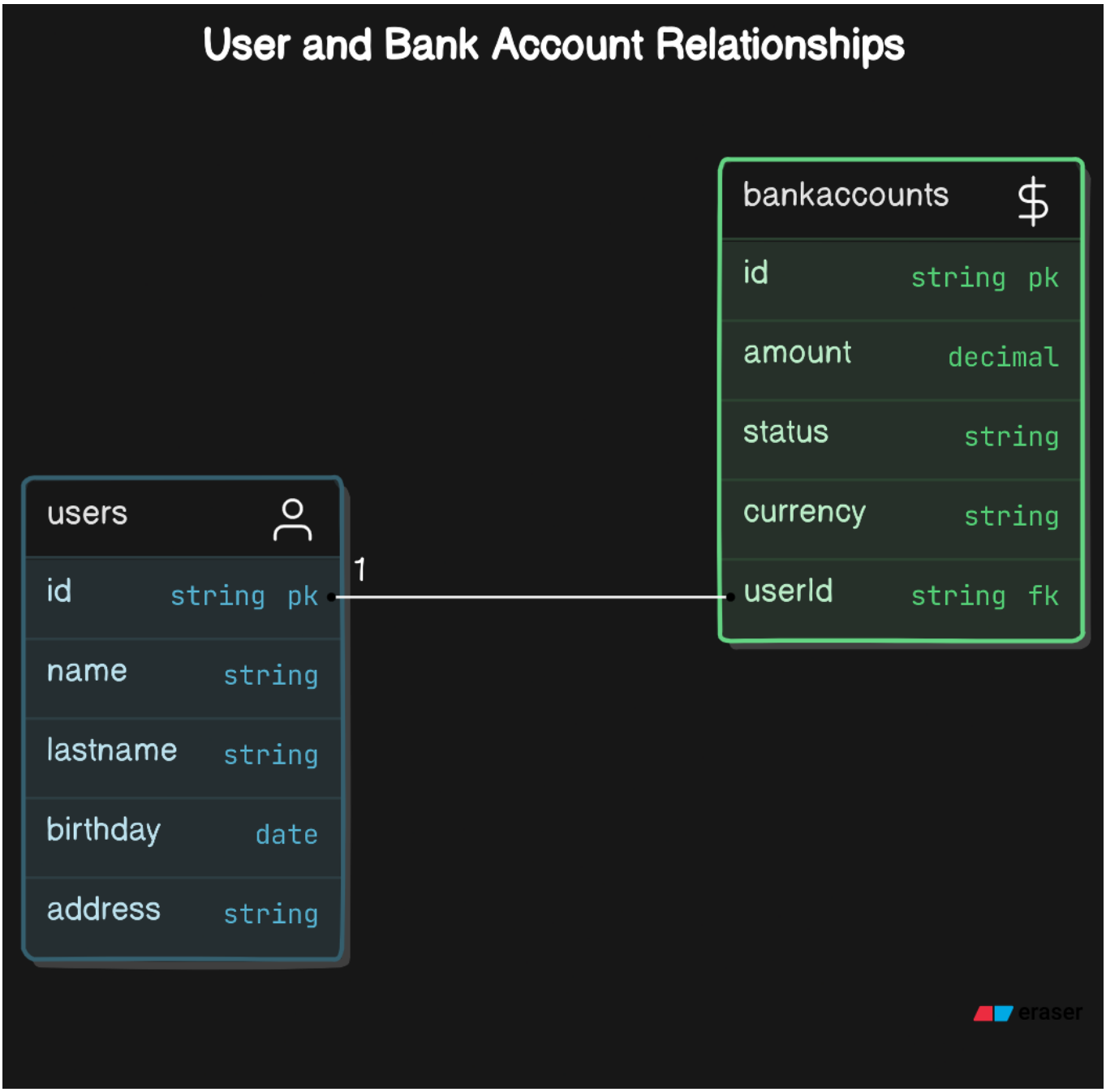
```
class Program
{
        static void Main(string[] args){

                Coordenada punto1 = new Coordenada(3, 4);
                Coordenada punto2 = new Coordenada(6, 8);
                Console.WriteLine("Coordenada 1: " + punto1);
                Console.WriteLine("Coordenada 2: " + punto2);
                double distancia = punto1.CalcularDistancia(punto2);
                Console.WriteLine("Distancia entre los puntos: " +
distancia)
                double angulo = punto1.CalcularAngulo(punto2);
                Console.WriteLine("Ángulo entre los puntos en radianes: " +
angulo);

        }

}
```

Relación de Dos Entidades


User and Bank Account Relationships
bankaccounts
id string pk
amount decimal
status string
currency string
userId string fk
users
id string pk
name string
lastname string
birthday date
address string
1
eraser