Actividad poo

A. Realizar los siguientes algoritmos aplicando la programación orientada a objetos

1.Crea una clase Racional que permita trabajar con números racionales (fracciones). Incluye los siguientes métodos: constructores (por defecto y parametrizado), accedentes, leer(), suma, resta, multiplicación, división.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace actividad\_poo

{

internal class Program

{

static void Main(string[] args)

{

Racional r1 = new Racional();

r1.Leer();

Racional r2 = new Racional();

r2.Leer();

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

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

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

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

Console.ReadKey();

}

}

public class Racional

{

public int Numerador { get; set; }

public int Denominador { get; set; }

public Racional()

{

Numerador = 1;

Denominador = 1;

}

public Racional(int numerador, int denominador)

{

Numerador = numerador;

Denominador = denominador == 0 ? 1 : denominador;

}

public void Leer()

{

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

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

Console.Write("Ingrese el denominador: ");

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

Denominador = d == 0 ? 1 : d;

}

public Racional Sumar(Racional otro)

{

int num = Numerador \* otro.Denominador + otro.Numerador \* Denominador;

int den = Denominador \* otro.Denominador;

return new Racional(num, den);

}

// Resta de dos racionales

public Racional Restar(Racional otro)

{

int num = Numerador \* otro.Denominador - otro.Numerador \* Denominador;

int den = Denominador \* otro.Denominador;

return new Racional(num, den);

}

// Multiplicación de dos racionales

public Racional Multiplicar(Racional otro)

{

int num = Numerador \* otro.Numerador;

int den = Denominador \* otro.Denominador;

return new Racional(num, den);

}

// División de dos racionales

public Racional Dividir(Racional otro)

{

int num = Numerador \* otro.Denominador;

int den = Denominador \* otro.Numerador;

return new Racional(num, den);

}

public override string ToString()

{

return $"{Numerador}/{Denominador}";

}

}

}

2 Crea una clase Cuenta (bancaria) con atributos para el número de cuenta (un entero largo), el DNI del cliente (otro entero largo), el saldo actual y el interés anual que se aplica a la cuenta (porcentaje). Define en la clase los siguientes métodos:

— Constructor por defecto y constructor con DNI, saldo e interés

- Accedentes y mutadores. Para el número de cuenta no habrá mutador.

- actualizarSaldo(): actualizará el saldo de la cuenta aplicándole el interés diario (interés anual dividido entre 365 aplicado al saldo actual).

- ingresar(double): permitirá ingresar una cantidad en la cuenta.

- retirar(double): permitirá sacar una cantidad de la cuenta (si hay saldo). –

- Método que nos permita mostrar todos los datos de la cuenta.

class Program

{

static void Main(string[] args)

{

Console.Write("Ingrese su DNI: ");

long dni = long.Parse(Console.ReadLine());

Console.Write("Ingrese saldo inicial: ");

double saldo = double.Parse(Console.ReadLine());

Console.Write("Ingrese interés anual (%): ");

double interes = double.Parse(Console.ReadLine());

Cuenta miCuenta = new Cuenta(dni, saldo, interes);

int opcion;

do

{

Console.WriteLine("\n=== MENÚ CUENTA BANCARIA ===");

Console.WriteLine("1. Ver datos de la cuenta");

Console.WriteLine("2. Ingresar dinero");

Console.WriteLine("3. Retirar dinero");

Console.WriteLine("0. Salir");

Console.Write("Elija una opción: ");

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

switch (opcion)

{

case 1:

miCuenta.MostrarDatos();

break;

case 2:

Console.Write("Cantidad a ingresar: ");

double ingreso = double.Parse(Console.ReadLine());

miCuenta.Ingresar(ingreso);

break;

case 3:

Console.Write("Cantidad a retirar: ");

double retiro = double.Parse(Console.ReadLine());

miCuenta.Retirar(retiro);

break;

case 0:

Console.WriteLine("Saliendo...");

break;

default:

Console.WriteLine("Opción inválida.");

break;

}

} while (opcion != 0);

}

}

public class Cuenta

{

private long numeroCuenta;

private long dniCliente;

private double saldoActual;

private double interesAnual;

public Cuenta(long dniCliente, double saldoInicial, double interesAnual)

{

numeroCuenta = GenerarNumeroCuenta();

this.dniCliente = dniCliente;

this.saldoActual = saldoInicial;

this.interesAnual = interesAnual;

}

private long GenerarNumeroCuenta()

{

Random rnd = new Random();

string num = "";

for (int i = 0; i < 10; i++)

num += rnd.Next(0, 10);

return long.Parse(num);

}

public void Ingresar(double cantidad)

{

if (cantidad > 0)

{

saldoActual += cantidad;

Console.WriteLine($"Has ingresado {cantidad:C2}. Nuevo saldo: {saldoActual:C2}");

}

else

Console.WriteLine("La cantidad a ingresar debe ser positiva.");

}

public void Retirar(double cantidad)

{

if (cantidad > 0)

{

if (cantidad <= saldoActual)

{

saldoActual -= cantidad;

Console.WriteLine($"Has retirado {cantidad:C2}. Nuevo saldo: {saldoActual:C2}");

}

else

Console.WriteLine("No hay suficiente saldo para retirar esa cantidad.");

}

else

Console.WriteLine("La cantidad a retirar debe ser positiva.");

}

public void MostrarDatos()

{

Console.WriteLine("\n=== DATOS DE LA CUENTA ===");

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

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

Console.WriteLine($"Saldo Actual: {saldoActual:C2}");

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

Console.WriteLine("==========================");

}

}

}

3. Crea las siguientes clases :

- Motor: con métodos para arrancar el motor y apagarlo.

- Rueda: con métodos para inflar la rueda y desinflarla. Ventana: con métodos para abrirla y cerrarla.

- Puerta: con una ventana y métodos para abrir la puerta y cerrar la puerta.

- Coche: con un motor, cuatro ruedas y dos puertas; con los métodos que te parezcan adecuados

/\* class Program

{

static void Main(string[] args)

{

Coche miCoche = new Coche();

miCoche.Motor.Arrancar();

miCoche.Ruedas[0].Inflar();

miCoche.Ruedas[1].Inflar();

miCoche.Puertas[0].Abrir();

miCoche.Puertas[0].Ventana.Abrir();

miCoche.MostrarEstado();

miCoche.Puertas[0].Cerrar();

miCoche.Motor.Apagar();

miCoche.MostrarEstado();

Console.ReadKey();

}

}

namespace VehiculoApp

{

public class Motor

{

public bool Encendido { get; private set; }

public void Arrancar()

{

if (!Encendido)

{

Encendido = true;

Console.WriteLine("Motor arrancado.");

}

else

Console.WriteLine("El motor ya está encendido.");

}

public void Apagar()

{

if (Encendido)

{

Encendido = false;

Console.WriteLine("Motor apagado.");

}

else

Console.WriteLine("El motor ya estaba apagado.");

}

}

public class Rueda

{

public bool Inflada { get; private set; }

public void Inflar()

{

Inflada = true;

Console.WriteLine("Rueda inflada.");

}

public void Desinflar()

{

Inflada = false;

Console.WriteLine("Rueda desinflada.");

}

}

public class Ventana

{

public bool Abierta { get; private set; }

public void Abrir()

{

Abierta = true;

Console.WriteLine("Ventana abierta.");

}

public void Cerrar()

{

Abierta = false;

Console.WriteLine("Ventana cerrada.");

}

}

public class Puerta

{

public Ventana Ventana { get; private set; } = new Ventana();

public bool Abierta { get; private set; }

public void Abrir()

{

Abierta = true;

Console.WriteLine("Puerta abierta.");

}

public void Cerrar()

{

Abierta = false;

Console.WriteLine("Puerta cerrada.");

}

}

public class Coche

{

public Motor Motor { get; private set; } = new Motor();

public Rueda[] Ruedas { get; private set; } = new Rueda[4];

public Puerta[] Puertas { get; private set; } = new Puerta[2];

public Coche()

{

for (int i = 0; i < Ruedas.Length; i++)

Ruedas[i] = new Rueda();

for (int i = 0; i < Puertas.Length; i++)

Puertas[i] = new Puerta();

}

public void MostrarEstado()

{

Console.WriteLine("\n=== ESTADO DEL COCHE ===");

Console.WriteLine($"Motor: {(Motor.Encendido ? "Encendido" : "Apagado")}");

for (int i = 0; i < Ruedas.Length; i++)

Console.WriteLine($"Rueda {i + 1}: {(Ruedas[i].Inflada ? "Inflada" : "Desinflada")}");

for (int i = 0; i < Puertas.Length; i++)

Console.WriteLine($"Puerta {i + 1}: {(Puertas[i].Abierta ? "Abierta" : "Cerrada")} - Ventana: {(Puertas[i].Ventana.Abierta ? "Abierta" : "Cerrada")}");

Console.WriteLine("========================\n");

}

}

}

}\*/