using Curso\_C\_;

using CursoCSharp.Exercicios;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Runtime.Intrinsics.X86;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace CursoCSharpBackup.Exercicios

{

     1. Classe e Instanciação Básica

     Crie uma classe chamada Pessoa com as propriedades Nome e Idade.Adicione um construtor que inicialize essas propriedades e um método Falar que imprima uma mensagem dizendo "Olá, meu nome é [Nome] e eu tenho [Idade] anos.". Em seguida, crie uma instância da classe Pessoa, defina os valores das propriedades e chame o método Falar.

    public class Pessoa

    {

        public string Nome { get; set; }

        public int Idade { get; set; }

        public Pessoa(string nome, int idade)

        {

            Nome = nome;

            Idade = idade;

        }

        public void Falar()

        {

            Console.WriteLine($"Olá, meu nome é {Nome} e eu tenho {Idade} anos.");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Pessoa pessoa = new Pessoa("Davi", 21);

            pessoa.Falar();

        }

    }

     2. Métodos com Parâmetros

     Crie uma classe Calculadora com um método Soma que recebe dois parâmetros inteiros e retorna a soma deles.Adicione também um método Subtracao que realiza a subtração de dois números.Instancie a classe Calculadora, use os métodos para fazer algumas operações e exiba os resultados.

    public class Calculadora

    {

        public int Soma(int numero1, int numero2)

        {

            return numero1 + numero2;

        }

        public int Subtracao(int numero1, int numero2)

        {

            return numero1 - numero2;

        }

        static void Main(string[] args)

        {

            Calculadora calculadora = new Calculadora();

            int soma = calculadora.Soma(5, 3);

            Console.WriteLine($"A soma de 5 e 3 é: {soma}");

            int subtracao = calculadora.Subtracao(5, 3);

            Console.WriteLine($"A subtração de 5 e 3 é: {subtracao}");

        }

    }

     3. Encapsulamento e Propriedades

    Crie uma classe Carro com as propriedades Modelo, Ano e Preco.Utilize propriedades automáticas para encapsular essas informações. Adicione um método ExibirInfo que exiba os detalhes do carro.Instancie a classe Carro, defina valores para as propriedades e chame o método ExibirInfo.

    public class Carro

    {

        public string Modelo { get; set; }

        public int Ano { get; set; }

        public decimal Preco { get; set; }

        public Carro(string modelo, int ano, decimal preco)

        {

            Modelo = modelo;

            Ano = ano;

            Preco = preco;

        }

        public void ExibirInfo()

        {

            Console.WriteLine($"Modelo: {Modelo}, Ano: {Ano}, Preço: R${Preco}");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Carro carro = new Carro("Civic", 2012, 38400);

            carro.ExibirInfo();

        }

    }

     4. Métodos Estáticos

     Crie uma classe Matematica com um método estático Fatorial que calcula o fatorial de um número inteiro não-negativo.O método deve retornar o resultado.Utilize o método Fatorial para calcular o fatorial de alguns números e exiba os resultados.

    public class Matematica

    {

*// Método estático para calcular o fatorial*

        public static long Fatorial(int numero)

        {

            if (numero < 0)

                throw new ArgumentException("O número deve ser não-negativo.");

            if (numero == 0 || numero == 1)

                return 1;

            long resultado = 1;

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

            {

                resultado \*= i;

            }

            return resultado;

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

*// Calculando o fatorial de alguns números*

            int[] numeros = { 0, 1, 5, 7 };

            foreach (int numero in numeros)

            {

                long fatorial = Matematica.Fatorial(numero);

                Console.WriteLine($"O fatorial de {numero} é: {fatorial}");

            }

        }

    }

     5. Construtores Sobrecarrregados

     Crie uma classe Retangulo com propriedades Largura e Altura.Implemente dois construtores: um que aceita largura e altura como parâmetros e outro que usa valores padrão(largura e altura de 1). Adicione um método CalcularArea que retorna a área do retângulo.Instancie a classe Retangulo usando ambos os construtores e exiba a área calculada.

    public class Retangulo

    {

        public double Largura { get; set; }

        public double Altura { get; set; }

        public Retangulo(double largura, double altura)

        {

            Largura = largura;

            Altura = altura;

        }

        public Retangulo() : *this*(1, 1) // Chama o construtor com parâmetros com valores padrão

        {

        }

        public double CalcularArea()

        {

            return Largura \* Altura;

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Retangulo retangulo1 = new Retangulo(5, 3);

            Console.WriteLine($"Área do retângulo 1 (Largura: {retangulo1.Largura}, Altura: {retangulo1.Altura}): {retangulo1.CalcularArea()}");

            Retangulo retangulo2 = new Retangulo();

            Console.WriteLine($"Área do retângulo 2 (Largura: {retangulo2.Largura}, Altura: {retangulo2.Altura}): {retangulo2.CalcularArea()}");

        }

    }

     6. Métodos de Instância e Estáticos

    Crie uma classe Pessoa com um método de instância AumentarIdade que incrementa a idade da pessoa e um método estático CriarPessoa que cria uma nova instância de Pessoa com idade inicial de 0. Instancie a classe usando o método estático e chame o método AumentarIdade.

    public class Pessoa

    {

        public int Idade { get; private set; }

        public void AumentarIdade()

        {

            Idade++;

        }

        public static Pessoa CriarPessoa()

        {

            return new Pessoa { Idade = 0 }; *// Inicializa a idade como 0*

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

            Pessoa pessoa = Pessoa.CriarPessoa();

            Console.WriteLine($"Idade inicial: {pessoa.Idade}");

            pessoa.AumentarIdade();

            Console.WriteLine($"Idade após aumento: {pessoa.Idade}");

        }

    }

     7. Herança Básica

     Crie uma classe base chamada Animal com um método EmitirSom.Em seguida, crie duas classes derivadas, Cachorro e Gato, que sobrescrevem o método EmitirSom para emitir sons específicos para cada animal. Instancie as classes Cachorro e Gato, e chame o método EmitirSom para cada uma delas.

    public class Animal

    {

*// Método que será sobrescrito pelas classes derivadas*

        public virtual void EmitirSom()

        {

            Console.WriteLine("O animal emitiu um som.");

        }

    }

    public class Cachorro : Animal

    {

*// Sobrescrevendo o método EmitirSom*

        public override void EmitirSom()

        {

            Console.WriteLine("O cachorro faz: Au Au!");

        }

    }

    public class Gato : Animal

    {

*// Sobrescrevendo o método EmitirSom*

        public override void EmitirSom()

        {

            Console.WriteLine("O gato faz: Miau!");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

*// Instanciando a classe Cachorro*

            Animal cachorro = new Cachorro();

            cachorro.EmitirSom(); *// Chama o método EmitirSom do Cachorro*

*// Instanciando a classe Gato*

            Animal gato = new Gato();

            gato.EmitirSom(); *// Chama o método EmitirSom do Gato*

        }

    }

     8. Polimorfismo

     Crie uma classe base Forma com um método abstrato CalcularArea.Em seguida, crie classes derivadas Circulo e Quadrado que implementam o método CalcularArea.Instancie cada forma e utilize uma referência da classe base para chamar o método CalcularArea polimorficamente.

    public abstract class Forma

    {

*// Método abstrato que deve ser implementado nas classes derivadas*

        public abstract double CalcularArea();

    }

    public class Circulo : Forma

    {

        public double Raio { get; set; }

        public Circulo(double raio)

        {

            Raio = raio;

        }

*// Implementação do método CalcularArea para o círculo*

        public override double CalcularArea()

        {

            return Math.PI \* Raio \* Raio;

        }

    }

    public class Quadrado : Forma

    {

        public double Lado { get; set; }

        public Quadrado(double lado)

        {

            Lado = lado;

        }

*// Implementação do método CalcularArea para o quadrado*

        public override double CalcularArea()

        {

            return Lado \* Lado;

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

*// Instanciando um círculo e um quadrado*

            Forma circulo = new Circulo(5);

            Forma quadrado = new Quadrado(4);

*// Chamando o método CalcularArea polimorficamente*

            Console.WriteLine($"Área do círculo: {circulo.CalcularArea()}");

            Console.WriteLine($"Área do quadrado: {quadrado.CalcularArea()}");

        }

    }

     9. Interfaces

     Crie uma interface IVeiculo com um método Acelerar.Em seguida, crie uma classe Carro que implementa a interface IVeiculo e define o comportamento do método Acelerar.Instancie a classe Carro e chame o método Acelerar.

    public interface IVeiculo

    {

        void Acelerar();

    }

    public class Carro : IVeiculo

    {

        public void Acelerar()

        {

            Console.WriteLine("O carro está acelerando!");

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

*// Instanciando a classe Carro*

            IVeiculo meuCarro = new Carro();

*// Chamando o método Acelerar*

            meuCarro.Acelerar();

        }

    }

     10. Propriedades Calculadas

     Crie uma classe ContaBancaria com as propriedades Saldo e NomeTitular.Adicione uma propriedade calculada chamada SaldoComBonus que retorna o saldo atual mais um bônus de 10%. Crie uma instância da classe ContaBancaria, defina o saldo e exiba o valor de SaldoComBonus.

    public class ContaBancaria

    {

*// Propriedades*

        public double Saldo { get; set; }

        public string NomeTitular { get; set; }

*// Propriedade calculada que retorna o saldo com um bônus de 10%*

        public double SaldoComBonus

        {

            get { return Saldo \* 1.10; }

        }

    }

    class Program

    {

        static void Main(string[] args)

        {

*// Criando uma instância da classe ContaBancaria*

            ContaBancaria conta = new ContaBancaria();

*// Definindo o saldo e o nome do titular*

            conta.Saldo = 1000.00; *// Exemplo de saldo*

            conta.NomeTitular = "João Silva";

*// Exibindo o saldo com bônus*

            Console.WriteLine($"Titular: {conta.NomeTitular}");

            Console.WriteLine($"Saldo atual: {conta.Saldo}");

            Console.WriteLine($"Saldo com bônus: {conta.SaldoComBonus}");

        }

    }

}