PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Breno Vieira Soares

Daniel Jorge Pacheco Gonzaga

Robert Victor Souza Cunha

Arthur Henrique de Paulo Abreu

João Gontijo de Oliveira Júnior

RELATÓRIO LABORATÓRIO 2 PROGRAMAÇÃO ORIENTADA POR OBJETOS

Belo Horizonte

2017

Breno Vieira Soares

Daniel Jorge Pacheco Gonzaga

Robert Victor Souza Cunha

Arthur Henrique de Paulo Abreu

João Gontijo de Oliveira Júnior

RELATÓRIO LABORATÓRIO 2 PROGRAMAÇÃO ORIENTADA POR OBJETOS

Relatório das atividades desenvolvidas no laboratório 2 da disciplina de programação orientada por objetos ministrada pelo professor Paulo Cesar do Amaral Ferreira como requisito parcial para obtenção de nota para o segundo período do curso de Sistemas de Informação.

Belo Horizonte

2017

SUMÁRIO

1 INTRODUÇÃO

O trabalho que será apresentado mais adiante foi realizado de acordo com as normas de padronização de documentos da ABNT (Associação Brasileira de Normas Técnicas) em paralelo ao padrão de normalização de documentos acadêmicos da Pontifícia Universidade Católica de Minas Gerais. O trabalho é divido em subtítulos que representam cada exercício proposto pelo professor e títulos de nível terciário que contém informações dos exercícios tais como código fonte, impressão da tela e valor das variáveis.

2 DESENVOLVIMENTO

2.1 Ex3.1

2.1.1 Código fonte Ex3.1

2.1.1.1 Classe 1

```
//

// nome do programa: Ex31

//

// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo

// data: 24/09/2017

// entrada(s): Tipo de acao a ser realizada, e datas

// saida(s): imprime diferenca de datas, proximo dia de uma data, data por extenso, quantidade de dias tem um mes,

// resultado se o ano é bissexto, resultado se a data é valida, dia da data naquele ano.

// para testar digite:

// Ex31.exe

// descricao: Executa varias funcoes com data, como dar a diferenca entre duas datas, verificar se um ano é bissexto

// dentre outras.

//

using System;

using System.Threading;

using System.Text;

using Date_5;


namespace Ex_31_LAB2

{

  class Program
```

```
        {
          static void Main(string[] args)
          {
                menu();
          }
        public static void menu()
        {
                int ano = 0,mes = 0,dia = 0;
                int op = 0;
                bool dataValida = true;
                string data  = " ";
                Date5 data1 = new Date5();
                Date5 data2 = new Date5();
                Console.Clear();
                Console.WriteLine("********************** Testes Classe Date5 **********************");
                Console.WriteLine("**********************                Menu **********************");
                Console.WriteLine("1 - Diferença de datas");
                Console.WriteLine("2 - Próximo dia de uma data");
                Console.WriteLine("3 - Data por extenso");
                Console.WriteLine("4 - Quantidade de dias tem o mes");
                Console.WriteLine("5 - Verificar se o ano é bissexto");
                Console.WriteLine("6 - Verificar se uma data é válida");
                Console.WriteLine("7 - Dia da data");
                Console.WriteLine("0 - Sair");
        op = int.Parse(Console.ReadLine());
```

```
            switch(op)
            {
                case 1:
                    Console.Clear();
                    Console.WriteLine("Digite a primeira data, com dia,mes e ano respectivamente separados por '/'");
                    data = Console.ReadLine();
                    if (quebraData(ref ano, ref mes, ref dia, data))
                    {
                        data1 = new Date5(ano, mes, dia);
                    }
                    else
                    {
                        menu();
                        break;
                    }
                    if (data1.dataValida())
                    {
                        Console.WriteLine("\nDigite a segunda data, com dia,mes e ano respectivamente separados por '/'");
                        data = Console.ReadLine();
                        if (quebraData(ref ano, ref mes, ref dia, data))
                        {
                            data2 = new Date5(ano, mes, dia);
                        }
                        else
                        {
                            menu();
```

```
                                        break;
                                }
                                if (!data2.dataValida())
                                {
                                        dataValida =  false;
                                }
                        }
                        else
                        {
                                dataValida = false;
                        }
                        if (dataValida)
                        {
                                Console.WriteLine("\n"+diferencaDias(data1,
data2));
                                voltarMenu();
                        }
                        else
                        {
                                Console.WriteLine("Data inválida");
                        }
                        break;
                case 2:
                        Console.Clear();
                        Console.WriteLine("Digite uma data, com dia,mes e ano
respectivamente separados por '/'");
                        data = Console.ReadLine();
                        if (quebraData(ref ano, ref mes, ref dia, data))
```

```
                {
                        data1 = new Date5(ano, mes, dia);
                }
                else
                {
                        menu();
                        break;
                }
                if (data1.dataValida())
                {
                        data1.proximoDia();
                        Console.WriteLine("O próximo dia da data é o dia "+data1.getDia());
                        voltarMenu();
                }
                else
                {
                        Console.WriteLine("Data inválida");
                }
                break;
        case 3:
                Console.Clear();
                Console.WriteLine("Digite uma data, com dia,mes e ano respectivamente separados por '/'");
                data = Console.ReadLine();
                if (quebraData(ref ano, ref mes, ref dia, data))
                {
                        data1 = new Date5(ano, mes, dia);
```

```
                }
                else
                {
                        menu();
                        break;
                }
                if (data1.dataValida())
                {
                        Console.WriteLine(data1.extenso());
                        voltarMenu();
                }
                else
                {
                        Console.WriteLine("Data inválida");
                }
                break;
        case 4:
                Console.Clear();
                Console.WriteLine("Digite uma data, com dia,mes e ano respectivamente separados por '/'");
                data = Console.ReadLine();
                if (quebraData(ref ano, ref mes, ref dia, data))
                {
                        data1 = new Date5(ano, mes, dia);
                }
                else
                {
```

```
                menu();
                break;
            }
            if (data1.dataValida())
            {
                string mesExtenso = System.Globalization.DateTimeFormatInfo.CurrentInfo.GetMonthName(mes).ToLower();
                Console.WriteLine(mesExtenso+" de "+ano+" tem "+data1.diasMes()+" dias");
                voltarMenu();
            }
            else
            {
                Console.WriteLine("Data inválida");
            }
            break;
        case 5:
            Console.Clear();
            Console.WriteLine("Digite um ano");
            ano = int.Parse(Console.ReadLine());
            if (ano > 0)
            {
                if (data1.bissexto(ano))
                {
                    Console.WriteLine(ano+" é bissexto!");
                    voltarMenu();
                }
```

```
                    else
                    {
                        Console.WriteLine(ano+" não é bissexto!");
                        voltarMenu();
                    }
                }
                else
                {
                    Console.WriteLine("Ano inválido");
                }
                break;
            case 6:
                Console.Clear();
                Console.WriteLine("Digite uma data, com dia,mes e ano respectivamente separados por '/'");
                data = Console.ReadLine();
                if (quebraData(ref ano, ref mes, ref dia, data))
                {
                    data1 = new Date5(ano, mes, dia);
                }
                else
                {
                    menu();
                    break;
                }
                if (data1.dataValida())
                {
```

```
                    Console.WriteLine("A data é válida !");
                    voltarMenu();
                }
                else
                {
                    Console.WriteLine("Data inválida");
                    voltarMenu();
                }
                break;
            case 7:
                Console.Clear();
                Console.WriteLine("Digite uma data, com dia,mes e ano respectivamente separados por '/'");
                data = Console.ReadLine();
                if (quebraData(ref ano, ref mes, ref dia, data))
                {
                    data1 = new Date5(ano, mes, dia);
                }
                else
                {
                    menu();
                    break;
                }
                if (data1.dataValida())
                {
                    Console.WriteLine(data1.extenso()+" é o dia "+data1.diaData()+" do ano");
                    voltarMenu();
```

```
                        }
                        else
                        {
                                Console.WriteLine("Data inválida");
                        }
                        break;
                case 0:
                        sair();
                        break;
                default:
                        Console.WriteLine("Digite uma opção válida, aperte qualquer tecla para voltar ao menu");
                        Console.ReadKey();
                        menu();
                        break;
            }
        }

        public static void sair()
        {
            Console.Write("Saindo em ");
            for (int i  = 3; i >= 1; i--)
            {
                Console.Write(i+" ");
                Thread.Sleep(500);
            }
            Console.Clear();
```

```
            Environment.Exit(0);
        }

        public static void voltarMenu()
        {
            char resposta = ' ';
            Console.WriteLine("Deseja voltar ao menu inicial ? s/n");
            resposta = char.Parse(Console.ReadLine());
            if (resposta == 's' || resposta == 'S')
            {
                menu();
            }
            else
            {
                sair();
            }
        }

        public static bool quebraData(ref int ano, ref int mes, ref int dia, string data)
        {
            string[] arrayData = data.Split('/');
            if (arrayData.Length != 3)
            {
                Console.WriteLine("Data com argumentos inválidos !, aperte qualquer tecla para voltar ao menu");
                Console.ReadKey();
                return false;
```

```
            }
            for (int i = 0; i < arrayData.Length; i++)
            {
                if (!validaInt(arrayData[i]))
                {
                    Console.WriteLine("Data com argumentos inválidos !, aperte qualquer tecla para voltar ao menu");
                    Console.ReadKey();
                    return false;
                }
            }
            ano = int.Parse(arrayData[2]);
            mes = int.Parse(arrayData[1]);
            dia = int.Parse(arrayData[0]);
            return true;
        }

        public static bool validaInt(string numero)
        {
            uint verificaConversao = 0;
            if (uint.TryParse(numero, out verificaConversao ))
            {
                return true;
            }
            else
            {
                return false;
```

```
            }
        }
        public static string diferencaDias(Date5 data1, Date5 data2)
        {
            string dataInicial      = data1.extenso();
            string dataFinal        = data2.extenso();
            string retorno              = "";
            string day                  = "dias";
            int quantDias               = 0;
            Date5 aux = new Date5();

            DateTime dt1 = new DateTime(data1.getAno(),
data1.getMes(),data1.getDia());
            DateTime dt2 = new DateTime(data2.getAno(),
data2.getMes(),data2.getDia());
            if (dt1 > dt2)
            {
                dataFinal = data1.extenso();
                dataInicial = data2.extenso();
                aux = data2;
                data2 = data1;
                data1 = aux;
            }
            quantDias = data1.diffDias(data2,data1);
            if (quantDias == 1)
            {
                day = "dia";
            }
```

```
            retorno = "O numero de dias entre "+dataInicial+" até "+dataFinal+" é de "+quantDias+" "+day;
            return retorno;
        }
         }
        }
```

2.1.1.2 Classe 2

```
        using System;
        using System.Threading;
        using System.Globalization;

        namespace Date_5 {
        class Date5
        {
            public int dia;
            public int mes;
            public int ano;

            public Date5()
            {
                this.dia = 1;
                this.mes = 1;
                this.ano = 2000;
            }
            public Date5 (int ano, int mes, int dia)
```

```
        {
            this.dia = dia;
            this.mes = mes;
            this.ano = ano;
        }

        public void proximoDia()
        {
            if (this.dia >= diasMes())
            {
                this.dia = 1;
                if (this.mes >= 12)
                {
                    this.mes = 1;
                    this.ano++;
                }
                else
                {
                    this.mes++;
                }
            }
            else
            {
                this.dia++;
            }
        }
```

```
        public int proximoDia(int ano, int mes, int dia)
        {
            if (dia >= diasMes(ano, mes))
            {
                dia = 1;
            }
            else
            {
                dia++;
            }
            return dia;
        }
        public string extenso()
        {
            CultureInfo culture = new CultureInfo("pt-BR");
            DateTime myDateTime = new System.DateTime(this.ano,
this.mes, this.dia);
            return myDateTime.ToLongDateString();
        }

        public string extenso(int ano, int mes, int dia)
        {
            CultureInfo culture = new CultureInfo("pt-BR");
            DateTime myDateTime = new System.DateTime(ano, mes, dia);
            return myDateTime.ToLongDateString();
        }
        public int diasMes()
```

```
        {
            int[] diasMes = new int[] { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31 };
            if (this.mes == 2 && bissexto())
            {
                diasMes[1] = 29;
            }

            return diasMes[this.mes - 1];
        }

        public int diasMes(int ano, int mes)
        {
            int[] diasMes = new int[] { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31 };
            if (mes == 2 && bissexto(ano))
            {
                diasMes[1] = 29;
            }

            return diasMes[mes - 1];
        }

        public bool bissexto()
        {
            if(((this.ano % 400 == 0) || (this.ano % 4 == 0)) && (this.ano %
100 != 0))
            {
```

```
                return true;
            }
            else
            {
                return false;
            }
        }

        public bool bissexto(int ano)
        {
            if(((ano % 400 == 0) || (ano % 4 == 0)) && (ano % 100 != 0))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        public int diffDias(Date5 objetoDataFinal, Date5 objetoDataInicial)
        {
            int quantidadeDias = 0;
            DateTime dataFinal = new DateTime(objetoDataFinal.getAno(),
objetoDataFinal.getMes(), objetoDataFinal.getDia());
            DateTime dataInicial = new DateTime(objetoDataInicial.getAno(),
objetoDataInicial.getMes(), objetoDataInicial.getDia());
            TimeSpan quantidadeDiasTs = dataFinal - dataInicial;
```

```
            return quantidadeDias = quantidadeDiasTs.Days;
        }

        public bool dataValida()
        {
            if (this.ano < 1)
            {
                return false;
            }
            else if (this.mes > 12 || this.mes < 1)
            {
                return false;
            }
            else if (this.dia > diasMes(this.ano, this.mes) || dia < 1)
            {
                return false;
            }
            else
            {
                return true;
            }
        }

        public bool dataValida(int ano, int mes, int dia)
        {
            if (dia > diasMes(ano,mes) || dia < 1)
            {
```

```
                return false;
            }
            else if (mes > 12 || mes < 1)
            {
                return false;
            }
            else if (ano < 1)
            {
                return false;
            }
            else
            {
                return true;
            }
        }

        public int diaData()
        {
            int quantidadeDias = 0;
            for (int i = 1; i <= this.mes; i++)
            {
                if (i == this.mes)
                {
                    quantidadeDias += this.dia;
                }
                else
                {
```

```
                        quantidadeDias += diasMes(this.ano, i);
                    }
                }
                return quantidadeDias;
            }

            public int diaData(int ano, int mes, int dia)
            {
                int quantidadeDias = 0;
                for (int i = 1; i <= mes; i++)
                {
                    if (i == mes)
                    {
                        quantidadeDias += dia;
                    }
                    else
                    {
                        quantidadeDias += diasMes(ano, i);
                    }
                }
                return quantidadeDias;
            }

            public int getDia()
            {
                return dia;
            }
```

```
        public int setDia(int dia)
        {
            return this.dia = dia;
        }
        public int getMes()
        {
            return mes;
        }
        public int setMes(int mes)
        {
            return this.mes = mes;
        }
        public int getAno()
        {
            return ano;
        }
    }
    }
```

2.2 **Ex3.2**

2.2.1 Código fonte Ex3.2

2.2.1.1 Classe 1

```
    // Programa: Conta
    // Programadores: João Gontijo, Daniel Jorge, Robert Victor, Breno Vieira, Arthur Hentirque
    // Data: 22/09/2017
    // Descrição: Programa usando classes para simular várias contas bancarias
    // Entradas: 1 a 7 para as opções 8 para sair
    // Saidas: Informação ou ação soliciatada pelo menu
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EX3._2
{
    class Program
    {
        static void Main(string[] args)
        {
            string auxTeste, nome;

            int opcao = 0,conta;

            double valor;

            Conta[] vetorConta = new Conta[100];

            while (opcao != 8)
            {
                Console.Clear();

                Console.Write("\n 1 - Criar conta\n 2 - Excluir conta existente\n 3 - Depositar em uma conta\n 4 - Sacar em uma conta\n 5 - Imprimir saldo da conta\n 6
```

```
- Imprimir a relação das contas existentes listando o número e o nome do titular da conta\n 7 - Mostrar informações de uma conta\n 8 - Sair\n\n  Digite sua opção: ");
                    auxTeste = Console.ReadLine();

                    bool teste = int.TryParse(auxTeste, out opcao);

                    if (teste == true && opcao > 0 || opcao < 8)
                    {
                        switch (opcao)
                        {
                            case 1: // Criar Conta
                                Console.Clear();

                                Console.Write(" Para criar sua conta, digite seu nome: ");
                                nome = Console.ReadLine();

                                vetorConta[Conta.contador] = new Conta(1, Conta.contador + 1, nome);

                                Console.Write("\n Nome: {2}\n Nº da conta: {0}\n Agência: {1}", vetorConta[Conta.contador].NumConta, vetorConta[Conta.contador].Agencia, vetorConta[Conta.contador].Titular);

                                Console.ReadKey();

                                Conta.contador++;
                                break;
```

```
                        case 2: // Excluir conta existente
                            Console.Clear();


                            Console.Write(" Digite o número da conta que deseija
excluir: ");
                            conta = int.Parse(Console.ReadLine());


                            Console.Write("\n\n Confirme a exclusão (S/N): ");
                            char confirmação = char.Parse(Console.ReadLine());


                            if (confirmação == 's' || confirmação == 'S')
                            {
                                if (conta <= Conta.contador && conta > 0)
                                {
                                    vetorConta[conta - 1].excluirConta();
                                    Console.Write("\n\n Conta excluída com sucesso");
                                    Console.ReadKey();
                                }
                                else
                                {
                                    Console.Write("\n Conta inexistente"); // Erro
                                    Console.ReadKey();
                                }
                            }
                            else
                            {
                                Console.Write("\n Conta não excluída"); // Erro /
Confirmação negada
```

```
                    Console.ReadKey();
                }
                break;
            case 3: // Depositar em uma conta
                Console.Clear();

                Console.Write(" Digite o número da conta que deseija depositar: ");
                conta = int.Parse(Console.ReadLine());

                if (conta <= Conta.contador && conta > 0)
                {
                    Console.Write("\n\n Digite o valor à depositar: ");
                    valor = double.Parse(Console.ReadLine());

                    if (valor > 0)
                    {
                        vetorConta[conta - 1].depositar(valor);
                        Console.Write("\n Valor depositado com suceeso");
                        Console.ReadKey();
                    }
                    else
                    {
                        Console.Write("\n\n Valor Inválido"); // Erro
                        Console.ReadKey();
                    }
                }
```

```
                    else
                    {
                        Console.Write("\n Conta inexistente"); // Erro
                        Console.ReadKey();
                    }

                    break;
                case 4: // Sacar em uma conta
                    Console.Clear();

                    Console.Write(" Digite o número da conta que deseija sacar:
");
                    conta = int.Parse(Console.ReadLine());

                    if (conta <= Conta.contador && conta > 0)
                    {
                        Console.Write("\n\n Digite o valor à sacar: ");
                        valor = double.Parse(Console.ReadLine());

                        if (valor < vetorConta[conta - 1].Saldo)
                        {
                            vetorConta[conta - 1].sacar(valor);

                            Console.Write("\n Valor retirado com suceeso");
                            Console.ReadKey();
                        }
                        else
```

```
                    {
                        Console.Write("\n\n Saldo inválido"); // Erro
                        Console.ReadKey();
                    }
                }
                else
                {
                    Console.Write("\n Conta inexistente"); // Erro
                    Console.ReadKey();
                }
                break;
            case 5: // Imprimir saldo da conta
                Console.Clear();

                Console.Write(" Digite o número da conta que deseija verificar o saldo: ");
                conta = int.Parse(Console.ReadLine());

                if (conta <= Conta.contador && conta > 0)
                {
                    Console.Write("\n O saldo da conta {0} é de R${1}", conta, vetorConta[conta - 1].Saldo);
                    Console.ReadKey();
                }
                else
                {
                    Console.Write("\n Conta inexistente"); // Erro
                    Console.ReadKey();
```

```
                    }
                    break;
                case 6: // Imprimir a relação das contas existentes listando o número e o nome do titular da conta
                    Console.Clear();

                    for (int i = 0; i < Conta.contador; i++)
                    {
                        if (vetorConta[i].NumConta > 0)
                        {
                            Console.Write("\n\n Nome: {0}\n Nº da conta:{1}", vetorConta[i].Titular, vetorConta[i].Agencia);
                        }
                        else
                        {
                            Console.Write("\n Não possui contas no sistema"); // Erro
                        }
                    }

                    Console.ReadKey();
                    break;
                case 7: // Mostrar informações de uma conta
                    Console.Clear();

                    Console.Write(" Digite o número da conta que deseja verificar as informações: ");
                    conta = int.Parse(Console.ReadLine());
```

```
                        if (conta <= Conta.contador && conta > 0)
                        {
                            Console.Write("\n Nome: {0}\n Nº da conta:{1}\n Agência:{2}\n Saldo: R${3}", vetorConta[conta - 1].Titular, vetorConta[conta - 1].Agencia, vetorConta[conta - 1].NumConta, vetorConta[conta - 1].Saldo);
                            Console.ReadKey();
                        }
                        else
                        {
                            Console.Write("\n Conta inexistente"); // Erro
                            Console.ReadKey();
                        }
                        break;
                    }
                }
                else
                {
                    Console.Clear();
                    Console.Write(" Opção inválida"); // Erro
                    opcao = 0;
                }

            }
        }
    }
}
```

2.2.1.2 Classe 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace EX3._2
{
    class Conta
    {
        // Atributos
        private string titular;
        public string Titular
        {
            get
            {
                return titular;
            }
        }

        private int agencia;
        public int Agencia
        {
            get
            {
                return agencia;
```

```
        }
    }

    private int numConta;
    public int NumConta
    {
        get
        {
            return numConta;
        }
    }

    private double saldo = 0;
    public double Saldo
    {
        get
        {
            return saldo;
        }
        set
        {
            if (value > 0.0)
            {
                saldo = value;
            }
            else
            {
```

```
                saldo = 0.0;
            }
        }
    }

    public Conta(int agencia, int numConta, string titular) // Construtor
    {
        this.agencia = agencia;
        this.numConta = numConta;
        this.titular = titular;
    }

    public void excluirConta() // Excluir conta
    {
        agencia = 0;
        numConta = 0;
        titular = null;
        saldo = 0;
    }

    public void depositar(double valor) // Depositar
    {
        saldo = valor;
    }

    public void sacar(double valor) // sacar
    {
```

```
                saldo -= valor;
            }


            public static int contador = 0; // Contador
        }
    }
```

2.3 Ex3.3

2.3.1 Código fonte Ex3.3

2.3.1.1 Classe 1

```
    //
    // nome do programa: Ex33
    //
    // programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
    // data: 24/09/2017
    // entrada(s): Tipo de acao a ser realizada, escolhida pelo menu
    // saida(s): Devolver a opção desejada, realizando a ação
    // descricao: Executa varias funcoes relacionada a pessoa, como UMC, salvar dados, calcular idade e entre outras
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;


    namespace Exercicios
    {
        class Ex33
```

```
        {
            const int MAXPESSOAS = 100;
            static Pessoa [] vetPessoas = new Pessoa [MAXPESSOAS];
            static void Main(string[] args)
            {
                int op = 0;
                int d, m, a, x;
                double pe, al, idade, imc;
                string data;
                string[] datas;
                do
                {
                    Console.Clear();
                    Menu();
                    op = int.Parse(Console.ReadLine());

                    switch (op)
                    {
                        case 1:
                            Console.Clear();
                            Console.WriteLine("1. Incluir nova pessoa");
                            Console.WriteLine("--------------------------------");
                                                if (Pessoa.contador <= MAXPESSOAS)
                                                {
                            Console.Write("\nDigite a data de nascimento no seguinte formato: dia/mes/ano: ");
                            data = Convert.ToString(Console.ReadLine());
```

```
                    if (Verifica(data, out datas))
                    {
                        d = Convert.ToInt32(datas[0]);
                        m = Convert.ToInt32(datas[1]);
                        a = Convert.ToInt32(datas[2]);
                        if (Pessoa.contador <= MAXPESSOAS)
                        {
                            if (Verifica(d, m, a))
                            {
                                DateTime Data = new DateTime(a, m, d);
                                Console.Write("\nDigite a altura em cm: ");
                                al = double.Parse(Console.ReadLine());
                                Console.Write("\nDigite o peso em kg: ");
                                pe = double.Parse(Console.ReadLine());
                                if (Verifica(al, pe))
                                {
                                    vetPessoas[Pessoa.contador - 1] = new
Pessoa(Data, al, pe);
                                    Console.WriteLine("\nPessoa cadastrada, número de
registro: {0} - Pressione qualquer tecla para sair", Pessoa.contador);
                                    Pessoa.contador++;
                                }

                                else
                                {
                                    Console.WriteLine("\nAltura ou peso informado
inválidos - Pressione qualquer tecla para sair");
                                }
```

```
                    }


                    else
                    {
                        Console.WriteLine("\nData em formato correto, porém inválida - Pressione qualquer tecla para sair");
                    }
                }


                else
                {
                    Console.WriteLine("\nLimite máximo de pessoas já foi atingido - Pressione qualquer tecla para sair");
                }
            }
            else
            {
                Console.WriteLine("\nData em formato inválido - Pressione qualquer tecla para sair");
            }
                                }
                                else
                                {
                                    Console.WriteLine("Limite de pessoas já foi atingido - Pressione qualquer tecla para sair");
                                }
            Console.ReadKey();
            break;
```

```
                default:
                    break;


                case 2:
                    Console.Clear();
                    Console.WriteLine("2. Alterar data de nascimento");
                    Console.WriteLine("--------------------------------");
                    Console.Write("\nDigite o número da pessoa: ");
                    x = int.Parse(Console.ReadLine());
                    if (Verifica(x))
                    {
                        Console.Write("\nDigite a data de nascimento no seguinte formato: dia/mes/ano: ");
                        data = Convert.ToString(Console.ReadLine());
                        if (Verifica(data, out datas))
                        {
                            d = Convert.ToInt32(datas[0]);
                            m = Convert.ToInt32(datas[1]);
                            a = Convert.ToInt32(datas[2]);
                            DateTime Data = new DateTime(a, m, d);
                            vetPessoas[x - 1].Nasc = Data;
                            Console.WriteLine("\nData de nascimento alterada - Pressione qualquer tecla para sair");
                        }


                        else
                        {
```

```
                    Console.WriteLine("\nData em formato inválido - Pressione qualquer tecla para sair");
                }
            }
            else
            {
                Console.WriteLine("\nNúmero de pessoa inválido - Pressione qualquer tecla para sair");
            }
            Console.ReadKey();
            break;

        case 3:
            Console.Clear();
            Console.WriteLine("3. Alterar o peso");
            Console.WriteLine("---------------------------------");
            Console.Write("\nDigite o número da pessoa: ");
            x = int.Parse(Console.ReadLine());
            if (Verifica(x))
            {
                Console.Write("\nDigite o novo peso: ");
                pe = double.Parse(Console.ReadLine());
                al = vetPessoas[x - 1].Altura;
                if (Verifica(al, pe))
                {
                    vetPessoas[x - 1].Peso = pe;
                    Console.WriteLine("\nPeso alterado - Pressione qualquer tecla para sair");
```

```
                    }


                    else
                    {
                        Console.WriteLine("\nPeso informado inválido - Pressione
qualquer tecla para sair");
                    }
                }


                else
                {
                    Console.WriteLine("\nNúmero de pessoa inválido - Pressione
qualquer tecla para sair");
                }
                Console.ReadKey();
                break;


            case 4:
                Console.Clear();
                Console.WriteLine("4. Alterar a altura");
                Console.WriteLine("--------------------------------");
                Console.Write("\nDigite o número da pessoa: ");
                x = int.Parse(Console.ReadLine());
                if (Verifica(x))
                {
                    Console.Write("\nDigite a nova altura: ");
                    pe = vetPessoas[x - 1].Peso;
                    al = double.Parse(Console.ReadLine());
```

```
                        if (Verifica(al, pe))
                        {
                            vetPessoas[x - 1].Altura = al;
                            Console.WriteLine("\nAltura alterada - Pressione qualquer
tecla para sair");
                        }

                        else
                        {
                            Console.WriteLine("\nAltura informada inválida - Pressione
qualquer tecla para sair");
                        }
                    }
                    else
                    {
                        Console.WriteLine("\nNúmero de pessoa inválido - Pressione
qualquer tecla para sair");
                    }
                    Console.ReadKey();
                    break;

                case 5:
                    Console.Clear();
                    Console.WriteLine("5. Informar idade atual");
                    Console.WriteLine("-------------------------------");
                    Console.Write("\nDigite o número da pessoa: ");
                    x = int.Parse(Console.ReadLine());
                    if (Verifica(x))
```

```
                {
                    idade = DateTime.Now.Subtract(vetPessoas[x - 1].Nasc).TotalDays;
                    idade /= 365;
                    idade = Convert.ToInt64(idade);
                    Console.WriteLine("\nNúmero da pessoa: {0}\nIdade: {1}\nPressione qualquer tecla para sair", x, idade);
                }
                else
                {
                    Console.WriteLine("\nNúmero de pessoa inválido - Pressione qualquer tecla para sair");
                }
                Console.ReadKey();
                break;

            case 6:
                Console.Clear();
                Console.WriteLine("6. Informar IMC");
                Console.WriteLine("--------------------------------");
                Console.Write("\nDigite o número da pessoa: ");
                x = int.Parse(Console.ReadLine());
                if (Verifica(x))
                {
                    imc = vetPessoas[x - 1].Peso / (Math.Pow(vetPessoas[x - 1].Altura / 100, 2));
                    Console.WriteLine("\nIMC: {0}", imc);
                }
```

```
                        else
                        {
                            Console.WriteLine("\nNúmero de pessoa inválido - Pressione qualquer tecla para sair");
                        }
                        Console.ReadKey();
                        break;

                    case 7:
                        Console.Clear();
                        Console.WriteLine("7. Sair do programa");
                        Console.WriteLine("---------------------------------");
                        Console.WriteLine("\nAperte qualquer tecla para sair");
                        Console.ReadKey();
                        break;
                }


            } while (op != 7);
        }

        static void Menu()
        {
            Console.WriteLine("Menu de opções\n");
            Console.WriteLine("1. Incluir nova pessoa");
            Console.WriteLine("2. Alterar a data de nascimento de uma pessoa cadastrada");
```

```
            Console.WriteLine("3. Alterar o peso de uma pessoa cadastrada");
            Console.WriteLine("4. Alterar a altura de uma pessoa cadastrada");
            Console.WriteLine("5. Informar a idade atual de uma pessoa cadastrada");
            Console.WriteLine("6. Informar IMC de uma pessoa cadastrada");
            Console.WriteLine("7. Sair do programa");
        }

        static bool Verifica(int dia, int mes,int ano)
        {


            if ((dia <= 0) || (dia > 31) || (mes % 2 == 0 && dia > 30 && mes < 8) || (mes == 2 && dia > 29) || (mes == 2 && dia > 28 && ano % 4 != 0) || (mes <= 0) || (mes > 12) || (ano < 0))
            {
                return false;
            }

            return true;
        }

        static bool Verifica (double altura, double peso)
        {
            if (peso < 0 || altura < 0 || peso > 180 || altura > 250)
            {
                return false;
            }
```

```
    return true;
}

static bool Verifica(int x)
{
    if (!(x > Pessoa.contador || x < 1 || x > 100))
    {
        return true;
    }

    return false;
}

static bool Verifica(string data, out string [] datas)
{
    int[] c = new int[3];
 datas = data.Split('/');
   for (int i = 0; i < datas.Length; i++)
    {
       bool testa = Int32.TryParse(datas[0], out c[0]);
        if (!testa)
        {
            return false;
        }
    }
```

```
            return true;
        }
    }
}
```

2.3.1.2 Classe 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exercicios
{
    class Pessoa
    {
        private DateTime nasc = new DateTime(01/01/2017);
        private double peso, altura;
        public static int contador=1;

        public Pessoa (DateTime nasc, double altura, double peso)
        {
            if (!(nasc > DateTime.Now))
            {
                this.nasc = nasc;
            }
            this.peso = peso;
            this.altura = altura;
```

```
        }

        public DateTime Nasc
        {
            get
            {
                return this.nasc;
            }

            set
            {
                this.nasc = value;
            }
        }

        public double Peso
        {
            get
            {
                return this.peso;
            }

            set
            {
                this.peso = value;
            }
        }
```

```
        public double Altura
        {
            get
            {
                return this.altura;
            }

            set
            {
                this.altura = value;
            }
        }
    }
}
```

2.4 Ex3.4

2.4.1 Código fonte Ex3.4

2.4.1.1 Classe 1

```
// programa: Televisao
// programadores: Daniel Jorge, Robert Victor, Breno Vieira, Arthur Henrique, João Gontijo
// data: 24/09/2017
// descricao: exemplo de programa utilizando classes, para simular uma televisão e suas funções.
// entrada(s): digitar de 1 a 4 para selecionar opção do menu
// saida(s): informação ou ação solicitada pelo menu

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Televisao
{
    class TestaTelevisao
    {
        static int verificarStatus()
        {
            if(vetTv.status == "Ligada")
            {
                return 1;
            }
            else
            {
                Console.Write("Opção invalida, sua televisão está desligada.\n");
                return 0;
            }
        }
        static Televisao vetTv;
        static void Main(string[] args)
        {
            int repetidor = 1, opcao;
            vetTv = new Televisao();
            while (repetidor != 0)
```

```
                {
                    Console.Write("Status:{0}\nVolume:{1}\nCanal :{2}\nOpções do programa.\n1.Ligar ou desligar a televisão.\n2.Aumentar ou abaixar o volume.\n3.Subir ou abaixar o canal.\n4.Fechar.\nQual a opção desejada:", vetTv.status,vetTv.Volume,vetTv.Canal);
                    opcao = int.Parse(Console.ReadLine());
                    Console.Clear();
                    switch (opcao)
                    {
                        case 1:
                            vetTv.StatusTV();
                            Console.Clear();
                            break;
                        case 2:
                            if (verificarStatus() == 1)
                            {
                                vetTv.VolumeTv();
                            }
                            else
                            {
                                Console.Write("Digite alguma tecla para continuar.\n");
                                Console.ReadKey();
                                Console.Clear();
                            }
                            break;
                        case 3:
                            if (verificarStatus() == 1)
                            {
```

```
                        vetTv.CanalTv();
                    }
                    else
                    {
                        Console.Write("Digite alguma tecla para continuar.\n");
                        Console.ReadKey();
                        Console.Clear();
                    }
                    break;
                case 4:
                    repetidor = 0;
                    break;
                default:
                    Console.Write("Erro, opção invalida.\nDigite alguma tecla para
continuar.\n");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            }
        }
    }
```

2.4.1.2 Classe 2

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Televisao
{
    class Televisao
    {
        //Atributos e Gets/Sets
        private int canal;
        public int Canal
        {
            get
            {
                return canal;
            }
            set
            {
                canal = value;
            }
        }
        private int volume;
        public int Volume
        {
            get
            {
                return volume;
            }
```

```
        set
        {
            volume = value;
        }
    }
    public string status;
    //Metodos
    public Televisao()
    {
        status = "Desligado";
        canal = 1;
        volume = 0;
    }
    public Televisao(string status,int canal, int volume)
    {
        this.status = status;
        this.canal = canal;
        this.volume = volume;
    }
    public void StatusTV()
    {
        if (status == "Ligada")
        {
            status = "Desligada";
        }
        else
        {
```

```
                    status = "Ligada";
                }
            }
            public void VolumeTv()
            {
                char opcaoVolume;
                int repetidor = 1;
                while (repetidor != 0)
                {
                    Console.Write("Volume atual:{0}\nDigite (M) para voltar ao menu,(U) para aumentar e (D) para abaixar o volume:", volume);
                    opcaoVolume = char.Parse(Console.ReadLine());
                    if (opcaoVolume == 'd')
                    {
                        if (volume == 0)
                        {
                            Console.Write("Volume minimo atingido.");
                            Console.ReadKey();
                        }
                        else
                        {
                           volume--;
                        }
                    }
                    if (opcaoVolume == 'u')
                    {
                        if (volume == 100)
```

```
                    {
                        Console.Write("Volume maximo atingido.");
                        Console.ReadKey();
                    }
                    else
                    {
                        volume++;
                    }
                }
                if (opcaoVolume == 'm')
                {
                    repetidor = 0;
                }
                Console.Clear();
            }
        }
        public void CanalTv()
        {
            char opcaoCanal;
            int repetidor = 1;
            while (repetidor != 0)
            {
                Console.Write("Canal atual:{0}\nDigite (M) para voltar ao menu,(U) para aumentar e (D) para abaixar o volume:", canal);
                opcaoCanal = char.Parse(Console.ReadLine());
                if (opcaoCanal == 'd')
                {
```

```
                    if(canal == 1)
                    {
                        canal = 83;
                    }
                    else
                    {
                        canal--;
                    }
                }
                if (opcaoCanal == 'u')
                {
                    if(canal == 83)
                    {
                        canal = 1;
                    }
                    else
                    {
                        canal++;
                    }
                }
                if (opcaoCanal == 'm')
                {
                    repetidor = 0;
                }
                Console.Clear();
            }
        }
```

```
    }
}
```

2.5 Ex4.1

2.5.1 Código fonte Ex4.1

2.5.1.1 Classe 1

```
// programa: Ex41
// programadores: Daniel Jorge, Robert Victor, João Gontijo, Arthur Henrique, Breno Vieira
// data: 24/09/2017
// descricao: exemplo de programa utilizando classes, para simular uma empresa com classe vendedor e administrador.
// entrada(s): digitar para selecionar opção do menu
// saida(s): informação ou ação solicitada pelo menu
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Exercicios
{
    class Ex41
    {
        const int MAXFUNC = 100;
         static Vendedores[] vetVendedores = new Vendedores[MAXFUNC];
        static Administrativos[] vetAdministrativos = new Administrativos[MAXFUNC];
        static void Main(string[] args)
```

```
        {
            int op = 0, n;
            string receptor, nome, rg;
            double salfixo, x;
            char tipo;
            do
            {
                Console.Clear();
                Menu();
                receptor = Console.ReadLine();
                if (int.TryParse(receptor, out op))
                {
                    op = int.Parse(receptor);
                    switch (op)
                    {
                        case 1:
                            Console.Clear();
                            Console.WriteLine("\n1. Incluir funcionário");
                            Console.WriteLine("----------------------------------------");
                            Console.Write("\nDigite o nome do novo funcionário: ");
                            nome = Console.ReadLine();
                            Console.Write("\nDigite o rg do novo funcionário:");
                            rg = Console.ReadLine();
                            Console.Write("\nDigite o salário fixo do novo funcionário: ");
                            receptor = Console.ReadLine();
                            if (double.TryParse(receptor, out salfixo))
                            {
```

```
                    salfixo = double.Parse(receptor);
                    Console.Write("\nDigite o tipo do funcionário - V para vendedor, A para administrativo: ");
                    receptor = Console.ReadLine();
                    receptor = receptor.ToUpper();
                    if (char.TryParse(receptor, out tipo))
                    {
                        tipo = char.Parse(receptor);
                        switch (tipo)
                        {
                            case 'V':
                                vetVendedores[Vendedores.contador - 1] = new Vendedores(nome, rg, salfixo);
                                Console.WriteLine("\nVendedor incluído \nNº do vendedor: {0} Pressione qualquer tecla para sair", Vendedores.contador);
                                Vendedores.contador++;
                                break;

                            case 'A':
                                vetAdministrativos[Administrativos.contador - 1] = new Administrativos(nome, rg, salfixo);
                                Console.WriteLine("\nAdministrador inclúido \nNº do administrador: {0} Pressione qualquer tecla para sair", Administrativos.contador);
                                Administrativos.contador++;
                                break;

                            default:
                                Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
```

```
                                    break;
                                }
                            }

                            else
                            {
                                Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
                            }
                        }

                        else
                        {
                            Console.WriteLine("\nValor digitado inválido - Pressione qualquer tecla para sair");
                        }
                        Console.ReadKey();
                        break;

                    case 2:
                        Console.Clear();
                        Console.WriteLine("\n2. Alterar dados de um funcionário");
                        Console.WriteLine("----------------------------------------");
                        Console.Write("\nDigite o tipo do funcionário - V para vendedor, A para administrador: ");
                        receptor = Console.ReadLine();
                        receptor = receptor.ToUpper();
                        if (char.TryParse(receptor, out tipo))
```

```
                        {
                            tipo = char.Parse(receptor);
                            switch (tipo)
                            {
                                case 'V':
                                    Console.Write("\nDigite o número do vendedor: ");
                                    receptor = Console.ReadLine();
                                    if (int.TryParse(receptor, out n))
                                    {
                                        n = int.Parse(receptor);
                                        if (n < Vendedores.contador)
                                        {
                                            Console.Write("\nDigite o nome do funcionário: ");
                                            vetVendedores[n - 1].Nome = Console.ReadLine();
                                            Console.Write("\nDigite o rg do funcionário:");
                                            vetVendedores[n - 1].Rg = Console.ReadLine();
                                            Console.Write("\nDigite o salário fixo do funcionário: ");
                                            receptor = Console.ReadLine();
                                            if (double.TryParse(receptor, out salfixo))
                                            {
                                                vetVendedores[n - 1].SalFixo = double.Parse(receptor);
                                                Console.WriteLine("\nDados alterados - Pressione qualquer tecla para sair");
                                            }
```

```
                                    else
                                    {
                                        Console.WriteLine("\nValor digitado inválido - Pressione qualquer tecla para sair");
                                    }
                                }

                                else
                                {
                                    Console.WriteLine("\nValor digitado inválido - Pressione qualquer tecla para sair");
                                }

                            }

                            else
                            {
                                Console.WriteLine("\nNúmero inválido - Pressione qualquer tecla para sair");
                            }
                            break;

                        case 'A':
                            Console.Write("\nDigite o número do administrador: ");
                            receptor = Console.ReadLine();
                            if (int.TryParse(receptor, out n))
                            {
```

```
                                    n = int.Parse(receptor);
                                    if (n < Administrativos.contador)
                                    {
                                        Console.Write("\nDigite o nome do funcionário: ");
                                        vetAdministrativos[n - 1].Nome = Console.ReadLine();
                                        Console.Write("\nDigite o rg do funcionário:");
                                        vetAdministrativos[n - 1].Rg = Console.ReadLine();
                                        Console.Write("\nDigite o salário fixo do funcionário: ");
                                        receptor = Console.ReadLine();
                                        if (double.TryParse(receptor, out salfixo))
                                        {
                                            vetAdministrativos[n - 1].SalFixo = double.Parse(receptor);
                                            Console.WriteLine("\nDados alterados - Pressione qualquer tecla para sair");
                                        }

                                        else
                                        {
                                            Console.WriteLine("\nValor digitado inválido - Pressione qualquer tecla para sair");
                                        }
                                    }

                                    else
```

```
                                    {
                                        Console.WriteLine("\nValor digitado inválido - Pressione qualquer tecla para sair");
                                    }
                                }

                                else
                                {
                                    Console.WriteLine("\nNúmero inválido- Pressione qualquer tecla para sair");
                                }
                                break;

                            default:
                                Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
                                break;
                        }
                    }

                    else
                    {
                        Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
                    }
                    Console.ReadKey();
                    break;
```

```
                    case 3:
                        Console.Clear();
                        Console.WriteLine("\n3. Adicionar vendas para um Vendedor");
                        Console.WriteLine("----------------------------------------");
                        Console.Write("\nDigite o número do vendedor: ");
                        receptor = Console.ReadLine();
                        if (int.TryParse(receptor, out n))
                        {
                            n = int.Parse(receptor);
                            if (n < Vendedores.contador)
                            {
                                Console.Write("\nDigite o total de vendas em R$: ");
                                receptor = Console.ReadLine();
                                if (double.TryParse(receptor, out x))
                                {
                                    x = double.Parse(receptor);
                                    if (x > 0)
                                    {
                                        vetVendedores[n - 1].Acumula(x);
                                        Console.WriteLine("\nVenda registrada - Pressione qualquer tecla para sair");
                                    }

                                    else
                                    {
                                        Console.WriteLine("\nO valor da venda deve ser superior a 0 - Pressione qualquer tecla para sair");
```

```
                    }


                }


                else
                {
                    Console.WriteLine("\nValor inválido - Pressione qualquer tecla para sair");
                }
            }


            else


            {
                Console.WriteLine("\nNúmero de vendedor inválido - Pressione qualquer tecla para sair");
            }
        }
        else
        {
            Console.WriteLine("\nNúmero de vendedor inválido - Pressione qualquer tecla para sair");
        }
        Console.ReadKey();
        break;


    case 4:
        Console.Clear();
```

```
                    Console.WriteLine("\n4. Adicionar horas extras para um Administrador");
                    Console.WriteLine("----------------------------------------");
                    Console.Write("\nDigite o número do administrador: ");
                    receptor = Console.ReadLine();
                    if (int.TryParse(receptor, out n))
                    {
                        n = int.Parse(receptor);

                        if (n < Administrativos.contador)
                        {
                            Console.Write("\nDigite quantas horas extras deseja adicionar: ");
                            receptor = Console.ReadLine();
                            if (int.TryParse(receptor, out n))
                            {
                                n = int.Parse(receptor);
                                vetAdministrativos[n - 1].Acumula(n);
                                Console.WriteLine("\nHora extra adicionada - Pressione qualquer tecla para sair");
                            }

                            else
                            {
                                Console.WriteLine("Número de horas inválidas - Pressione qualquer tecla para sair");
                            }
                        }
```

```
                    else
                    {
                        Console.WriteLine("\nNúmero do administrador inválido - Pressione qualquer tecla para sair");
                    }
                }

                else
                {
                    Console.WriteLine("\nNúmero do administrador inválido - Pressione qualquer tecla para sair");
                }
                Console.ReadKey();
                break;

            case 5:
                Console.Clear();
                Console.WriteLine("\n5. Excluir um funcionário");
                Console.WriteLine("----------------------------------------");
                Console.Write("\nDigite o tipo do funcionário - V para vendedor, A para administrativo: ");
                receptor = Console.ReadLine();
                receptor = receptor.ToUpper();
                if (char.TryParse(receptor, out tipo))
                {
                    tipo = char.Parse(receptor);
                    switch (tipo)
```

```
                {
                    case 'A':
                        Console.Write("\nDigite o número de administrativo: ");
                        receptor = Console.ReadLine();
                        if (int.TryParse(receptor, out n))
                        {
                            n = int.Parse(receptor);

                            if (n < Administrativos.contador)
                            {
                                vetAdministrativos[n - 1].Nome = null;
                                vetAdministrativos[n - 1].Rg = null;
                                vetAdministrativos[n - 1].SalFixo = 0;
                                Console.WriteLine("\nFuncionário excluido - Pressione qualquer tecla para sair");
                            }

                            else
                            {
                                Console.WriteLine("\nNúmero administrativo inválido - Pressione qualquer tecla para sair");
                            }
                        }

                        else
                        {
```

```
                    Console.WriteLine("\nNúmero administrativo
inválido - Pressione qualquer tecla para sair");
                }



                break;


            case 'V':
                Console.Write("\nDigite o número de vendedor: ");
                receptor = Console.ReadLine();
                if (int.TryParse(receptor, out n))
                {
                    n = int.Parse(receptor);
                    if (n < Vendedores.contador)
                    {
                        vetVendedores[n - 1].Nome = null;
                        vetVendedores[n - 1].Rg = null;
                        vetVendedores[n - 1].SalFixo = 0;
                        Console.WriteLine("\nFuncionario excluido -
Pressione qualquer tecla para sair");
                    }


                    else
                    {
                        Console.WriteLine("\nNúmero de vendedor
inválido - Pressione qualquer tecla para sair");
                    }
```

```
                                    }


                                    else
                                    {
                                        Console.WriteLine("\nNúmero de vendedor inválido - Pressione qualquer tecla para sair");
                                    }
                                    break;


                                default:
                                    Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
                                    break;
                                }
                            }


                            else
                            {
                                Console.WriteLine("\nTipo inválido - Pressione qualquer tecla para sair");
                            }
                            Console.ReadKey();
                            break;


                        case 6:
                            Console.Clear();
                            Console.WriteLine("\n6. Imprimir todas as informações de um funcionário");
```

```
                    Console.WriteLine("----------------------------------------");
                    Console.Write("\nDigite o tipo do funcionário - V para vendedor, A para administrativo: ");
                    receptor = Console.ReadLine();
                    receptor = receptor.ToUpper();
                    if (char.TryParse(receptor, out tipo))
                    {
                        tipo = char.Parse(receptor);
                        switch (tipo)
                        {
                            case 'A':
                                Console.Write("\nDigite o número do funcionário administrativo: ");
                                receptor = Console.ReadLine();
                                if (int.TryParse(receptor, out n))
                                {
                                    n = int.Parse(receptor);

                                    if (n < Administrativos.contador)
                                    {
                                        Console.WriteLine("\nNome: {0}", vetAdministrativos[n-1].Nome);
                                        Console.WriteLine("\nRG: {0}", vetAdministrativos[n - 1].Rg);
                                        Console.WriteLine("\nSalário fixo: {0}\n", vetAdministrativos[n - 1].SalFixo);
                                        vetAdministrativos[n - 1].ExibeSal();
                                    }
```

```
                                    else
                                    {
                                        Console.WriteLine("\nNúmero administrativo
inválido - Pressione qualquer tecla para sair");
                                    }

                                }

                                else
                                {
                                    Console.WriteLine("\nNúmero administrativo
inválido - Pressione qualquer tecla para sair");
                                }
                                    break;

                            case 'V':
                                Console.Write("\nDigite o número de vendedor: ");
                                receptor = Console.ReadLine();
                                if (int.TryParse(receptor, out n))
                                {
                                    n = int.Parse(receptor);

                                    if (n < Vendedores.contador)
                                    {
                                        Console.WriteLine("\nNome: {0}",
vetVendedores[n - 1].Nome);
                                        Console.WriteLine("\nRG: {0}", vetVendedores[n
- 1].Rg);
```

```
                    Console.WriteLine("\nSalário fixo: {0}\n", vetVendedores[n-1].SalFixo);
                    vetVendedores[n - 1].ExibeSal();
                }

                else
                {
                    Console.WriteLine("\nNúmero de vendedor inválido - Pressione qualquer tecla para sair");
                }
            }

            else
            {
                Console.WriteLine("\nNúmero de vendedor inválido - Pressione qualquer tecla para sair");
            }
            break;

                                default:
            Console.WriteLine("Tipo inválido - Pressione qualquer tecla para sair");
            break;

        }
    }

    else
```

```
                    {
                        Console.WriteLine("\nTipo inválido - Pressione qualquer
tecla para sair");
                    }
                    Console.ReadKey();
                    break;

                case 7:
                    Console.Clear();
                    Console.WriteLine("\n7. Imprimir relação de todos os
funcionários");
                    Console.WriteLine("----------------------------------------");
                    Console.WriteLine("\nVendedores");
                    for (int i = 0; i < Vendedores.contador - 1; i++)
                    {
                        Console.WriteLine("\nNome:                    {0}",
vetVendedores[i].Nome);
                        Console.WriteLine("\nRG: {0}", vetVendedores[i].Rg);
                        Console.WriteLine("\nSalário fixo: {0}\n",
vetVendedores[i].SalFixo);
                        vetVendedores[i].ExibeSal();
                    }
                    Console.WriteLine("----------------------------------------");
                    Console.WriteLine("\nAdministradores");
                    for (int i = 0; i < Administrativos.contador - 1; i++)
                    {
                        Console.WriteLine("\nNome:                    {0}",
vetAdministrativos[i].Nome);
                        Console.WriteLine("\nRG: {0}", vetAdministrativos[i].Rg);
```

```
                    Console.WriteLine("\nSalário fixo: {0}\n",
vetAdministrativos[i].SalFixo);
                    vetAdministrativos[i].ExibeSal();
                }
                Console.ReadKey();
                break;

            case 8:
                Console.Clear();
                Console.WriteLine("\n8. Sair do programa");
                Console.WriteLine("----------------------------------------");
                                Console.WriteLine("Pressione
qualquer tecla para sair");
                Console.ReadKey();
                break;

            default:
                Console.WriteLine("Opção inválida - Pressione qualquer
tecla para sair");
                Console.ReadKey();
                break;
            }
        }

        else
        {
            Console.WriteLine("\nOpção em formato inválido - Pressione
qualquer tecla para sair");
```

```
                    }
                }


                while (op != 8);
            }


            static void Menu ()
            {
                Console.WriteLine("\nMENU DE OPÇÕES");
                Console.WriteLine("\n1. Incluir funcionário");
                Console.WriteLine("\n2. Alterar dados de um funcionário");
                Console.WriteLine("\n3. Adicionar vendas para um Vendedor");
                Console.WriteLine("\n4. Adicionar hora extra para um Administrador");
                Console.WriteLine("\n5. Excluir funcionário");
                Console.WriteLine("\n6. Imprimir todas as informações de um funcionário");
                Console.WriteLine("\n7. Imprimir a relação de todos os funcionários do sistema");
                Console.WriteLine("\n8. Sair do programa");
            }


        }
    }
```

2.5.1.2 Classe 2

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
```

```
using System.Threading.Tasks;

namespace Exercicios
{
    class Funcionarios
    {
        protected string nome, rg;
        protected double salfixo;

        public string Nome
        {
            get
            {
                return this.nome;
            }

            set
            {
                nome = value;
            }
        }

        public string Rg
        {
            get
            {
                return this.rg;
```

```
            }

            set
            {
                this.rg = value;
            }
        }

        public double SalFixo
        {
            get
            {
                return this.salfixo;
            }

            set
            {
                this.salfixo = value;
            }
        }

    }
}
```

2.5.1.3 Classe 3

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Exercicios
{
    class Administrativos: Funcionarios
    {
        double hExtra, comissao;
        public static int contador = 1;

        public double Comissao
        {
            get
            {
                return comissao;
            }

            set
            {
                comissao = value;
            }
        }

        public double Hxtra
        {
            get
            {
```

```
            return hExtra;
        }

        set
        {
            hExtra = value;
        }
    }

    public Administrativos(string nome, string rg, double salfixo)
    {
        this.nome = nome;
        this.rg = rg;
        this.salfixo = salfixo;
    }

    public void Acumula(double n)
    {
        Comissao += n;
    }

    public double ExtraSS()
    {
        comissao = salfixo / 100 * hExtra;
        return comissao;
    }
```

```
        public void ExibeSal()
        {
            Console.WriteLine("Salário a receber: {0}", SalFixo + ExtraSS());
            hExtra = 0;
            comissao = 0;
        }
    }
}
```

2.5.1.4 Classe 4

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exercicios
{
    class Vendedores: Funcionarios
    {
        double totalVendas, comissao;
        public static int contador=1;

        public Vendedores(string nome, string rg, double salfixo)
        {
            this.nome = nome;
            this.rg = rg;
            this.salfixo = salfixo;
```

```
        }

        public void Acumula(double n)
        {
            TotalVendas += n;
        }

        public double ExtraSS()
        {
            comissao = totalVendas * 5 / 100;
            return comissao;
        }

        public void ExibeSal()
        {
            Console.WriteLine("Salário a receber: {0}", SalFixo + ExtraSS());
            totalVendas = 0;
            comissao = 0;
        }

        public double TotalVendas
        {
            get
            {
                return totalVendas;
            }
```

```
            set
            {
                totalVendas = value;
            }
        }
    }
}
```

2.6 Ex4.2

2.6.1 Código fonte Ex4.2

2.6.1.1 Classe 1

```
// programa: Imposto
// programadores: Daniel Jorge, Robert Victor, João Gontijo, Breno Vieira, Arthur Henrique
// data: 24/09/2017
// descricao: exemplo de programa utilizando classes, para calcular imposto de pessoa fisica e juridica
// entrada(s): digitar para selecionar opção do menu
// saida(s): informação ou ação solicitada pelo menu
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Imposto
{
    class TestarContribuinte
    {
```

```
        static int verificarCadastro(string registro)
        {
            for (int i = 0; i < cont; i++)
            {
                if (vetContribuinte[i].Comprarar(registro) == 1)
                {
                    return 1;
                }
            }
            return 0;
        }
        static void cadastrar()
        {
            double dinheiro;
            string nome, endereço, registro;
            int tipo;
            Console.Write("O contribuinte é:\n1 - Pessoa Juridica:\n2 - Pessoa Fisica\nQual a opção:");
            tipo = int.Parse(Console.ReadLine());
            Console.Write("Informe o CNPJ ou CPF do contribuinte:");
            registro = Console.ReadLine();
            if (verificarCadastro(registro) == 0)
            {
                if (tipo == 1)
                {
                    Console.Write("Qual o nome do constribuinte:");
                    nome = Console.ReadLine();
```

```
                    Console.Write("Qual o endereço:");
                    endereço = Console.ReadLine();
                    Console.Write("Qual o faturamento:");
                    dinheiro = double.Parse(Console.ReadLine());
                    PJuridica aux = new PJuridica(nome, endereço, registro,
dinheiro);
                    vetContribuinte[cont] = aux;
                    dadosJuridica = dadosJuridica + vetContribuinte[cont].Imprimir();
                    cont++;
                }
                else if (tipo == 2)
                {
                    Console.Write("Qual o nome do constribuinte:");
                    nome = Console.ReadLine();
                    Console.Write("Qual o endereço:");
                    endereço = Console.ReadLine();
                    Console.Write("Qual o salario:");
                    dinheiro = double.Parse(Console.ReadLine());
                    PFisica aux = new PFisica(nome, endereço, registro, dinheiro);
                    vetContribuinte[cont] = aux;
                    dadosFisica = dadosFisica +  vetContribuinte[cont].Imprimir();
                    cont++;
                }
            }
            else if(verificarCadastro(registro) == 1)
            {
                Console.Write("Erro, contribuinte já cadastrado.\n");
```

```
            }
        }
        static void excluirContribuinte()
        {
            string registro;
            int tipo;
            Console.Write("O contribuinte é:\n1 - Pessoa Juridica:\n2 - Pessoa Fisica\nQual a opção:");
            tipo = int.Parse(Console.ReadLine());
            Console.Write("Informe o CNPJ ou CPF do contribuinte que deseja escluir:");
            registro = Console.ReadLine();
            if (verificarCadastro(registro) == 1)
            {
                for (int i = 0; i < cont; i++)
                {
                    if (vetContribuinte[i].Comprarar(registro) == 1)
                    {
                        vetContribuinte[i].Excluir();
                    }
                }
            }
            else if (verificarCadastro(registro) == 0)
            {
                Console.Write("Erro, contribuinte não cadastrado.\n");
            }
        }
        static void imprimirContribuinte()
```

```
        {
            string registro;
            int tipo;
            Console.Write("O contribuinte é:\n1 - Pessoa Juridica:\n2 - Pessoa Fisica\nQual a opção:");
            tipo = int.Parse(Console.ReadLine());
            Console.Write("Informe o CNPJ ou CPF do contribuinte que deseja escluir:");
            registro = Console.ReadLine();
            if (verificarCadastro(registro) == 1)
            {
                for (int i = 0; i < cont; i++)
                {
                    if (vetContribuinte[i].Comprarar(registro) == 1)
                    {
                        Console.Write(vetContribuinte[i].Imprimir());
                    }
                }
            }
            else if (verificarCadastro(registro) == 0)
            {
                Console.Write("Erro, contribuinte não cadastrado.\n");
            }
        }
        static void imposto()
        {
            string registro;
            int tipo;
```

```
                Console.Write("O contribuinte é:\n1 - Pessoa Juridica:\n2 - Pessoa Fisica\nQual a opção:");
                tipo = int.Parse(Console.ReadLine());
                Console.Write("Informe o CNPJ ou CPF do contribuinte que deseja escluir:");
                registro = Console.ReadLine();
                if (verificarCadastro(registro) == 1)
                {
                   if (tipo == 1)
                   {
                      for (int i = 0; i < cont; i++)
                      {
                         if (vetContribuinte[i].Comprarar(registro) == 1)
                         {
                            Console.Write("Imposto a ser pago é de:R$ {0}",vetContribuinte[i].CalcImposto());
                            i = cont;
                         }
                      }
                   }
                }
                static void imprimirFisicos()
                {
                   Console.Write(dadosFisica);
                }
                static void imprimirJuridica()
                {
```

```
            Console.Write(dadosJuridica);
        }
        static string dadosFisica ="";
        static string dadosJuridica = "";
        static int cont = 0;
        static Contribuinte[] vetContribuinte;
        static void Main(string[] args)
        {
            int maxContribuintes = 100;
            vetContribuinte = new Contribuinte[maxContribuintes];
            int repetidor = 1, opcao;
            while (repetidor != 0)
            {
                Console.Write("Menu:\n1.Incluir um contribuinte.\n2.Excluir um contribunte.\n3.Exibir os dados de um contribuinte:CPF/CNPJ, nome, endereço e salario/faturamenteo.\n4.Calcular e exibir o imposto a ser pago por um contribinte.\n5.Imprimir uma realção dos contribuintes Pessoa Fisica cadastrados, mostrando os dados:CPF, nome e endereço.\n6Imprimir uma realção dos contribuintes Pessoa Juridica cadastrados, mostrando os dados:CNPJ, nome e endereço.\n7.Sair do programa.\nQual a opção desejada:");
                opcao = int.Parse(Console.ReadLine());
                Console.Clear();
                switch (opcao)
                {
                    case 1:
                        cadastrar();
                        Console.Write("Digite alguma tecla para continuar.");
                        Console.ReadKey();
                        Console.Clear();
```

```
                    break;
                case 2:
                    excluirContribuinte();
                    Console.Write("\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                case 3:
                    imprimirContribuinte();
                    Console.Write("\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                case 4:
                    imposto();
                    Console.Write("\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                case 5:
                    imprimirFisicos();
                    Console.Write("\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                case 6:
                    imprimirJuridica();
```

```
                    Console.Write("\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    Console.Clear();
                    break;
                case 7:
                    repetidor = 0;
                    break;
                default:
                    Console.Write("Erro, opção invalida.\nDigite alguma tecla para continuar.");
                    Console.ReadKey();
                    break;
            }
        }
    }
}
```

2.6.1.2 Classe 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Imposto
{
    class Contribuinte
```

```
{
    private string nome;
    public string Nome
    {
        get
        {
            return nome;
        }
        set
        {
            nome = value;
        }
    }
    private string enderco;
    public string Enderco
    {
        get
        {
            return enderco;
        }
        set
        {
            enderco = value;
        }
    }

    public virtual double CalcImposto()
```

```
        {
            return 0;
        }
        public virtual void Excluir()
        {

        }
        public virtual string Imprimir()
        {
            string dados = "";
            return dados;
        }
        public virtual int Comprarar(string registro)
        {
            return 0;
        }
    }
}
```

2.6.1.3 Classe 3

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Imposto
{
```

```
class PFisica : Contribuinte
{
    private string cpf;
    public string Cpf
    {
        get
        {
            return cpf;
        }
        set
        {
            cpf = value;
        }
    }
    private double salario;
    public double Salario
    {
        get
        {
            return salario;
        }
        set
        {
            salario = value;
        }
    }
    public PFisica()
```

```
        {
            Nome = string.Empty;
            Enderco = string.Empty;
            cpf = string.Empty;
            salario = 0;
        }
        public PFisica(string nome, string endereço, string cps, double salario)
        {
            Nome = nome;
            Enderco = endereço;
            this.cpf = cps;
            this.salario = salario;
        }
        public override double CalcImposto()
        {
            if (salario >= 0 && salario <= 1400)
            {
                return 0;
            }
            else if (salario > 1400 && salario <= 2100)
            {
                return ((salario * 0.1) - 100);
            }
            else if (salario > 2100 && salario <= 2800)
            {
                return ((salario * 0.15) - 100);
            }
```

```
                else if (salario > 2800 && salario <= 3600)
                {
                    return ((salario * 0.25) - 100);
                }
                else
                {
                    return ((salario * 0.30) - 100);
                }
            }
            public override void Excluir()
            {
                Nome = null;
                Enderco = null;
                cpf = null;
                salario = 0;
            }
            public override string Imprimir()
            {
                string dados = "Nome:" + Nome + "\nEndereço:" + Enderco + "\nCNPJ:" + cpf + "\nFaturamento:" + salario + "\n";
                return dados;
            }
            public override int Comprarar(string registro)
            {
                if (registro == cpf)
                {
                    return 1;
```

```
            }
            else
            {
                return 0;
            }
        }
    }
}
```

2.6.1.4 Classe 4

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Imposto
{
    class PJuridica : Contribuinte
    {
        private string cnpj;
        public string Cnpj
        {
            get
            {
                return cnpj;
            }
            set
```

```
            {
                cnpj = value;
            }
        }
        private double faturamento;
        public double Faturamento
        {
            get
            {
                return faturamento;
            }
            set
            {
                faturamento = value;
            }
        }
        public PJuridica()
        {
            Nome = string.Empty;
            Enderco = string.Empty;
            cnpj = string.Empty;
            faturamento = 0;
        }
        public PJuridica(string nome, string endereço, string cnpj, double faturamento)
        {
            Nome = nome;
```

```
            Enderco = endereço;
            this.cnpj = cnpj;
            this.faturamento = faturamento;
        }
        public override double CalcImposto()
        {
            return (faturamento * 0.1);
        }
        public override void Excluir()
        {
            Nome = null;
            Enderco = null;
            cnpj = null;
            faturamento = 0;
        }
        public override string Imprimir()
        {
            string dados = "Nome:" + Nome + "\nEndereço:" + Enderco + "\nCNPJ:" + cnpj + "\nFaturamento:" + faturamento + "\n";
            return dados;
        }
        public override int Comprarar(string registro)
        {
            if(registro == cnpj)
            {
                return 1;
            }
```

```
                else
                {
                    return 0;
                }
            }
        }
    }
```

2.7 Ex4.3

2.7.1 Código fonte Ex4.3

2.7.1.1 Classe 1

```
    // programa: Exer4.3
    // programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
    // data: 24/09/2017
    // Descrição: programa principal que utiliza as classes imovel ,novo e velho
    // Entrada(s):Digitar 1 para imovel novo e 2 para imovel velho , e depois entre com o preço
    // Saida:retorna preço final
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;


    namespace Exer4._3
    {
        class Program
        {
```

```
        static void Main(string[] args)
        {
            Console.WriteLine("Escolha se seu imovel é: \n 1- Novo  \n 2- Velho");


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


            if (imovel == 1)
            {
                Novo im1 = new Novo();
                Console.WriteLine("Escreva o Preço inicial do imovel:");
                im1.Preco = double.Parse(Console.ReadLine());
                Console.WriteLine("Preço Final:" + im1.PrecoFinal());
                Console.ReadKey();
            }
            if (imovel == 2)
            {
                Velho im2 = new Velho();
                Console.WriteLine("Escreva o Preço inicial do imovel:");
                im2.Preco = double.Parse(Console.ReadLine());
                Console.WriteLine("Preço Final:" + im2.PrecoFinal());
                Console.ReadKey();
            }
            else
            {
                Console.WriteLine("Opçao Invalida!");
                Console.ReadKey();
            }
```

```
            }
        }
    }
```

2.7.1.2 Classe 2

```
    // programa: Imovel
    // programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
    // data: 24/09/2017
    // Descrição: classe pai das classes velho e novo , com a funçao de salvar o preco do imovel.
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;


    namespace Exer4._3
    {
        class Imovel
        {
            private double preco;// definiçao de atributo
            public double Preco// get e set para conseguir alterar o preço depois
            {
                get
                {
                    return preco;
                }
                set
```

```
            {
                preco = value;
            }
        }

        public virtual double PrecoFinal()// funçao virtual para ser sobrescrita nas classes filhas
        {
            return 0;
        }

        protected int RandomNumber(int min, int max)// funçao que gera numeros aleatorios
        {
            Random random = new Random();
            int resultado = random.Next(min, max);
            return resultado;
        }
    }
}
```

2.7.1.3 Classe 3

```
// programa: Novo
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 24/09/2017
// Descrição: classe filha que pega o valor do preço e adiciona um numero aleatorio a ele.
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Exer4._3
{
    class Novo : Imovel
    {
        public Novo()// construtor padrao
        {
            Preco = 0;
        }


        public override double PrecoFinal()// funçao que retorna o preço do imovel novo
        {
            double resultado = Preco + RandomNumber(50000, 400000);// adiciona numero aleatorio de 50000 a 400000
            return resultado;
        }
    }
}
```

2.7.1.4 Classe 4

```
// programa: Velho
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 24/09/2017
```

```
        // Descrição: classe filha que pega o valor do preço e subtrai um numero aleatorio a ele
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;


        namespace Exer4._3
        {
            class Velho : Imovel
            {
                public Velho()// contrutor padrao
                {
                    Preco = 0;
                }


                public override double PrecoFinal()//funçao que retorna o preço do imovel velho
                {
                    double resultado = Preco - RandomNumber(50000, 400000);// numero entre 50000 e 400000


                    if (Preco >= 0)
                    {
                        return resultado;
                    }
                    else
```

```
                {
                    return 0;
                }
            }
        }
    }
```

2.8 Ex4.4

2.8.1 Código fonte Ex4.4

2.8.1.1 Classe 1

```
    //
    // nome do programa: Ex44
    //
    // programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
    // data: 24/09/2017
    // entrada(s): Tipo do ingresso, localizacao do camarote
    // saida(s): Imprime o tipo do ingresso, ou a localizacao do camarote do tipo VIP e o preço do ingresso
    // para executar  e testar digite:
    // Ex44.exe
    // descricao: Pergunta ao usuario se ele quer comprar um ingresso do tipo Normal ou VIP,
    // caso escolha normal, imprimo o tipo do ingresso e o valor.
    // caso escolha Vip, o valor recebe um acrécimo de 20% do valor, é solicitado qual camarote o cliente quer, superior ou inferior,
    // se o cliente quiser o superior tem mais um preco adicional no ingresso, que é de 10%
    // se o cliente escolher o inferior, o acrécimo é somente o default do VIP.
    //
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ing;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            int op;
            double valorIngresso = 50;
            Console.WriteLine("****Menu de Ingresso******");
            Console.WriteLine("1 - Ingresso Normal");
            Console.WriteLine("2 - Ingresso Vip");
            Console.WriteLine("Digite o ingresso desejado:");
            op = int.Parse(Console.ReadLine());

            switch (op)
            {
                case 1:
                    Normal ingresso = new Normal(valorIngresso);
```

```
                    Console.WriteLine("Tipo de ingresso: {0}", ingresso.impremeIngressoNormal());
                    Console.WriteLine("Valor do ingresso: R${0}", ingresso.Valor);
                    Console.ReadKey();
                    break;
                case 2:
                    menuVip();
                    op = int.Parse(Console.ReadLine());
                    switch (op)
                    {
                        case 1:
                            CamaroteInferior camInf = new CamaroteInferior(valorIngresso);
                            Console.WriteLine("Localizacao: {0}", camInf.impreLocalizacao());
                            Console.WriteLine("Valor do ingresso: R${0}", camInf.Valor);
                            Console.ReadKey();
                            break;
                        case 2:
                            CamaroteSuperior camSup = new CamaroteSuperior(valorIngresso);
                            Console.WriteLine("Localizacao: {0}", camSup.impreLocalizacao());
                            Console.WriteLine("Valor do ingresso: R${0}", camSup.Valor);
                            Console.ReadKey();
                            break;
                        default:
                            Console.WriteLine("Opcao Inválida");
```

```
                    Console.ReadKey();
                    break;
                }
                break;
            default:
                Console.WriteLine("Opcao Inválida");
                Console.ReadKey();
                break;
        }
    }
    public static void menuVip()
    {
        Console.WriteLine("*****Menu Ingresso VIP*****");
        Console.WriteLine("1 - Camarote Inferior");
        Console.WriteLine("1 - Camarote Superior");
        Console.WriteLine("Digite o camarote desejado:");
    }
  }
}
```

2.8.1.2 Classe 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace ing
```

```
{
    class Ingresso
    {
        double valor;

        public Ingresso() { }

        public double Valor
        {
            get
            {
                return valor;
            }
            set
            {
                valor = value;
            }
        }
    }
}
```

2.8.1.3 Classe 3

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ing
{
    class Normal : Ingresso
    {
        public string impremeIngressoNormal()
        {
            return "Ingresso Normal";
        }

        public Normal(double valor)
        {
            Valor = valor;
        }
    }
}
```

2.8.1.4 Classe 4

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ing
{
    class Vip : Ingresso
    {
        double valorAdicional;
```

```
        public double impremeValorComAdicional()
        {
            return Valor + valorAdicional;
        }
        public void adicionaValorAdicional()
        {
            valorAdicional = 0.2 * Valor;
        }
        public void alteraValor()
        {
            Valor = Valor + valorAdicional;
        }
    }
}
```

2.8.1.5 Classe 5

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ing
{
    class CamaroteInferior : Vip
    {
        string localizacao = "Camarote Inferior";
```

```
        public CamaroteInferior(double valor)
        {
            Valor = valor;
            adicionaValorAdicional();
            alteraValor();
        }
        public string impreLocalizacao()
        {
            return localizacao;
        }
    }
}
```

2.8.1.6 Classe 6

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ing
{
    class CamaroteSuperior : Vip
    {
        string localizacao = "Camarote Superior";
        double valorAdicionalCamaroteSup;
```

```
        public CamaroteSuperior(double valor)
        {
            Valor = valor;
            adicionaValorAdicional();
            alteraValor();
            adicionaValorAdicionalCamaroteSup();
            alteraValorCamroteSup();
        }

        public string impreLocalizacao()
        {
            return localizacao;
        }

        public void adicionaValorAdicionalCamaroteSup()
        {
            valorAdicionalCamaroteSup = 0.1 * Valor;
        }
        public void alteraValorCamroteSup()
        {
            Valor = Valor + valorAdicionalCamaroteSup;
        }
    }
}
```

2.9 Ex5.5

2.9.1 Código fonte Ex5.5

2.9.1.1 Programa 1

```
// programa: Copiar binário

// programadores: Daniel Jorge, Robert Victor, João Gontijo, Arthur Henrique, Breno Vieira

// data: 24/09/2017

// descricao: Programa que faz a copia de um arquivo qualquer (texto ou binário).

// entrada(s): linha de comando ambos arquivos, para copiar

// saida(s): arquivo copiado


using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

namespace copia_bin_cada_byte

{

    class copia_bin_cada_byte

    {

        static void Main(string[] args)

        {

            // args[0]: nome do arquivo de origem (ja existente): arq_origem

            // args[1]: nome do arquivo de destino(a ser criado): arq_destino

            FileInfo fi = new FileInfo(args[0]); // cria objeto Fileinfo para obter atributos do arquivo

            Console.WriteLine("Arquivo de origem: " + fi.Name); // exibe alguns atributos do arquivo
```

```
                Console.WriteLine("Diretorio: " + fi.Directory);
                Console.WriteLine("Tamanho do arquivo: " + fi.Length);
                Console.WriteLine("Arquivo de destino: " + args[1]);
                Stream entrada = File.Open(args[0], FileMode.Open); // abrir arquivo
                        Stream saida = File.Open(args[1], FileMode.Create); // criar arquivo
                        BinaryReader f1 = new BinaryReader(entrada);
                        BinaryWriter f2 = new BinaryWriter(saida);
                        byte[ ] buf = new byte[1]; // buffer para armazenar byte(s) lido(s)
                        int lidos; // byte(s) lido(s) ou escritos em cada etapa
                        while (true)
                        {
                                lidos = f1.Read(buf, 0, 1); // ler um byte de cada vez
                                if (lidos < 0) break; // erro no arquivo
                                if (lidos == 0) break; // fim de arquivo
                                f2.Write(buf, 0, lidos); // escrever byte(s) lido(s)
                        }
                        f1.Close(); // fechar arquivo de leitura
                        f2.Close(); // fechar arquivo de escrita
                }
            }
        }
```

2.9.1.2 Programa 2

```
        // programa: Copiar binário
        // programadores: Daniel Jorge, Robert Victor, João Gontijo, Arthur Henrique, Breno Vieira
        // data: 24/09/2017
        // descricao: Programa que faz a copia de um texto qualquer.
```

```
// entrada(s): linha de comando ambos arquivos, para copiar
// saida(s): arquivo copiado


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
namespace copia_bin_cada_byte_sga_s116 {


  class copia_bin_cada_byte {


static void Main(string[] args){
        int i,b;
        // args[0]: nome do arquivo de origem (ja existente)
        // args[1]: nome do arquivo de destino(a ser criado)
        FileInfo fi = new FileInfo(args[0]); // cria objeto Fileinfo para obter atributos do arquivo
        Console.WriteLine("Arquivo de origem: " + fi.Name); // exibe alguns atributos do arquivo
        Console.WriteLine("Diretorio: " +  fi.Directory);
        Console.WriteLine("Tamanho do arquivo: " + fi.Length);
        Console.WriteLine("Arquivo de destino: " + args[1]);
        Stream entrada  = File.Open(args[0], FileMode.Open); // abre arquivo de origem ja existente
        Stream saida = File.Open(args[1], FileMode.Create); // criar arquivo de destino
        for (i = 0; i < fi.Length ; i++) {
```

```
                    b = entrada.ReadByte();
                    saida.WriteByte( (byte)b);
                }
                Console.WriteLine("Bytes copiados: " + i );
            entrada.Close();
                saida.Close();
        }


          }
        }
```

2.10**Ex5.6**

2.10.1 Código fonte Ex5.6

```
        // programa: Testar Arquivos
        // programadores: Daniel Jorge, Robert Victor, João Gontijo, Breno Vieira, Arthur Henrique
        // data: 24/09/2017
        // descricao: exemplo de programa para manipular arquivo
        // entrada(s): digitar para selecionar opção do menu
        // saida(s): informação ou ação solicitada pelo menu
        using System;
        using System.Collections;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
        using System.IO;
```

```
namespace Exercicios
{
    class Ex56
    {
        static void Main(string[] args)
        {
            List<string> c = new List<string>();
            ConsoleKeyInfo receptor = new ConsoleKeyInfo();
            StreamWriter entradaE = new StreamWriter(args[0]);
            StreamWriter saidaE = new StreamWriter(args[1]);
            StreamReader entradaL;
            StreamReader saidaL;
            int i = 0;
            string x = "0";

            Console.WriteLine("Programa de teste do sistema de arquivos \nAlunos Robert, Daniel, Breno \nArquivo texto de entrada: entrada.txt \nArquivo texto de saída: saida.txt \nEntre com os caracteres: a x r s q w z *");

            while (!(string.Equals(x, "*")))
            {
                Console.Write("Informe o caracter: ");
                receptor = Console.ReadKey();
                c.Add(Convert.ToString(receptor.KeyChar));
                Console.WriteLine("\nCaracter adicionada\n");
                x = Convert.ToString(c[i]);
                i++;
            }
```

```
i = 0;

for (i = 0; i < c.Count - 1; i++)
{
    entradaE.WriteLine(c[i]);
}

entradaE.Close();

string[] chars = new string[c.Count];
string[] charsM = new string[c.Count];

entradaL = new StreamReader(args[0]);

Console.Write("\nCaracteres digitados: ");
for (i = 0; i < chars.Length - 1; i++)
{
    chars[i] = entradaL.ReadLine();
    Console.Write(chars[i] + " ");
    saidaE.WriteLine(chars[i].ToUpper());
}
saidaE.Close();
entradaL.Close();

saidaL = new StreamReader(args[1]);
```

```
            Console.Write("\nCaracteres convertidos: ");
            for (i = 0; i < charsM.Length - 1; i++)
            {
                charsM[i] = saidaL.ReadLine();
                Console.Write(charsM[i] + " ");
            }

            Console.ReadKey();
        }
    }
}
```

2.11**Ex5.8**

2.11.1 Código fonte Ex5.8

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
// copia_texto_linha_sga_s217
// copia_binario_cada_byte_sga_s217.exe arq_origem arq_destino
namespace copia_texto_linha_s217 {

  class copia_texto_linha_s217 {

static void Main(string[] args){
        int i=0;
        String linha;  //para ler ou escrever linhas do ou para o arquivo
```

```
            // args[0]: nome do arquivo de origem (ja existente): arq_origem
            // args[1]: nome do arquivo de destino(a ser criado): arq_destino
            FileInfo fi = new FileInfo(args[0]); // cria objeto Fileinfo para obter atributos do arquivo
            Console.WriteLine("Arquivo de origem: " + fi.Name); // exibe alguns atributos do arquivo
            Console.WriteLine("Diretorio: " +  fi.Directory);
            Console.WriteLine("Tamanho do arquivo: " + fi.Length);
            Console.WriteLine("Arquivo de destino: " + args[1]);

            if (File.Exists(args[0]))  //se existe o arquivo...
          {// Aqui se tem certeza que o arquivo existe

                StreamReader entrada = new StreamReader(args[0]); //abrir o arquivo origem
                StreamWriter saida   = new StreamWriter(args[1]); //abre arquivo de destino
                linha = entrada.ReadLine();     //ler 1a linha
            while (linha != null)          //enquanto houver dados...
            {
               saida.WriteLine(linha);      //escreve no arquivo
                        linha = entrada.ReadLine(); //ler proxima linha
                        i++;
            }
            entrada.Close();       //fecha arquivo de leitura
                saida.Close();  //fecha o arquivo de escrita
          }
            Console.WriteLine("Bytes copiados: " + i );
```

```
        }


          }
        }
```

2.12**Ex5.9**

2.12.1 Código fonte Ex5.9

```
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
        using System.IO;


        namespace ConsoleApp69
        {
            class Program
            {
                static void Main(string[] args)
                {
                    string caracteres, aux;
                    FileInfo informacoes = new FileInfo(args[0]);
                    Console.Write("Arquivo de entrada:{0}\nArquivo de saida{1}\n", args[0],
args[1]);
                    Console.Write("Digite uma sequencia de caractereres, depois digite *
no ultimo caracter:");
                    caracteres = Console.ReadLine();
                    string[] separa = caracteres.Split('*');
```

```
            aux = separa[0];
            Console.Write("Caracter digitados:{0}\n",aux);
            char[] caracSeparado = new char[aux.Length];
            for (int i = 0; i < aux.Length; i++)
            {
                caracSeparado[i] = aux[i];
            }
            File.Create(args[0]);
            StreamWriter arqEntrada = new StreamWriter(args[0]);
            for (int i = 0; i < aux.Length; i++)
            {
                arqEntrada.WriteLine(caracSeparado[i]);
            }
            arqEntrada.Close();
            Console.ReadKey();

            if (File.Exists(args[0]))
            {
                int contLinhas = 0;
                StreamReader arqLeitura = new StreamReader(args[0]);
                string linha;
                linha = arqLeitura.ReadLine();
                string dados = "";
                while(linha != null)
                {
                    dados = linha + ',';
                    contLinhas++;
```

```
                    linha = arqLeitura.ReadLine();
                }
                string[] dadosColetados = dados.Split(',');
                File.Create(args[1]);
                StreamWriter arqEscrita = new StreamWriter(args[1]);
                for (int i = 0; i < contLinhas; i++)
                {
                    Console.Write(dadosColetados[i].ToUpper());
                    arqEscrita.WriteLine(dadosColetados[i].ToUpper());
                }
                arqEscrita.Close();
            }
            else
            {
                Console.Write("Erro, arquivo não existe.");
                Console.ReadKey();
            }
            Console.ReadKey();
        }
    }
}
```

2.13**Ex6.1**

2.13.1 Código fonte Ex6.1

```
// programa: Arquivos
// programadores: Daniel Jorge, Robert Victor, João Gontijo, Breno Vieira, Arthur Henrique
// data: 24/09/2017
```

```
// descricao: exemplo de programa para manipular arquivo
// entrada(s): digitar para selecionar opção do menu
// saida(s): informação ou ação solicitada pelo menu
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Exercicios
{
    class Ex61
    {
        static void Main(string[] args)
        {
            int op;
            string receptor, novoNome;
            string dir = Environment.CurrentDirectory;
            ConsoleKeyInfo k = new ConsoleKeyInfo();
                Console.Clear();
                Menu();
                k = Console.ReadKey();
                receptor = Convert.ToString(k.KeyChar);
                if (int.TryParse(receptor, out op))
                {
                    op = int.Parse(receptor);
```

```
                    switch (op)
                    {
                        case 1:
                            Console.Clear();
                            Console.WriteLine("\n1. Excluir o arquivo");
                            Console.WriteLine("-------------------------------------");
                            if (File.Exists(dir + @"\" +args[0]))
                            {
                                File.Delete(dir + @"\" + args[0]);
                                Console.WriteLine("\nArquivo excluido com sucesso - Pressione qualquer tecla para sair");
                            }

                            else
                            {
                                Console.WriteLine("\nArquivo inexistente - Pressione qualquer tecla para sair");
                            }

                            break;

                        case 2:
                            Console.Clear();
                            Console.WriteLine("\n2. Renomear o arquivo");
                            Console.WriteLine("-------------------------------------");
                            Console.Write("\nDigite o nome do arquivo a ser renomeado: ");
```

```
                        if (File.Exists(dir + @"\" + args[0]))
                        {
                            Console.Write("\nDigite o novo nome do arquivo: ");
                            novoNome = Console.ReadLine();
                            File.Move(dir + @"\" + args[0], dir + @"\" + novoNome);
                            Console.WriteLine("\nNome do arquivo alterado com
sucesso - Pressione qualquer tecla para sair");
                        }

                        else
                        {
                            Console.WriteLine("\nArquivo inexistente - Pressione
qualquer tecla para sair");
                        }

                        break;

                    case 3:
                        Console.Clear();
                        Console.WriteLine("\n3. Copiar um arquivo");
                        Console.WriteLine("-------------------------------------");
                    Console.Write("\nDigite o nome da cópia do arquivo: ");
                    novoNome = Console.ReadLine();
                    if (!string.Equals(args[0], novoNome))
                    {
                        File.Copy(args[0], novoNome);
                        Console.Write("\nArquivo copiado com sucesso - Pressione
qualquer tecla para sair");
```

```
                        }
                            break;


                        case 4:
                        Console.Clear();
                            Console.WriteLine("\n4. Sair do programa");
                            Console.WriteLine("-------------------------------------");
                            Console.WriteLine("\nPressione qualquer tecla para sair");
                            break;


                        default:
                            Console.Clear();
                            Console.WriteLine("Opção digitada inválida - Pressione
qualquer tecla para sair");
                            break;
                       }
                    }


                    else
                    {
                       Console.WriteLine("\nOpção digitada inválida - Pressione
qualquer tecla para sair");
                    }
                    Console.ReadKey();



                }
```

```
            static void Menu()
            {
                Console.WriteLine("MENU DE OPÇÕES");
                Console.WriteLine("\n1. Excluir o arquivo");
                Console.WriteLine("\n2. Renomear o arquivo");
                Console.WriteLine("\n3. Copiar o arquivo");
                Console.WriteLine("\n4. Sair do programa");
            }
        }
    }
```

2.14 **Ex6.2**

2.14.1 Código fonte Ex6.2

2.14.1.1 Programa 1

```
    //
    // nome do programa: Ex621
    //
    // programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
    // data: 24/09/2017
    // entrada(s): Caminho do diretorio a ser criado com o nome do diretorio
    // saida(s): Imprime se o diretorio foi criado, ou se ja é existente
    // para executar  e testar digite:
    // Ex621.exe Diretorio01
    // descricao: Cria um diretório no caminho especificado, se conter somente o nome
    // cria o diretorio onde está o arquivo exe do programa
    //
```

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ex_621
{
    class Ex621
    {
        static void Main(string[] args)
        {
            string nomeDir = args[0];
            if (!Directory.Exists(nomeDir))
            {
                Directory.CreateDirectory(nomeDir);
                        Console.WriteLine("Diretorio criado");
            }
                else
                {
                        Console.WriteLine("Diretório já existente !");
                }
        }
    }
}
```

2.14.1.2 Programa 2

```
//
// nome do programa: Ex622
//
// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
// data: 24/09/2017
// entrada(s): Nao tem
// saida(s): Imprime o diretorio absoluto onde o programa esta sendo executado
// para executar  e testar digite:
// Ex622.exe
// descricao: Mostra o nome do diretorio onde o programa foi executado
//


using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Ex_622
{
    class Ex622
    {
        static void Main(string[] args)
        {
```

```
                Console.WriteLine("Diretorio absoluto: {0}",Directory.GetCurrentDirectory());
            }
        }
    }
```

2.14.1.3 Programa 3

```
    //
    // nome do programa: Ex623
    //
    // programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
    // data: 24/09/2017
    // entrada(s): Caminho do diretorio
    // saida(s): Imprime as pastas e arquivos do diretorio
    // para executar  e testar digite:
    // Ex623.exe Diretorio01
    // descricao: O programa recebe um caminho pela lc e imprime as pastas e arquivos que estao dentro dele
    //


    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;


    namespace Ex_623
```

```
{
    class Ex623
    {
        static void Main(string[] args)
        {
            string path = args[0];

            if (Directory.Exists(path))
            {
                string[] diretorios = Directory.GetDirectories(path);
                string[] arquivos = Directory.GetFiles(path);
                if (diretorios.Length != 0)
                {
                    Console.WriteLine("**********Diretórios**********\n");
                    foreach (var dir in diretorios)
                    {
                        Console.WriteLine(dir);
                    }
                }

                if (arquivos.Length != 0)
                {
                    Console.WriteLine("\n**********Arquivos**********\n");

                    foreach (var arq in arquivos)
                    {
                        Console.WriteLine(arq);
```

```
                    }
                }


                    if (arquivos.Length == 0 && diretorios.Length == 0)
                    {
                        Console.WriteLine("Diretorio vazio!");
                    }
                Console.ReadKey();
            }
            else
            {
                Console.WriteLine("Diretório inexistente!");
            }
        }
    }
}
```

2.14.1.4 Programa 4

```
//
// nome do programa: Ex624
//
// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
// data: 24/09/2017
// entrada(s): Caminho do diretorio
// saida(s): Imprime se o diretorio foi deletado ou se é inexistente
// para executar  e testar digite:
// Ex624.exe Diretorio01
```

```
        // descricao: Recebe o caminho de um diretorio pela lc, se ele existir, ele é deletado
        //


        using System;
        using System.Collections.Generic;
        using System.IO;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;


        namespace Ex_624
        {
            class Ex624
            {
                static void Main(string[] args)
                {
                    string path = args[0];


                    if (Directory.Exists(path))
                    {
                        Directory.Delete(path, true);
                                    Console.WriteLine("Diretorio deletado!");
                    }
                    else
                    {
                        Console.WriteLine("Diretório inexistente!");
```

```
                }
            }
        }
    }
```

2.14.1.5 Programa 5

```
    //
    // nome do programa: Ex625
    //
    // programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
    // data: 24/09/2017
    // entrada(s): Caminho do diretorio, novo nome do diretorio
    // saida(s): Imprime se o diretorio teve o nome alterado, ou se é inexistente
    // para executar  e testar digite:
    // Ex625.exe Diretorio01 DiretorioNovo
    // descricao: Renomeia um diretorio
    //


    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;


    namespace Ex_625
    {
```

```
class Ex625
{
   static void Main(string[] args)
   {
      string pathAntigo = args[0];
      string pathNovo = args[1];
            string pathRaiz = "";
            pathRaiz = Directory.GetCurrentDirectory()+"\\";
      if (Directory.Exists(pathAntigo))
      {
                  if (Path.GetDirectoryName(pathAntigo) != "")
                  {
                        pathRaiz =
Path.GetDirectoryName(pathAntigo)+"\\";
                  }
                  Directory.Move(pathAntigo, pathRaiz+pathNovo);
         Console.WriteLine("Nome alterado");
      }
            else
            {
                  Console.WriteLine("Diretório inexistente!");
            }
      Console.ReadKey();
   }
}
```

2.14.1.6 Programa 6

```
//

// nome do programa: Ex626

//

// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo

// data: 24/09/2017

// entrada(s): Caminho do diretorio a ser copiado, caminho para onde vai copiar o diretorio

// saida(s): Imprime se o diretorio foi copiado, ou se ja é existente

// para executar  e testar digite:

// Ex626.exe Diretorio01 DiretorioCopiado

// descricao:Copia um diretorio existente para um caminho especificado

//


using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Ex_626

{

    class Ex626

    {

        static void Main(string[] args)

        {
```

```
            string pathAntigo = args[0];
            string pathNovo = args[1];
            if (Directory.Exists(pathAntigo))
            {
                copiaDiretorio(pathAntigo,pathNovo);
                            Console.WriteLine("Diretorio copiado");
            }

            else
            {
                Console.WriteLine("Diretório inexistente!");
            }
                Console.ReadKey();
        }
        public static void copiaDiretorio(string input, string output)
        {
            if (!Directory.Exists(output))
            {
                Directory.CreateDirectory(output);
            }

            if (!string.IsNullOrEmpty(input))
            {
                string[] arquivos = Directory.GetFiles(input);
                foreach (string arquivo in arquivos)
                {
```

```
                    File.Copy(input + @"\" + Path.GetFileName(arquivo), output + @"\" + Path.GetFileName(arquivo), true);
                }
                string[] diretorios = Directory.GetDirectories(input);
                foreach (string diretorio in diretorios)
                {
                    copiaDiretorio(diretorio, output + @"\" + diretorio.Split(Convert.ToChar(@"\"))[diretorio.Split(Convert.ToChar(@"\")).Length - 1]);
                }
            }

        }

    }
}
```

2.15**Ex6.3**

2.15.1 Código fonte Ex6.3

```
//
// nome do programa: Ex63
//
// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
// data: 24/09/2017
// entrada(s): Nao tem
// saida(s): Imprime dados dos sistemas de arquivos
// para executar  e testar digite:
// Ex63.exe
```

```
        // descricao: Pega os sistemas de arquivos da máquina e retorna as informacoes
        //




        using System;
        using System.Collections.Generic;
        using System.IO;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
        using System.Management;


        namespace Ex_623
        {
            class Ex63
            {
                static void Main(string[] args)
                {
                    DriveInfo[] allDrives = DriveInfo.GetDrives();


                    foreach (var d in allDrives)
                    {
                        Console.WriteLine("\nNome do drive: {0}",d.Name);
                        if (d.IsReady ==  true)
                        {
                            Console.WriteLine("\tSistema de Arquivo: {0}", d.DriveFormat);
```

```
                    Console.WriteLine("\tTipo de drive: {0}", d.DriveType);
                    Console.WriteLine("\tDiretorio: {0}",d.RootDirectory);
                    Console.WriteLine("\tCapacidade total: {0} bytes", d.TotalSize);
                    Console.WriteLine("\tEspaço livre: {0} bytes",d.AvailableFreeSpace);
                }
                else
                {
                    Console.WriteLine("\tDrive nao está em funcionamento no momento");
                }

            }
            Console.ReadKey();
        }
    }
}
```

3 CONCLUSÃO

Com a execução desse trabalho fica evidente o quanto a orientação a objetos é importante no contexto de sistemas de informação, porque além de estar inserida atualmente no desenvolvimento de sistemas em geral, a orientação a objetos proporciona uma melhor organização de código e sua reutilização e torna o processo de desenvolvimento mais próximo da realidade, podendo abstrair somente coisas necessárias e que consequentemente deixam o código de certa forma mais intuitivo com relação a regra de negócio estabelecida. Sendo assim conclui-se que o aprendizado dessa maneira de trabalhar é uma boa prática e é de suma importância para o desenvolvimento de sistemas de informação.