1.

public class Racional

{

private int numerador;

private int denominador;

public Racional()

{

numerador = 0;

denominador = 1;

}

public Racional(int numerador, int denominador)

{

if (denominador == 0)

throw new ArgumentException("El denominador no puede ser cero.");

this.numerador = numerador;

this.denominador = denominador;

Simplificar();

}

public int Numerador => numerador;

public int Denominador => denominador;

public static Racional Leer()

{

Console.Write("Introduce el numerador: ");

int num = int.Parse(Console.ReadLine());

Console.Write("Introduce el denominador (no puede ser 0): ");

int den = int.Parse(Console.ReadLine());

return new Racional(num, den);

}

private void Simplificar()

{

int mcd = MaximoComunDivisor(Math.Abs(numerador), Math.Abs(denominador));

numerador /= mcd;

denominador /= mcd;

if (denominador < 0)

{

numerador = -numerador;

denominador = -denominador;

}

}

private int MaximoComunDivisor(int a, int b)

{

while (b != 0)

{

int temp = b;

b = a % b;

a = temp;

}

return a;

}

public Racional Sumar(Racional otro)

{

int nuevoNumerador = numerador \* otro.denominador + otro.numerador \* denominador;

int nuevoDenominador = denominador \* otro.denominador;

return new Racional(nuevoNumerador, nuevoDenominador);

}

public Racional Restar(Racional otro)

{

int nuevoNumerador = numerador \* otro.denominador - otro.numerador \* denominador;

int nuevoDenominador = denominador \* otro.denominador;

return new Racional(nuevoNumerador, nuevoDenominador);

}

public Racional Multiplicar(Racional otro)

{

int nuevoNumerador = numerador \* otro.numerador;

int nuevoDenominador = denominador \* otro.denominador;

return new Racional(nuevoNumerador, nuevoDenominador);

}

public Racional Dividir(Racional otro)

{

if (otro.numerador == 0)

throw new DivideByZeroException("No se puede dividir por un racional cero.");

int nuevoNumerador = numerador \* otro.denominador;

int nuevoDenominador = denominador \* otro.numerador;

return new Racional(nuevoNumerador, nuevoDenominador);

}

public override string ToString()

{

return $"{numerador}/{denominador}";

}

}

public class Program

{

public static void Main()

{

Console.WriteLine("Creación de dos números racionales:");

Racional r1 = Racional.Leer();

Racional r2 = Racional.Leer();

Console.WriteLine($"\nRacional 1: {r1}");

Console.WriteLine($"Racional 2: {r2}");

Console.WriteLine($"\nSuma: {r1.Sumar(r2)}");

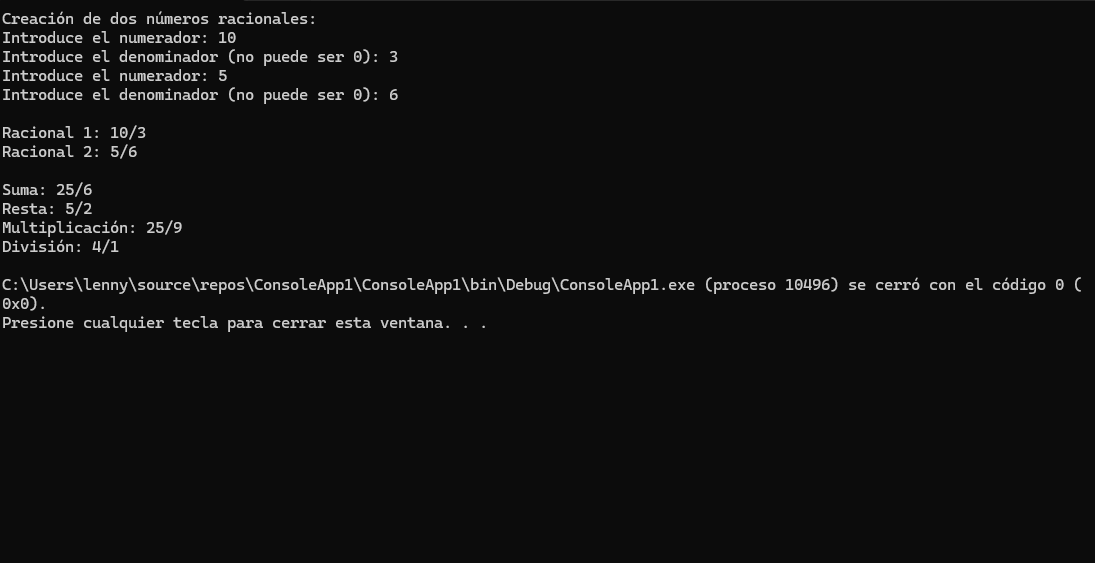
Console.WriteLine($"Resta: {r1.Restar(r2)}");

Console.WriteLine($"Multiplicación: {r1.Multiplicar(r2)}");

Console.WriteLine($"División: {r1.Dividir(r2)}");

}

}



2.

public class Cuenta

{

private long numeroCuenta;

private long dniCliente;

private double saldo;

private double interesAnual;

private static long contadorCuentas = 1000;

public Cuenta()

{

numeroCuenta = contadorCuentas++;

dniCliente = 0;

saldo = 0;

interesAnual = 0;

}

public Cuenta(long dni, double saldoInicial, double interes)

{

numeroCuenta = contadorCuentas++;

dniCliente = dni;

saldo = saldoInicial;

interesAnual = interes;

}

public long NumeroCuenta => numeroCuenta;

public long DniCliente => dniCliente;

public double Saldo => saldo;

public double InteresAnual => interesAnual;

public void SetDniCliente(long dni) => dniCliente = dni;

public void SetInteresAnual(double interes) => interesAnual = interes;

public void ActualizarSaldo()

{

double interesDiario = interesAnual / 365 / 100;

saldo += saldo \* interesDiario;

}

public void Ingresar(double cantidad)

{

if (cantidad > 0)

saldo += cantidad;

}

public bool Retirar(double cantidad)

{

if (cantidad > 0 && saldo >= cantidad)

{

saldo -= cantidad;

return true;

}

return false;

}

public void MostrarDatos()

{

Console.WriteLine($"Número de cuenta: {numeroCuenta}");

Console.WriteLine($"DNI cliente: {dniCliente}");

Console.WriteLine($"Saldo actual: {saldo:C}");

Console.WriteLine($"Interés anual: {interesAnual}%");

}

}

public class Program

{

public static void Main()

{

Cuenta cuenta1 = new Cuenta(12345678, 1000, 1.5);

Console.WriteLine("Datos iniciales:");

cuenta1.MostrarDatos();

cuenta1.Ingresar(500);

cuenta1.Retirar(200);

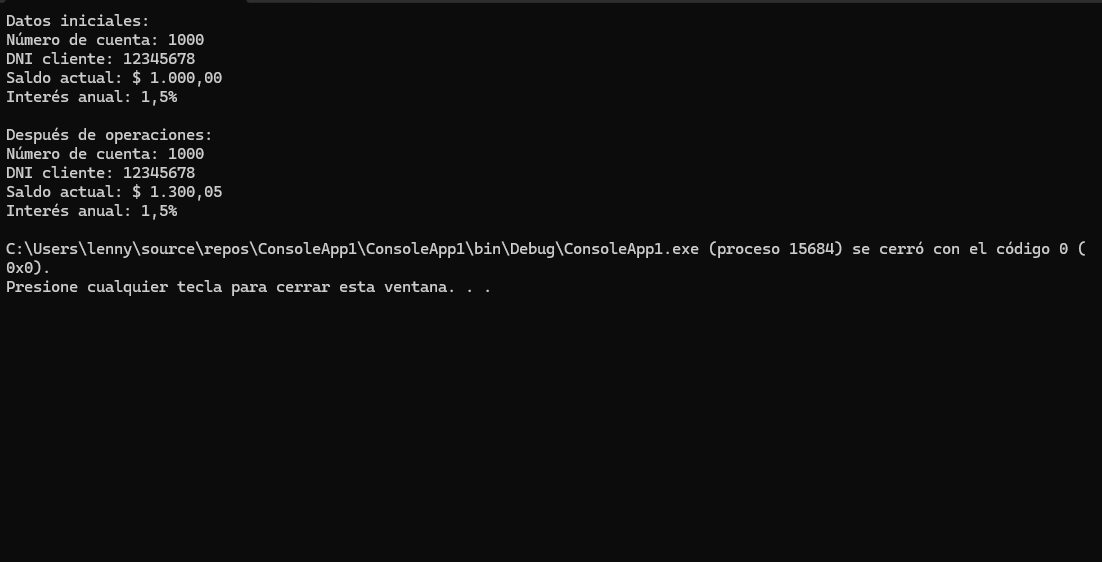
cuenta1.ActualizarSaldo();

Console.WriteLine("\nDespués de operaciones:");

cuenta1.MostrarDatos();

}

}



3.

using System;

public class Motor

{

public void Arrancar()

{

Console.WriteLine("Motor arrancado");

}

public void Apagar()

{

Console.WriteLine("Motor apagado");

}

}

public class Rueda

{

public void Inflar()

{

Console.WriteLine("Rueda inflada");

}

public void Desinflar()

{

Console.WriteLine("Rueda desinflada");

}

}

public class Ventana

{

public void Abrir()

{

Console.WriteLine("Ventana abierta");

}

public void Cerrar()

{

Console.WriteLine("Ventana cerrada");

}

}

public class Puerta

{

private Ventana ventana = new Ventana();

public void Abrir()

{

Console.WriteLine("Puerta abierta");

}

public void Cerrar()

{

Console.WriteLine("Puerta cerrada");

}

public void AbrirVentana()

{

ventana.Abrir();

}

public void CerrarVentana()

{

ventana.Cerrar();

}

}

public class Coche

{

private Motor motor = new Motor();

private Rueda[] ruedas = new Rueda[4];

private Puerta[] puertas = new Puerta[2];

public Coche()

{

for (int i = 0; i < 4; i++)

ruedas[i] = new Rueda();

for (int i = 0; i < 2; i++)

puertas[i] = new Puerta();

}

public void Arrancar()

{

motor.Arrancar();

}

public void Apagar()

{

motor.Apagar();

}

public void InflarRuedas()

{

foreach (var rueda in ruedas)

rueda.Inflar();

}

public void AbrirPuertas()

{

foreach (var puerta in puertas)

puerta.Abrir();

}

}

class Program

{

static void Main()

{

Coche miCoche = new Coche();

miCoche.Arrancar();

miCoche.InflarRuedas();

miCoche.AbrirPuertas();

miCoche.Apagar();

}

}

