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 3 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 3 PROGRAMAÇÃO ORIENTADA POR OBJETOS

Relatório das atividades desenvolvidas no laboratório 3 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 Ex1.1

2.1.1 Código fonte Ex1.1

2.1.1.1 Classe 1

```
// programa: TestaConta.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para executar ações de um banco
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ex_11
{
    class TestaConta
    {
        static void Main(string[] args)
        {
            int d, option;
            double valor;
            Console.Write("\nNova conta deve ser criada - Digite o dia de aniversário: ");
            d = int.Parse(Console.ReadLine());
            if (d > 31 || d < 1)
            {
```

```
                    while (d > 31 || d < 1)
                    {
                        Console.WriteLine("\nDia inválido - Dia deve ser maior que 0 e menor que 31");
                        Console.Write("\nDigite o dia de aniversário: ");
                        d = int.Parse(Console.ReadLine());
                    }
                }

                Conta c = new ContaPoupança(d);

                do
                {
                    Console.Clear();
                    Menu();
                    option = int.Parse(Console.ReadLine());

                    switch (option)
                    {

                        case 1:
                            Console.Clear();
                            Console.Write("\nDigite o valor a ser sacado: ");
                            valor = double.Parse(Console.ReadLine());
                            c.Saque(ref valor);
                            if (valor == -1)
                            {
```

```
                    Console.WriteLine("\nImpossível efetuar saque maior que o saldo disponível - Pressione qualquer tecla para sair");
                }

                else
                {
                    Console.WriteLine("Saque efetuado com sucesso - Pressione qualquer tecla para sair");
                }
                Console.ReadKey();
                break;

            case 2:
                Console.Clear();
                Console.Write("Digite um valor para ser depositado: ");
                valor = double.Parse(Console.ReadLine());
                c.Deposito(ref valor);
                if (valor == -1)
                {
                    Console.WriteLine("\nImpossível efetuar depósito menor que 1 - Pressione qualquer tecla para sair");
                }

                else
                {
                    Console.WriteLine("Depósito efetuado com sucesso - Pressione qualquer tecla para sair");
                }
```

```
                    Console.ReadKey();
                    break;

                case 3:
                    Console.Clear();
                    c.ExibeExtrato();
                    Console.ReadKey();
                    break;

                case 4:
                    break;

                default:
                    break;
            }
        }
        while (option != 4);

    }

    static void Menu()
    {
        Console.WriteLine("MENU DE OPÇÕES");
        Console.WriteLine("\n1- Efetuar saque");
        Console.WriteLine("\n2- Efetuar depósito");
        Console.WriteLine("\n3- Imprimir extrato");
    }
```

```
    }
}
```

2.1.1.2 Classe 2

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

namespace Ex_11
{
    abstract class Conta
    {
        protected double saldo;

        public double Saldo
        {
            get
            {
                return saldo;
            }

            set
            {
                if (value > 0)
                {
                    saldo = value;
```

```
            }
        }
    }

    public abstract void ExibeExtrato();
    public abstract void Saque(ref double valor);
    public abstract void Deposito(ref double valor);
  }
}
```

2.1.1.3 Classe 3

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

namespace Ex_11
{
  class ContaPoupança: Conta
  {
    int diaAniv;

    public int DiaAniv
    {
      get
      {
        return diaAniv;
```

```
            }

            set
            {
                diaAniv = value;
            }
        }

        public ContaPoupança(int diaAniv)
        {
            saldo = 0;
            this.diaAniv = diaAniv;
        }

        public override void ExibeExtrato()
        {
            Console.WriteLine("EXTRATO DETALHADO DE CONTA POUPANÇA");
            DateTime now = DateTime.Now;
            Console.WriteLine("\nDATA: {0}", now);
            Console.WriteLine("\nSALDO: {0}", Saldo);
            Console.WriteLine("\nANIVERSÁRIO: {0}", diaAniv);
        }

        public override void Saque(ref double valor)
        {
            if (saldo - valor >= 0 && valor > 0)
```

```
            {
                saldo -= valor;
            }

            else
            {
                valor = -1;
            }
        }

        public override void Deposito(ref double valor)
        {
            if (valor > 0)
            {
                saldo += valor;
            }

            else
            {
                valor = -1;
            }
        }
    }
}
```

2.2 Ex1.2

2.2.1 Código fonte Ex1.2

2.2.1.1 Classe 1

```
// programa: Program.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para executar cadastros e calculos
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;


namespace Imposto
{
    class Program
    {
        const int maxContribuintes = 100;
        static Contribuinte[] lst = new Contribuinte[maxContribuintes];
        //Verifacor de CPF/CNPJ
        static bool Verificador(string registro)
        {
            bool verificador = false;
            StreamReader arqContriLeitura = new StreamReader("cadastros.txt");
            string linha;
            linha = arqContriLeitura.ReadLine();
            while (linha != null)
```

```
            {
                if(linha == "")
                {
                    linha = arqContriLeitura.ReadLine();
                }
                else
                {
                    string[] separado = linha.Split(',');
                    if (separado[1] == registro)
                    {
                        verificador = true;
                    }
                    linha = arqContriLeitura.ReadLine();
                }
            }
            arqContriLeitura.Close();
            if (verificador)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        //Cadastro de Contribuinte
        static void CadContribuinte()
```

```
            {
                char tipo;
                string registro, nome, endereco;
                double valor;
                if(!File.Exists("cadastros.txt"))
                {
                    StreamWriter arqCadastros = new StreamWriter("cadastros.txt");
                    arqCadastros.Close();
                }
                Console.Write("Informe abaixo os dados do contribuinte.\nO contribuinte é Pessoa Fisica ou Juridica (Digite F ou J):");
                tipo = char.Parse(Console.ReadLine());
                if(tipo == 'f' || tipo == 'F')
                {
                    Console.Write("Digite o CPF do contribuinte:");
                    registro = Console.ReadLine();
                    if (Verificador(registro) == true)
                    {
                        Console.Write("Erro, contribuinte já registrado.\n");
                    }
                    else
                    {
                        Console.Write("Qual o nome do Contribuinte:");
                        nome = Console.ReadLine();
                        Console.Write("Qual o enderço do Contribuinte(Digite o Rua , numero)):");
                        endereco = Console.ReadLine();
                        Console.Write("Qual salario do Contribuinte:");
```

```
                        valor = double.Parse(Console.ReadLine());
                        Fisica PFisica = new Fisica(nome, endereco, registro, valor);
                        StreamReader arq = new StreamReader("cadastros.txt");
                        string dados = arq.ReadToEnd();
                        arq.Close();
                        StreamWriter arqEscrever = new StreamWriter("cadastros.txt");
                        string linha = dados + "PF" + "," + PFisica.Cpf + "," + PFisica.Nome + "," + PFisica.Endereco + "," + PFisica.Salario;
                        arqEscrever.WriteLine(linha);
                        arqEscrever.Close();
                        Contribuinte.Cont++;
                        lst[Contribuinte.Cont] = PFisica;
                    }
                }
                else if (tipo == 'J' || tipo == 'j')
                {
                    Console.Write("Digite o CNPJ do contribuinte:");
                    registro = Console.ReadLine();
                    if (Verificador(registro))
                    {
                        Console.Write("Erro, contribuinte já registrado.\n");
                    }
                    else
                    {
                        Console.Write("Qual o nome do Contribuinte:");
                        nome = Console.ReadLine();
                        Console.Write("Qual o enderço do Contribuinte(Digite o Rua , numero)):");
```

```
                    endereco = Console.ReadLine();
                    Console.Write("Qual faturamento do Contribuinte:");
                    valor = double.Parse(Console.ReadLine());
                    Juridica PJuridica = new Juridica(nome, endereco, registro, valor);
                    StreamReader arq = new StreamReader("cadastros.txt");
                    string dados = arq.ReadToEnd();
                    arq.Close();
                    StreamWriter arqEscrever = new StreamWriter("cadastros.txt");
                    string linha = dados + "PJ" + "," + PJuridica.Cnpj + "," + PJuridica.Nome + "," + PJuridica.Endereco + "," + PJuridica.Faturamento;
                    arqEscrever.WriteLine(linha);
                    arqEscrever.Close();
                    Contribuinte.Cont++;
                    lst[Contribuinte.Cont] = PJuridica;
                }
            }
            else
            {
                Console.Write("Erro, opção invalida.\nDigite alguma tecla para continuar.\n");
                Console.ReadKey();
                Console.Clear();
            }
        }
        static void ExcluirContribuinte()
        {
            string registro, linhas = "", todosDados;
            char tipo;
```

```
            Console.Write("O contribuinte que deseja excluir é Pessoa Fisica ou Juridica (Digite F ou J):");
            tipo = char.Parse(Console.ReadLine());
            if (tipo == 'f' || tipo == 'F')
            {
                Console.Write("Digite o CPF do contribuinte:");
                registro = Console.ReadLine();
                if (Verificador(registro) == false)
                {
                    Console.Write("Erro, contribuinte não registrado.\n");
                }
                else
                {
                    StreamReader arqExcluir = new StreamReader("cadastros.txt");
                    todosDados = arqExcluir.ReadToEnd();
                    string[] separa = todosDados.Split('\n');
                    for (int i = 0; i < separa.Length - 1; i++)
                    {
                        string[] excluir = separa[i].Split(',');
                        if (!(excluir[1] == registro))
                        {
                            if(separa[i] != "")
                            {
                                linhas = linhas + separa[i];
                                if (i < (separa.Length -2))
                                {
                                    linhas = linhas + "\n";
```

```
                    }
                }
            }
        }
        arqExcluir.Close();
        StreamWriter arqNovo = new StreamWriter("cadastros.txt");
        arqNovo.WriteLine(linhas);
        arqNovo.Close();
    }
}
else if (tipo == 'j' || tipo == 'J')
{
    Console.Write("Digite o CNPJ do contribuinte:");
    registro = Console.ReadLine();
    if (Verificador(registro) == false)
    {
        Console.Write("Erro, contribuinte não registrado.\n");
    }
    else
    {
        StreamReader arqExcluir = new StreamReader("cadastros.txt");
        todosDados = arqExcluir.ReadToEnd();
        string[] separa = todosDados.Split('\n');
        for (int i = 0; i < separa.Length - 1; i++)
        {
            string[] excluir = separa[i].Split(',');
            if (!(excluir[1] == registro))
```

```
                        {
                            if(separa[i] != "")
                            {
                                linhas = linhas + separa[i];
                                if (i < (separa.Length -2))
                                {
                                    linhas = linhas + "\n";
                                }
                            }
                        }
                    }
                    arqExcluir.Close();
                    StreamWriter arqNovo = new StreamWriter("cadastros.txt");
                    arqNovo.WriteLine(linhas);
                    arqNovo.Close();
                }
            }
            else
            {
                Console.Write("Erro, opção invalida.\nDigite alguma tecla para continuar.\n");
                Console.ReadKey();
                Console.Clear();
            }
        }
        //Imprimir Contribuinte
        static void ImprimirContri()
```

```
            {
                string registro, dados;
                Console.Write("Digite o CPF ou CNPJ do contribuinte:");
                registro = Console.ReadLine();
                if (Verificador(registro) == false)
                {
                    Console.Write("Erro, contribuinte não registrado.\n");
                }
                else
                {
                    StreamReader arqImprimi = new StreamReader("cadastros.txt");
                    dados = arqImprimi.ReadToEnd();
                    string[] separaLinha = dados.Split('\n');
                    for (int i = 0; i < separaLinha.Length - 1; i++)
                    {
                        string[] dadosContri = separaLinha[i].Split(',');
                        if (dadosContri[1] == registro)
                        {
                            if (dadosContri[0]== "PF")
                            {
                                Console.Write("Tipo:                                Pessoa
Fisicia\nCPF:{0}\nNome:{1}\nEndereço:{2},{3}\nSalario:{4}\n",dadosContri[1],
dadosContri[2], dadosContri[3], dadosContri[4], dadosContri[5]);
                            }
                            else
                            {
```

```
                    Console.Write("Tipo: Pessoa Juridica\nCNPJ:{0}\nNome:{1}\nEndereço:{2},{3}\nFaturamento:{4}\n", dadosContri[1], dadosContri[2], dadosContri[3], dadosContri[4], dadosContri[5]);
                }
            }
        }
        arqImprimi.Close();
    }
}
static void ImprimirFisico()
{
    string dadosFisico;
    StreamReader arqImprimiFisico = new StreamReader("cadastros.txt");
    dadosFisico = arqImprimiFisico.ReadToEnd();
    string[] separaLinha = dadosFisico.Split('\n');
    for (int i = 0; i < separaLinha.Length - 1; i++)
    {
        string[] dadosContri = separaLinha[i].Split(',');
        if (dadosContri[0] == "PF")
        {
            Console.Write("Tipo: Pessoa Fisicia\nCPF:{0}\nNome:{1}\nEndereço:{2},{3}\nSalario:{4}\n", dadosContri[1], dadosContri[2], dadosContri[3], dadosContri[4], dadosContri[5]);
        }
    }
    arqImprimiFisico.Close();
}
static void ImprimirJuridico()
```

```
            {
                string dadosJuridico;
                StreamReader arqImprimiJuridi = new StreamReader("cadastros.txt");
                dadosJuridico = arqImprimiJuridi.ReadToEnd();
                string[] separaLinha = dadosJuridico.Split('\n');
                for (int i = 0; i < separaLinha.Length - 1; i++)
                {
                    string[] dadosContri = separaLinha[i].Split(',');
                    if (dadosContri[0] == "PJ")
                    {
                        Console.Write("Tipo: Pessoa Juridica\nCNPJ:{0}\nNome:{1}\nEndereço:{2},{3}\nFaturamento:{4}\n",
dadosContri[1], dadosContri[2], dadosContri[3], dadosContri[4], dadosContri[5]);
                    }
                }
                arqImprimiJuridi.Close();
            }
            static void CalcImposto()
            {
                double valor;
                string registro, dados;
                Console.Write("Digite o CPF ou CNPJ do contribuinte:");
                registro = Console.ReadLine();
                if (Verificador(registro) == false)
                {
                    Console.Write("Erro, contribuinte não registrado.\n");
                }
                else
```

```
        {
            StreamReader arqImposto = new StreamReader("cadastros.txt");
            dados = arqImposto.ReadToEnd();
            string[] separaLinha = dados.Split('\n');
            for (int i = 0; i < separaLinha.Length - 1; i++)
            {
                string[] dadosContri = separaLinha[i].Split(',');
                if (dadosContri[1] == registro)
                {
                    if(dadosContri[0] == "PF")
                    {
                        valor = double.Parse(dadosContri[5]);
                        if (valor > 1400 && valor <= 2100)
                        {
                            Console.Write("Imposto a ser pago é:{0}\n", (valor * 0.1));
                        }
                        else if (valor > 2100 && valor <= 2800)
                        {
                            Console.Write("Imposto a ser pago é:{0}\n", (valor * 0.15));
                        }
                        else if (valor > 2800 && valor <= 3600)
                        {
                            Console.Write("Imposto a ser pago é:{0}\n", (0.25 * valor));
                        }
                        else if (valor > 3600)
                        {
                            Console.Write("Imposto a ser pago é:{0}\n", (0.30 * valor));
```

```
                        }
                        else
                        {
                            Console.Write("Imposto a ser pago é:{0}\n", 0);
                        }


                    }
                    else
                    {
                        valor = double.Parse(dadosContri[5]);
                        Console.Write("Imposto a ser pago é:{0}\n", (valor * 0.1));
                    }
                }
            }
            arqImposto.Close();
        }
    }
    static void Main(string[] args)
    {
        int repetidor = 1, opcao;
        while (repetidor != 0)
        {
            Console.Write("Menu de opções abaixo:\n1.Incluir um contribuinte.\n2.Excluir um contribuinte.\n3.Exibir os dados de um contribuinte: CPF/CNPJ, nome, endereço e salario/faturamento.\n4.Calcular e exibir o imposto a ser pago por um contribuinte.\n5.Imprimir uma relação dos contribuintes Pessoa Física cadastrados, mostrando os dados:CPF, nome e endereço.\n6.Imprimir uma relação dos contribuintes Pessoa Jurídica cadastrados, mostrando os dados: CNPJ, nome e endereço.\n7.Sair do programa.\nQual opção desejada:");
```

```
opcao = int.Parse(Console.ReadLine());
Console.Clear();
switch (opcao)
{
    case 1:
        CadContribuinte();
        Console.Write("Digite alguma tecla para continuar.\n");
        Console.ReadKey();
        Console.Clear();
        break;
    case 2:
        ExcluirContribuinte();
        Console.Write("Digite alguma tecla para continuar.\n");
        Console.ReadKey();
        Console.Clear();
        break;
    case 3:
        ImprimirContri();
        Console.Write("Digite alguma tecla para continuar.\n");
        Console.ReadKey();
        Console.Clear();
        break;
    case 4:
        CalcImposto();
        Console.Write("Digite alguma tecla para continuar.\n");
        Console.ReadKey();
        Console.Clear();
```

```
                        break;
                    case 5:
                        ImprimirFisico();
                        Console.Write("Digite alguma tecla para continuar.\n");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case 6:
                        ImprimirJuridico();
                        Console.Write("Digite alguma tecla para continuar.\n");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case 7:
                        repetidor = 0;
                        break;
                    default:
                        Console.Write("Erro, opção invalida.\nDigite alguma tecla para
continuar.\n");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                }
            }
        }
    }
}
```

2.2.1.2 Classe 2

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

namespace Imposto
{
    public abstract class Contribuinte
    {
        static private int cont = -1;
        static public int Cont
        {
            get
            {
                return cont;
            }
            set
            {
                cont = value;
            }
        }
        //Atributos
        protected string nome;
        public string Nome
        {
```

```
            get
            {
                return nome;
            }
        }
        protected string endereco;
        public string Endereco
        {
            get
            {
                return endereco;
            }
        }
        //Metodos
        abstract public double CalcImposto(double salario);
        abstract public void Excluir();
    }
}
```

2.2.1.3 Classe 3

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

namespace Imposto
{
```

```
class Fisica : Contribuinte
{
    //Atributos
    protected string cpf;
    public string Cpf
    {
        get
        {
            return cpf;
        }
    }
    protected double salario;
    public double Salario
    {
        get
        {
            return salario;
        }
    }
    //Metodos
    public Fisica(string nome, string endereco, string cpf, double salario)
    {
        this.nome = nome;
        this.endereco = endereco;
        this.cpf = cpf;
        this.salario = salario;
    }
```

```
public override double CalcImposto(double salario)
{
    if (salario > 1400 && salario <= 2100 )
    {
        return (0.1 * salario);
    }
    else if (salario > 2100 && salario <= 2800)
    {
        return (0.15 * salario);
    }
    else if (salario > 2800 && salario <= 3600)
    {
        return (0.25 * salario);
    }
    else if (salario > 3600)
    {
        return (0.30 * salario);
    }
    else
    {
        return 0;
    }
}
public override void Excluir()
{
    nome = null;
    endereco = null;
```

```
            cpf = null;
            salario = 0;
        }
    }
}
```

2.2.1.4 Classe 4

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

namespace Imposto
{
    class Juridica : Contribuinte
    {
        //Atributos
        protected string cnpj;
        public string Cnpj
        {
            get
            {
                return cnpj;
            }
        }
        protected double faturamento;
        public double Faturamento
```

```
        {
            get
            {
                return faturamento;
            }
        }
        //Metodos
        public Juridica(string nome, string endereco, string cnpj, double faturamento)
        {
            this.nome = nome;
            this.endereco = endereco;
            this.cnpj = cnpj;
            this.faturamento = faturamento;
        }
        public override double CalcImposto(double faturamento)
        {
            return (0.1 * faturamento);
        }
        public override void Excluir()
        {
            nome = null;
            endereco = null;
            cnpj = null;
            faturamento = 0;
        }
    }
```

```
}
```

2.3 Ex1.3

2.3.1 Código fonte Ex1.3

2.3.1.1 Classe 1

```
//
// nome do programa: Ex1.3
//
// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
// data: 12/10/2017
// entrada(s): Nao tem
// saida(s): Imprime os dados do cliente e do funcionario criado
// para testar digite:
// Ex13.cs
// descricao: Cria um cliente e um funcionario e busca seus respectivos dados.
//
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace ConsoleApp6
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
                // Criando funcionario
                Funcionario f =  new Funcionario("Breno","Teste","05/03/1999",2000);


                // Criando cliente
                Cliente c = new Cliente("Joao", "Rua teste 01", "01/01/2000", 0524, 200);


                Console.WriteLine("Dados do funcionario:");
                Console.WriteLine("Nome: {0}\nEndereco: {1}\nData Nascimento: {2}\nSalario: {3}",f.InformaNome(),f.Endereco,f.Nascimento,f.InformarSalario());
                Console.WriteLine();
                Console.WriteLine("Dados do Cliente:");
                Console.WriteLine("Nome: {0}\nEndereco: {1}\nData Nascimento: {2}\nCartao: {3}\nDebito: {4}", c.InformaNome(), c.Endereco, c.Nascimento, c.InformarCartao(), c.InformarDebito());
                Console.ReadKey();
            }
        }
    }
```

2.3.1.2 Classe 2

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


    namespace ConsoleApp6
    {
```

```
abstract class Pessoa
{
    protected string nome;
    protected string endereco;
    protected string nascimento;

    public string InformaNome()
    {
        return nome;
    }

    public string Endereco
    {
        get
        {
            return endereco;
        }
        set
        {
            endereco = value;
        }
    }
    public string Nascimento
    {
        get
        {
            return nascimento;
```

```
            }
            set
            {
                nascimento = value;
            }
        }

    }
}
```

2.3.1.3 Classe 3

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

namespace ConsoleApp6
{
    class Funcionario: Pessoa
    {
        private double salario;

        public Funcionario(string nome, string endereco, string nascimento, double salario)
        {
            this.nome = nome;
            this.endereco = endereco;
```

```
            this.nascimento = nascimento;
            this.salario = salario;
        }

        public double InformarSalario()
        {
            return salario;
        }
        public int CalcularIdade()
        {
            DateTime DataNascimento = DateTime.Parse(nascimento);
            int anos = DateTime.Now.Year - DataNascimento.Year;

            if (DateTime.Now.Month < DataNascimento.Month ||
(DateTime.Now.Month == DataNascimento.Month && DateTime.Now.Day <
DataNascimento.Day))

                anos--;

            return anos;
        }
    }
}
```

2.3.1.4 Classe 4

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

```
using System.Threading.Tasks;


namespace ConsoleApp6
{
    class Cliente : Pessoa
    {
        private int cartao;
        private double debito;


        public Cliente(string nome, string endereco, string nascimento, int cartao,
double debito)
        {
            this.nome = nome;
            this.endereco = endereco;
            this.nascimento = nascimento;
            this.cartao = cartao;
            this.debito = debito;
        }
        public int InformarCartao()
        {
            return cartao;
        }
        public double InformarDebito()
        {
            return debito;
        }
    }
```

```
}
```

2.4 Ex2.1

2.4.1 Código fonte Ex2.1

2.4.1.1 Classe 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace _2._1_Lab_3
{
    static class Program
    {
        /// <summary>
        /// Ponto de entrada principal para o aplicativo.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

2.4.1.2 Classe 2

```
// Programa: Exercício 2.1 do LAB 3
// Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.
// Data: 19/10/2017
// Descrição: Programa que simula um controle de TV e DVD. Com funções de ligar, desligar, Volume, Aumentar e Diminuir o Canal,
// Inserir o Disco, Play, Pause, Stop.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace _2._1_Lab_3
{
    class cDVD : IControleRemoto
    {
        public bool statusPower = false; // Atributos
        public int volume = 0;
        public bool statusFilme = false;
        public bool statusCD = false;


        public void Volume(int vol)
        {
            volume = vol; // Volume DVD
        }
```

```
public void Power() // Status Power (on/off)
{
    if (!statusPower)
    {
        statusPower = true;
    }
    else
    {
        statusPower = false;
    }
}

public void Filme() // Status do filme, se ele está rodando ou não
{
    if (!statusFilme)
    {
        statusFilme = true;
    }
    else
    {
        statusFilme = false;
    }
}

public void CD() // Status do CD, se ele está inserido ou não
{
    if (!statusCD)
```

```
            {
                statusCD = true;
            }
            else
            {
                statusCD = false;
            }
        }
    }
}
```

2.4.1.3 Classe 3

```
// Programa: Exercício 2.1 do LAB 3
// Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.
// Data: 19/10/2017
// Descrição: Programa que simula um controle de TV e DVD. Com funções de ligar, desligar, Volume, Aumentar e Diminuir o Canal,
// Inserir o Disco, Play, Pause, Stop.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _2._1_Lab_3
{
    class cTelevisor : IControleRemoto
    {
```

```
public bool statusPower = false; // Atributos
public int volume = 0;
public int canal = 1;

public void Volume(int vol)
{
    volume = vol; // Volume Televisor
}

public void Power() //  Status da televisão (on/off)
{
    if (!statusPower)
    {
        statusPower = true;
    }
    else
    {
        statusPower = false;
    }
}

public void aumentarCanal() // Aumentar canal
{
    if (canal >= 1 && canal <= 82)
    {
        canal++;
    }
```

```
                else
                {
                    canal = 1;
                }
            }
            public void diminuirCanal() // Diminuir canal
            {
                if (canal >= 2 && canal <= 83)
                {
                    canal--;
                }
                else
                {
                    canal = 83;
                }
            }
        }
    }
```

2.4.1.4 Classe 4

```
    // Programa: Exercício 2.1 do LAB 3
    // Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.
    // Data: 19/10/2017
    // Descrição: Programa que simula um controle de TV e DVD. Com funções de ligar, desligar, Volume, Aumentar e Diminuir o Canal,
    // Inserir o Disco, Play, Pause, Stop.
    using System;
    using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace _2._1_Lab_3
{
    public partial class Form1 : Form
    {
        cTelevisor televisor = new cTelevisor(); // Objetos
        cDVD dvd = new cDVD();

        int seg = 0, min = 0, hr = 0; // Variaveis

        public Form1()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
```

```
        private void powerTV_Click(object sender, EventArgs e)
        {
            televisor.Power(); // Método de Ligar a TV

            if (televisor.statusPower) // Mudar label e enables
            {
                statusTV.Text = "On";
                volumeTV.Enabled = true;
                aumentarCanal.Enabled = true;
                diminuirCanal.Enabled = true;
            }
            else
            {
                statusTV.Text = "Off";
                volumeTV.Enabled = false;
                aumentarCanal.Enabled = false;
                diminuirCanal.Enabled = false;
            }
        }

        private void PowerDVD_Click(object sender, EventArgs e)
        {
            dvd.Power(); // Método de ligar o DVD

            if (dvd.statusPower)
            {
                statusDVD.Text = "On";
```

```
            volumeDVD.Enabled = true;
            inserirButton.Enabled = true;
            if (dvd.statusCD == true)
            {
                playButton.Enabled = true;
                stopButton.Enabled = true;
            }
        }
        else
        {
            statusDVD.Text = "Off";
            volumeDVD.Enabled = false;
            inserirButton.Enabled = false;
            playButton.Enabled = false;
            stopButton.Enabled = false;
            timer1.Enabled = false;
            statusFilme.Text = "00:00:00";
            dvd.Filme();
            playButton.Text = "PLAY";
        }
    }

    private void volumeTV_Scroll(object sender, EventArgs e)
    {
        televisor.Volume(volumeTV.Value);
        statusVolumeTV.Text = Convert.ToString(televisor.volume);
    }
```

```
private void canalTV_Click(object sender, EventArgs e)
{

}

private void VolumeDVD_Scroll(object sender, EventArgs e)
{
    dvd.Volume(volumeDVD.Value);
    statusVolumeDVD.Text = Convert.ToString(dvd.volume);
}

private void aumentarCanal_Click(object sender, EventArgs e)
{
    televisor.aumentarCanal();
    statusCanal.Text = Convert.ToString(televisor.canal);
}

private void diminuirCanal_Click(object sender, EventArgs e)
{
    televisor.diminuirCanal();
    statusCanal.Text = Convert.ToString(televisor.canal);
}

private void playButton_Click(object sender, EventArgs e)
{
    dvd.Filme();
```

```
            if (dvd.statusFilme)
            {
                playButton.Text = "PAUSE";
                inserirButton.Enabled = false;
                timer1.Enabled = true;
            }
            else
            {
                playButton.Text = "PLAY";
                inserirButton.Enabled = true;
                timer1.Enabled = false;
            }
        }

        private void timer1_Tick(object sender, EventArgs e) // Cronometro usado
no filme
        {
            seg++;
            if (seg == 60)
            {
                min++;
                seg = 0;
            }
            if (min == 60)
            {
                hr++;
```

```
                min = 0;
            }
            statusFilme.Text = hr.ToString().PadLeft(2, '0') + ":" + min.ToString().PadLeft(2, '0') + ":" + seg.ToString().PadLeft(2, '0');
        }

        private void inserirButton_Click(object sender, EventArgs e) // Metodo usado para verificar a existencia de disco
        {
            dvd.CD();

            if (dvd.statusCD)
            {
                statusCD.Text = "DISCO INSERIDO";
                inserirButton.Text = "Ejetar";
                statusFilme.Text = "00:00:00";
                playButton.Enabled = true;
                stopButton.Enabled = true;
            }
            else
            {
                statusCD.Text = "INSIRA O DISCO";
                inserirButton.Text = "Inserir";
                statusFilme.Text = "00:00:00";
                playButton.Enabled = false;
                stopButton.Enabled = false;
            }
        }
```

```
            private void stopButton_Click(object sender, EventArgs e) // botão de parada
            {
                statusFilme.Text = "00:00:00";
                dvd.Filme();
                playButton.Text = "PLAY";
                inserirButton.Enabled = true;
                timer1.Enabled = false;
                seg = 0;
                min = 0;
                hr = 0;
            }
        }
    }
```

2.4.1.5 Interface 1

```
        // Programa: Exercício 2.1 do LAB 3
        // Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.
        // Data: 19/10/2017
        // Descrição: Programa que simula um controle de TV e DVD. Com funções de ligar, desligar, Volume, Aumentar e Diminuir o Canal,
        // Inserir o Disco, Play, Pause, Stop.
        using System;
        using System.Collections.Generic;
        using System.Linq;
        using System.Text;
        using System.Threading.Tasks;
```

```
namespace _2._1_Lab_3
{
    interface IControleRemoto // Interface do controle
    {
        void Power();

        void Volume(int volume);
    }
}
```

2.5 Ex2.2

2.5.1 Código fonte Ex2.2

2.5.1.1 Classe 1

```
// programa: Exer2.2.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para armazenar os diferentes objetos em arraylist e depois printar.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Collections;

namespace Exer2._2
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Quantas Formas deseja criar?(quadrado,retangulo ou circulo:");

            ArrayList ListaFormas = new ArrayList();

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

            for (int x = 0; x < quantidade; x++) // criar arraylist de objetos
            {
                Console.WriteLine("Qual forma deseja criar?:");

                string nome = Console.ReadLine();

                if(nome == "quadrado" || nome == "Quadrado")
                {
                    Console.WriteLine("Entre com o Lado do quadrado:");

                    double lado = double.Parse(Console.ReadLine());
                    var quadrado = new Quadrado();

                    quadrado.Lado = lado;
```

```
            ListaFormas.Add(quadrado);


        }
        else if(nome =="retangulo" || nome == "Retangulo")
        {

            Console.WriteLine("Entre com o primeiro lado:");
            double lado1 = double.Parse(Console.ReadLine());
            Console.WriteLine("Entre com o segundo lado:");
            double lado2 = double.Parse(Console.ReadLine());

            var retangulo = new Retangulo();

            retangulo.Lado1 = lado1;
            retangulo.Lado2 = lado2;

            ListaFormas.Add(retangulo);
        }
        else if(nome == "circulo" || nome == "Circulo")
        {
            Console.WriteLine("Entre com o raio do circulo:");
            double raio = double.Parse(Console.ReadLine());

            var circulo = new Circulo();
            circulo.Raio = raio;

            ListaFormas.Add(circulo);
```

```
                    }
                    else
                    {
                        Console.WriteLine("Erro opcao invalida");
                    }



                }
                for (int x = 0; x < quantidade; x++) // testar e printar os objetos
                {
                    var forma = ListaFormas[x];
                    Console.Write($"Forma[{x}] => {forma.GetType()}");


                    if (forma is Circulo)
                    {
                        Console.Write($" com Raio : {(forma as Circulo).Raio}");
                        Console.WriteLine(" e perimetro :" + (forma as
Circulo).Perimetro() + " e Area :" + (forma as Circulo).Area());
                    }


                    else if (forma is Quadrado)
                    {
                        Console.Write($" com Lado : {(forma as Quadrado).Lado}");
                        Console.WriteLine(" com area de :" + (forma as Quadrado).Area()
+ " e perimetro de : " + (forma as Quadrado).Perimetro());
                    }
                    else if (forma is Retangulo)
                    {
```

```
                    Console.Write($" com Lado 1 : {(forma as Retangulo).Lado1} e Lado 2 : {(forma as Retangulo).Lado2}");

                    Console.WriteLine(" com area de :" + (forma as Retangulo).Area() + " e perimetro de : " + (forma as Retangulo).Perimetro());

                }

                Console.WriteLine();

            }

            Console.ReadKey();



            }

            }


        }
```

2.5.1.2 Classe 2

```
        // programa: Circulo.cs

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

        // data: 19/10/2017

        // Descrição: Classe para definir as caracteristicas de um circulo e suas funçoes.

        using System;

        using System.Collections.Generic;

        using System.Linq;

        using System.Text;

        using System.Threading.Tasks;
```

```
namespace Exer2._2
{
    class Circulo:IForma
    {
        private double raio;

        public Circulo(double raio)
        {
            raio = this.raio;
        }
        public Circulo()
        {

        }
        public double Raio
        {
            get
            {
                return raio;
            }
            set
            {
                raio = value;
            }
        }
```

```
        public double Area()
        {
            return (3.14*(raio*raio));// pi*raio^2
        }

        public double Perimetro()
        {
            return ((2*3.14)*raio);// 2*pi*raio
        }
    }
}
```

2.5.1.3 Classe 3

```
// programa: Quadrado.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe para definir as caracteristicas de um quadrado e suas funçoes.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exer2._2
{
    class Quadrado : IForma
    {
```

```
        private double lado;
        public Quadrado(double lado)
        {
            lado = this.lado;
        }

        public Quadrado()
        {
            lado = 0;
        }

        public double Lado
        {
            get
            {
                return lado;
            }
            set
            {
                lado = value;
            }
        }

        public double Area()
        {
            return (lado * lado);
        }
```

```
        public double Perimetro()
        {
            return (lado * 4);
        }
    }
}
```

2.5.1.4 Classe 4

```
// programa: Retangulo.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe para definir as caracteristicas de um retangulo e suas funçoes.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exer2._2
{
    class Retangulo:IForma
    {
        private double lado1,lado2;

        public Retangulo(double lado1 , double lado2)
        {
```

```
        lado1 = this.lado1;
        lado2 = this.lado2;
    }
    public Retangulo ()
    {
        lado1 = 0;
        lado2 = 0;
    }
    public double Lado1
    {
        get
        {
            return lado1;
        }
        set
        {
            lado1 = value;
        }
    }
    public double Lado2
    {
        get
        {
            return lado2;
        }
        set
        {
```

```
                lado2 = value;
            }
        }
        public double Area()
        {
            return (lado1*lado2);
        }

        public double Perimetro()
        {
            return ((lado1*2)+(lado2*2));
        }
    }
}
```

2.5.1.5 Interface 1

```
// programa: IForma.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Interface para definir os metodos a serem trabalhados nas classes.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Exer2._2
```

```
{
    interface IForma
    {
        double Area();
        double Perimetro();
    }
}
```

2.6 Ex3.1

2.6.1 Código fonte Ex3.1

2.6.1.1 Classe 1

```
// programa: Ex31.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para executar funções de uma empresa e prover registros de entrada e saida
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Ex_31
{
    class Ex31
    {
        const int nFunc = 100;
```

```
            static void Main(string[] args)
            {
                Funcionario[] f = new Funcionario[nFunc];
                int op, matricula;
                string nome, receptor;
                char tipo;
                ControleDePonto controle = new ControleDePonto();
                do
                {
                    Console.Clear();
                    Menu();
                    op = int.Parse(Console.ReadLine());
                    Console.Clear();
                    switch (op)
                    {
                        case 1:
                            if (Funcionario.contador <= 100)
                            {
                                Console.Write("\nDigite o nome do funcionário: ");
                                nome = Console.ReadLine();
                                Console.Write("Selecione o tipo de funcionário - G para
gerente, T para telefonista: ");
                                receptor = Console.ReadLine();
                                tipo = char.Parse(receptor.ToUpper());

                                switch (tipo)
                                {
```

```
                        case 'G':
                            f[Funcionario.contador - 1] = new Gerente();
                            f[Funcionario.contador - 1].Nome = nome;
                            Console.WriteLine("\nGerente incluído, sua matrícula é {0} - Pressione qualquer tecla para sair", Funcionario.contador);
                            Funcionario.contador++;
                            break;

                        case 'T':
                            f[Funcionario.contador - 1] = new Telefonista();
                            f[Funcionario.contador - 1].Nome = nome;
                            Console.WriteLine("\nTelefonista incluída, sua matrícula é {0} - Pressione qualquer tecla para sair", Funcionario.contador);
                            Funcionario.contador++;
                            break;

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

                else
                {
                    Console.WriteLine("\nLimite de funcionários já foi atingido - Pressione qualquer tecla para sair");
                }
```

```
                    break;


                case 2:
                    Console.Write("\nDigite o número de matrícula do funcionário: ");
                    matricula = int.Parse(Console.ReadLine());
                    if (matricula > 0 && matricula <= Funcionario.contador)
                    {
                        if (!f[matricula - 1].Trabalhando)
                        {
                            controle.Entrada(f[matricula - 1]);
                        }

                        else
                        {
                            Console.WriteLine("\nFuncionário já está em serviço - Pressione qualquer tecla para sair");
                        }
                    }

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

                case 3:
```

```
                    Console.Write("\nDigite o número de matrícula do funcionário: ");
                    matricula = int.Parse(Console.ReadLine());
                    if (matricula > 0 && matricula <= Funcionario.contador)
                    {
                        if (f[matricula - 1].Trabalhando)
                        {
                            controle.Saida(f[matricula - 1]);
                        }

                        else
                        {
                            Console.WriteLine("\nFuncionário não está em serviço - Pressione qualquer tecla para sair");
                        }
                    }

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

                    break;

                case 4:
                    for (int i = 1; i < Funcionario.contador; i++)
                    {
```

```
                        Console.WriteLine("FUNCIONÁRIO: {0}\nCARGO: {1}", f[i - 1].Nome, f[i-1].ToString());
                        for (int x = 0; x < f[i - 1].historicoE.Count; x++)
                        {
                            Console.WriteLine("\nENTRADA: {0}", f[i - 1].historicoE.ElementAt(x));
                            Console.WriteLine("\nSAÍDA: {0}", f[i - 1].historicoS.ElementAt(x));
                        }

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

                    }
                    break;

                case 5:
                    Console.WriteLine("\nPressione qualquer tecla para sair");
                    break;

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

            Console.ReadKey();
        } while (op != 5);
    }
```

```
        static void Menu()
        {
            Console.WriteLine("MENU DE OPÇÕES");
            Console.WriteLine("\n1- Incluir novo Funcionário");
            Console.WriteLine("\n2- Registrar entrada do funcionário");
            Console.WriteLine("\n3- Registrar saída do funcionário");
            Console.WriteLine("\n4- Exibir histórico de ponto dos funcionários");
            Console.WriteLine("\n5- Sair do programa");
        }
    }
}
```

2.6.1.2 Classe 2

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

namespace Ex_31
{
    abstract class Funcionario
    {
        public static int contador = 1;
       protected string nome;
        public List<string> historicoE = new List<string>();
```

```
        public List<string> historicoS = new List<string>();
        protected bool trabalhando = false;

        public string Nome
        {
            get
            {
                return nome;
            }

            set
            {
                nome = value;
            }
        }

        public bool Trabalhando
        {
            get
            {
                return trabalhando;
            }

            set
            {
                trabalhando = value;
            }
```

```
        }

        public override string ToString()
        {
            string[] vetString;
            string x = base.ToString();

            vetString = x.Split('.');
            return vetString[1];
        }

    }
    }
```

2.6.1.3 Classe 3

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

namespace Ex_31
{
    class Gerente: Funcionario
    {
        string usuario, senha;

        public string Usuario
```

```
        {
            get
            {
                return usuario;
            }

            set
            {
                usuario = value;
            }
        }

        public string Senha
        {
            get
            {
                return senha;
            }

            set
            {
                senha = value;
            }
        }
    }
}
```

2.6.1.4 Classe 4

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

namespace Ex_31
{
    class Telefonista: Funcionario
    {
        int ramal;

        public int Ramal
        {
            get
            {
                return ramal;
            }

            set
            {
                if (value > 0)
                ramal = value;
            }
        }
    }
```

```
}
```

2.6.1.5 Classe 5

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

namespace Ex_31
{
    class ControleDePonto
    {
        public void Entrada (Funcionario f)
        {
            DateTime agora = DateTime.Now;
            string horario = agora.ToString();
            Console.WriteLine("ENTRADA: {0}", f.Nome);
            Console.WriteLine("DATA: {0}", horario);
            f.Trabalhando = true;
            f.historicoE.Add(horario);
        }

        public void Saida (Funcionario f)
        {
            DateTime agora = DateTime.Now;
            string horario = agora.ToString();
            Console.WriteLine("SAÍDA: {0}", f.Nome);
```

```
                Console.WriteLine("DATA: {0}", horario);
                f.Trabalhando = false;
                f.historicoS.Add(horario);
            }
        }
    }
```

2.7 Ex3.2

2.7.1 Código fonte Ex3.2

2.7.1.1 Classe 1

```
// programa: Program.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para imprimir extratos de uma conta escolhida pelo usuario
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Banco
{
    class Program
    {
        static GeradorDeExtrato extratos;
        static void ContaCorrente()
        {
```

```
            ContaCorrente corrente = new ContaCorrente();
            corrente.Saldo = 1000;
            extratos.ImprimeExtratoBasico(corrente);
        }
        static void ContaPoupanca()
        {
            ContaPoupanca poupanca = new ContaPoupanca();
            poupanca.Saldo = 2000;
            extratos.ImprimeExtratoBasico(poupanca);
        }
        static void Main(string[] args)
        {
            extratos = new GeradorDeExtrato();
            int opcao, repetidor = 1;
            while(repetidor !=0)
            {
                Console.Write("Menu:\n1-Conta Corrente\n2-Conta Poupança\n3-
Sair\nQual a opção desejada:");
                opcao = int.Parse(Console.ReadLine());
                Console.Clear();
                switch(opcao)
                {
                    case 1:
                        ContaCorrente();
                        Console.ReadKey();
                        Console.Clear();
                        break;
```

```
                    case 2:
                        ContaPoupanca();
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case 3:
                        repetidor = 0;
                        break;
                    default:
                        Console.Write("Erro, opção invalida, digite algo para
continuar.");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                }
            }
        }
    }
}
```

2.7.1.2 Classe 2

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


namespace Banco
```

```
{
    class Conta
    {
        public double Saldo
        {
            set;
            get;
        }
    }
}
```

2.7.1.3 Classe 3

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


namespace Banco
{
    class ContaCorrente : Conta
    {
        public double Limite
        {
            get;
            set;
        }
    }
```

```
}
```

2.7.1.4 Classe 4

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

namespace Banco
{
    class ContaPoupanca : Conta
    {
        public int DiaDoAniversario
        {
            get;
            set;
        }
    }
}
```

2.7.1.5 Classe 5

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

namespace Banco
```

```
{
    class GeradorDeExtrato
    {
        public void ImprimeExtratoBasico(Conta c)
        {
            DateTime agora = DateTime.Now;
            string horario = String.Format("{0:d/M/yyyy HH:mm:ss}", agora);
            System.Console.WriteLine("Data: " + horario);
            System.Console.WriteLine("Saldo: " + c.Saldo);
        }
    }
}
```

2.8 Ex3.3

2.8.1 Código fonte Ex3.3

2.8.1.1 Classe 1

```
//
// nome do programa: Ex33
//
// programador(es): Breno Vieira, Daniel Jorge, Robert Victor, Arthur Henrique e João Gontijo
// data: 22/10/2017
// entrada(s): Nao tem
// saida(s): Imprime dados da conta
// para testar digite:
// Ex33.cs
// descricao: Sistema de contas, criacao, deposito, saque extrato e relatorio de contas por numero e titular.
//
```

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

namespace ConsoleApplication5
{
    class Program
    {
        const int MAXCONTAS = 100; // número máximo de contas suportado
        public static Conta[] vetContas = new Conta[MAXCONTAS]; //vetor de contas
        public static int numContas = 0;

        static void Main(string[] args)
        {
            populaObjetodeContas();
            menu();
        }
        public static void menu()
        {
            Console.Clear();
            int op = 0, agencia,conta, tipoConta;
            string nome;
```

```
                Console.WriteLine("******** Menu ******** ");
                Console.WriteLine("1 - Criar uma nova conta");
                Console.WriteLine("2 - Excluir uma conta existente");
                Console.WriteLine("3 - Depositar em uma conta ");
                Console.WriteLine("4 - Sacar de em uma conta ");
                Console.WriteLine("5 - Imprimir o extrato de uma conta");
                Console.WriteLine("6 - Imprimir uma relação das contas existentes informando o número da conta e o nome do titular da conta");
                Console.WriteLine("7 - Sair do programa");
                Console.WriteLine("Digite a opcao desejada: ");
                op = int.Parse(Console.ReadLine());
                switch (op)
                {
                    case 1:
                        Console.Write("Nome do Cliente: ");
                        nome = Console.ReadLine();
                        Console.Write("Agencia: ");
                        agencia = int.Parse(Console.ReadLine());
                        Console.Write("Tipo conta: ");
                        tipoConta = int.Parse(Console.ReadLine());

                        /*
                         * Tipo conta = 1 => Conta Corrente
                         * Tipo conta = 2 => Conta Poupança
                         */
                        if (tipoConta == 1)
                        {
```

```
                    numContas = 1;
                    while (Conta.verificaExistenciaConta(vetContas, numContas))
                    {
                        numContas++;
                        Console.WriteLine("Ja existe, virou "+numContas);


                    }
                    ContaCorrente contaCorrente = new
ContaCorrente(nome,agencia,numContas,tipoConta, true);
                    vetContas[numContas-1] = contaCorrente;




                }
                else if (tipoConta == 2)
                {
                    numContas = 1;
                    while (Conta.verificaExistenciaConta(vetContas, numContas))
                    {
                        numContas++;
                    }
                    ContaPoupanca contaPoupanca = new ContaPoupanca(nome,
agencia, numContas, tipoConta, true);
                    vetContas[numContas-1] = contaPoupanca;
                }
                else
                {
                    Console.WriteLine("Digite um tipo de conta Válido!");
                    Console.WriteLine("1 - Conta Corrente");
```

```
            Console.WriteLine("2 - Conta Poupança");
            voltaMenu();
        }
        Console.WriteLine("Conta criada com sucesso!");
        voltaMenu();
        break;
    case 2:
        Console.WriteLine("Digite o numero da conta para excluíla");
        conta = int.Parse(Console.ReadLine());
        if (Conta.verificaExistenciaConta(vetContas, conta))
        {
            excluiConta(conta);
            Console.WriteLine("Conta excluída com sucesso!");
        }
        else
        {
            Console.WriteLine("Conta inexistente!");
        }
        voltaMenu();
        break;
    case 3:
        double valor;
        Console.WriteLine("Digite o numero da conta:");
        conta = int.Parse(Console.ReadLine());
        if (Conta.verificaExistenciaConta(vetContas, conta))
        {
            Console.WriteLine("Digite um valor para depositar");
```

```
                    valor = double.Parse(Console.ReadLine());
                    if (valor > 0)
                    {
                        vetContas[conta - 1].Deposita(valor);
                        if (vetContas[conta - 1].TipoConta == 1)
                        {
                            vetContas[conta - 1].taxaPorOperacao(0.45);
                        }
                        Console.WriteLine("Deposito realizado!");
                    }
                    else
                    {
                        Console.WriteLine("Valor invalido!");
                    }
                }
                else
                {
                    Console.WriteLine("Conta inexistente!");
                }
                voltaMenu();
                break;
            case 4:
                Console.WriteLine("Digite o numero da conta");
                conta = int.Parse(Console.ReadLine());
                if (Conta.verificaExistenciaConta(vetContas, conta))
                {
                    Console.WriteLine("Digite o valor a ser sacado:");
```

```
                    valor = double.Parse(Console.ReadLine());
                    if (valor <= 0 || valor > vetContas[conta -1].Saldo)
                    {
                        Console.WriteLine("Você nao pode sacar esse valor!");
                    }
                    else
                    {
                        vetContas[conta - 1].Saca(valor);
                        if (vetContas[conta - 1].TipoConta == 1)
                        {
                            vetContas[conta - 1].taxaPorOperacao(0.45);
                        }
                        Console.WriteLine("Saque realizado");
                    }
                }
                else
                {
                    Console.WriteLine("Conta inexistente");
                }
                voltaMenu();
                break;
            case 5:
                Console.WriteLine("Digite o numero da conta");
                conta = int.Parse(Console.ReadLine());
                if (Conta.verificaExistenciaConta(vetContas, conta))
                {
                    GeradorDeExtrato geraExtrato = new GeradorDeExtrato();
```

```
                                        if (vetContas[conta - 1].TipoConta == 1)
                    {
                        vetContas[conta - 1].taxaPorOperacao(0.45);
                    }
                    geraExtrato.GeraExtrato(vetContas[conta - 1]);
                }
                else
                {
                    Console.WriteLine("Conta inexistente");
                }
                voltaMenu();
                break;
            case 6:
                imprimeRelacaoContasExistentes();
                voltaMenu();
                break;
            case 7:
                Environment.Exit(1);
                break;
            default:
                Console.WriteLine("Opcao invalida!");
                voltaMenu();
                break;
        }
    }
    static void voltaMenu()
    {
```

```
                Console.WriteLine("Aperte qualquer tecla para voltar ao menu!");
                Console.ReadKey();
                menu();
            }
            public static void imprimeRelacaoContasExistentes()
            {
                int cont = 0;
                for (int i = 0; i <= numContas; i++)
                {
                    if (vetContas[i] != null)
                    {
                        cont++;
                        Console.WriteLine("Numero: {0}\tTitular: {1}",
vetContas[i].NumConta, vetContas[i].Nome);
                    }
                }
                if (cont == 0)
                {
                    Console.WriteLine("Nao existe nenhuma conta");
                }
            }
            public static void populaObjetodeContas()
            {
                string dadosLinha = "";
                string line = "";
                string[] arrayDados = new string[5];
                if (Directory.Exists(Conta.path))
```

```
            {
                string[] arquivos = Directory.GetFiles(Conta.path);
                if (arquivos.Length != 0)
                {
                    foreach (var arq in arquivos)
                    {
                        StreamReader leArq = new StreamReader(arq);
                        while ((line = leArq.ReadLine()) != null)
                        {
                            dadosLinha = line;
                            arrayDados = dadosLinha.Split(';');
                            if (int.Parse(arrayDados[3]) == 1)
                            {
                                vetContas[int.Parse(arrayDados[2]) - 1] = new
ContaCorrente(arrayDados[0],int.Parse(arrayDados[1]),int.Parse(arrayDados[2]),int.P
arse(arrayDados[3]), double.Parse(arrayDados[4]));
                            }
                            else
                            {
                                vetContas[int.Parse(arrayDados[2]) - 1] = new
ContaPoupanca(arrayDados[0], int.Parse(arrayDados[1]), int.Parse(arrayDados[2]),
int.Parse(arrayDados[3]), double.Parse(arrayDados[4]));
                            }
                            numContas++;
                        }
                        leArq.Close();
                    }
                }
```

```
        }
    }

    public static void excluiConta(int conta)
    {
        vetContas[conta - 1] = null;
        string dadosLinha = "";
        string[] arrayDados = new string[5];
        if (Directory.Exists(Conta.path))
        {
            string[] arquivos = Directory.GetFiles(Conta.path);
            string line = "";
            if (arquivos.Length != 0)
            {
                foreach (var arq in arquivos)
                {
                    StreamReader leArq = new StreamReader(arq);
                    string[] novoArq = new string[File.ReadAllLines(arq).Length];
                    int count = 0;
                    while ((line = leArq.ReadLine()) != null)
                    {
                        dadosLinha = line;
                        arrayDados = dadosLinha.Split(';');
                        if (arrayDados[2] != Convert.ToString(conta))
                        {
                            novoArq[count] = dadosLinha;
```

```
                        count++;
                    }
                }
                leArq.Close();
                StreamWriter grava = new StreamWriter(arq);
                for (int i = 0; i < novoArq.Length; i++)
                {
                    if (novoArq[i] != null)
                    {
                        grava.WriteLine(novoArq[i]);
                    }
                }
                grava.Close();
            }
        }
    }
    populaObjetodeContas();
  }
 }
}
```

2.8.1.2 Classe 2

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

```
namespace ConsoleApplication5
{
    abstract class Conta : IConta
    {
        protected string nome;
        protected int agencia, conta, tipoConta;
        protected double saldo = 0;
        public static string path = @"Contas\";

        public string Nome
        {
            get
            {
                return nome;
            }
            set
            {
                nome = value;
            }
        }
        public int NumConta
        {
            get
            {
                return conta;
            }
```

```
        set
        {
            conta = value;
        }
    }

    public int Agencia
    {
        get
        {
            return agencia;
        }
        set
        {
            agencia = value;
        }
    }

    public double Saldo
    {
        get
        {
            return saldo;
        }
        set
        {
            saldo = value;
```

```
            }
        }
        public int TipoConta
        {
            get
            {
                return tipoConta;
            }
            set
            {
                tipoConta = value;
            }
        }
        public void CriaArquivo()
        {
            if (!Directory.Exists(@"Contas"))
            {
                Directory.CreateDirectory(@"Contas");
            }
            string path = @"Contas\agencia" + agencia + ".txt";
            string content = nome + ";" + agencia + ";" + conta + ";" + tipoConta +
";" + saldo;
            StreamWriter escreve = new StreamWriter(path, true);
            if (!File.Exists(path))
            {
                File.Create(path);
                escreve.WriteLine(content);
```

```
            }
            else
            {
                escreve.WriteLine(content);
            }
            escreve.Close();
        }

        public void Deposita(double valor)
        {
            this.saldo += valor;
            alteraDadosArquivo();
        }

        public void Saca(double valor)
        {
            saldo -= valor;
            alteraDadosArquivo();
        }
        public static bool verificaExistenciaConta(Conta[] vetConta, int conta)
        {
            if (vetConta[conta - 1] != null)
            {
                return true;
            }
            return false;
        }
```

```
public void taxaPorOperacao(double valor)
{
    saldo -= valor;
    alteraDadosArquivo();
}

private void alteraDadosArquivo()
{
    string dadosLinha = "";
    string[] arrayDados = new string[5];
    if (Directory.Exists(Conta.path))
    {
        string[] arquivos = Directory.GetFiles(Conta.path);
        string line = "";
        if (arquivos.Length != 0)
        {
            foreach (var arq in arquivos)
            {
                StreamReader leArq = new StreamReader(arq);
                string[] novoArq = new string[File.ReadAllLines(arq).Length];
                int count = 0;

                while ((line = leArq.ReadLine()) != null)
                {
                    dadosLinha = line;
                    arrayDados = dadosLinha.Split(';');
```

```
                            if (arrayDados[2] != Convert.ToString(conta))
                            {
                                novoArq[count] = dadosLinha;
                            }
                            else
                            {
                                novoArq[count] = arrayDados[0] + ";" + arrayDados[1] + ";"
+ arrayDados[2] + ";" + arrayDados[3] + ";" + saldo;
                            }
                            count++;
                        }
                        leArq.Close();
                        StreamWriter grava = new StreamWriter(arq);
                        for (int i = 0; i < novoArq.Length; i++)
                        {
                            if (novoArq[i] != null)
                            {
                                grava.WriteLine(novoArq[i]);
                            }
                        }
                        grava.Close();
                    }
                }
            }
        }
    }
```

```
}
```

2.8.1.3 Classe 3

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


namespace ConsoleApplication5
{
    class ContaCorrente : Conta, IConta
    {
        public ContaCorrente(string nome, int agencia, int conta, int tipoConta, bool novaConta)
        {
            this.nome = nome;
            this.agencia = agencia;
            this.conta = conta;
            this.tipoConta = tipoConta;
            CriaArquivo();
        }
        public ContaCorrente(string nome, int agencia, int conta, int tipoConta, double saldo)
        {
            this.nome = nome;
            this.agencia = agencia;
            this.conta = conta;
```

```
                this.tipoConta = tipoConta;
                this.saldo = saldo;
            }
        }
    }
```

2.8.1.4 Classe 4

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


    namespace ConsoleApplication5
    {
        class ContaPoupanca : Conta, IConta
        {
            public ContaPoupanca(string nome, int agencia, int conta, int tipoConta,
bool novaConta)
            {
                this.nome = nome;
                this.agencia = agencia;
                this.conta = conta;
                this.tipoConta = tipoConta;
                CriaArquivo();
            }
            public ContaPoupanca(string nome, int agencia, int conta, int tipoConta,
double saldo)
```

```
        {
            this.nome = nome;
            this.agencia = agencia;
            this.conta = conta;
            this.tipoConta = tipoConta;
            this.saldo = saldo;
        }
    }
}
```

2.8.1.5 Classe 5

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

namespace ConsoleApplication5
{
    class GeradorDeExtrato
    {
        public void GeraExtrato(IConta c)
        {
            System.Console.WriteLine(" EXTRATO ");
            System.Console.WriteLine(" SALDO : " + c.Saldo);
        }
    }
}
```

2.8.1.6 Interface 1

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


namespace ConsoleApplication5
{
    interface IConta
    {
        void Deposita(double valor);
        void Saca(double valor);
        double Saldo { get; set; }
        void CriaArquivo();
    }
}
```

2.9 Ex3.4

2.9.1 Código fonte Ex3.4

2.9.1.1 Classe 1

```
// Programa: Exercício 3.4 do LAB 3
// Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.
// Data: 19/10/2017
// Descrição: Programa que Simula dois tipo de toque, analógico e digital. Usando Override, Polimorfismo e Herança
using System;
using System.Collections.Generic;
```

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


namespace Ex3._4_LAB_3
{
    class Program
    {
        static void Main(string[] args)
        {


            Console.Write(" Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.\n\n");


            cTelefoneEletronico TelefoneEletronico = new cTelefoneEletronico(); // Objetos
            cTelefone Telefone = new cTelefone();


            Telefone.Ring(); // Chamar métodos
            TelefoneEletronico.Ring();


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

2.9.1.2 Classe 2

```
// Programa: Exercício 3.4 do LAB 3
```

```
// Programadores: João Gontijo, Daniel Jorge, Arthur Henrique, Robert Victor, Breno Vieira.

// Data: 19/10/2017

// Descrição: Programa que Simula dois tipo de toque, analógico e digital. Usando Override, Polimorfismo e Herança

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Ex3._4_LAB_3

{

    class cTelefone

    {

        protected string tipoDoTelefone = "Analógico"; // Atributo


        public virtual void Ring()

        {

            Console.Write(" Tocando o {0}\n", tipoDoTelefone); // Método principal

        }

    }

}
```

2.9.1.3 Classe 3

```
// Programa: Exercício 3.4 do LAB 3

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

// Data: 19/10/2017
```

```
// Descrição: Programa que Simula dois tipo de toque, analógico e digital. Usando Override, Polimorfismo e Herança

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace Ex3._4_LAB_3

{

    class cTelefoneEletronico : cTelefone

    {

        public cTelefoneEletronico()

        {

            tipoDoTelefone = "Digital"; // Construtor

        }


        public override void Ring()

        {

            Console.Write(" Tocando o {0}\n", tipoDoTelefone); // Método override Ring

        }

    }

}
```

2.10 **Ex3.5**

2.10.1 Código fonte Ex3.5

2.10.1.1 Classe 1

```
// programa: Ex3.5.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Programa principal para executar as operaçoes
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Ex3._5
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.");
            int num1, num2;
            int opcao = -1;


            while(opcao != 0)
            {
                Console.WriteLine("\t CALCULO \n 1 - SOMA \n 2 - SUBTRAÇAO \n 3 - MULTIPLICAÇÃO \n 4 - DIVISÃO \n OPÇAO:");
```

```
                opcao = int.Parse(Console.ReadLine());

                if(opcao == 1)
                {
                    Console.WriteLine("Entre com o primeiro numero:");
                    num1 = int.Parse(Console.ReadLine());
                    Console.WriteLine("Entre com o segundo numero:");
                    num2 = int.Parse(Console.ReadLine());

                    Soma soma = new Soma();

                    Console.WriteLine("Resultado:" + soma.Calcula(num1, num2));

                    Console.ReadKey();
                    Console.Clear();
                }
                else if(opcao == 2)
                {
                    Console.WriteLine("Entre com o primeiro numero:");
                    num1 = int.Parse(Console.ReadLine());
                    Console.WriteLine("Entre com o segundo numero:");
                    num2 = int.Parse(Console.ReadLine());

                    Subtraçao subtraçao = new Subtraçao();

                    Console.WriteLine("Resultado:"      +      subtraçao.Calcula(num1,
num2));
```

```
                    Console.ReadKey();
                    Console.Clear();
                }
                else if(opcao == 3)
                {
                    Console.WriteLine("Entre com o primeiro numero:");
                    num1 = int.Parse(Console.ReadLine());
                    Console.WriteLine("Entre com o segundo numero:");
                    num2 = int.Parse(Console.ReadLine());

                    Multiplicacao multiplicaçao = new Multiplicacao();

                    Console.WriteLine("Resultado:"    +    multiplicaçao.Calcula(num1,
num2));

                    Console.ReadKey();
                    Console.Clear();
                }
                else if (opcao == 4)
                {
                    Console.WriteLine("Entre com o primeiro numero:");
                    num1 = int.Parse(Console.ReadLine());
                    Console.WriteLine("Entre com o segundo numero:");
                    num2 = int.Parse(Console.ReadLine());
                    if ((num1 / num2 == 0) && (num1 != 0 && num2 != 0))
                    {
```

```
                    Console.WriteLine("ERRO DIVISAO INVALIDA");
                }
                else
                {
                    Divisao divisao = new Divisao();

                    Console.WriteLine("Resultado:" + divisao.Calcula(num1, num2));

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

2.10.1.2 Classe 2

```
// programa: Divisao.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe com funçao para divisao
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Ex3._5
{
    class Divisao : IOperacaoMatematica
    {
        public Divisao()
        {

        }
        public int Calcula(int a, int b)
        {
            return  ( a / b );
        }
    }
}
```

2.10.1.3 Classe 3

```
// programa: Multiplicaçao.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe com a funçao de multiplicar
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Ex3._5
{
    class Multiplicacao:IOperacaoMatematica
    {
        public Multiplicacao()
        {

        }

        public int Calcula(int a, int b)
        {
            return (a*b);
        }
    }
}
```

2.10.1.4 Classe 4

```
// programa: Soma.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe com a funçao de somar
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Ex3._5
{
    class Soma : IOperacaoMatematica
    {
        public Soma()
        {


        }
        public int Calcula(int a, int b)
        {
            return  (a + b);
        }
    }
}
```

2.10.1.5 Classe 5

```
// programa: subtraçao.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: Classe com a funçao de subtrair
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Ex3._5
```

```
{
    class Subtraçao:IOperacaoMatematica
    {
        public Subtraçao()
        {

        }

        public int Calcula(int a, int b)
        {
            return (a - b);
        }
    }
}
```

2.10.1.6 Interface

```
// programa: IOperacaoMatematica.cs
// programadores: Daniel Jorge,Robert Victor,Breno Vieira,Arthur Henrique,João Gontijo.
// data: 19/10/2017
// Descrição: interfaçe que delimita os metodos a serem trabalhados nas classes
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ex3._5
```

```
{
    interface IOperacaoMatematica
    {
        int Calcula(int a, int b);
    }
}
```

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.