



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Graduação em Sistemas de Informação

Bryan Diniz Rodrigues
Luiz Henrique Gomes Guimarães
Thais Barcelos Lorentz

**RELATÓRIO DO LABORATÓRIO 1 DE PROGRAMAÇÃO ORIENTADA POR
OBJETOS**

Belo Horizonte
2019/2

SUMÁRIO

| | |
|--|-----------|
| 1. INTRODUÇÃO | 2 |
| 1.1 OBJETIVO | 2 |
| 2. CLASSES | 2 |
| 2.1 OBJETO | 3 |
| 2.2 MÉTODOS | 3 |
| 2.3 TRY-CATCH (TRATAMENTO DE ERROS) | 5 |
| 2.4 GET E SET COMO AUTO PROPERTIES | 6 |
| 2.5 INTERFACE ICOMPARABLE | 7 |
| EXERCÍCIO 3.1 CALCULADORA STATIC | 8 |
| EXERCÍCIO 3.2 VETOR NA ORDEM, MAIOR E MENOR VALOR E MÉDIA | 14 |
| EXERCÍCIO 3.3 HAMMING DISTANCE ARRAY | 18 |
| EXERCÍCIO 3.4 VOTAÇÃO PRIMEIRO E SEGUNDO TURNO | 24 |
| EXERCÍCIO 3.5 CONVERSOR DE TEMPERATURAS | 40 |
| EXERCÍCIO 5.1 CALCULADORA CLASSE | 46 |
| EXERCÍCIO 5.5 CONVERSOR DE TEMPERATURA COM CLASSE | 53 |
| EXERCÍCIO 5.6 CONVERSOR DE TEMPERATURA COM MENU E CLASSE | 57 |
| EXERCÍCIO 5.7 CONTA, DEPÓSITO E SAQUE | 66 |
| 3. CONCLUSÃO | 76 |
| 4. REFERÊNCIAS | 77 |

RESUMO

1. INTRODUÇÃO

Este relatório trata da primeira lista de exercícios da disciplina Laboratório de Programação Orientada por objetos tendo o foco de colocar em prática o conteúdo ensinado no primeiro período da disciplina Laboratório de Algoritmos e Técnicas de Programação e introdução de Classes.

Está presente neste relatório os código dos programas desenvolvidos junto a explicações sobre os mesmos.

1.1 OBJETIVO

Este relatório tem por objetivo abordar todos os conhecimentos que foram usados para desenvolver os programas propostos e aperfeiçoar habilidades que foram adquiridas. Ao longo deste documento procuramos deixar bem detalhado.

2. CLASSES

Uma Classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. Sendo assim uma especificação para a criação de um objeto na memória do computador. Serve como modelo para armazenar essas informações, realizar tarefas. Um sistema completo é composto, geralmente, por muitas classes, que são copiadas na memória do computador durante a execução do programa. Essa cópia é feita na memória do computador no momento em que o programa está sendo executado chama-se objeto.

Uma classe é definida pela palavra reservada `class` e é composta por atributos e métodos:

Atributos de uma classe também conhecido como propriedades, descrevem um intervalo de valores que as instâncias da classe podem apresentar. Um atributo

é uma variável que pertence a um objeto. Os dados de um objeto são armazenados nos seus atributos. Informações sobre o objeto. Dados que posso armazenar.

Os métodos são procedimentos ou funções que realizam as ações próprias do objeto. Assim, os métodos são as ações que o objeto pode realizar. Tudo o que o objeto faz é através de seus métodos, pois é através dos seus métodos que um objeto se manifesta, através deles que o objeto interage com os outros objetos. Sendo mais conhecidos como: Método Construtor, Métodos Get e Set, Métodos do usuário e Método sobrescrito.

Exemple de declaração de uma classe:

| | | | |
|-----------------------------|---|-------------|--------|
| Nome da classe | → | ROUPA | |
| | | | |
| | | - cor: | string |
| Características / Atributos | → | - tamanho: | string |
| | | - textura: | string |
| | | - material: | string |
| | | | |
| Funções / Métodos | → | + vestir() | |

2.1 OBJETO

Objetos são instâncias de classes, que determinam qual informação um objeto contém e como ele pode manipulá-la. É uma entidade capaz de reter um estado (informação) e que oferece uma série de informações (comportamento) ou para examinar ou para afetar este estado. É através deles que praticamente todo o processamento ocorre em sistemas implementados com linguagens de programação orientada a objetos. Ou seja, um objeto é uma entidade lógica que contém dados e o código para manipular esses dados. Objetos são “instanciados” ou criados de uma classe.

2.2 MÉTODOS

O conjunto de funcionalidades da classe. Para cada método, especifica-se sua assinatura, composta por:

- nome: um identificador para o método.
- tipo: quando o método tem um valor de retorno, o tipo desse valor.
- lista de argumentos: quando o método recebe parâmetros para sua execução, o tipo e um identificador para cada parâmetro.
- visibilidade: como para atributos, define o quão visível é um método a partir de objetos de outras classes.

MÉTODO CONSTRUTOR

Um método construtor é responsável por instanciar (criar) e inicializar os atributos de um objeto.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5
6  namespace ExemploClasses
7  {
8      class Pessoa
9      {
10         private String nome;
11         private String rg;
12         private int idade;
13
14         public Pessoa() { }
15
16         public Pessoa(String nome, String rg, int idade)
17         {
18             this.nome = nome;
19             this.rg = rg;
20             this.idade = idade;
21         }
22     }
23 }
```

As linhas 14 e 16 declaram os métodos construtores. Todo método construtor deve ser public, não possuir nenhum retorno e ter o mesmo nome da classe.

Criamos dois métodos construtores, um vazio (linha 14) e um recebendo todos os parâmetros para inicializar o objeto (linha 16). As boas práticas de orientação a

objetos recomenda que seja criado um método construtor vazio e um ou vários métodos construtores com os parâmetros mais importantes para inicializar o objeto.

MÉTODOS ESTÁTICOS

Os métodos estáticos, quando declaramos, não necessitam que uma classe seja instanciada para que possa ser utilizado, ou seja, um método estático, ele pode ser utilizado juntamente com a classe Main();

2.3 TRY-CATCH (TRATAMENTO DE ERROS)

Um tratamento de exceções que será usado em todos os códigos desse relatório, ele funciona de forma que se inicia pelo comando try e se encerra com o comando catch. Entre esses dois comandos há um bloco entre chaves denominado área protegida, em que qualquer exceção será tratada.

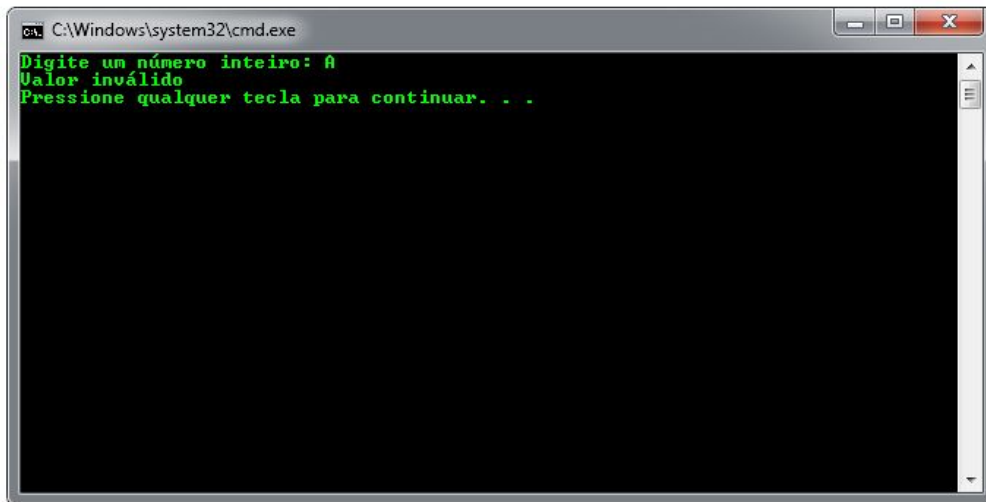
Segue abaixo um exemplo do uso do Try-Catch:

```
7      try
8      {
9          Console.WriteLine("Digite um número inteiro: ");
10         int num = int.Parse(Console.ReadLine());
11
12         Console.WriteLine("O número digitado foi: " + num);
13     }
14     catch
15     {
16         Console.WriteLine("Valor inválido");
17     }
18 }
19 }
```

A mensagem presente dentro do catch será retornada em alguns casos, como;

Se no campo ReadLine () for digitado uma palavra, um caractere ou até mesmo um número não inteiro.

SAÍDA:



Sem esse tratamento de exceção, simplesmente seria emitido uma mensagem encerrando o programa.

2.4 GET E SET COMO AUTO PROPERTIES

Essa forma de adicionar o get e set como uma propriedade diretamente do atributo, facilita na implementação das mesmas, e tornando ainda mais fácil a utilização desses atributos, já que não será preciso utilizar variantes de nomes.

Abaixo podemos ver a sintaxe dessa propriedade.

```
5 public int Idade { get; private set; }
```

Dessa forma temos um get e set, mas com um set sendo privado, então só é possível usar o get em outras classes.

Uma desvantagem de utilizar essas propriedades é que elas não aceitam lógicas, por exemplo: Só aceitar uma idade se o valor for > 0 , sendo assim, terá que se optar por outras formas de implementar um get e set.

Foi utilizado essa propriedade no Exercício 3.4.

2.5 INTERFACE ICOMPARABLE

É uma interface que auxilia na comparação entre objetos de uma classe, nela há um método que deve ser implementado na classe chamado `CompareTo` que será desenvolvido uma lógica para a comparação ordenação.

Em nosso programa a interface é utilizada para fazer o ordenamento dos candidatos após as votações, primeiramente pelos números de votos e em caso de números de votos iguais a idade do candidato entrará para desempatar.

Abaixo está a classe `Candidato` e seu método `CompareTo`:

```
class Candidato : IComparable<Candidato>
{
    public string Nome { get; private set; }
    public int Numero { get; private set; }
    public int Votos { get; private set; }
    public int Idade { get; private set; }
```

```
public int CompareTo(Candidato cand)
{
    // Se o número de votos for igual então faz a ordenação de acordo com a maior idade
    if (this.Votos == cand.Votos)
    {
        return cand.Idade.CompareTo(this.Idade);
    }
    // Ordenação padrão : do maior número de votos para o menor
    return cand.Votos.CompareTo(this.Votos);
}
```

Após ser aplicado essa interface, podemos utilizar o método `Array.Sort()` para ordenar um possível vetor de objetos do tipo `Candidato`.

EXERCÍCIO 3.1 CALCULADORA STATIC

ENUNCIADO

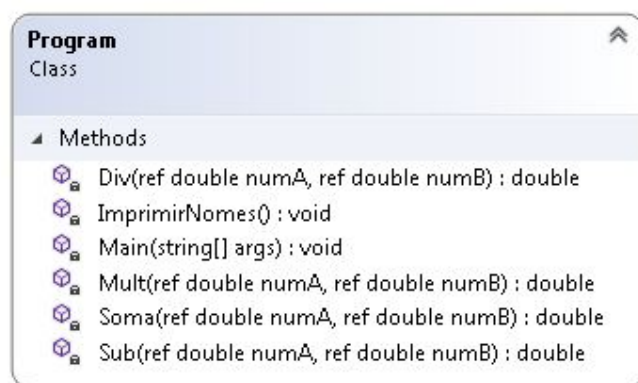
Fazer um programa codificado em C#, usando obrigatoriamente o csc.exe, que implemente uma calculadora com as seguintes especificações:

- Deve efetuar as quatro operações aritméticas básicas (+, -, x e /) com números reais.
- Os operandos e o operador são recebidos obrigatoriamente através de parâmetros passados na linha de comando (Args[]).
- Deve implementar obrigatoriamente um método estático para cada operação aritmética básica (+, -, x e /).
- Deve-se fazer o tratamento de erros nos casos de parâmetros inválidos ou inadequados.
- Exemplo:

ENTRADA: ./calcs.exe 2.5 x 3

SAÍDA: 2 x 3 = 7.5

DIAGRAMA UML



CÓDIGO

```
//  
// nome do programa: Ex31.cs  
//  
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais  
Barcelos Lorentz  
// data: 09/08/2019  
// entrada(s): Operandos e operadores matemáticos  
// o número de matrícula (6 dígitos) e o nome completo do aluno  
// saída(s): imprime a operação e o resultado  
// para executar e testar digite:  
// Ex31.exe 9 x 7  
// descrição: Recebe dois números e um operador matemático (+, -, x ou *, /),  
e retorna essa operação  
// com seu respectivo resultado.  
//  
  
using System;  
using System.Collections.Generic;  
using System.Globalization;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
  
namespace Ex31  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            ImprimirNomes(); // chamada de metodo para mostrar nomes do  
integrantes do trabalho  
            try // try usado para prevenir erros ao receber parametros  
invalidos pelo args  
            {  
                double numA =  
double.Parse(args[0].ToString(CultureInfo.InvariantCulture));  
                char operador = char.Parse(args[1]);  
                double numB =  
double.Parse(args[2].ToString(CultureInfo.InvariantCulture));  
                if ((numB == 0) && ((operador == '/') || (operador == '÷'))) //  
impedir a entrada de divisão por zero  
                {  
                    Console.WriteLine("Impossível divisão por zero");  
                    System.Environment.Exit(0);  
                }  
  
                Console.Write("{0} {1} {2} = ", numA, operador,  
numB); // imprimir base para exibir operação  
                if (operador == '+') // chamando método somar  
                {  
                    Console.Write(Soma(ref numA, ref numB));  
                }  
                else if (operador == '-') // chamando método subtrair  
                {  

```

```

        Console.WriteLine(Sub(ref numA, ref numB));
    }
    else if ((operador == 'x') || (operador == '*')) // chamando
método multiplicar
    {
        Console.WriteLine(Mult(ref numA, ref numB));
    }
    else if ((operador == '/') || (operador == '÷')) // chamando
método divisão
    {
        Console.WriteLine(Div(ref numA, ref numB));
    }
    else Console.WriteLine("Operador Inválido");
    Console.WriteLine();
}
catch
{
    Console.WriteLine("Valores inseridos inválidos, tente algo como:
4 x 3");
}
}

static double Soma(ref double numA, ref double numB) // método somar
{
    return (numA + numB);
}

static double Sub(ref double numA, ref double numB) // metodo subtrair
{
    return (numA - numB);
}

static double Mult(ref double numA, ref double numB) // método
multiplicar
{
    return (numA * numB);
}

static double Div(ref double numA, ref double numB) // método divisão
{
    return (numA / numB);
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.WriteLine("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

EXPLICANDO O CÓDIGO

```
24 static void Main(string[] args)
25 {
26     ImprimirNomes(); // chamada de metodo para mostrar nomes do integrantes do trabalho
27
28     try // try usado para prevenir erros ao receber parametros invalidos pelo args
29     {
30         double numA = double.Parse(args[0].ToString(CultureInfo.InvariantCulture));
31         char operador = char.Parse(args[1]);
32         double numB = double.Parse(args[2].ToString(CultureInfo.InvariantCulture));
33     }
```

Como todos os comandos do programa será repassado por linha de comando Args[], da linha 30 a 32, será selecionado os campos do vetor Args, e armazenados em variáveis, double para os números operando e char para o operador.

Args[0] = primeiro operando

Args[1] = operador

Args[2] = segundo operando

Exemplo: Args[0] = 3; Args[1] = '+'; Args[2] = 7;

Nesse trecho do código é criado uma condição para que se, operador == '/', que é o representante da divisão no nosso código, e numB == 0, irá retornar uma mensagem de erro informando que é impossível a divisão por zero, e fechará o programa como o comando Environment.Exit(0) (linha 37);

```
34 if ((numB == 0) && ((operador == '/') || (operador == '÷'))) // impedir a entrada de divisao por zero
35 {
36     Console.WriteLine("Impossível divisão por zero");
37     System.Environment.Exit(0);
38 }
39
```

Nessa linha apenas será impressa a operação digitada pelo usuário.

```
40 Console.Write("{0} {1} {2} = ", numA, operador, numB); // imprimir base para exibir operacao
```

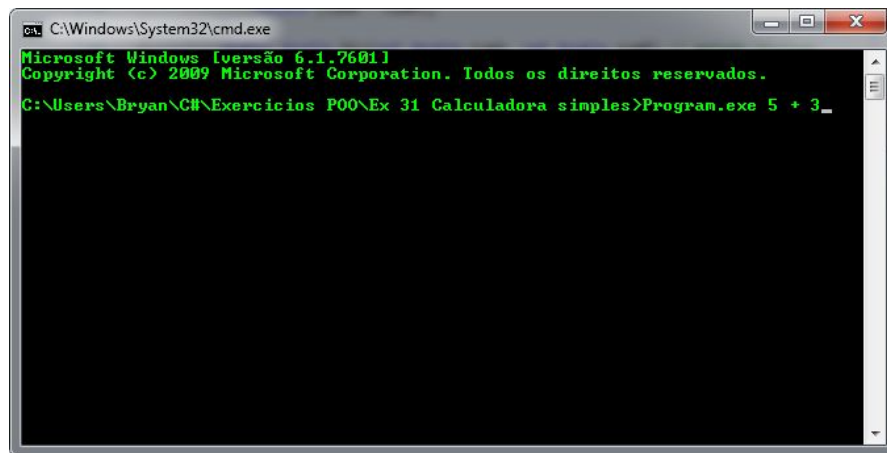
Agora de acordo com o operador, se for válido será chamado um método específico para cada tipo de operação.

```
42         if (operador == '+') // chamando metodo somar
43         {
44             Console.Write(Soma(ref numA, ref numB));
45         }
46         else if (operador == '-') // chamando metodo subtrair
47         {
48             Console.Write(Sub(ref numA, ref numB));
49         }
50         else if ((operador == 'x') || (operador == '*')) // chamando metodo multiplicar
51         {
52             Console.Write(Mult(ref numA, ref numB));
53         }
54         else if ((operador == '/') || (operador == '+')) // chamando metodo divisao
55         {
56             Console.Write(Div(ref numA, ref numB));
57         }
58         else Console.WriteLine("Operador Inválido");
59     }
```

Seguindo o código com os métodos citados acima, todos recebem como parâmetros, referências do primeiro e do segundo operando. Esses métodos retornarão o resultado de suas respectivas operações, completando o que será exibido para o usuário.

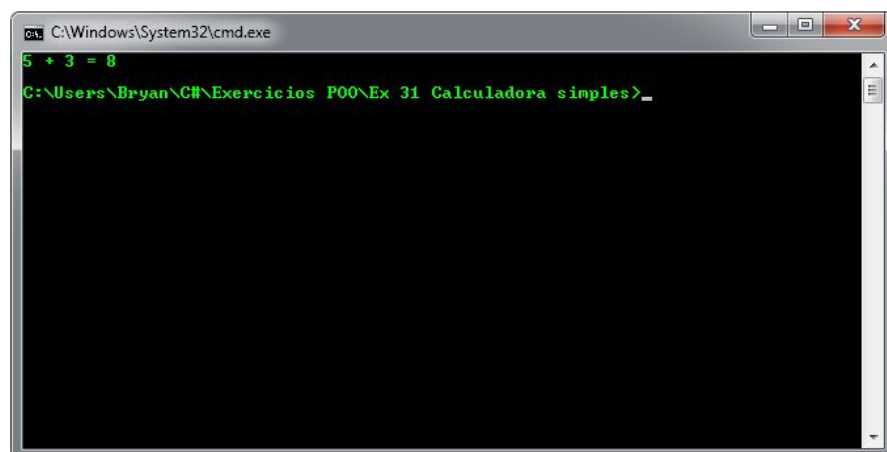
```
68     static double Soma(ref double numA, ref double numB) // metodo somar
69     {
70         return (numA + numB);
71     }
72     static double Sub(ref double numA, ref double numB) // metodo subtrair
73     {
74         return (numA - numB);
75     }
76     static double Mult(ref double numA, ref double numB) // metodo multiplicar
77     {
78         return (numA * numB);
79     }
80     static double Div(ref double numA, ref double numB) // metodo divisao
81     {
82         return (numA / numB);
83     }
```

ENTRADA:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Bryan\C#\Exercicios P00\Ex 31 Calculadora simples>Program.exe 5 + 3_
```

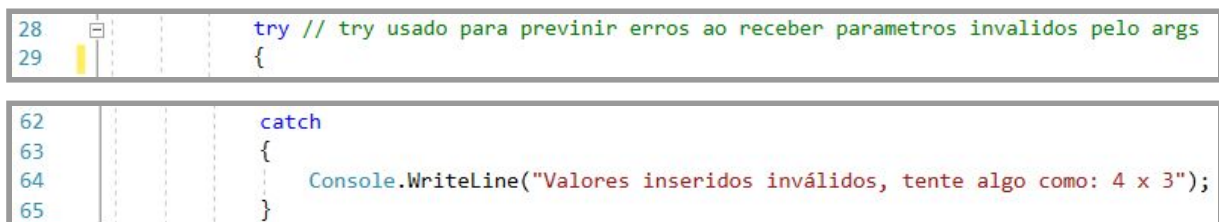
SAÍDA:



```
C:\Windows\System32\cmd.exe
5 + 3 = 8
C:\Users\Bryan\C#\Exercicios P00\Ex 31 Calculadora simples>_
```

TRATAMENTO DE ERROS

Try-Catch usado em caso de parâmetros inválidos recebidos e convertidos do Args[].



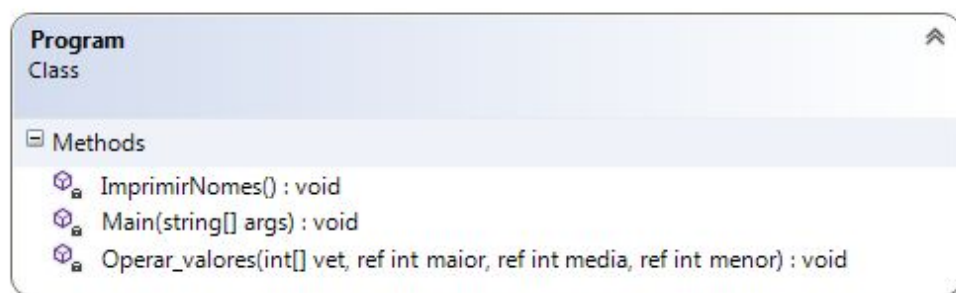
```
28 try // try usado para prevenir erros ao receber parametros invalidos pelo args
29 {
62 catch
63 {
64     Console.WriteLine("Valores inseridos inválidos, tente algo como: 4 x 3");
65 }
```

EXERCÍCIO 3.2 VETOR NA ORDEM, MAIOR E MENOR VALOR E MÉDIA

ENUNCIADO

Escreva um programa em C#, usando obrigatoriamente o csc.exe, que receba n números naturais através da linha de comando, armazene em um vetor e que chame um método estático que receba esses números inteiros (armazenados no vetor), ordene este vetor na ordem crescente de valores, e retorne: o vetor ordenado, o menor, o maior deles e a média de seus valores. Depois de acionar este método, deve-se imprimir o vetor de números ordenado, o menor, o maior e a média destes valores. Deve-se tratar erros comuns como parâmetros inválidos, inadequados ou inexistentes. Código exemplo de programa em C# com solução parcial (arquivo fonte disponível no SGA) Ex32_Lab1_s218p.cs

DIAGRAMA UML



CÓDIGO

```
//
// nome do programa: Ex32.cs
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
// Barcelos Lorentz
// data: 05/08/2019
// entrada(s): n numeros inteiros através da linha de comando
// o numero de matricula (6 dígitos)e o nome completo do aluno
// saída(s): imprime o vetor de números ordenado, o maior, e o menor
// para executar e testar digite:
// Ex32.exe 9 3 8 5
// descricao: recebe n numeros inteiros através da linha de comando,
// armazena em um vetor e chama um método estático que receba estes
// números inteiros (armazenados no vetor), e retorna o maior
// deles e a média de seus valores.
//
```

```

using System;
using System.Linq;
using System.Text;
namespace Ex32
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
            integrantes do trabalho
            try // try usado para prevenir erros ao receber parâmetros
            inválidos pelo args
            {
                int maior = 0, media = 0, menor = 0; // inicialização de
                variáveis para cálculo
                int i, j; // inicialização de variáveis contadoras
                j = args.Length; // numero de parametros recebidos na LC
                int[] vetor = new int[j]; // alocação de vetor para receber
                numeros inteiros
                Console.WriteLine("\nPrograma Ex32.exe com " + j + "
                parametros:\n");
                for (i = 0; i < j; i++)
                { // mostra e converte parâmetros recebidos na LC
                    vetor[i] = int.Parse(args[i]); // converte parâmetro recebidos
                    na LC de string para inteiro e armazena no vetor
                    Console.Write(" args[{0}] = {1}", i, args[i]);
                    Console.WriteLine(" vetor[{0}] = {1}", i, vetor[i]);
                }
                // mostra vetor
                Operar_valores(vetor, ref maior, ref media, ref menor); // chama
                metodo para fazer as operações solicitadas
                Console.WriteLine("\nMaior : " + maior); // mostra maior valor
                Console.WriteLine("Media : " + media); // mostra a média dos
                valores
                Console.WriteLine("\nMenor : " + menor); // falta implementar
                Console.WriteLine("Vetor : " + String.Join(" ", vetor)); //
                mostrar vetor ordenado
            }
            catch
            {
                Console.WriteLine("Parâmetros inválidos, para executar o
                programa digite Ex32.exe 9 3 8 5");
            }
        }

        static void Operar_valores(int[] vet, ref int maior, ref int media, ref
        int menor)
        {
            int soma = 0;
            media = 0;
            maior = vet[0];
            menor = vet[0];
            for (int i = 0; i < vet.Length; i++)
            {

```



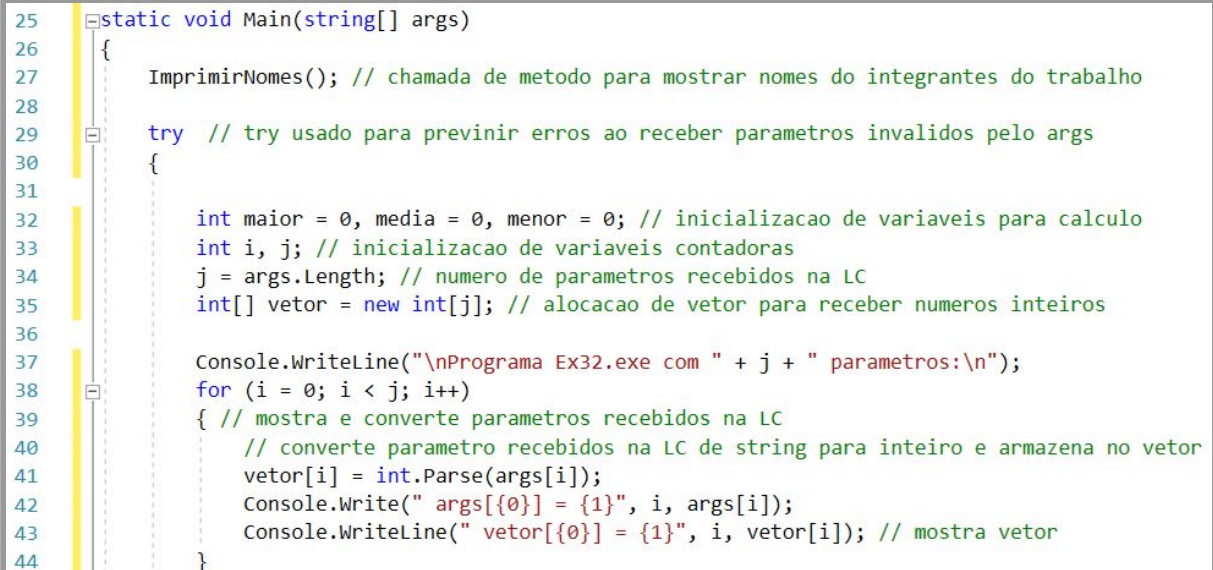
```

        soma += vet[i]; // acumular valores
        if (vet[i] > maior) // atualizar maior valor
        {
            maior = vet[i];
        }
        if (vet[i] < menor) // atualizar menor valor
        {
            menor = vet[i];
        }
    }
    Array.Sort(vet); // ordenar vetor
    media = soma / vet.Length; // calcular média dos valores
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.WriteLine("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

EXPLICANDO O CÓDIGO



```

25 static void Main(string[] args)
26 {
27     ImprimirNomes(); // chamada de metodo para mostrar nomes do integrantes do trabalho
28
29     try // try usado para prevenir erros ao receber parametros invalidos pelo args
30     {
31
32         int maior = 0, media = 0, menor = 0; // inicializacao de variaveis para calculo
33         int i, j; // inicializacao de variaveis contadoras
34         j = args.Length; // numero de parametros recebidos na LC
35         int[] vetor = new int[j]; // alocao de vetor para receber numeros inteiros
36
37         Console.WriteLine("\nPrograma Ex32.exe com " + j + " parametros:\n");
38         for (i = 0; i < j; i++)
39         { // mostra e converte parametros recebidos na LC
40             // converte parametro recebidos na LC de string para inteiro e armazena no vetor
41             vetor[i] = int.Parse(args[i]);
42             Console.WriteLine(" args[{0}] = {1}", i, args[i]);
43             Console.WriteLine(" vetor[{0}] = {1}", i, vetor[i]); // mostra vetor
44         }
45     }
46 }

```

Os comandos do programa serão repassados por linha de comando Args[], nas linha 34 a 35, será recebido o número total de parâmetros e definir o tamanho

do vetor. Após o tamanho do defino foi criado um for na linha 38 para que os valores digitados fossem salvos no vetor.

Um método foi criado para que através do vetor que foi preenchido, e de outras três variáveis que foram inicializadas com o valor 0 calcular o que foi pedido no exercício. Há um for que percorre por todo o vetor para somar todos as suas posições e dois if: um para achar o menor e outro para achar o maior valor do vetor.

```
55
56
57 static void Operar_valores(int[] vet, ref int maior, ref int media, ref int menor)
58 {
59     int soma = 0;
60     media = 0;
61     maior = vet[0];
62     menor = vet[0];
63     for (int i = 0; i < vet.Length; i++)
64     {
65         soma += vet[i]; // acumular valores
66         if (vet[i] > maior) // atualizar maior valor
67         {
68             maior = vet[i];
69         }
70         if (vet[i] < menor) // atualizar menor valor
71         {
72             menor = vet[i];
73         }
74     }
75     Array.Sort(vet); // ordenar vetor
76
77     media = soma / vet.Length; // calcular media dos valores
78 }
79
```

Nesta parte é chamado o método que faz as operações necessárias que printam na tela o maior, menor e média a partir dos números que o usuário digitou através da linha de comando.

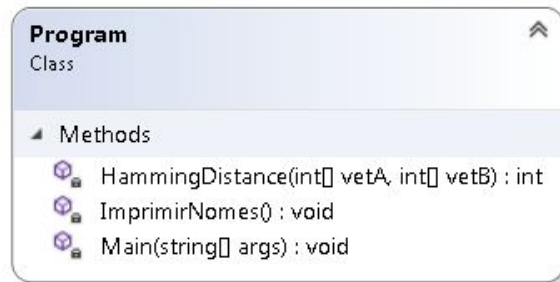
```
46 Operar_valores(vetor, ref maior, ref media, ref menor); // chama metodo para fazer as operacoes solicitadas
47 Console.WriteLine("\nMaior : " + maior); // mostra maior valor
48 Console.WriteLine("Media : " + media); // mostra media dos valores
49 Console.WriteLine("\nMenor : " + menor); // falta implementar
50 Console.WriteLine("Vetor : " + String.Join(" ", vetor)); // mostrar vetor ordenado
```

EXERCÍCIO 3.3 HAMMING DISTANCE ARRAY

ENUNCIADO

Escreva um programa em C#, usando obrigatoriamente o csc.exe, receba duas sequências de números inteiros através da linha de comando, as quais são separados pelo caractere flag "h", armazene em dois vetores e chame um método estático que receba estes vetores retorne a distância de Hamming entre tais vetores. Depois de acionar este método, deve-se imprimir os dois vetores e a distância de Hamming entre eles. Deve-se tratar erros comuns como parâmetros inválidos, inadequados ou inexistentes.

DIAGRAMA UML



CÓDIGO CLASS PROGRAM

```
//
// nome do programa:
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 15/08/2019
// entrada(s): 1 1 0 1 1 h 1 0 0 1 1
// o numero de matricula (6 digitos)e o nome completo do aluno
// saida(s): Distância de Hamming: 1
// para executar e testar digite: Program.exe 1 1 0 1 1 h 1 0 0 1 1
// descricao: o usuário entrará com duas sequências de inteiros separados por
um "h" e o programa
// retornará a distância de hamming entre esses dois vetores
//

using System;

namespace HammingDistance
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
            integrantes do trabalho
        }
    }
}
```

```

        int cont = 0;
        int tam = (args.Length - 1) / 2; // tamanho dos vetores
individuais
        if (args.Length % 2 == 0) // em caso de tamanhos diferentes entre
os vetores
        {
            Console.WriteLine("Os dois vetores devem ter o mesmo
tamanho");
            Environment.Exit(0);
        }
        try
        {
            int[] vetA = new int[tam]; // vetor que armazenará a
primeira sequência de inteiros
            int[] vetB = new int[tam]; // vetor que armazenará a
segunda sequência de inteiros

            for (int i = 0; i < args.Length; i++) // adicionando
elementos ao primeiro vetor
            {
                if (args[i] != "h")
                {
                    vetA[i] = int.Parse(args[i]);
                    Console.Write(vetA[i] + " ");
                    cont++;
                }
                else break;
            }

            Console.WriteLine();
            cont++; // conta para continuar contagem a partir de onde
terminou o for anterior
            for (int i = 0; i < vetB.Length; i++) // adicionando
elementos ao segundo vetor
            {
                vetB[i] = int.Parse(args[cont]);
                Console.Write(vetB[i] + " ");
                cont++;
            }
            Console.WriteLine("\nDistância de Hamming: " +
HammingDistance(vetA, vetB));
        }
        catch
        {
            Console.WriteLine("\nParâmetros inválidos, para executar o
programa digite: Program.exe 1 0 0 h 1 0 1");
        }
    }

    static int HammingDistance(int[] vetA, int[] vetB) // metodo para
calcular a distância de Hamming
    {
        int distance = 0;
        for (int i = 0; i < vetA.Length; i++)
        {
            if (vetA[i] != vetB[i]) // verificando diferenças entre os
elementos dos dois vetores
            {
                distance++;
            }
        }
        return distance;
    }
}

```

```

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.WriteLine("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

O QUE É A DISTÂNCIA DE HAMMING?

Antes de começarmos a explicar o programa, precisamos deixar claro o que é essa distância. É o número de posições (bits) em que as palavras diferem.

Exemplo: $p_1 = 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$
 $p_2 = 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1$

$\rightarrow h(p_1, p_2) = 3$

EXPLICANDO O PROGRAMA

Temos na linha 24 uma variável int cont que iremos usar para fazer uma contagem para popular os vetores, mais a frente.

Na linha 25 usamos uma variável para salvar o tamanho dos dois vetores que irá separar cada sequência para medir a distância de Hamming.

A partir da linha 27 temos uma estrutura de condição para testarmos se os dois vetores, antes e depois da tag flag 'h' tenham os mesmos tamanhos. Já que se o número de elementos inseridos no vetor args for par quer dizer que uma das sequências entre o 'h' é maior. Se retornar true, a mensagem de erro será exibida e o programa será fechado.

```

18 class Program
19 {
20     static void Main(string[] args)
21     {
22         ImprimirNomes();
23
24         int cont = 0;
25         int tam = (args.Length - 1) / 2;
26
27         if (args.Length % 2 == 0)
28         {
29             Console.WriteLine("Os dois vetores devem ter o mesmo tamanho");
30             Environment.Exit(0);
31         }
32

```

Na linha 35 e 36 nós instanciamos dois vetores para armazenar as duas sequências para a comparação. Já no for da linha 38, iremos percorrer o vetor args e salvar seus valores no vetA até acharmos a nossa tag 'h', ela sendo achada o for será quebrado na linha 46, mas antes temos usamos aquele contador declarado no início do código, o cont na linha 44.

```

33     try
34     {
35         int[] vetA = new int[tam];
36         int[] vetB = new int[tam];
37
38         for (int i = 0; i < args.Length; i++)
39         {
40             if (args[i] != "h")
41             {
42                 vetA[i] = int.Parse(args[i]);
43                 Console.Write(vetA[i] + " ");
44                 cont++;
45             }
46             else break;
47         }
48

```

Na linha 51 adicionamos mais 1 ao cont, por que queremos que no próximo for comece depois da tag 'h'. Já no for da linha 53 fazemos o mesmo do for anterior só que agora adicionando elementos ao. Na linha 60 chamamos o método e mostramos seu retorno.

```

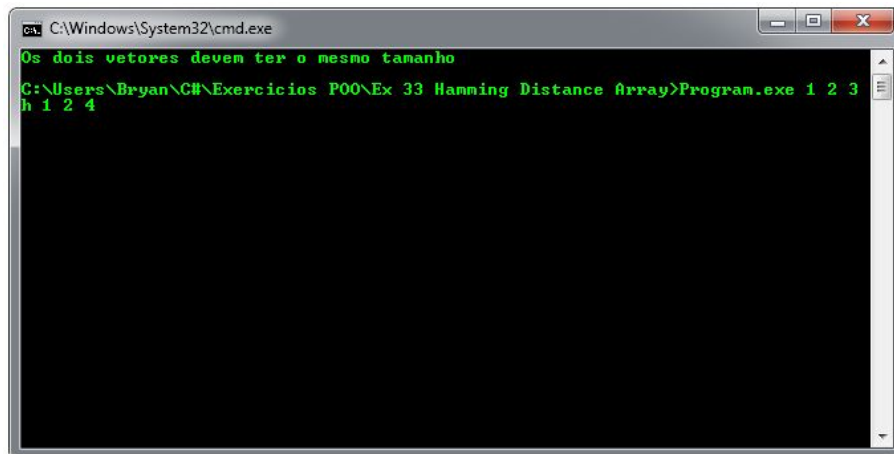
51         cont++;
52
53         for (int i = 0; i < vetB.Length; i++)
54         {
55             vetB[i] = int.Parse(args[cont]);
56             Console.Write(vetB[i] + " ");
57             cont++;
58         }
59
60         Console.WriteLine("\nDistância de Hamming: " + HammingDistance(vetA, vetB));
61     }

```

No método abaixo fazemos comparações entre os dois vetores, vetA e vetB, e sempre que os elementos dentro desses vetores forem diferentes um ao outro iremos adicionar +1 a nossa variável distance. No final é retornado a distância de hamming entre os dois vetores.

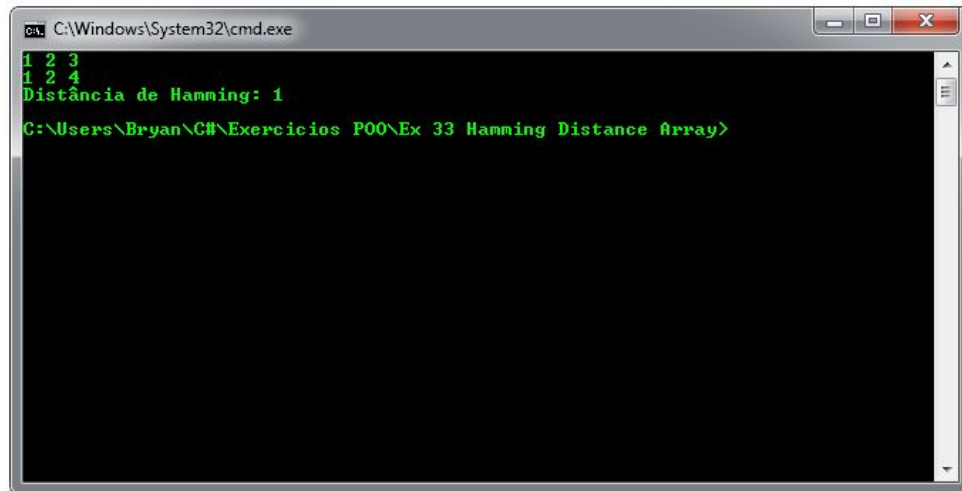
```
68 static int HammingDistance(int[] vetA, int[] vetB)
69 {
70     int distance = 0;
71
72     for (int i = 0; i < vetA.Length; i++)
73     {
74         if (vetA[i] != vetB[i])
75         {
76             distance++;
77         }
78     }
79     return distance;
80 }
```

ENTRADA:



```
C:\Windows\System32\cmd.exe
Os dois vetores devem ter o mesmo tamanho
C:\Users\Bryan\C#\Exercicios P00\Ex 33 Hamming Distance Array>Program.exe 1 2 3
h 1 2 4
```

SAÍDA:



```
cmd C:\Windows\System32\cmd.exe
1 2 3
1 2 4
Distância de Hamming: 1
C:\Users\Bryan\C#\Exercicios P00\Ex 33 Hamming Distance Array>
```


EXERCÍCIO 3.4 VOTAÇÃO PRIMEIRO E SEGUNDO TURNO

ENUNCIADO

Escreva um programa em C#, usando obrigatoriamente o csc.exe, que receba através da linha de comando o número de eleitores votantes e depois solicite e receba os votos para presidente dos eleitores votantes.

Os candidatos são:

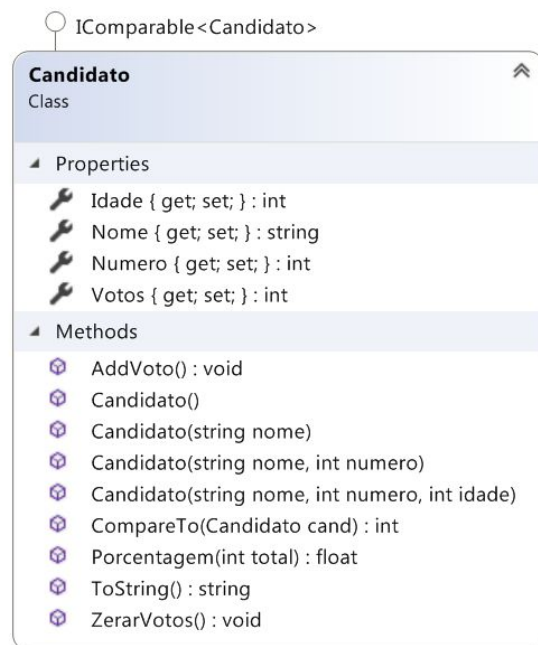
19 – Machado de Assis

21 – Guimarães Rosa

33 – Cecília Meireles

Ao final da votação, o algoritmo deve mostrar o resultado final em número de votos e em porcentagem. Devem ser tratados erros comuns tais como: parâmetros inválidos, inadequados ou inexistentes.

DIAGRAMA UML CLASS CANDIDATO



CÓDIGO PROGRAM

```
//
// nome do programa:
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 14/08/2019
// entrada(s): numero de eleitores e numero no qual cada um irá votar
// o numero de matricula (6 digitos)e o nome completo dos alunos
// saida(s): resultado dos votos inseridos em numero de votos e porcentagem
// para executar e testar digite: Ex34.exe 5
// descricao: um programa que funciona a partir de linhas de comandos em que
sera inserido um numero de eleitors
// apos todos esses eleitores efetuarem seus votos, sera mostrado o
resultado, e caso de um segundo turno, uma
// segunda eleição será chamada
//

using System;

namespace Ex34
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de método para mostrar nomes do
integrantes do trabalho

            try // try usado para previnir erros ao receber parametros
invalidos pelo args
            {
                int eleitores = int.Parse(args[0]); // converter entrada args
para inteiro
                AddCandidatos(eleitores); // chamada de método inicial
            }
            catch
            {
                Console.WriteLine("Valor inválido");
            }
        }

        static void AddCandidatos(int eleitores) // método para adicionar os
candidatos
        {
            // para adicionar um candidato basta instânciar um novo
objeto, e adiciona-lo no vetor Candidatos
            // contrutor Candidato: (string nome, int numero, int
idade)
            Candidato Machado = new Candidato("Machado de Assis", 19, 69);
            Candidato Guimaraes = new Candidato("Guimarães Rosa", 21, 59);
            Candidato Cecilia = new Candidato("Cecília Meireles", 33, 63);

            Candidato Drummond = new Candidato("Carlos Drummond de Andrade", 48,
84);

            Candidato[] Candidatos = new Candidato[] { Machado, Guimaraes, Cecilia,
Drummond }; // vetor para armazenar candidatos
        }
    }
}
```

```

        Candidato Branco = new Candidato("Branco", 0); // instanciando objeto
para o voto Branco
        Candidato Nulo = new Candidato("Nulo"); // instanciando objeto para o
voto Nulo

        Candidato[] Invalidos = new Candidato[] { Branco, Nulo }; //
vetor para armazenar votos nulos e brancos

        Votacao(eleitores, Candidatos, Invalidos); // chamada do método
para iniciar a votação
    }

    static void Votacao(int eleitores, Candidato[] candidatos,
Candidato[] invalidos) // método para apresentar opções para a votacao
    {
        Console.Clear();

        int opc = 0; // variável para armazenar opção do usuário

        Console.WriteLine("Número de eleitores: " + eleitores); //
mostrar quantidade de eleitores digitada

        foreach (var item in candidatos) // zerar números de votos
válidos para o segundo turno
        {
            item.ZerarVotos();
        }
        foreach (var item in invalidos) // zerar números de votos nulos e
brancos para o segundo turno
        {
            item.ZerarVotos();
        }

        for (int i = 0; i < eleitores; i++) // repetição de acordo com
número de eleitores
        {
            bool votoConfirmado = false; // verificar votos nulos

            Console.WriteLine();

            foreach (var item in candidatos) // mostrar números e nome
dos candidatos
            {
                Console.WriteLine(item);
            }
            Console.WriteLine("0 - Branco");

            Console.Write("\nEleitor {0}, digite o número do candidato em
que deseja votar: ", i + 1);

            try // try para realizar tratamento de erros
            {
                opc = int.Parse(Console.ReadLine()); // receber opção de
voto do eleitor
            }
            catch
            {
                opc = 999; // retorna valor para voto nulo
            }

            foreach (var item in candidatos) // percorrer lista
candidatos para adicionar os votos

```

```

        {
            if (item.Numero == opc) // verificar voto do eleitor
            {
                item.AddVoto(); // adicionar voto
                Console.Clear();
                Console.WriteLine("Voto confirmado");
                votoConfirmado = true;
                System.Threading.Thread.Sleep(200);
                break;
            }
        }

        if (opc == 0) // adicionar voto branco
        {
            foreach (var pos in invalidos)
            {
                if (pos.Nome == "Branco")
                {
                    Console.Clear();
                    Console.WriteLine("Voto Branco");
                    System.Threading.Thread.Sleep(200);
                    pos.AddVoto();
                }
            }
        }

        if ((votoConfirmado == false) && (opc != 0)) // adicionar
voto nulo
        {
            foreach (var pos in invalidos)
            {
                if (pos.Nome == "Nulo")
                {
                    Console.Clear();
                    Console.WriteLine("Voto nulo");
                    System.Threading.Thread.Sleep(200);
                    pos.AddVoto();
                }
            }
        }

        int votosValidos = VotosValidos(invalidos, eleitores); //
armazenar número de votos válidos
        MostrarResultados(candidatos, invalidos, eleitores,
votosValidos); // método para mostrar resultados
        Apuracao(candidatos, invalidos, eleitores, votosValidos); //
chamada de método para apurar os votos
    }

    static int VotosValidos(Candidato[] invalidos, int eleitores) //
método para filtrar votos válidos
    {
        int votosInvalidos = 0;

        foreach (var item in invalidos) // filtrar votos válidos,
excluindo votos nulos e brancos
        {
            votosInvalidos = votosInvalidos + item.Votos;
        }

        return eleitores - votosInvalidos;
    }

```

```

    }

    static void MostrarResultados(Candidato[] candidatos, Candidato[]
invalidos, int eleitores, int validos) // métodos resultados primeiro e
segundo turno
    {
        Console.Clear();

        Array.Sort(candidatos); // ordenar vetor de acordo com o método
IComparable.CompareTo

        foreach (var item in candidatos) // mostrar votos totais e
válidos
        {
            Console.WriteLine("{0} - {1} votos, sendo {2} do total,
votos validos {3}", item.Nome, item.Votos,
item.Porcentagem(eleitores).ToString("F1"),
item.Porcentagem(validos).ToString("F1"));
        }
        Console.WriteLine();

        foreach (var item in invalidos) // mostrar votos nulos e brancos
        {
            Console.WriteLine("{0} - teve {1} votos, sendo {2} do
total", item.Nome, item.Votos, item.Porcentagem(eleitores).ToString("F1"));
        }
        Console.WriteLine();
    }

    static void Apuracao(Candidato[] candidatos, Candidato[] invalidos,
int eleitores, int validos)
    {
        bool segTurno = false; // variável que indica se terá ou não
segundo turno

        if (candidatos.Length == 2) // caso já esteja no segundo turno,
ira mostrar o ganhador
        {
            Console.WriteLine("Ganhador: " + candidatos[0]);
            Console.ReadKey();
            Environment.Exit(0); // fechar programa
        }

        foreach (var item in candidatos) // percorrer lista para definir
se terá ou não segundo turno
        {
            if (item.Porcentagem(validos) > 50) // se % de votos válidos
for maior que 50%, exibir ganhador no primeiro turno
            {
                Console.WriteLine("Ganhador: " + item);
                Console.WriteLine();
                segTurno = false;
                break;
            }
            else segTurno = true;
        }

        if (segTurno == true) // chamar votacao para segundo turno
        {
            Candidato[] segundoTurno = new Candidato[] { candidatos[0],
candidatos[1] }; // selecionando participantes do segundo turno de acordo com
a ordenação

```

```

        Console.WriteLine("Terá segundo turno entre {0} e {1}",
segundoTurno[0], segundoTurno[1]);

        Console.Write("\nPressione qualquer tecla para continuar");
        Console.ReadKey();

        Console.WriteLine("Votação segundo turno");

        Votacao(eleitores, segundoTurno, invalidos); // chamada de
metodo de votação, agora para o segundo turno
    }
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.Write("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

CÓDIGO CLASSE CANDIDATO

```

using System;

namespace Ex34
{
    class Candidato : IComparable<Candidato>
    {
        public string Nome { get; private set; }
        public int Numero { get; private set; }
        public int Votos { get; private set; }
        public int Idade { get; private set; }

        public Candidato() {} // construtor padrão

        public Candidato(string nome, int numero, int idade)
        {
            Nome = nome;
            Numero = numero;
            Idade = idade;
        }

        public Candidato(string nome, int numero)
        {
            Nome = nome;
            Numero = numero;
        }

        public Candidato(string nome)
        {

```

```

        Nome = nome;
    }

    public override string ToString()
    {
        return "" + Numero + " - " + Nome;
    }

    public int CompareTo(Candidato cand)
    {
        // Se o número de votos for igual então faz a ordenação de acordo
        com a maior idade
        if (this.Votos == cand.Votos)
        {
            return cand.Idade.CompareTo(this.Idade);
        }
        // Ordenação padrão : do maior número de votos para o menor
        return cand.Votos.CompareTo(this.Votos);
    }

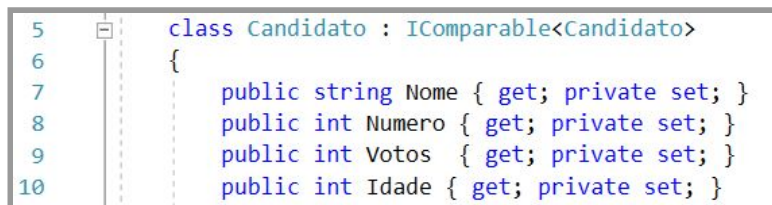
    public void AddVoto() // adiciona votos de acordo com o num recebido
    {
        Votos++;
    }

    public void ZerarVotos() // zera votos para um segundo turno
    {
        Votos = 0;
    }

    public float Porcentagem(int total) // calcular porcentagem de acordo
    com o total de votos
    {
        return (100f / total) * Votos;
    }
}

```

EXPLICANDO O CÓDIGO CLASS CANDIDATO



```

5  class Candidato : IComparable<Candidato>
6  {
7      public string Nome { get; private set; }
8      public int Numero { get; private set; }
9      public int Votos { get; private set; }
10     public int Idade { get; private set; }

```

Atributo Nome onde será armazenado o nome completo do candidato, em Número será salvo a número de campanha do candidato, Votos, o total de votos, e em Idade a idade dele.

Foi inserido a interface `Comparable` já explicada previamente, mas ela irá fazer a ordenação em um vetor que será armazenado todos os candidatos.

Construtores:

```
12 public Candidato(){} // construtor padrão
13
14 public Candidato(string nome, int numero, int idade)
15 {
16     Nome = nome;
17     Numero = numero;
18     Idade = idade;
19 }
20
21 public Candidato(string nome, int numero)
22 {
23     Nome = nome;
24     Numero = numero;
25 }
26
27 public Candidato(string nome)
28 {
29     Nome = nome;
30 }
```

Temos três construtores além do padrão, o construtor com 3 parâmetros é onde será instanciado o candidato em si;

```
Candidato Drummond = new Candidato("Carlos Drummond de Andrade", 48, 84);
```

No primeiro parâmetro será o nome, número de campanha e idade em sequência.

O próximo construtor com dois parâmetros será para instanciar um objeto de voto Branco, já que para esse objeto não precisamos de uma idade.

```
Candidato Branco = new Candidato("Branco", 0);
```

E por último um construtor com apenas um parâmetro para instanciar o voto Nulo, que será necessário apenas o nome.

```
Candidato Nulo = new Candidato("Nulo");
```

Temos também o método `ToString()` implementado a nosso objeto, para retornar de forma fácil o nome e o número de campanha do objeto desejado.


```

32 public override string ToString()
33 {
34     return "" + Numero + " - " + Nome;
35 }

```

Um método **CompareTo** para ordenar de acordo com o número de votos e idade.

```

public int CompareTo(Candidato cand)
{
    // Se o número de votos for igual então faz a ordenação de acordo com a maior idade
    if (this.Votos == cand.Votos)
    {
        return cand.Idade.CompareTo(this.Idade);
    }
    // Ordenação padrão : do maior número de votos para o menor
    return cand.Votos.CompareTo(this.Votos);
}

```

Por fim, um método para adicionar votos, outro para zerar os votos para um segundo turno, e outro para calcular a porcentagem com relação ao número de eleitores.

```

48 public void AddVoto()
49 {
50     Votos++;
51 }
52
53 public void ZerarVotos()
54 {
55     Votos = 0;
56 }
57
58 public float Porcentagem(int total)
59 {
60     return (100f / total) * Votos;
61 }

```

EXPLICANDO O CÓDIGO CLASSE PROGRAM

Um Main com um tratamento de exceções para evitar erros de parâmetros, ele começa chamando o método para imprimir os nomes dos integrantes, após isso é armazenado o número de eleitores recebido por linha de comando no Args[] na variável int eleitores. Em sequência é chamado o método AddCandidatos(), que dará sequência ao programa.

```

static void Main(string[] args)
{
    ImprimirNomes(); // chamada de método para mostrar nomes do integrantes do trabalho

    try // try usado para prevenir erros ao receber parametros invalidos pelo args
    {
        int eleitores = int.Parse(args[0]); // converter entrada args para inteiro
        AddCandidatos(eleitores); // chamada de método inicial
    }
    catch
    {
        Console.WriteLine("Valor inválido");
    }
}

```

O método AddCandidatos irá instanciar os candidatos a partir da linha 39 até a 43 através do construtor de três parâmetros e armazená-los em um vetor na linha 46-47, depois será instanciado também o objeto para voto Branco e Nulo pelo construtor de dois e um parâmetro respectivamente. Após isso será criado um outro vetor somente para armazenar os objetos de votos nulos, que só terão serventia para fins estatísticos.

```

36 static void AddCandidatos(int eleitores) // método para adicionar os candidatos
37 {
38
39     Candidato Machado = new Candidato("Machado de Assis", 19, 69);
40     Candidato Guimaraes = new Candidato("Guimarães Rosa", 21, 59);
41     Candidato Cecilia = new Candidato("Cecília Meireles", 33, 63);
42
43     Candidato Drummond = new Candidato("Carlos Drummond de Andrade", 48, 84);
44
45
46     Candidato[] Candidatos = new Candidato[]
47     { Machado, Guimaraes, Cecilia, Drummond }; // vetor para armazenar candidatos
48
49     Candidato Branco = new Candidato("Branco", 0); // instanciando objeto para o voto Branco
50
51     Candidato Nulo = new Candidato("Nulo"); // instanciando objeto para o voto Nulo
52
53     Candidato[] Invalidos = new Candidato[]
54     { Branco, Nulo }; // vetor para armazenar votos nulos e brancos
55
56     Votacao(eleitores, Candidatos, Invalidos); // chamada do método para iniciar a votação
57 }

```

Agora será chamado o próximo método, o da votação, na linha 56.

Esse método receberá os seguintes parâmetros, o número de eleitores, a vetor com os candidatos que participaram das votações e também o vetor de inválidos (Branco e Nulos)

A partir da linha 67 até a 74 terá duas estruturas de repetições para zerar todos os votos dos objetos dos dois vetores, isso para podermos ter um segundo turno, mas também serve como forma de mostrar que não será possível um candidato começar com alguma quantidade de votos.

```

59 static void Votacao(int eleitores, Candidato[] candidatos, Candidato[] invalidos)
60 {
61     Console.Clear();
62
63     int opc = 0; // variável para armazenar opção do usuário
64
65     Console.WriteLine("Número de eleitores: " + eleitores); // mostrar quantidade de eleitores digitada
66
67     foreach (var item in candidatos) // zerar números de votos válidos para o segundo turno
68     {
69         item.ZerarVotos();
70     }
71     foreach (var item in invalidos) // zerar números de votos nulos e brancos para o segundo turno
72     {
73         item.ZerarVotos();
74     }
75 }

```

Aqui entraremos em um loop que será determinado pelo número de eleitores, na linha 78 temos uma variável bool que será usada para determinar se o voto foi ou não válido posteriormente.

Na linha 78 temos um foreach para exibir a lista de candidatos com os números e os nomes dos mesmos, e também a opção de votos em Branco digitando 0.

A partir da linha 90 temos um tratamento de exceção sobre a obtenção da opção do usuário, que em caso de parâmetro inválido, irá retornar um valor para variável opção que será considerada um voto Nulo.

```

76 for (int i = 0; i < eleitores; i++) // repetição de acordo com número de eleitores
77 {
78     bool votoConfirmado = false; // verificar votos nulos
79
80     Console.WriteLine();
81
82     foreach (var item in candidatos) // mostrar números e nome dos candidatos
83     {
84         Console.WriteLine(item);
85     }
86     Console.WriteLine("0 - Branco");
87
88     Console.WriteLine("\nEleitor {0}, digite o número do candidato em que deseja votar: ", i + 1);
89
90     try // try para realizar tratamento de erros
91     {
92         opc = int.Parse(Console.ReadLine()); // receber opção de voto do eleitor
93     }
94     catch
95     {
96         opc = 999; // retorna valor para voto nulo
97     }

```

Seguindo temos outro foreach para adicionar o voto de acordo com a opção armazenada na variável int opc, em caso de a opção ser compatível com os números presente na lista, uma mensagem de “voto confirmado” será exibida no console. E a variável bool na linha 106 retornará true.

```

99      foreach (var item in candidatos) // percorrer lista candidatos para adicionar os votos
100     {
101         if (item.Numero == opc) // verificar voto do eleitor
102         {
103             item.AddVoto(); // adicionar voto
104             Console.Clear();
105             Console.WriteLine("Voto confirmado");
106             votoConfirmado = true;
107             System.Threading.Thread.Sleep(200);
108             break;
109         }
110     }

```

Se a opção digitada pelo usuário for zero será adicionado o voto Branco, através de mais um foreach que irá percorrer agora o vetor inválidos.

```

112     if (opc == 0) // adicionar voto branco
113     {
114         foreach (var pos in invalidos)
115         {
116             if (pos.Nome == "Branco")
117             {
118                 Console.Clear();
119                 Console.WriteLine("Voto Branco");
120                 System.Threading.Thread.Sleep(200);
121                 pos.AddVoto();
122             }
123         }
124     }

```

E por fim o voto inválido, que se a variável bool do começo do método for igual a false e opc diferente de 0 será adicionado um voto ao objeto nulo, e aparecerá uma mensagem na tela indicado a operação.

```

126     if ((votoConfirmado == false) && (opc != 0)) // adicionar voto nulo
127     {
128         foreach (var pos in invalidos)
129         {
130             if (pos.Nome == "Nulo")
131             {
132                 Console.Clear();
133                 Console.WriteLine("Voto nulo");
134                 System.Threading.Thread.Sleep(200);
135                 pos.AddVoto();
136             }
137         }
138     }

```

Antes de finalizar o método Votacao() será feito mais algumas tarefas, como calcular o número de votos válidos pelo método de nome sugestivo:

```
int votosValidos = VotosValidos(invalidos, eleitores);
```

Método simples que receberá como parâmetros o vetor de inválidos e o número de eleitores.

Após isso na linha 148 é declarado uma variável para armazenar os votos inválidos, em sequência um foreach para coletar todos esses votos. E por fim é retornado os votos válidos, através da conta de eleitores totais menos votos inválidos.

```
146 static int VotosValidos(Candidato[] invalidos, int eleitores)
147 {
148     int votosInvalidos = 0;
149
150     foreach (var item in invalidos)
151     {
152         votosInvalidos = votosInvalidos + item.Votos;
153     }
154
155     return eleitores - votosInvalidos;
156 }
```

Depois de retornar os votos válidos é chamado o método para mostrar o resultado da votação.

```
MostrarResultados(candidatos, invalidos, eleitores, votosValidos);
```

Esse método receberá os dois vetores como parâmetros, o número total de votos (eleitores) e os votos válidos.

Na linha 162 será utilizado o método `Array.Sort()` que se comunicaram com a interface `IComparable` da nossa classe `candidato`, e ordenará o vetor de acordo com nossa lógica previamente estabelecida.

A partir da linha 164 terá um foreach para mostrar o resultado de votos, será exibido o nome do candidato, o número de votos, a porcentagem em relação ao número total de votos, e também a porcentagem em relação aos votos válidos, desconsiderando os nulos e brancos.

Abaixo teremos outro foreach semelhante, mas agora para exibir os votos nulos separadamente.


```

158 static void MostrarResultados(Candidato[] candidatos, Candidato[] invalidos, int eleitores, int validos)
159 {
160     Console.Clear();
161
162     Array.Sort(candidatos); // ordenar vetor de acordo com o método IComparable.CompareTo
163
164     foreach (var item in candidatos) // mostrar votos totais e válidos
165     {
166         Console.WriteLine("{0} - {1} votos, sendo {2} do total, votos validos {3}", item.Nome,
167             item.Votos, item.Porcentagem(eleitores).ToString("F1"), item.Porcentagem(validos).ToString("F1"));
168     }
169     Console.WriteLine();
170
171     foreach (var item in invalidos) // mostrar votos nulos e brancos
172     {
173         Console.WriteLine("{0} - teve {1} votos, sendo {2} do total", item.Nome, item.Votos,
174             item.Porcentagem(eleitores).ToString("F1"));
175     }
176     Console.WriteLine();
177 }

```

Após esse método ser finalizado, voltamos ao método votação para a chamada do último método, que será a apuração dos votos.

Que irá receber os mesmos parâmetros do método mostrado acima, mas agora com outras finalidades.

```

Apuracao(candidatos, invalidos, eleitores, votosValidos);

```

Como nosso método de apuração é utilizado tanto para o primeiro, quanto para um possível segundo turno, teremos uma verificação a partir da linha 183 até a 188 para conseguir diferenciar se está tendo o primeiro ou o segundo turno, funcionando da seguinte forma:

Caso o número de candidatos presente no vetor de mesmo nome for igual a 2 significa claramente que estamos em um segundo turno, já que em primeiro turno teríamos mais de dois candidatos, e se isso for confirmado, será mostrado o ganhador na linha 185 e na linha 187 encerrará o programa com o comando `Environment.Exit(0)`;

```

179 static void Apuracao(Candidato[] candidatos, Candidato[] invalidos, int eleitores, int validos)
180 {
181     bool segTurno = false; // variável que indica se terá ou não segundo turno
182
183     if (candidatos.Length == 2) // caso já esteja no segundo turno, ira mostrar o ganhador
184     {
185         Console.WriteLine("Ganhador: " + candidatos[0]);
186         Console.ReadKey();
187         Environment.Exit(0); // fechar programa
188     }
189 }

```

Já aqui iremos verificar se terá ou não um segundo turno, seguindo a regra que;

Se em nossa lista de candidatos não tiver um candidato com mais de 50% dos votos válidos, será retornado true para a boolean `segTurno`, mas em caso de existir um candidato

com mais de 50% já será exibido o ganhador em primeiro turno, e na linha 196 será definido false a variável de controle.

```
190     foreach (var item in candidatos)
191     {
192         if (item.Porcentagem(validos) > 50)
193         {
194             Console.WriteLine("Ganhador: " + item);
195             Console.WriteLine();
196             segTurno = false;
197             break;
198         }
199         else segTurno = true;
200     }
```

Por fim será testado se a variável segTurno foi definida como true para poder iniciar um segundo turno.

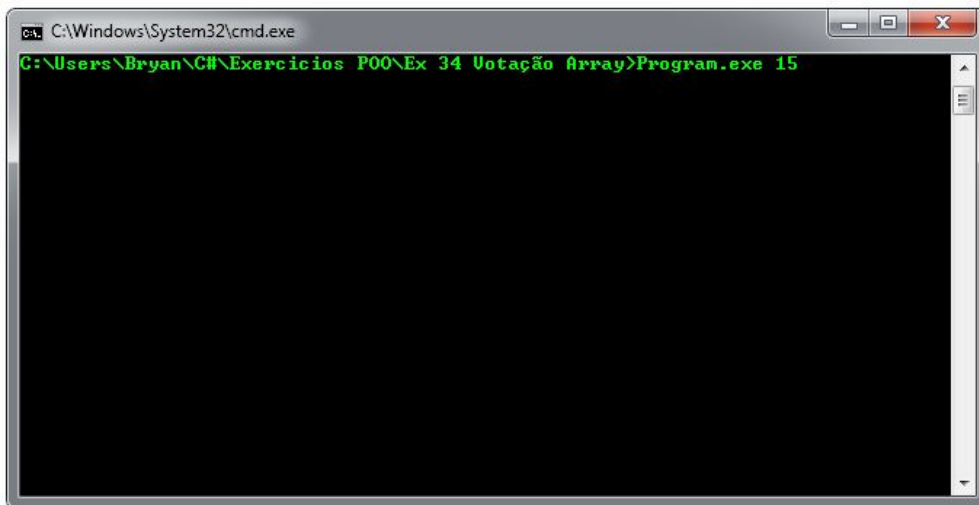
Caso for necessário o segundo turno, o candidato que estiver em primeiro e em segundo no vetor de candidato, que já foi ordenado anteriormente, será armazenado no vetor segundoTurno para participar do segundo turno, como podemos ver na linha 204.

Na linha 206 uma mensagem com que irá para o segundo turno será exibida.

Já na 213 será chamado o mesmo método que foi usado para a votação do primeiro turno, com uma diferença no segundo parâmetro, dos candidatos, que agora será composto apenas por duas pessoas, que serão os participantes do segundo turno.

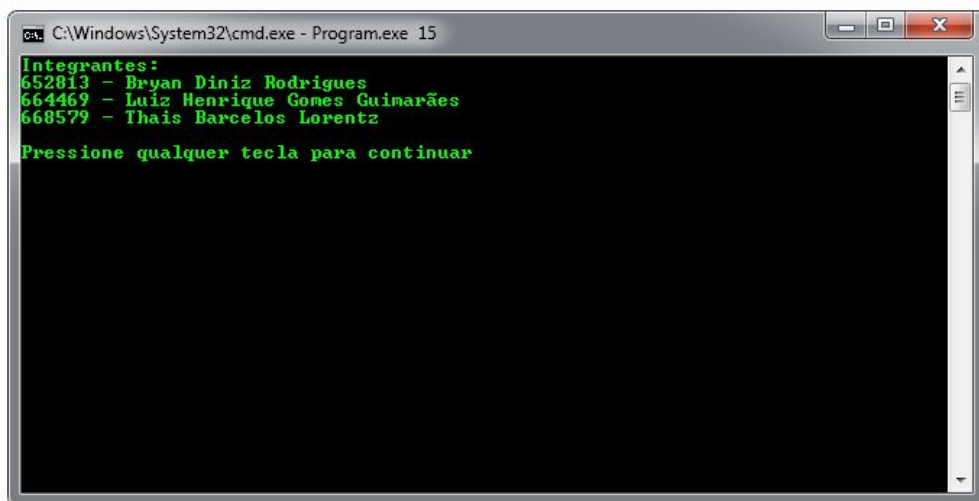
```
202     if (segTurno == true)
203     {
204         Candidato[] segundoTurno = new Candidato[] { candidatos[0], candidatos[1] };
205
206         Console.WriteLine("Terá segundo turno entre {0} e {1}", segundoTurno[0], segundoTurno[1]);
207
208         Console.WriteLine("\nPressione qualquer tecla para continuar");
209         Console.ReadKey();
210
211         Console.WriteLine("Votação segundo turno");
212
213         Votacao(eleitores, segundoTurno, invalidos);
214     }
215 }
```

ENTRADA:

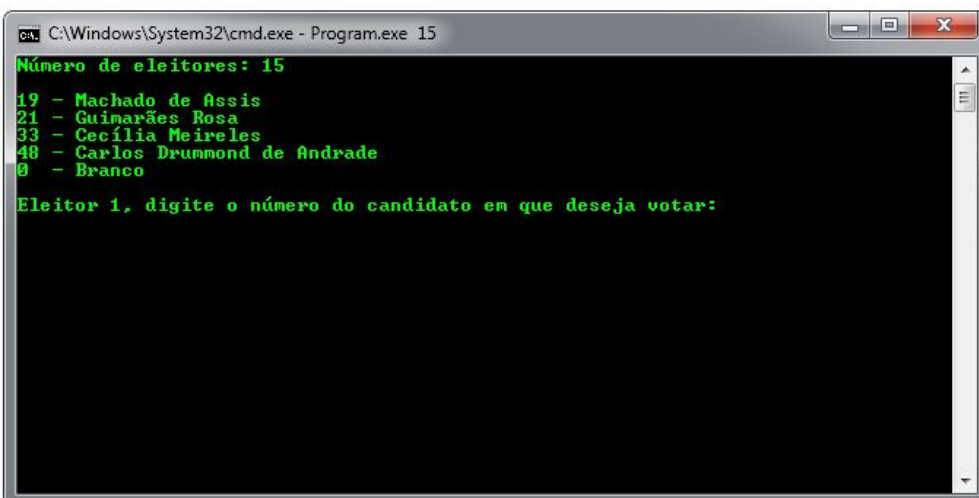


```
C:\Windows\System32\cmd.exe
G:\Users\Bryan\C#\Exercicios P00\Ex 34 Votação Array>Program.exe 15
```

SAÍDA:



```
C:\Windows\System32\cmd.exe - Program.exe 15
Integrantes:
652813 - Bryan Diniz Rodrigues
664469 - Luiz Henrique Gomes Guimarães
668579 - Thais Barcelos Lorentz
Pressione qualquer tecla para continuar
```



```
C:\Windows\System32\cmd.exe - Program.exe 15
Número de eleitores: 15
19 - Machado de Assis
21 - Guimarães Rosa
33 - Cecília Meireles
48 - Carlos Drummond de Andrade
0 - Branco
Eleitor 1, digite o número do candidato em que deseja votar:
```


EXERCÍCIO 3.5 CONVERSOR DE TEMPERATURAS

ENUNCIADO

Fazer um programa codificado em C#, usando obrigatoriamente o csc.exe, para conversão de temperaturas que recebe obrigatoriamente através de parâmetros passados na linha de comando (Args[]) os seguintes argumentos: a temperatura a ser convertida em graus Celsius (C) ou Fahrenheit (F).

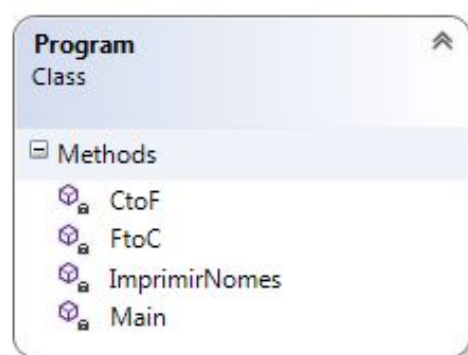
O programa converte de Celsius para Fahrenheit se a temperatura informada for em graus Celsius (C) ou converte de Fahrenheit para Celsius se a temperatura informada for em graus em Fahrenheit.

- Deve-se implementar obrigatoriamente métodos estáticos para converter a temperatura de Celsius para Fahrenheit e Fahrenheit para Celsius.

Exemplos: ENTRADA: ./ConvTemp.exe 0 C SAÍDA: 0 graus Celsius = 32 graus Fahrenheit

Devem ser tratados erros comuns tais como: parâmetros inválidos, inadequados ou inexistentes.

DIAGRAMA UML



CÓDIGO DO PROGRAMA

```
//
// nome do programa: Ex35.cs
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 07/08/2019
// entrada(s): um valor para temperatura em °C ou em °F, exemplo 75 C
// o numero de matricula (6 dígitos)e o nome completo do aluno
// saída(s): valor convertido para outra medida
// para executar e testar digite:
// Ex35.exe 75 C
// descricao: Recebe uma temperatura em °c e converte para °f ou o oposto.
//

using System;
using System.Globalization;

namespace Ex35
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
            integrantes do trabalho

            try // try usado para previnir erros ao receber parametros
            invalidos pelo args
            {
                float minC = (-273.15f), minF = (-459.67f); // limite minimo de
                conversao
                float graus = float.Parse(args[0]); // recebendo valor numerico
                para medida
                string medida = args[1].ToLower(); // recebendo medida para
                indicar a conversao

                if ((medida == "°c") || (medida == "c") || (medida ==
                "celsius")) && (graus >= minC))
                {
                    Console.WriteLine("{0} graus Fahrenheit",
                    CtoF(graus).ToString("F2"));
                }
                else if ((medida == "°f") || (medida == "f") || (medida ==
                "fahrenheit")) && (graus >= minF))
                {
                    Console.WriteLine("{0} graus Celsius",
                    FtoC(graus).ToString("F2"));
                }
                else Console.WriteLine("Medida inválida");
            }
            catch
            {
                Console.WriteLine("Parâmetros inválidos, tente algo como: 43 f
                ou 32 °C");
            }
        }
    }
}
```

```

    }

    static double FtoC(double graus)
    { // metodo de conversao de Fahrenheit para Celsius
        return (graus - 32) * 5 / 9;
    }
    static double CtoF(double graus)
    { // metodo de conversao de Celsius para Fahrenheit
        return (graus * 1.8f) + 32;
    }

    static void ImprimirNomes()
    {
        Console.Clear();
        Console.WriteLine("Integrantes:");
        Console.WriteLine("652813 - Bryan Diniz Rodrigues");
        Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
        Console.WriteLine("668579 - Thais Barcelos Lorentz");
        Console.WriteLine("\nPressione qualquer tecla para continuar");
        Console.ReadKey();
        Console.Clear();
    }
}
}

```

EXPLICANDO O CÓDIGO

A primeira parte programa recebe os valores e o tipo de temperatura através da linha de comando, e armazena como string no vetor args[]. As variáveis nas linhas 29 e 30 recebem os valores digitados, sendo que a variável graus (linha 29) converte o valor para float.

A primeira estrutura de condição recebe o valor digitado em Celsius, limitando uma temperatura mínima. Após isso, ele exibe a temperatura convertida para Fahrenheit, chamando o método CtoF. (método CtoF escrito na linha 54)

A segunda estrutura de condição recebe o valor digitado em Fahrenheit, também limitando uma temperatura mínima. Após isso, ele exibe a temperatura convertida para Celsius, chamando o método FtoC. (método FtoC escrito na linha 50)

```

22 static void Main(string[] args)
23 {
24     ImprimirNomes(); // chamada de metodo para mostrar nomes do integrantes do trabalho
25
26     try // try usado para prevenir erros ao receber parametros invalidos pelo args
27     {
28         float minC = (-273.15f), minF = (-459.67f); // limite minimo de conversao
29         float graus = float.Parse(args[0]); // recebendo valor numerico para medida
30         string medida = args[1].ToLower(); // recebendo medida para indicar a conversao
31
32         if (((medida == "°c") || (medida == "c") || (medida == "celsius")) && (graus >= minC))
33         {
34             Console.WriteLine("{0} graus Fahrenheit", CtoF(graus).ToString("F2"));
35         }
36         else if (((medida == "°f") || (medida == "f") || (medida == "fahrenheit")) && (graus >= minF))
37         {
38             Console.WriteLine("{0} graus Celsius", FtoC(graus).ToString("F2"));
39         }
40         else Console.WriteLine("Medida inválida");
41     }
42     catch
43     {
44         Console.WriteLine("Parâmetros inválidos, tente algo como: 43 f ou 32 °C");
45     }
46 }
47

```

A segunda parte do código é composta pelos métodos responsáveis pela conversão de Fahrenheit para Celsius, e Celsius para Fahrenheit, respectivamente.

Os métodos recebem o valor de temperatura digitado pelo usuário e retornam o valor convertido para a devida estrutura de condição escolhida pelo usuário.

```

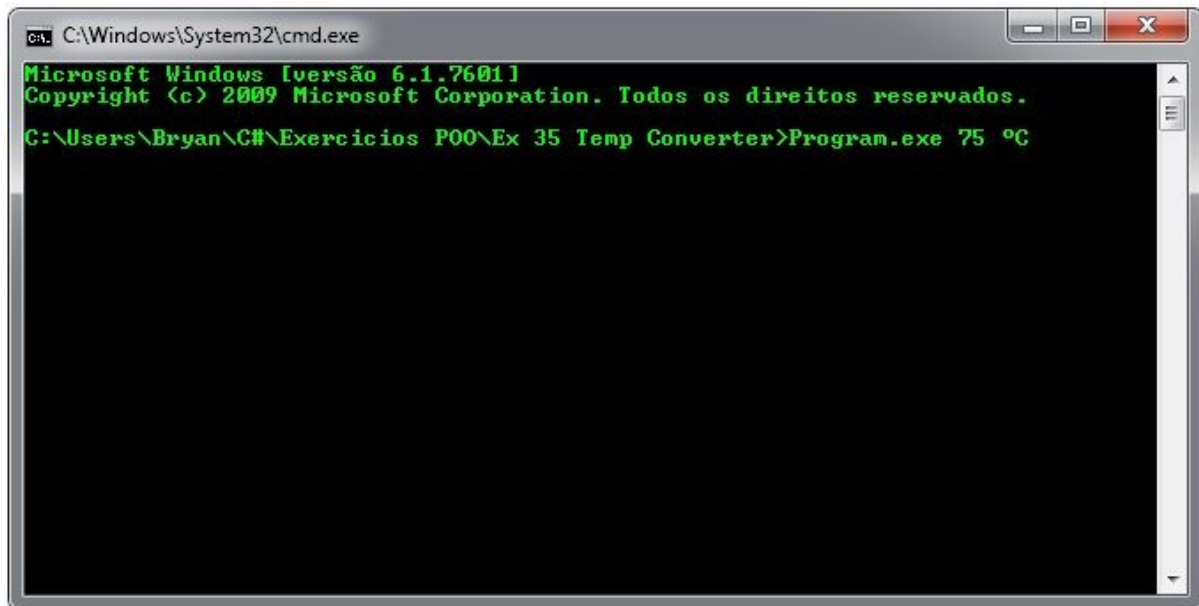
50 static double FtoC(double graus)
51 { // metodo de convesao de Fahrenheit para Celsius
52     return (graus - 32) * 5 / 9;
53 }
54 static double CtoF(double graus)
55 { // metodo de convesao de Celsius para Fahrenheit
56     return (graus * 1.8f) + 32;
57 }

```

O método FtoC recebe a temperatura em Fahrenheit, e retorna o valor em Celsius para a estrutura de condição formulada na linha 36.

O método CtoF recebe a temperatura em Celsius, e retorna o valor em Fahrenheit para a estrutura de condição formulada na linha 32.

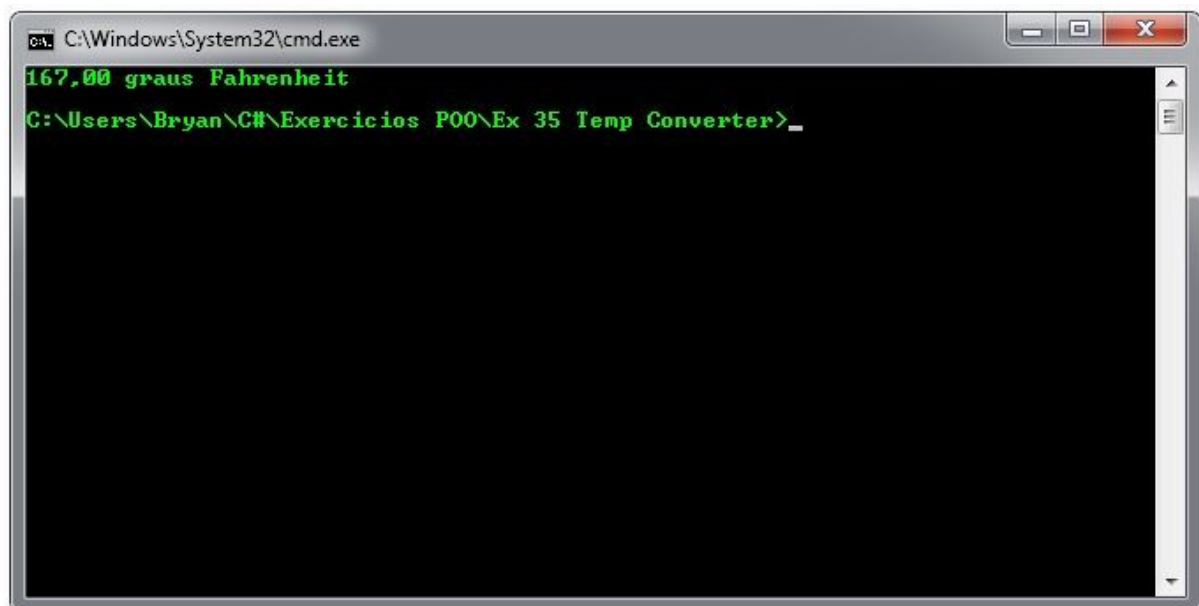
ENTRADA :



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\System32\cmd.exe". The window contains the following text in green: "Microsoft Windows [versão 6.1.7601]", "Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.", and the command prompt "C:\Users\Bryan\C#\Exercicios P00\Ex 35 Temp Converter>". The command "Program.exe 75 °C" has been entered and executed.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Bryan\C#\Exercicios P00\Ex 35 Temp Converter>Program.exe 75 °C
```

SAÍDA :



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\System32\cmd.exe". The window contains the following text in green: "167,00 graus Fahrenheit" and the command prompt "C:\Users\Bryan\C#\Exercicios P00\Ex 35 Temp Converter>". The output of the program is displayed on the first line.

```
C:\Windows\System32\cmd.exe
167,00 graus Fahrenheit
C:\Users\Bryan\C#\Exercicios P00\Ex 35 Temp Converter>
```

TRATAMENTOS DE ERRO

Caso o usuário digite algum caractere inválido, o programa envia uma mensagem mostrando que o parâmetro é inválido, através do Try Catch.

```
25  
26     try // try usado para prevenir erros ao receber parametros invalidos pelo args  
27     {
```

```
43         catch  
44         {  
45             Console.WriteLine("Parâmetros inválidos, tente algo como: 43 f ou 32 °C");  
46         }
```

EXERCÍCIO 5.1 CALCULADORA CLASSE

Fazer um programa codificado em C#, usando obrigatoriamente o csc.exe, que implemente uma calculadora com as seguintes especificações:]

-Deve efetuar as quatro operações aritméticas básicas(+, -, x e /) com números reais.

-Os operandos e o operador são recebidos obrigatoriamente através de parâmetros passados na linha de comando (Args[]).

-Deve ter obrigatoriamente uma classe que contenha um método de instância para cada operação aritmética básica (+, -, x e /).

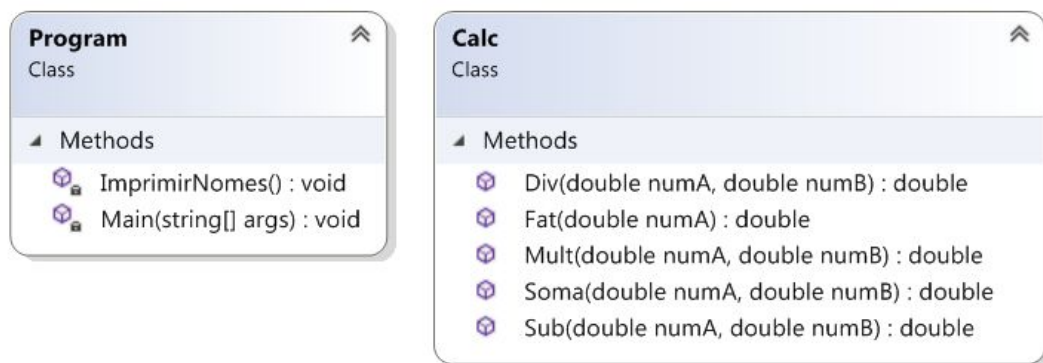
-Deve fazer o tratamento de erro nos casos de parâmetros inválidos ou inadequado. ou inexistentes

-Exemplo:

ENTRADA: ./calcs.exe 2.5 x 3 SAÍDA: 2 x 3 = 7.5

-As classes Program(que contém os métodos main e menu) e Calculadora(contém os métodos de instância)para as operações: (+, -, x e /) devem estar em arquivos separados..

DIAGRAMA UML



CÓDIGO

```
//
// nome do programa: Ex51.cs
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 07/08/2019
// entrada(s): Operandos e operadores matemáticos
// o numero de matricula (6 dígitos)e o nome completo do aluno
// saida(s): imprime o a operacao e o resultado
// para executar e testar digite:
// Ex51.exe 9 x 7
// descricao: Recebe dois numeros e um operador matematico (+, -, x ou *, /),
e retorna essa operacao
//com seu respectivo resultado.
//

using System;
using System.Globalization;

namespace Ex51
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
integrantes do trabalho

            Calc c = new Calc(); // instanciando objeto da calculadora

            try // try usado para previnir erros ao receber parametros
invalidos pelo args
            {
                double numA =
double.Parse(args[0].ToString(CultureInfo.InvariantCulture)); // variável
para armazenar primeiro operando
                char operador = char.Parse(args[1]); // variável para armazenar
operador

                if (operador == '!') // chamada para fatorial
                {
                    Console.WriteLine("{0} {1} = ", numA, operador);
// imprimir base para exhibir operacao
                    Console.WriteLine(c.Fat(numA));
                }
                else
                {

```



```

double numB =
double.Parse(args[2].ToString(CultureInfo.InvariantCulture)); // variável
para armazenar segundo operando

Console.Write("{0} {1} {2} = ", numA,
operador, numB); // imprimir base para exibir operacao

if ((numB == 0) && (operador == '/')) //
impossibilitar divisao por zero
{
    Console.WriteLine("Impossível divisão por zero");
    System.Environment.Exit(0);
}

if (operador == '+')
{
    Console.Write(c.Soma(numA, numB));
}
else if (operador == '-')
{
    Console.Write(c.Sub(numA, numB));
}
else if ((operador == 'x') || (operador == '*') ||
(operador == 'X'))
{
    Console.Write(c.Mult(numA, numB));
}
else if ((operador == '/') || (operador == '÷'))
{
    Console.Write(c.Div(numA, numB));
}
else Console.WriteLine("Operador Inválido");
}
}
catch
{
    Console.WriteLine("Valores inseridos inválidos, tente algo como:
4 x 3");
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.Write("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

CÓDIGO CLASSE CALCULADORA

```
namespace Ex51
{
    class Calc
    {
        public double Soma(double numA, double numB) // metodo somar
        {
            return (numA + numB);
        }
        public double Sub(double numA, double numB) // metodo subtrair
        {
            return (numA - numB);
        }
        public double Mult(double numA, double numB) // metodo multiplicar
        {
            return (numA * numB);
        }

        public double Div(double numA, double numB) // metodo divisao
        {

            return (numA / numB);

        }
    }
    public double Fat(double numA) // metodo fatoracao
    {
        double num, fatorial = 1;
        num = numA;

        while(num > 1)
        {
            fatorial = fatorial*num;
            num--;
        }
        return fatorial;
    }
}
```

EXPLICANDO O CÓDIGO CLASS CALC

```
3      class Calc
4      {
5          public double Soma(double numA, double numB) // metodo somar
6          {
7              return (numA + numB);
8          }
9          public double Sub(double numA, double numB) // metodo subtrair
10         {
11             return (numA - numB);
12         }
13         public double Mult(double numA, double numB) // metodo multiplicar
14         {
15             return (numA * numB);
16         }
17         public double Div(double numA, double numB) // metodo divisao
18         {
19             return (numA / numB);
20         }
21
22         public double Fat(double numA) // metodo fatoracao
23         {
24             double num, fatorial = 1;
25             num = numA;
26
27             while(num > 1)
28             {
29                 fatorial = fatorial*num;
30                 num--;
31             }
32             return fatorial;
33         }
34     }
35 }
```

Há cinco métodos nessa classe nas quais são responsáveis por suas respectivas funções operacionais. Um método relacionado para fazer somas, subtração, multiplicação, divisão e fatorial.

EXPLICANDO O CÓDIGO CLASSE PROGRAM

```
23     static void Main(string[] args)
24     {
25         ImprimirNomes();
26
27         Calc c = new Calc();
28
29         try // try usado para prevenir erros ao receber parametros invalidos pelo args
30         {
31             double numA = double.Parse(args[0].ToString(CultureInfo.InvariantCulture));
32             char operador = char.Parse(args[1]);
33
34
35             if (operador == '!')
36             {
37                 Console.Write("{0} {1} = ", numA, operador);
38                 Console.Write(c.Fat(numA));
39             }
40             else
41             {
42                 double numB = double.Parse(args[2].ToString(CultureInfo.InvariantCulture));
43
44                 Console.Write("{0} {1} {2} = ", numA, operador, numB);
```

```

45
46     if ((numB == 0) && (operador == '/'))
47     {
48         Console.WriteLine("Impossível divisão por zero");
49         System.Environment.Exit(0);
50     }

```

O Main inicializa instanciando um objeto Calc na linha 27. Na linha 31 é salvo o primeiro número que o usuário digitou em um double, na linha 32 é salvo o caractere que foi digitado uma variável tipo Char com o nome de operador. Em seguida é verificado se o operador é uma exclamação para instanciar o número e printar na tela o resultado.

Caso o caractere não seja uma exclamação é salvo em outra variável o segundo número digitado e em seguida um if é criado para verificar se o segundo número é 0 e caso esse número seja 0 e o operador for indicado para a operação ser uma divisão é printado na tela que não é possível fazer a divisão por zero.

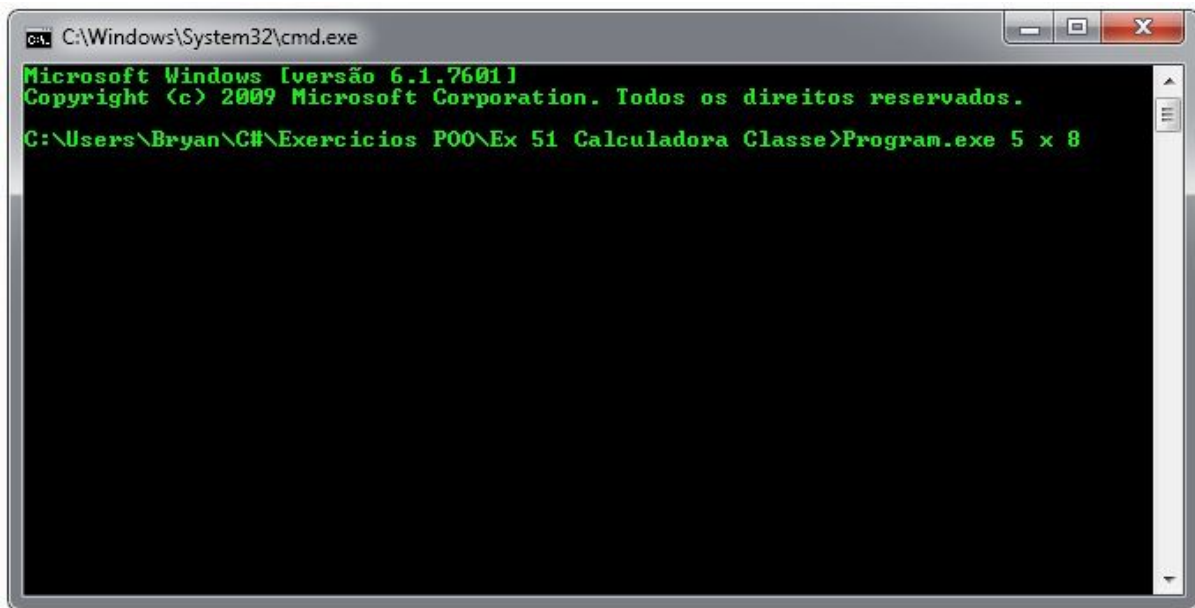
Logo após é criado um if para instanciar os números para cada método de acordo com o operador digitado pelo usuário e printado na tela o valor da operação.

```

52     if (operador == '+')
53     {
54         Console.Write(c.Soma(numA, numB));
55     }
56     else if (operador == '-')
57     {
58         Console.Write(c.Sub(numA, numB));
59     }
60     else if ((operador == 'x') || (operador == '*'))
61     {
62         Console.Write(c.Mult(numA, numB));
63     }
64     else if ((operador == '/') || (operador == '÷'))
65     {
66         Console.Write(c.Div(numA, numB));
67     }
68     else Console.WriteLine("Operador Inválido");
69 }
70 }

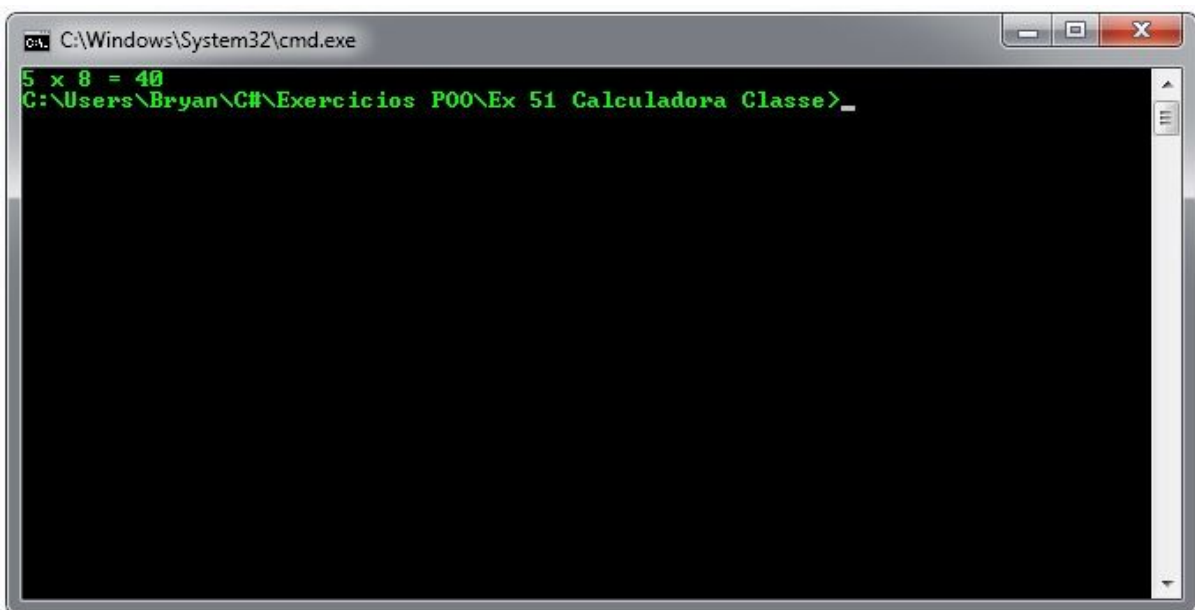
```

ENTRADA :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Bryan\C#\Exercicios P00\Ex 51 Calculadora Classe>Program.exe 5 x 8
```

SAÍDA :



```
C:\Windows\System32\cmd.exe
5 x 8 = 40
C:\Users\Bryan\C#\Exercicios P00\Ex 51 Calculadora Classe>_
```

EXERCÍCIO 5.5 CONVERSOR DE TEMPERATURA COM CLASSE

ENUNCIADO

Fazer um programa codificado em C#, usando obrigatoriamente o csc.exe, para conversão de temperaturas que recebe obrigatoriamente através de parâmetros passados na linha de comando (Args[]) os seguintes argumentos: a temperatura a ser convertida em graus Celsius (C) ou Fahrenheit (F).

O programa converte de Celsius para Fahrenheit se a temperatura informada for em graus Celsius (C) ou converte de Fahrenheit para Celsius se temperatura informada for em graus em Fahrenheit.

- Deve-se implementar obrigatoriamente métodos de instância para converter a temperatura de Celsius para Fahrenheit e Fahrenheit para Celsius.

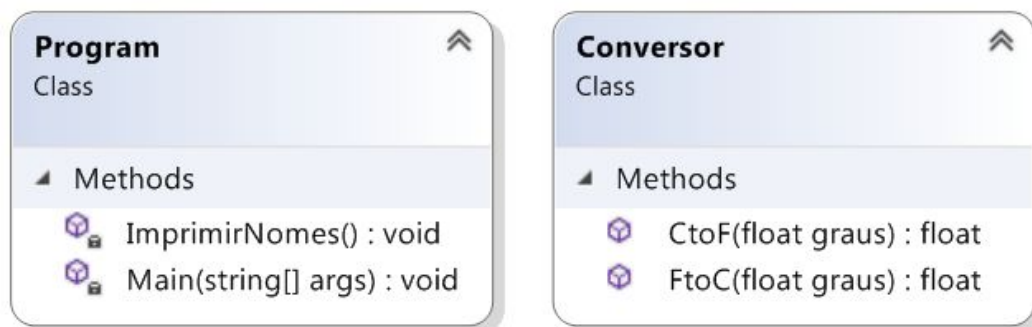
Exemplos:

ENTRADA: ./ConvTempC.exe 0 C

SAÍDA: 0 graus Celsius = 32 graus Fahrenheit

Devem ser tratados erros comuns tais como: parâmetros inválidos, inadequados ou inexistentes.

DIAGRAMA UML



CÓDIGO CLASS PROGRAM

```
//
// nome do programa: Ex35.cs
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 07/08/2019
// entrada(s): um valor para temperatura em °C ou em °F, exemplo 75 C
// o numero de matricula (6 dígitos)e o nome completo do aluno
// saida(s): valor convertido para outra medida
// para executar e testar digite:
// Ex35.exe 75 C
// descricao: Recebe uma temperatura em °c e converte para °f ou o oposto.
//

using System;

namespace Ex55
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
integrantes do trabalho

            Conversor c = new Conversor(); // instanciando objeto de
conversao

            try // try usado para previnir erros ao receber parametros
invalidos pelo args
            {
                float minC = (-273.15f), minF = (-459.67f); // limite minimo de
conversao

                float graus = float.Parse(args[0]); // recebendo valor numerico
para medida

                string medida = args[1].ToLower(); // recebendo medida para
indicar a conversao

                if ((medida == "°c") || (medida == "c") || (medida ==
"celsius")) && (graus >= minC))
                {
                    Console.WriteLine("{0} graus Fahrenheit",
c.CtoF(graus).ToString("F2"));
                }
                else if ((medida == "°f") || (medida == "f") || (medida ==
"fahrenheit")) && (graus >= minF))
                {
                    Console.WriteLine("{0} graus Celsius",
c.FtoC(graus).ToString("F2"));
                }
            }
        }
    }
}
```

```

        else Console.WriteLine("Medida inválida");
    }
    catch
    {
        Console.WriteLine("Parâmetros inválidos, tente algo como: 43 f ou
32 °C");
    }
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.Write("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

CÓDIGO CLASS CONVERTOR

```

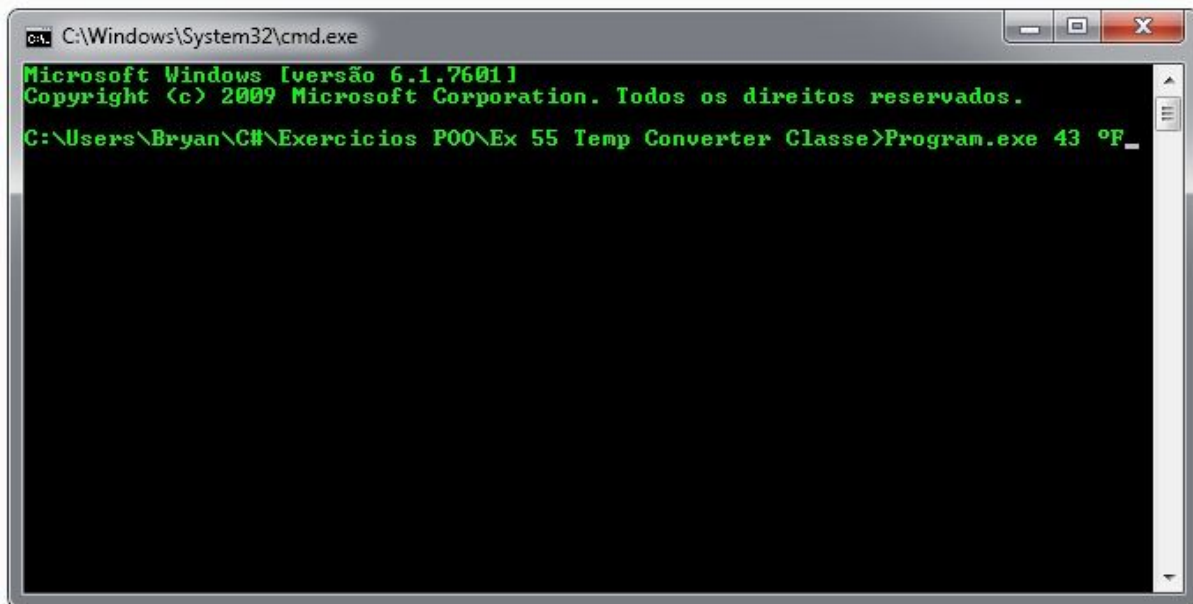
using System;

namespace Ex55
{
    class Conversor
    {
        public float FtoC(float graus)
        { // metodo de conversao de Fahrenheit para Celsius
            return (graus - 32) * 5 / 9;
        }
        public float CtoF(float graus)
        { // metodo de conversao de Celsius para Fahrenheit
            return (graus * 1.8f) + 32;
        }
    }
}

```

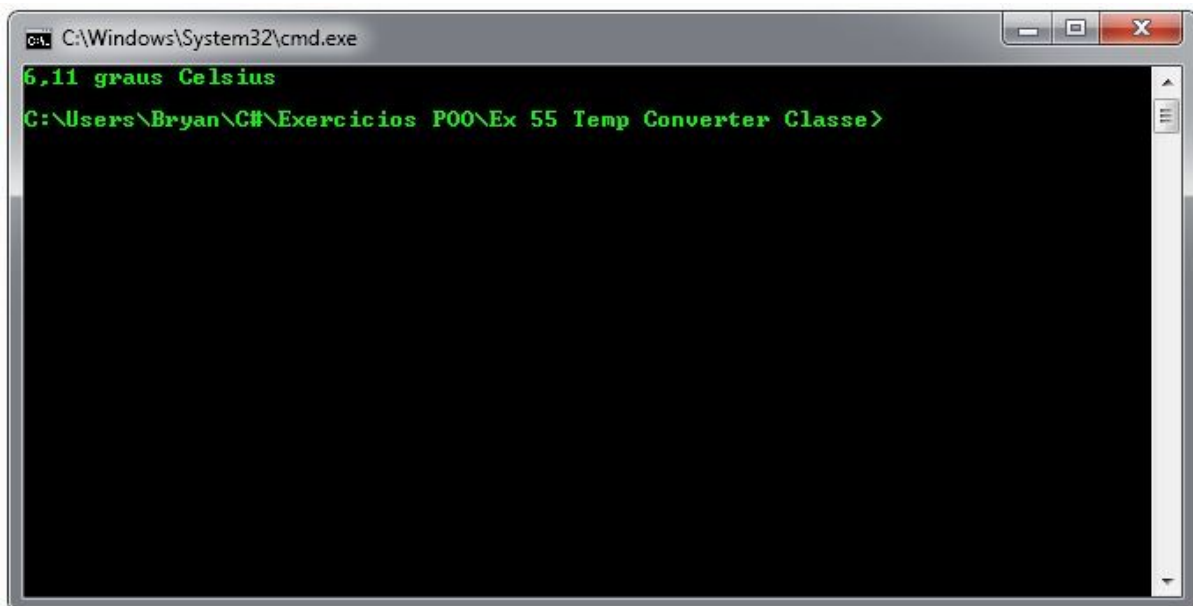
Como esse programa é uma adaptação do 3.5 com os métodos de conversão passados para métodos de instâncias e não mais estáticos, o programa funcionará da mesma forma, e também podemos ver a explicação da classe Conversor abaixo, no exercício 5.6.

ENTRADA :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.
C:\Users\Bryan\C#\Exercicios P00\Ex 55 Temp Converter Classe>Program.exe 43 °F_
```

SAÍDA :



```
C:\Windows\System32\cmd.exe
6,11 graus Celsius
C:\Users\Bryan\C#\Exercicios P00\Ex 55 Temp Converter Classe>
```

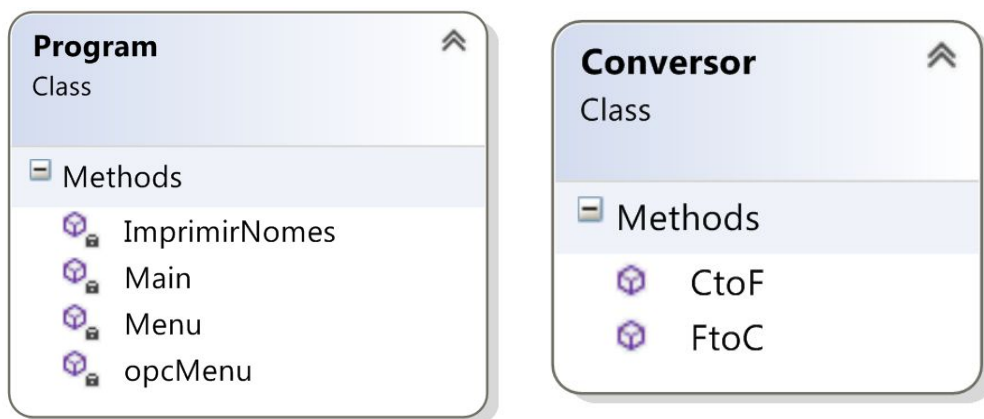
EXERCÍCIO 5.6 CONVERSOR DE TEMPERATURA COM MENU E CLASSE

ENUNCIADO

Fazer um programa codificado em C#, usando obrigatoriamente o csc.exe, para conversão de temperaturas com as seguintes especificações:

- Apresentar inicialmente na tela um menu com as seguintes opções:
 1. Converter de Celsius para Fahrenheit
 2. Converter de Fahrenheit para Celsius
 3. Sair do programa
- Obter a opção do usuário
 - Chamar o método correspondente, apresentar o resultado e sempre voltar ao menu inicial, exceto quando for selecionada a opção 3 (Sair do programa).
- Deve ter obrigatoriamente uma classe que contenha pelo menos os dois métodos para converter a temperatura em Celsius para Fahrenheit e Fahrenheit para Celsius.
- As classes Program (que contém os métodos main e menu) e ConversorTemp.cs (que contém os métodos para as operações para converter a temperatura para Celsius para Fahrenheit e Fahrenheit para Celsius) devem estar em arquivos separados.

DIAGRAMA UML



CÓDIGO CLASS PROGRAM

```
//
// nome do programa:
//
// programador(es): Bryan Diniz, Luiz Henrique Gomes Guimarães, Thais
Barcelos Lorentz
// data: 08/08/2019
// entrada(s): um valor para temperatura em °C ou em °F, exemplo 75 C
// o numero de matricula (6 dígitos)e o nome completo do aluno
// saida(s): valor convertido para outra medida
// para executar e testar digite:
// descricao: recebe uma temperatura em °c e converte para °f ou o oposto.
//

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

namespace Ex56
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
integrantes do trabalho

            while (10 > 5)
            {
                Menu();
            }

            static void opcMenu()
            {
                Console.WriteLine("1. Converter de Celsius para Fahrenheit");
                Console.WriteLine("2. Converter de Fahrenheit para Celsius");
                Console.WriteLine("3. Sair do programa");
            }

            static void Menu()
            {
                const float minC = (-273.15f), minF = (-459.67f);
                float graus; // variavel para armazenar valor em graus digitado
pelo usuário

                Conversor c = new Conversor(); // instanciando objeto conversor

                opcMenu(); // chamada de metodo com opcoes do menu

                try
                {
                    Console.Write("\nInsira uma opção: ");
                    byte opc = byte.Parse(Console.ReadLine()); //obter valor
digitado pelo usuario
                }
            }
        }
    }
}
```

```

        Console.Clear();
        switch (opc)
        {
            case 1:
                Console.Write("Digite um valor em °C para converter
em °F: ");

                graus = float.Parse(Console.ReadLine());
                if (graus >= minC)
                {
                    Console.WriteLine("{0} graus Fahrenheit \n",
c.CtoF(graus).ToString("F2")); // exibir valor convertido em °F
                }
                else { Console.WriteLine("Medida inválida, o Zero
Absoluto em celsius é -273,15°C"); }
                break;

            case 2:
                Console.Write("Digite um valor em °F para converter
em °C: ");

                graus = float.Parse(Console.ReadLine());
                if (graus >= minF)
                {
                    Console.WriteLine("{0} graus Celsius \n",
c.FtoC(graus).ToString("F2")); // exibir valor convertido em °C
                }
                else { Console.WriteLine("Medida inválida, o Zero
Absoluto em fahrenheit é -459,67°F"); }
                break;

            case 3:
                Environment.Exit(0);
                break;

            default:
                Console.Clear();
                Console.WriteLine("Opção inválida");
                System.Threading.Thread.Sleep(200);
                break;
        }
    }
    catch
    {
        Console.Clear();
        Console.WriteLine("Opção inválida, ou valor inválido");
        System.Threading.Thread.Sleep(200);
    }
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.Write("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

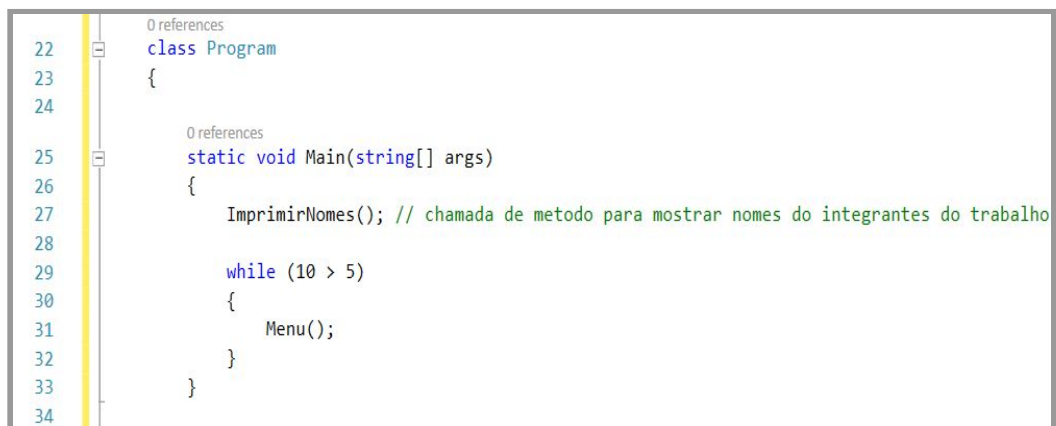
CÓDIGO CLASS CONVERTOR

```
using System;

namespace Ex56
{
    class Conversor
    {
        public float FtoC(float graus)
        { // metodo de conversao de Fahrenheit para Celsius
            return (graus - 32) * 5 / 9;
        }
        public float CtoF(float graus)
        { // metodo de conversao de Celsius para Fahrenheit
            return (graus * 1.8f) + 32;
        }
    }
}
```

EXPLICANDO A CLASSE PROGRAM

O método Main() imprime o nome dos integrantes do grupo e cria um loop sem fim, chamando o método Menu().



```
22 0 references
23 class Program
24 {
25 0 references
26 static void Main(string[] args)
27 {
28     ImprimirNomes(); // chamada de metodo para mostrar nomes do integrantes do trabalho
29
30     while (10 > 5)
31     {
32         Menu();
33     }
34 }
```

O método OpcMenu() é responsável por exibir as opções de escolha para o usuário.

```

35 1 reference
36 static void opcMenu()
37 {
38     Console.WriteLine("1. Converter de Celsius para Fahrenheit");
39     Console.WriteLine("2. Converter de Fahrenheit para Celsius");
40     Console.WriteLine("3. Sair do programa");
41 }

```

Logo em seguida, o método Menu(), que é a parte central do código, começa declarando um valor mínimo de temperatura para Celsius e Fahrenheit (linha 44).

Após isso, o método declara a variável que armazena os graus digitados pelo usuário e instancia o objeto responsável pela conversão de temperaturas.

Depois, chama o método responsável por exibir o menu : OpcMenu();

```

42 1 reference
43 static void Menu()
44 {
45     const float minC = (-273.15f), minF = (-459.67f);
46     float graus; // variavel para armazenar valor em graus digitado pelo usuário
47
48     Conversor c = new Conversor(); // instanciando objeto conversor
49
50     opcMenu(); // chamada de metodo com opcoes do menu
51
52     try
53     {

```

Após exibir o menu, o método pede para o usuário digitar uma opção, que será armazenada na variável opc.

- Caso o usuário escolha a opção 1, ele entrará no Case 1. Onde será pedido uma temperatura em Celsius e retornará o valor convertido para Fahrenheit chamando o método CtoF() da class Conversor, que é responsável pela conversão.

Se o usuário digitar uma temperatura abaixo do zero absoluto, o programa retorna uma mensagem de Medida inválida, voltando para o Menu;

- Caso o usuário escolha a opção 2, ele entrará no Case 2. Onde será pedido uma temperatura em Fahrenheit e retornará o valor convertido para Celsius chamando o método FtoC() da class Conversor, que é responsável pela conversão.

Se o usuário digitar uma temperatura abaixo do zero absoluto, o programa também retorna uma mensagem de Medida inválida, voltando para o Menu;

- Caso o usuário escolha a opção 3, ele entrará no Case 3. Onde o comando Environment.Exit(0) será responsável pela finalização do programa.
- Caso o usuário digite qualquer outro valor, ele entrará na opção default, que será responsável por enviar mensagem "Opção inválida", retornando para o Menu.

```
57 switch (opc)
58 {
59     case 1:
60         Console.Write("Digite um valor em °C para converter em °F: ");
61         graus = float.Parse(Console.ReadLine());
62         if (graus >= minC)
63         {
64             Console.WriteLine("{0} graus Fahrenheit \n", c.CtoF(graus).ToString("F2"));
65         }
66         else { Console.WriteLine("Medida inválida, o Zero Absoluto em celsius é -273,15°C"); }
67         break;
68
69     case 2:
70         Console.Write("Digite um valor em °F para converter em °C: ");
71         graus = float.Parse(Console.ReadLine());
72         if (graus >= minF)
73         {
74             Console.WriteLine("{0} graus Celsius \n", c.FtoC(graus).ToString("F2"));
75         }
76         else { Console.WriteLine("Medida inválida, o Zero Absoluto em fahrenheit é -459,67°F"); }
77         break;
```

```
79     case 3:
80         Environment.Exit(0);
81         break;
82
83     default:
84         Console.Clear();
85         Console.WriteLine("Opção inválida");
86         System.Threading.Thread.Sleep(200);
87         break;
88     }
89 }
```

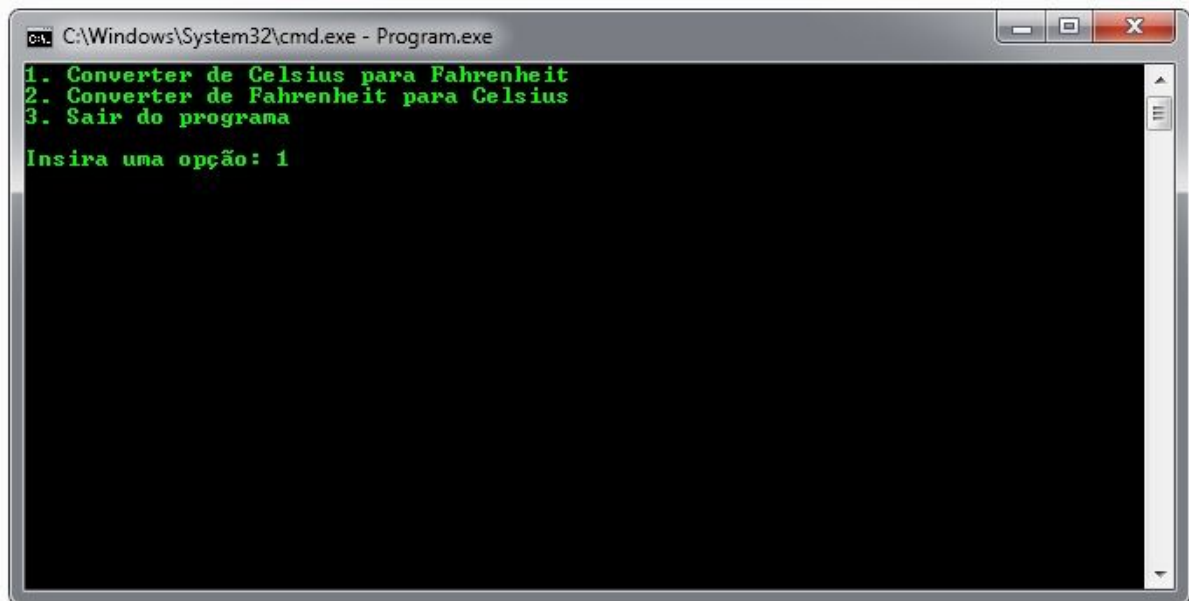
EXPLICANDO A CLASSE CONVERSOR

A classe Conversor armazena os dois métodos responsáveis por receber a temperatura digitada pelo usuário e retornar o tipo de temperatura escolhida pelo usuário.

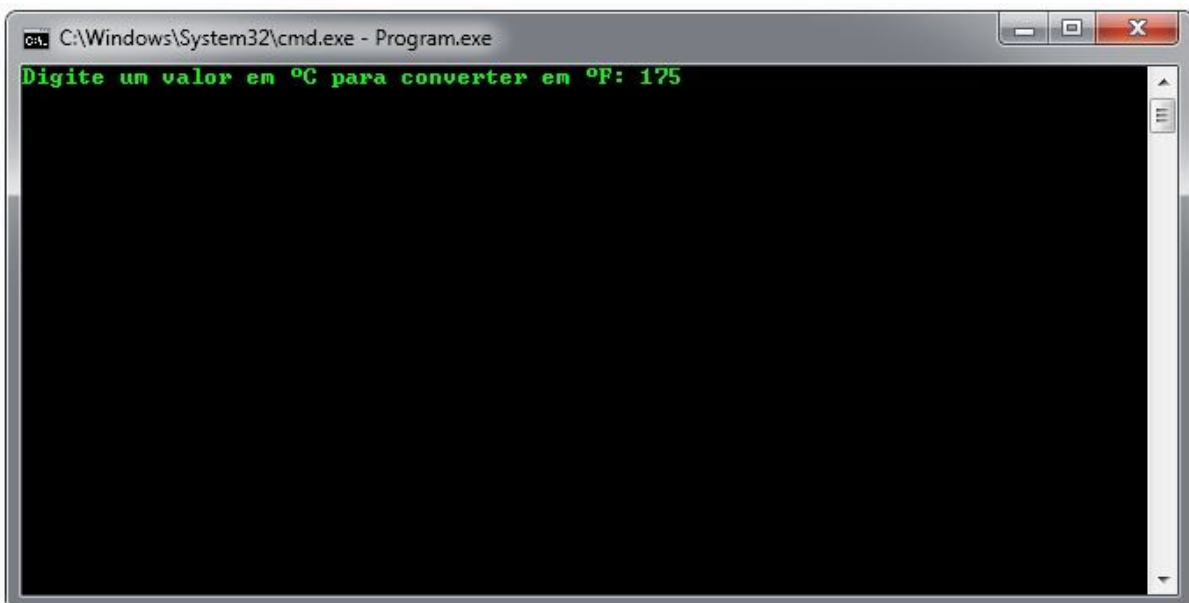
Ambos os métodos estão sendo chamados na função “Menu()” da class Program, com o intermédio do objeto “c”, criado a partir desta classe.

```
1  using System;
2
3  namespace Ex56
4  {
5      2 references
6      class Conversor
7      {
8          1 reference
9          public float FtoC(float graus)
10         { // metodo de convesao de Fahrenheit para Celsius
11             return (graus - 32) * 5 / 9;
12         }
13         1 reference
14         public float CtoF(float graus)
15         { // metodo de convesao de Celsius para Fahrenheit
16             return (graus * 1.8f) + 32;
17         }
18     }
19 }
```


ENTRADA :

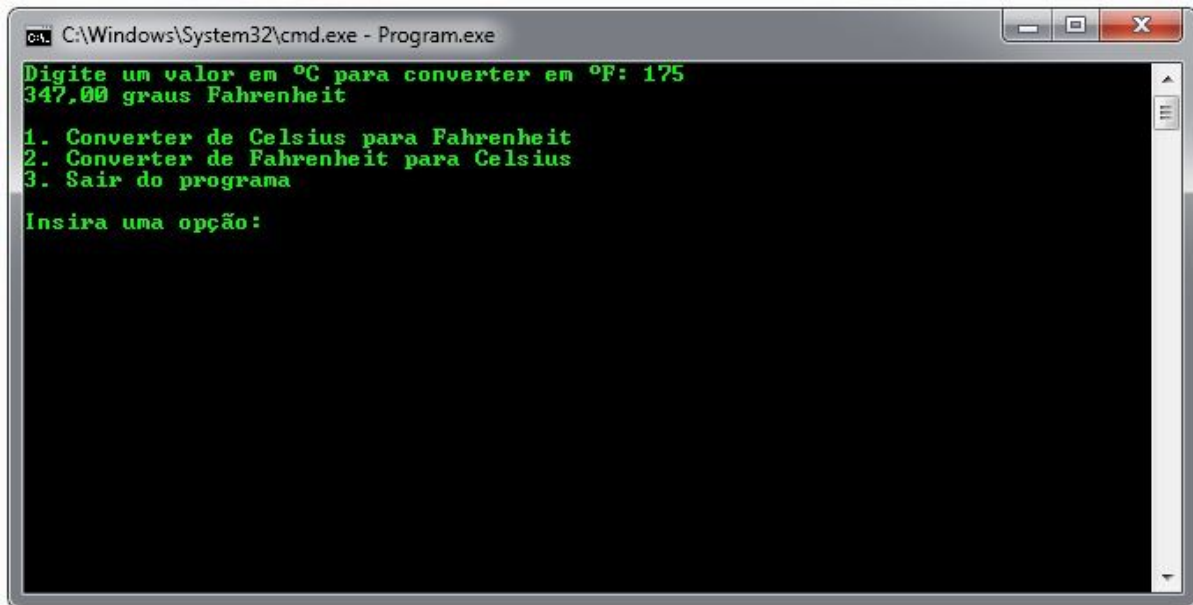


```
C:\Windows\System32\cmd.exe - Program.exe
1. Converter de Celsius para Fahrenheit
2. Converter de Fahrenheit para Celsius
3. Sair do programa
Insira uma opção: 1
```



```
C:\Windows\System32\cmd.exe - Program.exe
Digite um valor em °C para converter em °F: 175
```

SAÍDA :



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe - Program.exe". The window has standard Windows window controls (minimize, maximize, close) on the right. The text inside the window is as follows:

```
Digite um valor em °C para converter em °F: 175
347.00 graus Fahrenheit

1. Converter de Celsius para Fahrenheit
2. Converter de Fahrenheit para Celsius
3. Sair do programa

Insira uma opção:
```

EXERCÍCIO 5.7 CONTA, DEPÓSITO E SAQUE

ENUNCIADO

Fazer um programa, usando obrigatoriamente o `csc.exe`, codificado em C# que implemente o programa Conta com as seguintes especificações:

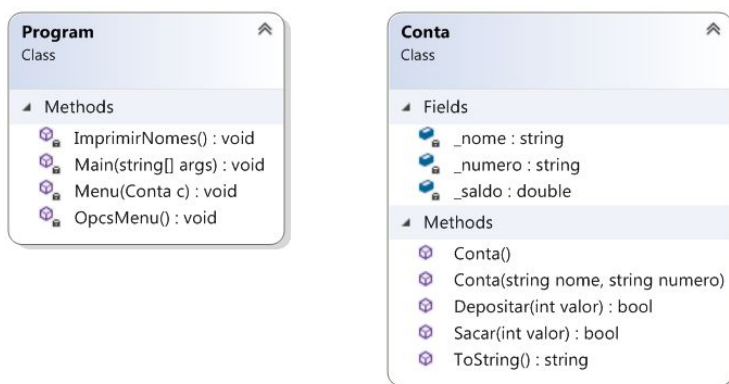
- Apresenta inicialmente na tela um menu com as seguintes opções:

1. Depositar na conta
2. Fazer retirada na conta
3. Imprimir saldo da conta
4. Sair do programa

- Obtêm a opção do usuário e chamar o método correspondente, apresenta o resultado e sempre voltará ao menu inicial, exceto quando for selecionada a opção 4 (Sair do programa).

- Deve ter obrigatoriamente um método de instância para cada operação na conta: depositar, fazer retirada e obter o saldo. Devem ser tratados erros comuns tais como: parâmetros inválidos, inadequados ou inexistentes

DIAGRAMA UML



CÓDIGO PROGRAM

```
//
// nome do programa: Ex57
//
// programador(es): Bryan Diniz
// data: 09/08/2019
// entrada(s): nome da conta, valor inicial, valor de deposito e valor de
saque
// o numero de matricula (6 digitos)e o nome completo do aluno
// saida(s): de acordo com opcoes escolhidas nos menus
// para executar e testar digite: Ex57.exe Nome 5000
// descricao:um programa para executar operacoes basicas em uma conta
bancaria
//

using System;

namespace Ex57
{
    class Program
    {
        static void Main(string[] args)
        {
            ImprimirNomes(); // chamada de metodo para mostrar nomes do
            integrantes do trabalho

            string nome = "Nome da Conta";
            string numero = "00000000-0";

            Conta c = new Conta(nome, numero); // intanciando objeto
            utilizando um construtor

            while (10 > 5) // loop infinito até usuário desejar sair do
            programa
            {
                Menu(c); // chamada de método inicial
                Console.Clear();
            }

            static void OpcoesMenu() // opções a serem exibidas no menu
            {
                Console.WriteLine("1. Depositar na conta");
                Console.WriteLine("2. Fazer retirada na conta");
                Console.WriteLine("3. Imprimir saldo da conta");
                Console.WriteLine("4. Sair do programa");
            }

            static void Menu(Conta c)
            {
                int valor; // variável que receberá valor para saque ou depósito

                OpcoesMenu(); // chamada de método para imprimir menu

                try // try usado para tratamento de erros, como valores inválidos
                {
                    int opc = int.Parse(Console.ReadLine()); // variável para
                    armazenar opção do usuário para os itens do menu

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

```

        switch (opc)
        {
            case 1:
                Console.Write("Digite um valor para o depósito: ");
                valor = int.Parse(Console.ReadLine()); // armazenar
valor de depósito do usuário
                if (c.Depositar(valor) == true)
Console.Write("Depósito realizado com sucesso"); // chamada de método de
instância para realizar depósito
                else Console.WriteLine("Valor inválido");
                break;

            case 2:
                Console.Write("Digite um valor para sacar: ");
                valor = int.Parse(Console.ReadLine()); // armazenar
valor de saque do usuário
                if (c.Sacar(valor) == true) Console.Write("Saque
realizado com sucesso"); // chamada de método de instância para realizar
saque
                else Console.Write("Saldo insuficiente ou valor
inválido");
                break;

            case 3:
                Console.WriteLine(c); // chamada de método de
instância ToString do objeto, para imprimir informações sobre o saldo
                break;

            case 4:
                Environment.Exit(0); // método usado para terminar a
aplicação
                break;

            default:
                Console.WriteLine("Opção inválida");
                break;
        }
    }
    catch
    {
        Console.Clear();
        Console.WriteLine("Valor inválido");
    }
    Console.ReadKey();
}

static void ImprimirNomes()
{
    Console.Clear();
    Console.WriteLine("Integrantes:");
    Console.WriteLine("652813 - Bryan Diniz Rodrigues");
    Console.WriteLine("664469 - Luiz Henrique Gomes Guimarães");
    Console.WriteLine("668579 - Thais Barcelos Lorentz");
    Console.Write("\nPressione qualquer tecla para continuar");
    Console.ReadKey();
    Console.Clear();
}
}
}

```

CÓDIGO CLASS CONTA

```

using System;

namespace Ex57
{
    class Conta
    {
        private string _nome;
        private string _numero;
        private double _saldo;

        public override string ToString() // retorno padrão do objeto em
String
        {
            return "" + _nome + " seu saldo é de " + "R$" +
            _saldo.ToString("F2");
        }

        public Conta() { } // construtor padrão

        public Conta(string nome, string numero) // construtor para criação
da conta
        {
            _nome = nome;
            _numero = numero;
        }

        public bool Depositar(int valor) // método para realizar depósito
        {
            if (valor > 0) // não permite depósito de valores negativos
            {
                _saldo += valor;
                return true;
            }
            else return false;
        }

        public bool Sacar(int valor) // método para realizar saque
        {
            if ((_saldo >= valor) && (valor > 0)) // só permite saque se
valor for menor que saldo e maior que zero
            {
                _saldo -= valor;
                return true;
            }
            else return false;
        }
    }
}

```

EXPLICANDO O CÓDIGO CLASS CONTA

Nossa classe conta possuía três atributos privados, sendo eles;

_nome: Para armazenar o nome do titular da conta. String

_numero: Para armazenar um número para a conta. Int

_saldo: Para armazenar o saldo da conta. Double

```
3 namespace Ex57
4 {
5     class Conta
6     {
7         private string _nome;
8         private string _numero;
9         private double _saldo;
```

A classe também possui um método ToString() para nos ajudar a retornar o saldo junto ao nome da conta.

```
11 public override string ToString() // retorno padrão do objeto em String
12 {
13     return "" + _nome + " seu saldo é de " + "R$" + _saldo.ToString("F2");
14 }
15
```

Temos dois construtores, o primeiro continua como padrão, já o segundo com dois argumentos para criarmos um objeto conta, um nome e um número.

```
16 public Conta() { }
17
18 public Conta(string nome, string numero)
19 {
20     _nome = nome;
21     _numero = numero;
22 }
```

Seguindo temos um método para realizar o depósito na conta, mas para isso o valor recebido por parâmetro deve ser maior que zero, para impedirmos de receber valores negativos. Com o valor sendo maior que zero será adicionado o valor a nosso atributo _saldo na linha 28.

Caso o valor seja menor que zero na linha 31 será retornado false, e usaremos essa informação em nossa classe principal mais tarde.

```
24 public bool Depositar(int valor) // método para realizar depósito
25 {
26     if (valor > 0) // não permite depósito de valores negativos
27     {
28         _saldo += valor;
29         return true;
30     }
31     else return false;
32 }
```

Para finalizarmos temos o método para realizar o saque, que também recebe como parâmetro um valor, mas só permitirá o saque se o valor for menor ou igual ao saldo, e o valor deve ser maior que zero. Também retornará true o false para usarmos depois.

```
34 public bool Sacar(int valor)
35 {
36     if ((_saldo >= valor) && (valor > 0))
37     {
38         _saldo -= valor;
39         return true;
40     }
41     else return false;
42 }
```

EXPLICANDO O CÓDIGO PRINCIPAL

No Main definimos um nome para a conta na linha 23 e na 24 um número, abaixo na linha 26 instanciamos um objeto para a conta com as variáveis acima como parâmetros.

A partir da linha 28 entramos em um loop e chamamos nosso método menu.


```

19 static void Main(string[] args)
20 {
21     ImprimirNomes();
22
23     string nome = "Nome da Conta";
24     string numero = "00000000-0";
25
26     Conta c = new Conta(nome, numero);
27
28     while (10 > 5)
29     {
30         Menu(c);
31         Console.Clear();
32     }
33 }

```

Logo no começo criamos uma variável para armazenarmos o valor que o usuário digitará para realizar saques ou depósitos. Já na linha 47 é chamado um método para exibir as opções do nosso menu.

```

43 static void Menu(Conta c)
44 {
45     int valor;
46
47     OpcoesMenu();

```

```

35 static void OpcoesMenu() // opções a serem exibidas no menu
36 {
37     Console.WriteLine("1. Depositar na conta");
38     Console.WriteLine("2. Fazer retirada na conta");
39     Console.WriteLine("3. Imprimir saldo da conta");
40     Console.WriteLine("4. Sair do programa");
41 }

```

Voltando ao método Menu(), na linha 51 temos uma variável que irá salvar a opção digitada pelo usuário que será usada em um switch.

```

49 try
50 {
51     int opc = int.Parse(Console.ReadLine());
52
53     Console.Clear();
54 }

```

Caso a opção seja igual a 1 irá pedir para ser digitado um valor para o depósito, e o salvará na variável valor, na linha 59, já na linha 60 temos uma

condição que se retornado como true será exibido a mensagem de sucesso na operação, já se for retornado false, irá retornar uma mensagem de valor inválido.

Se a opção for igual a 2 será semelhante ao caso anterior só que agora chamando o método de instância para o saque.

No case 3 na linha 71 será exibido o saldo através do método ToString() do nosso objeto.

```
55 switch (opc)
56 {
57     case 1:
58         Console.Write("Digite um valor para o depósito: ");
59         valor = int.Parse(Console.ReadLine());
60         if (c.Depositar(valor) == true) Console.Write("Deposito realizado com sucesso");
61         else Console.WriteLine("Valor inválido");
62         break;
63
64     case 2:
65         Console.Write("Digite um valor para sacar: ");
66         valor = int.Parse(Console.ReadLine());
67         if (c.Sacar(valor) == true) Console.Write("Saque realizado com sucesso");
68         else Console.WriteLine("Saldo insuficiente ou valor inválido");
69         break;
70
71     case 3:
72         Console.WriteLine(c);
73         break;
```

Já se a opção for quatro 4 irá fechar o programa como o comando da linha 76.

```
75 case 4:
76     Environment.Exit(0);
77     break;
78
79 default:
80     Console.WriteLine("Opção inválida");
81     break;
```

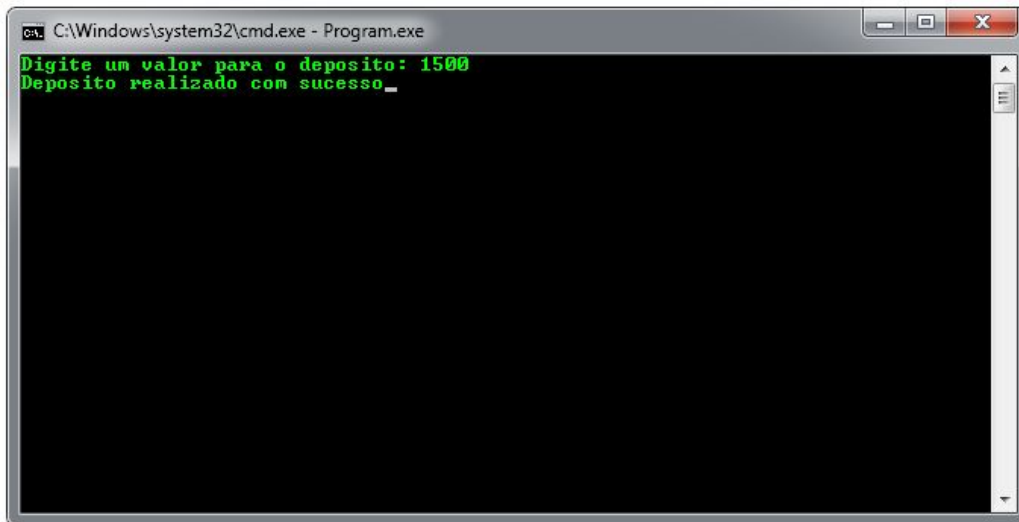
ENTRADAS:

```
C:\Windows\system32\cmd.exe - Program.exe
1. Depositar na conta
2. Fazer retirada na conta
3. Imprimir saldo da conta
4. Sair do programa
Digite uma opção: 1_
```

```
C:\Windows\system32\cmd.exe - Program.exe
Digite um valor para o depósito: 1500_
```

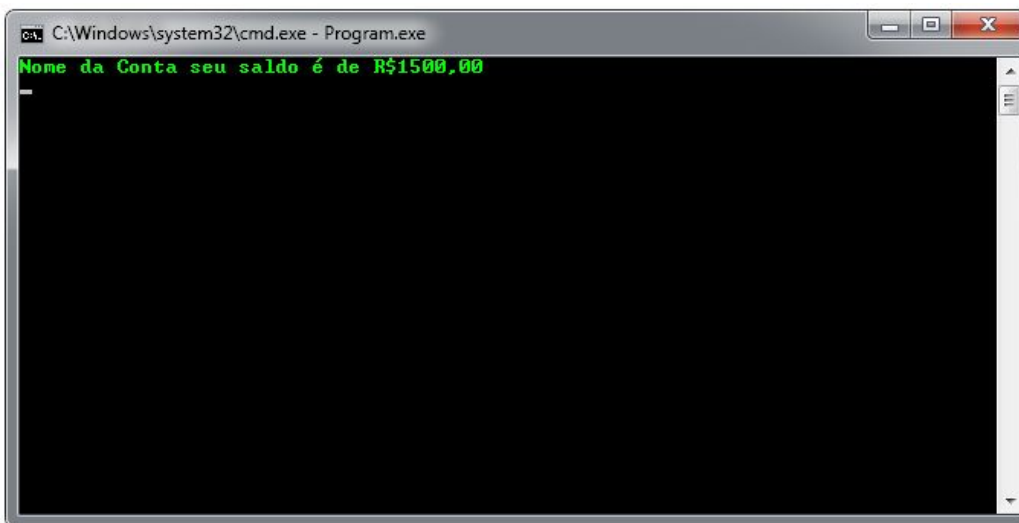
```
C:\Windows\system32\cmd.exe - Program.exe
1. Depositar na conta
2. Fazer retirada na conta
3. Imprimir saldo da conta
4. Sair do programa
Digite uma opção: 3
```

SAÍDAS:



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe - Program.exe". The command prompt shows two lines of green text: "Digite um valor para o depósito: 1500" and "Depósito realizado com sucesso_". The cursor is positioned at the end of the second line.

```
C:\Windows\system32\cmd.exe - Program.exe
Digite um valor para o depósito: 1500
Depósito realizado com sucesso_
```



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe - Program.exe". The command prompt shows a single line of green text: "Nome da Conta seu saldo é de R\$1500,00". The cursor is positioned at the end of the line.

```
C:\Windows\system32\cmd.exe - Program.exe
Nome da Conta seu saldo é de R$1500,00
```

3. CONCLUSÃO

Por meio deste trabalho da disciplina Programação Orientada a Objetos, tivemos a oportunidade de praticar e aprender novos conceitos de programação. Essa lista foi capaz de abranger todo o conteúdo visto de forma prática, deixando claro que é uma área bastante abrangente.

4. REFERÊNCIAS

Microsoft2019, Try-catch

Disponível em:

<<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch>>

Microsoft2019, Get e set

Disponível em:

<<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/using-properties>>

Microsoft2019, Guia de Programação em C#

Disponível em:

<<https://docs.microsoft.com/pt-br/>>

Microsoft2018, Suporte da Microsoft C# .

Disponível em:

<<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/>>