**POO**

1. **Escriba una clase que represente un vehículo con métodos y atributos. Dentro atributos cree uno llamado “placa” y en los métodos cree uno que permita determinar de acuerdo con el día (datetime), si el vehículo tiene restricción por pico y placa o no, en la ciudad de Bogotá. Los vehículos podrán circular así: días pares 6,7,8,9,0 y días impares 1,2,3,4,5.**

using System;

class Vehiculo

{

// Atributos de la clase

public string Placa { get; set; }

// Constructor para inicializar el vehículo con su placa

public Vehiculo(string placa)

{

Placa = placa;

}

// Método para verificar si el vehículo tiene restricción

public bool TieneRestriccion(DateTime fecha)

{

// Obtener el último dígito de la placa

char ultimoDigitoChar = Placa[^1];

int ultimoDigito = int.Parse(ultimoDigitoChar.ToString());

// Verificar si la fecha es día par o impar

bool esDiaPar = fecha.Day % 2 == 0;

// Días pares: 6, 7, 8, 9, 0 pueden circular

if (esDiaPar)

{

return !(ultimoDigito == 6 || ultimoDigito == 7 || ultimoDigito == 8 || ultimoDigito == 9 || ultimoDigito == 0);

}

// Días impares: 1, 2, 3, 4, 5 pueden circular

else

{

return !(ultimoDigito == 1 || ultimoDigito == 2 || ultimoDigito == 3 || ultimoDigito == 4 || ultimoDigito == 5);

}

}

// Método para mostrar el estado del vehículo

public void MostrarEstado(DateTime fecha)

{

if (TieneRestriccion(fecha))

{

Console.WriteLine($"El vehiculo con placa {Placa} tiene restriccion el {fecha:dd/MM/yyyy}.");

}

else

{

Console.WriteLine($"El vehiculo con placa {Placa} puede circular el {fecha:dd/MM/yyyy}.");

}

}

}

class Program

{

static void Main()

{

// Solicitar la placa al usuario

Console.WriteLine("Ingrese la placa del vehiculo:");

string placa = Console.ReadLine();

// Crear una instancia de Vehículo

Vehiculo miVehiculo = new Vehiculo(placa);

// Solicitar la fecha

Console.WriteLine("Ingrese la fecha (formato: dd/mm/yyyy):");

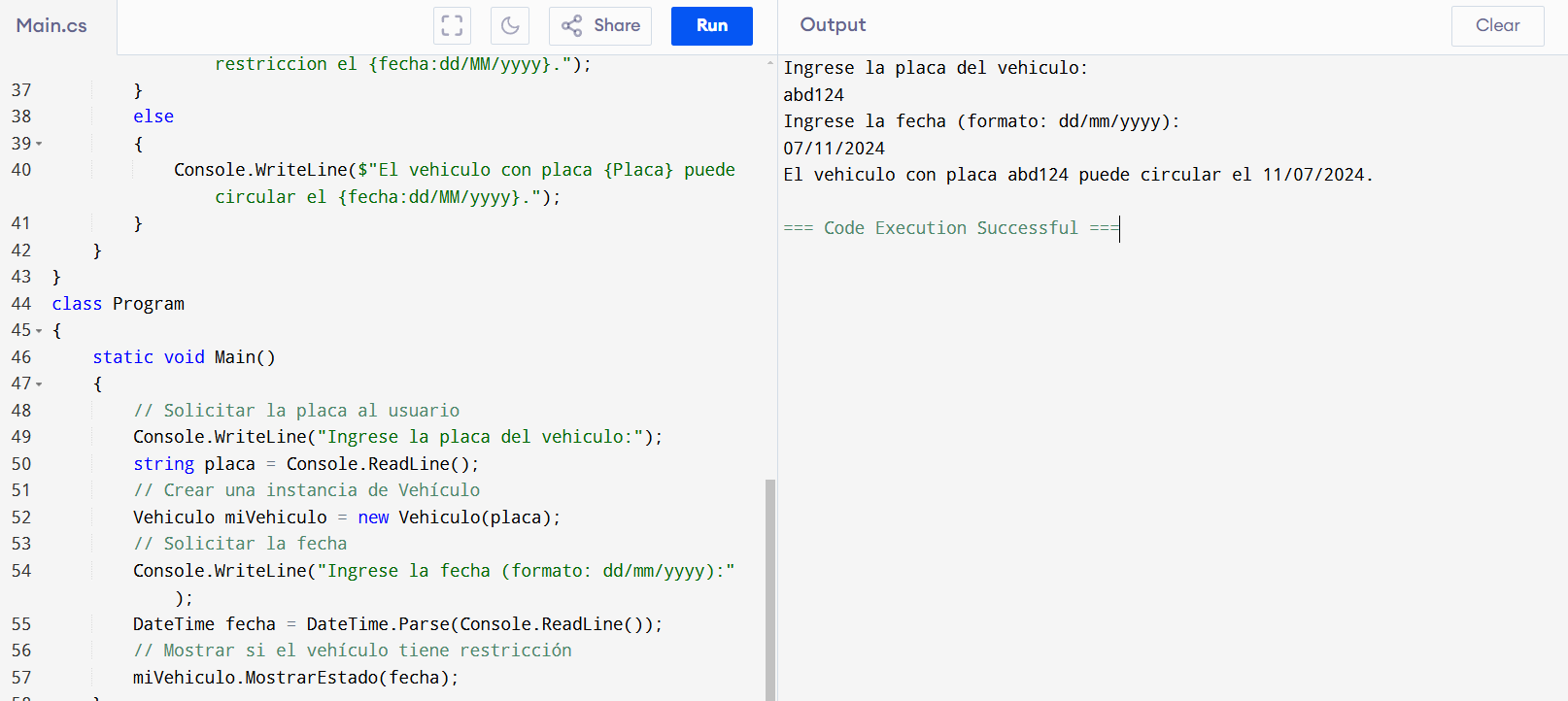
DateTime fecha = DateTime.Parse(Console.ReadLine());

// Mostrar si el vehículo tiene restricción

miVehiculo.MostrarEstado(fecha);

}

}



1. **Escriba una clase llamada Rectangle construida por una longitud y anchura y un método que calculará el área de un rectángulo.**

using System;

class Rectangle

{

// Atributos de la clase

public double Longitud { get; set; }

public double Anchura { get; set; }

// Constructor para inicializar la longitud y la anchura

public Rectangle(double longitud, double anchura)

{

Longitud = longitud;

Anchura = anchura;

}

// Método para calcular el área del rectángulo

public double CalcularArea()

{

return Longitud \* Anchura;

}

// Método para mostrar los detalles del rectángulo

public void MostrarDetalles()

{

Console.WriteLine($"Rectangulo con longitud {Longitud} y anchura {Anchura}:");

Console.WriteLine($"Area: {CalcularArea()}");

}

}

class Program

{

static void Main()

{

// Solicitar al usuario la longitud y la anchura

Console.WriteLine("Ingrese la longitud del rectangulo:");

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

Console.WriteLine("Ingrese la anchura del rectangulo:");

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

// Crear una instancia de Rectangle

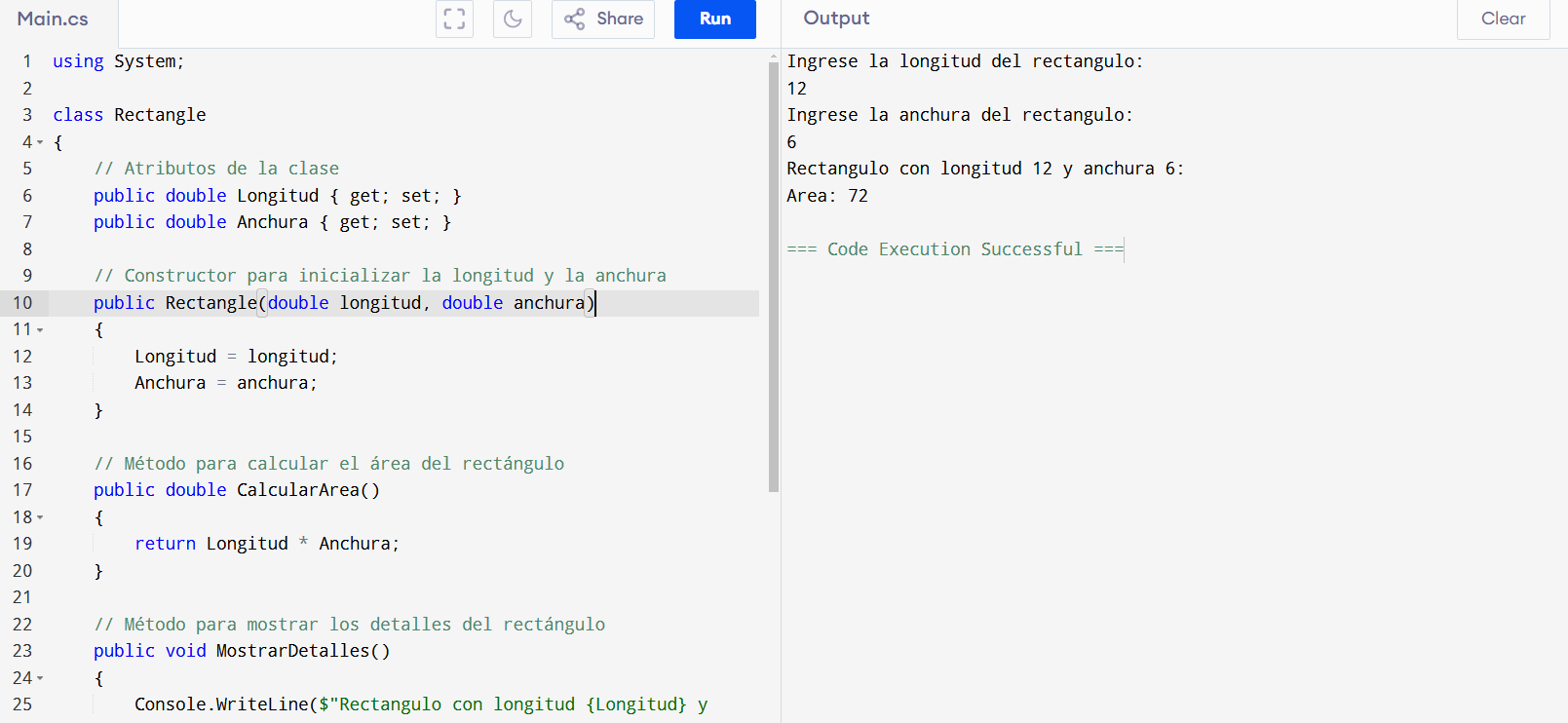
Rectangle miRectangulo = new Rectangle(longitud, anchura);

// Mostrar los detalles y el área del rectángulo

miRectangulo.MostrarDetalles();

}

}

****

1. **Escriba una clase llamada Circle construida por un radio y dos métodos que calcularán el área y el perímetro de un círculo.**

using System;

class Circle

{

// Atributo para el radio

public double Radio { get; set; }

// Constructor para inicializar el radio

public Circle(double radio)

{

Radio = radio;

}

// Método para calcular el área del círculo

public double CalcularArea()

{

return Math.PI \* Radio \* Radio;

}

// Método para calcular el perímetro del círculo

public double CalcularPerimetro()

{

return 2 \* Math.PI \* Radio;

}

// Método para mostrar los detalles del círculo

public void MostrarDetalles()

{

Console.WriteLine($"Circulo con radio {Radio}:");

Console.WriteLine($"Area: {CalcularArea():F2}");

Console.WriteLine($"Perimetro: {CalcularPerimetro():F2}");

}

}

class Program

{

static void Main()

{

// Solicitar al usuario el radio del círculo

Console.WriteLine("Ingrese el radio del circulo:");

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

// Crear una instancia de Circle

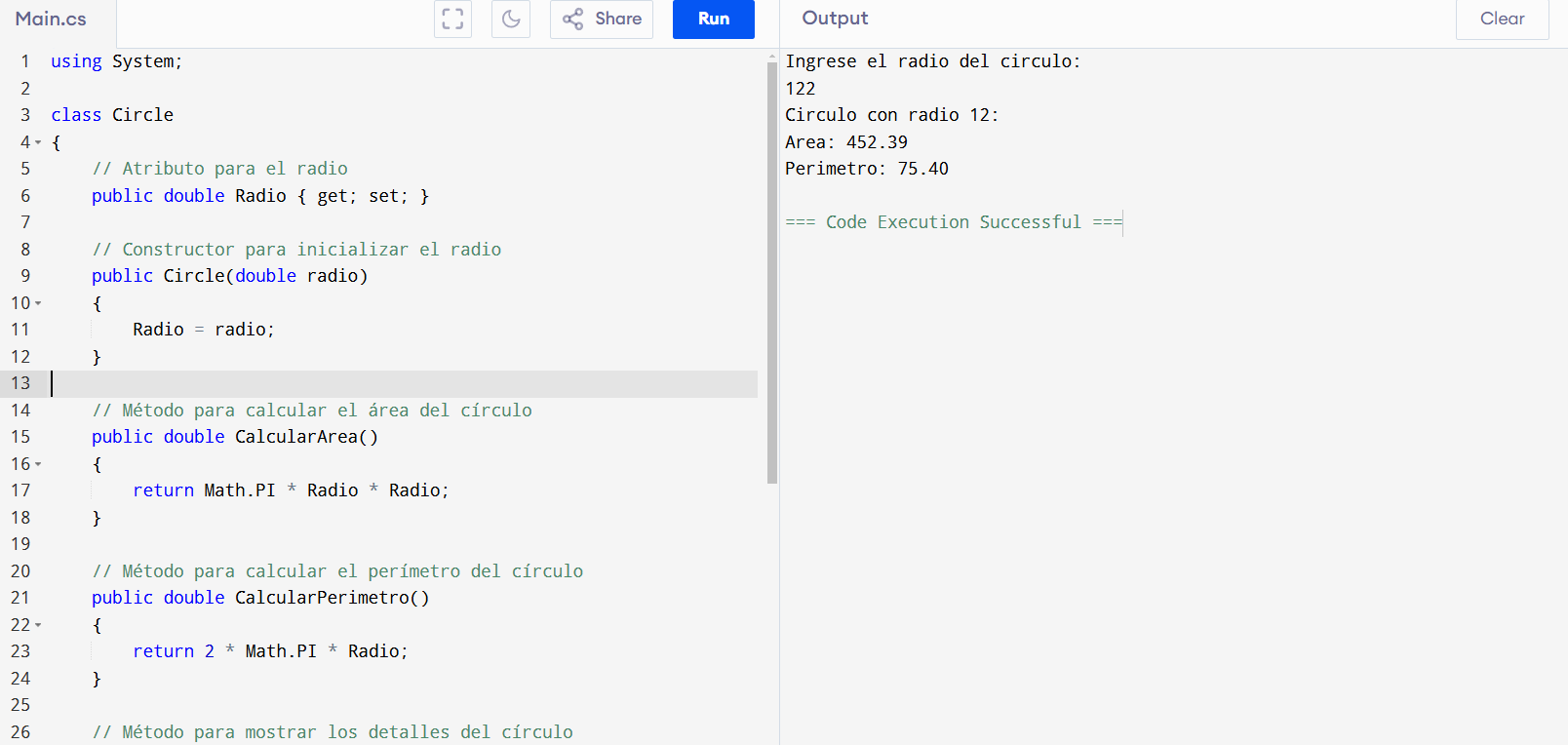
Circle miCirculo = new Circle(radio);

// Mostrar los detalles del círculo

miCirculo.MostrarDetalles();

}

}



1. **Escriba una clase para implementar pow(x, n).**

using System;

class PowerCalculator

{

// Método para calcular x elevado a la potencia n

public double Pow(double baseNumber, int exponent)

{

double result = 1;

// Caso para exponentes positivos

if (exponent > 0)

{

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

{

result \*= baseNumber;

}

}

// Caso para exponentes negativos

else if (exponent < 0)

{

for (int i = 0; i < Math.Abs(exponent); i++)

{

result \*= baseNumber;

}

result = 1 / result;

}

// Si el exponente es 0, el resultado es 1 (x^0 = 1)

return result;

}

// Método para mostrar el resultado

public void MostrarResultado(double baseNumber, int exponent)

{

double resultado = Pow(baseNumber, exponent);

Console.WriteLine($"{baseNumber}^{exponent} = {resultado}");

}

}

class Program

{

static void Main()

{

// Solicitar base y exponente al usuario

Console.WriteLine("Ingrese la base (x):");

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

Console.WriteLine("Ingrese el exponente (n):");

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

// Crear una instancia de PowerCalculator

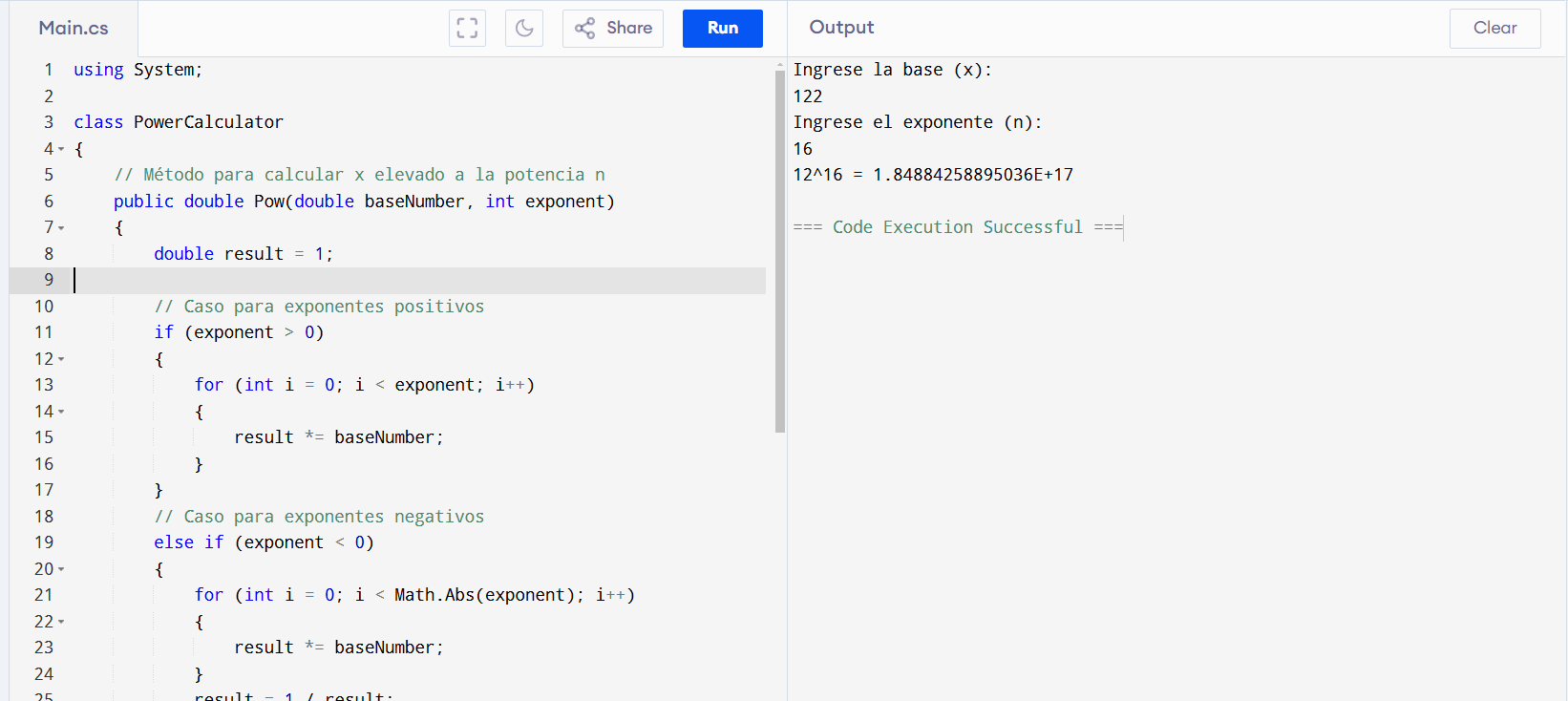
PowerCalculator calculator = new PowerCalculator();

// Mostrar el resultado

calculator.MostrarResultado(baseNumber, exponent);

}

}



1. **Escriba un clase padre llamada Ave que herede a clases hijas con tipos de aves.**

using System;

// Clase Padre: Ave

class Ave

{

// Atributos comunes para todas las aves

public string Nombre { get; set; }

public string Color { get; set; }

public double Tamano { get; set; } // Tamaño en cm

// Constructor para inicializar los atributos

public Ave(string nombre, string color, double tamano)

{

Nombre = nombre;

Color = color;

Tamano = tamano;

}

// Método para que el ave realice un sonido (puede ser sobreescrito por las clases hijas)

public virtual void EmitirSonido()

{

Console.WriteLine($"{Nombre} esta emitiendo un sonido generico.");

}

// Método para describir al ave

public void DescribirAve()

{

Console.WriteLine($"Nombre: {Nombre}, Color: {Color}, Tamano: {Tamano} cm.");

}

}

// Clase Hija: Loro

class Loro : Ave

{

public Loro(string color, double tamano)

: base("Loro", color, tamano) { }

// Sobrescribir el método EmitirSonido

public override void EmitirSonido()

{

Console.WriteLine("El loro dice: Hola! Hola!");

}

}

// Clase Hija: Águila

class Aguila : Ave

{

public Aguila(double tamano)

: base("Aguila", "Marron con blanco", tamano) { }

// Sobrescribir el método EmitirSonido

public override void EmitirSonido()

{

Console.WriteLine("El aguila emite un fuerte grito: Ki-aaah!");

}

}

// Clase Hija: Pingüino

class Pinguino : Ave

{

public Pinguino(double tamano)

: base("Pinguino", "Blanco y negro", tamano) { }

// Sobrescribir el método EmitirSonido

public override void EmitirSonido()

{

Console.WriteLine("El pinguino emite un sonido: Quack! Quack!");

}

}

class Program

{

static void Main()

{

// Crear instancias de diferentes aves

Loro miLoro = new Loro("Verde", 30);

Aguila miAguila = new Aguila(100);

Pinguino miPinguino = new Pinguino(60);

// Describir y emitir sonidos de cada ave

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

miLoro.DescribirAve();

miLoro.EmitirSonido();

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

miAguila.DescribirAve();

miAguila.EmitirSonido();

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

miPinguino.DescribirAve();

miPinguino.EmitirSonido();

}

}

