



Curso: Téc. em Informática.

Turno: Matutino.

Modalidade: Integrado.

Data: 31/08/2022

Disciplina:

Série: 2º ano.

Turma: Única.

Professor: Marcos Gomes Prado.

Linguagem de  
Programação.

Período Letivo: 2022.2

## **Relatório do Projeto - II Unidade:**

### **PROJETO DE LINGUAGEM DE PROGRAMAÇÃO I - MINIMUNDO ACADEMIA**

Autores:

Kayllane Santos Silva

Marco Túlio Silva Santos Alves

Maxsuel Aparecido Lima Santos

Milene Stephany Lima Ribeiro

Brumado - Bahia

Agosto – 2022



Curso: Téc. em Informática.

Turno: Matutino.

Modalidade: Integrado.

Data: 31/08/2022

Disciplina:

Série: 2º ano.

Turma: Única.

Professor: Marcos Gomes Prado.

Linguagem de  
Programação I.

Período Letivo: 2022.2

## PROJETO DE LINGUAGEM DE PROGRAMAÇÃO I - MINIMUNDO ACADEMIA

Kayllane Santos Silva

Marco Túlio Silva Santos Alves

Maxsuel Aparecido Lima Santos

Milene Stephany Lima Ribeiro

Relatório apresentado como parte do conteúdo avaliativo da disciplina Linguagem de Programação I, solicitado aos alunos do Curso Integrado em Informática – 2º ano do Instituto Federal de Educação, Ciência e Tecnologia da Bahia – *campus* Brumado, sob orientação do professor Marcos Gomes Prado.

Brumado - Bahia

Agosto – 2022

## SUMÁRIO

<b>1. Introdução.....</b>	<b>04</b>
<b>2. Projeto.....</b>	<b>04</b>
<b>3. Desenvolvimento.....</b>	<b>04</b>
<b>4. Conclusão.....</b>	<b>13</b>
<b>5. Referências.....</b>	<b>19</b>

## **1. Introdução:**

O atual trabalho foi realizado de acordo com as exigências do professor Marcos Gomes Prado da disciplina Linguagem de Programação I, na qual os alunos foram orientados a elaborar um projeto referenciando todo o conteúdo de aprendizagem ao longo da II Unidade, tal como a Programação Orientada a Objetos (POO), com o objetivo de criar um terreno comum entre as partes teórica e prática da temática supracitada.

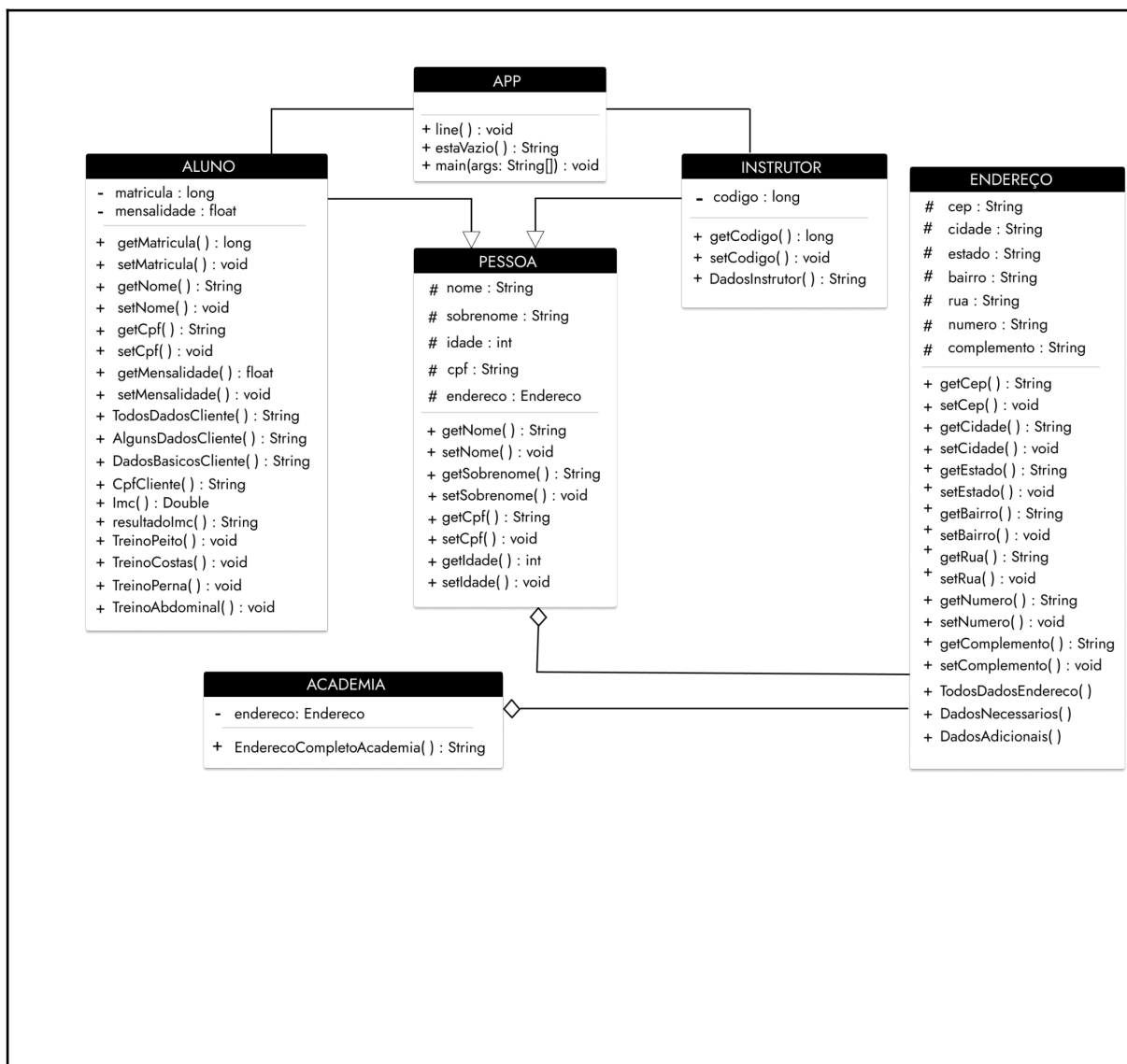
## **2. Projeto:**

Neste ínterim, os discentes Kayllane Santos Silva, Marco Túlio Silva Santos Alves, Maxsuel Aparecido Lima Santos e Milene Stephany Lima Ribeiro decidiram que seria feito um projeto envolvendo o minimundo de uma certa academia, à qual foram abstraídas as classes App, Academia, Aluno, Endereço, Instrutor e Pessoa.

Para que um indivíduo se torne aluno da instituição, faz-se necessário o seu cadastramento, ao passo que, do mesmo modo, é possível a visualização tanto dos treinamentos disponíveis quanto dos instrutores cadastrados por este último. Ao se cadastrar na academia, é solicitado, ao usuário, a inserção do seu nome, sobrenome, idade e CPF, assim como a escolha dentre três opções de mensalidade que, a depender do seu nível de treinamento constituem-se em “Civil Marombinha”, “Padawan Maromba” e “Mestre Jedi Marombão” custando 60, 80 e 100 reais, respectivamente, e o seu endereço completo, contendo as informações acerca da cidade, estado, bairro, logradouro, complemento, número do imóvel e CEP.

Ao mesmo tempo, o treinamento ministrado pela instituição é voltado para peito, costas, pernas e abdômen. Após a conclusão do cadastro, os indivíduos são automaticamente considerados alunos da academia e cada um possui um número de matrícula diferente, nome, sobrenome, CPF, mensalidade e endereço.

Outrossim, também é importante ressaltar que a “Maromba's Gym” também possui endereço próprio mediante a agregação de Endereco - como atributo do tipo Endereco - com a classe Academia; e, ao término de cada uma das opções disponíveis no menu principal, retorna-se ao mesmo.



Descrição: Diagrama feito para auxiliar na criação do código. Representa as classes do projeto (App, Pessoa, Instrutor, Aluno, Academia e Endereço), juntamente com seus atributos e métodos.

### 3. Desenvolvimento:

No intuito de possibilitar um melhor entendimento do funcionamento do programa proposto, decidiu-se que a explanação deste seria feita mediante composição das classes, sejam elas básicas ou derivadas, tendo em consideração seus atributos e seus respectivos tipos, construtores, assim como os métodos *get*, *set* e de retorno. Portanto, descreve-se a totalidade das partes de todas as classes em sua devida ordem.

A superclasse Pessoa é composta por cinco atributos, todos protegidos, denominados nome, sobrenome, idade, CPF e endereço, cujos tipos são, respectivamente, *String*, *int* e *Endereço*.

Quadro 02

```

1. public class Pessoa {
2.     protected String nome;
3.     protected String sobrenome;
4.     protected int idade;
5.     protected String cpf;
6.     protected Endereco endereco;
7.     ...

```

Descrição: Código dos atributos “nome”, “sobrenome”, “idade”, “cpf” e “endereco”, a superclasse Pessoa.

Por sua vez, no que diz respeito aos cinco construtores desta, é importante ressaltar que são diferentes os seus parâmetros. Em razão disto, pode-se construir o seguinte esquema: o primeiro construtor recebe, como parâmetro, todos os valores homônimos à todos os atributos supracitados; o segundo não recebe o valor CPF como parâmetro; o terceiro, apenas estes dois primeiros; o quarto, somente o CPF e o quinto, nenhum destes.

Quadro 03

```

1. // Construtores
2.     public Pessoa(String nome, String sobrenome, int idade, String
   cpf) {
3.         this.nome = nome;
4.         this.sobrenome = sobrenome;
5.         this.idade = idade;
6.         this.cpf = cpf;
7.     }
8.     public Pessoa(String nome, String sobrenome, int idade) {
9.         this.nome = nome;
10.        this.sobrenome = sobrenome;
11.        this.idade = idade;
12.    }
13.    public Pessoa(String nome, String sobrenome) {
14.        this.nome = nome;
15.        this.sobrenome = sobrenome;
16.    }
17.    public Pessoa(String cpf) {
18.        this.cpf = cpf;
19.    }
20.    public Pessoa(Endereco endereco) {
21.        this.endereco = endereco;
22.    }
23.    public Pessoa() {}
24.    ...

```

Descrição: Código dos construtores da superclasse Pessoa.

Por fim, tem-se a definição dos métodos *get* e *set*, cujas funções são, respectivamente, o acesso e a modificação, pelos programadores, dos atributos do tipo *private* e/ou *protected* em outra classe.

Quadro 04

```
1. // Métodos Get e Set
2.     public String getNome() {
3.         return nome;
4.     }
5.     public void setNome(String nome) {
6.         this.nome = nome;
7.     }
8.     public String getSobrenome() {
9.         return sobrenome;
10.    }
11.    public void setSobrenome(String sobrenome) {
12.        this.sobrenome = sobrenome;
13.    }
14.    public String getCpf() {
15.        return cpf;
16.    }
17.    public void setCpf(String cpf) {
18.        this.cpf = cpf;
19.    }
20.    public int getIdade() {
21.        return idade;
22.    }
23.    public void setIdade(int idade) {
24.        this.idade = idade;
25.    }
26.}
27. ...
```

Descrição: Código dos métodos *get* e *set* da superclasse Pessoa.

Neste ínterim, a classe Instrutor, subclasse derivada de Pessoa, somente possui código, privada e do tipo *long*, como constituinte de um dos seus atributos, posto que, como herda todos os outros da primeira classe, não é necessário repeti-los, pois isto tornaria o código redundante.

Quadro 05

```
1. public class Instrutor extends Pessoa {
2.     private long codigo;
3.     ...
```

Descrição: Código do atributo “código” na subclasse Instrutor.

Além disso, também é possível notar a semelhança entre os construtores dessas duas classes, que tão somente diferenciam-se no último valor passado como atributo homônimo a este primeiro.

Quadro 04

```
1. // Construtores
2.     public Instrutor(String nome, String sobrenome, int idade,
   String cpf, long codigo) {
3.         super(nome, sobrenome, idade, cpf);
4.         this.codigo = codigo;
5.     }
6.     public Instrutor(String nome, String sobrenome, int idade) {
7.         super(nome, sobrenome, idade);
8.     }
9.     public Instrutor(Endereco endereco) {
10.        this.endereco = endereco;
11.    }
12. ...
```

Descrição: Código de todos construtores da subclasse Instrutor.

Tal fato igualmente é aplicado à programação dos métodos *get* e *set* desta classe, que da mesma maneira, só há duas pequenas alterações: uma que apenas o valor código está sendo passado como parâmetro e a segunda é a impressão dos dados do instrutor, como nome, sobrenome e código, mediante os *gets* já predeterminados.

Quadro 05

```
1. // Métodos Get e Set
2.     public long getCodigo() {
3.         return codigo;
4.     }
5.     public void setCodigo(long codigo) {
6.         this.codigo = codigo;
7.     }
8.     public String DadosInstrutor() {
9.         return "Nome: " + getNome() + " " + getSobrenome() + " |
   Código: " + getCodigo();
10.    }
11. }
12. ...
```

Descrição: Código dos métodos *get* e *set* da subclasse Instrutor.

Semelhante à classe Instrutor, a classe Aluno também é derivada da classe pessoa. Seus atributos são todos privados, sendo formados por matrícula e mensalidade, dos tipos *long* e *float*, respectivamente.



#### Quadro 06

```
1. public class Aluno extends Pessoa {  
2.     private long matricula;  
3.     private float mensalidade;  
4. ...
```

Descrição: Código dos atributos “matricula” e “mensalidade” na subclasse Aluno.

Agora, os seguintes construtores e seus respectivos parâmetros.

#### Quadro 07

```
1. // Construtores  
2.     public Aluno() {}  
3.     public Aluno(String nome, String sobrenome, int idade, String  
   cpf, long matricula, float mensalidade, Endereco endereco) {  
4.         super(nome, sobrenome, idade, cpf);  
5.         this.matricula = matricula;  
6.         this.mensalidade = mensalidade;  
7.         this.endereco = endereco;  
8.     }  
9.     public Aluno(String nome, String sobrenome, int idade, String  
   cpf, float mensalidade) {  
10.        super(nome, sobrenome, idade, cpf);  
11.        this.matricula = matricula++;  
12.        this.mensalidade = mensalidade;  
13.    }  
14.    public Aluno(String nome, String cpf, float mensalidade,  
   Endereco endereco) {  
15.        super(nome, cpf);  
16.        this.mensalidade = mensalidade;  
17.        this.endereco = endereco;  
18.    }  
19. ...
```

Descrição: Código de todos os construtores da subclasse Aluno.

Por conseguinte, há os métodos *get* e *set* de cada atributo.

#### Quadro 08

```
1. // Métodos Get e Set  
2.     public long getMatricula() {  
3.         return matricula;  
4.     }  
5.     public void setMatricula(long matricula) {  
6.         this.matricula = matricula;  
7.     }  
8.     public String getNome() {  
9.         return nome;  
10.    }  
11.    public void setNome(String nome) {
```

```

12.         this.nome = nome;
13.     }
14.     public String getCpf() {
15.         return cpf;
16.     }
17.     public void setCpf(String cpf) {
18.         this.cpf = cpf;
19.     }
20.     public float getMensalidade() {
21.         return mensalidade;
22.     }
23.     public void setMensalidade(float mensalidade) {
24.         this.mensalidade = mensalidade;
25.     }
26. ...

```

Descrição: Código de todos os métodos get e set da subclasse Aluno.

Com ênfase na implementação de três tipos distintos de impressão dos dados do cliente.

#### Quadro 09

```

1.     // Métodos de retorno
2.     public String TodosDadosCliente() {
3.         return "Nome do aluno(a): " + this.nome + "\nSobrenome: " +
this.sobrenome + "\nIdade: " + this.idade + "\nCPF: " + this.cpf +
"\nMatrícula: " + this.matricula + "\nValor da mensalidade: R$ " +
this.mensalidade + "\nCEP: " + this.endereco.getCep() + "\nCidade:
" + this.endereco.getCidade() + "\nEstado: " +
this.endereco.getEstado() + "\nBairro: " +
this.endereco.getBairro() + "\nRua: " + this.endereco.getRua() +
"\nNúmero: " + this.endereco.getNumero() + "\nComplemento: " +
this.endereco.getComplemento();
4.     }
5.     public String AlgunsDadosCliente() {
6.         return "Nome do aluno(a): " + this.nome + "\nSobrenome: " +
this.sobrenome + "\nIdade: " + this.idade + "\nMatrícula: " +
this.matricula + "\nValor da mensalidade: R$ " + this.mensalidade +
"\nCEP:" + this.endereco.getCep() + "\nCidade: " +
this.endereco.getCidade() + "\nEstado: " +
this.endereco.getEstado();
7.     }
8.     public String DadosBasicosCliente() {
9.         return "Aluno(a): " + this.nome + " | Sobrenome: "+
this.getSobrenome() + " | Matrícula: " + this.getMatricula();
10.    }
11.    public String CpfCliente() {
12.        return "CPF: " + this.cpf;
13.    }
14. ...

```

Descrição: Código de todos os métodos de retorno da subclasse Aluno.

Não obstante, ainda é possível a visualização do cálculo de Índice de Massa Corporal (IMC) do aluno a fim de verificar em que estado encontra-se ele.

Quadro 10

```

1.    // IMC
2.    public double Imc(float peso, float altura) {
3.        double imc = peso / Math.pow(altura, 2);
4.        return imc;
5.    }
6.    public String resultadoImc(double imc) {
7.        String result;
8.        if (imc < 18.5)
9.            result = "ABAIXO DO PESO.";
10.       else if (imc < 25)
11.           result = "PESO IDEAL.";
12.       else if (imc < 30)
13.           result = "LEVEMENTE ACIMA DO PESO.";
14.       else if (imc < 35)
15.           result = "OBESIDADE GRAU 1.";
16.       else if (imc < 40)
17.           result = "OBESIDADE GRAU 2 (SEVERA).";
18.       else
19.           result = "OBESIDADE GRAU 3 (MÓRBIDA).";
20.       return result;
21.    }
22.    ...

```

Descrição: Código do cálculo do Índice de Massa Corporal (IMC) do usuário.

Enfim, também é possível observar todos os tipos de treinos disponíveis.

Quadro 11

```

1.    // Treinos
2.    public void TreinoPeito() {
3.        System.out.print("VOADOR PEITORAL 1X12, 1X10, 1X9, 1X8 ,
4.        1X6 (AUMENTANDO CARGA)\nSUPINO RETO 3X12 - DESCANSA 10SEG. - MAIS
5.        10 REPETIÇÕES (RESTPAUSE)\nFLAY INCLINADO 3X12.10.8
6.        (DROP)\nCRUCIFIXO COM HALTER 3X15\nTRÍCEPS FRANÇÊS - POLIA ALTA
7.        3X15\nTRÍCEPS PULLEY 3X12.10.8 (DROP)\nTRÍCEPS COICE
8.        3X15\nDESENVOLVIMENTO COM BARRA 3X15 (RESTPAUSE)\nREMADA ALTA NO
9.        CROSS 3X12\nELEVACÃO FRONTAL COM A BARRA H 3X15\n");
10.    }
11.    public void TreinoCostas() {
12.        System.out.print("PUXADA ALTA COM TRI NGULO 3X12.10.8 (DROP
13.        - REDUZINDO CARGA)\nCRUCIFIXO INVERSO COM HALTER 3X15\nREMADA
14.        CURVADA NO CROSS 3X12 - DESCANSA 10 SEG. - FAZER ATE A
15.        FALHA\nPUXADA ALTA 3X12 (RESTPAUSE)\nROSCA DIRETA COM A BARRA W
16.        3X12\nROSCA UNILATERAL NO BANCO SCOOT 3X15\nROSCA COMBINADA 3X15 +
17.        ENCOLHIMENTO DE OMBROS\nROSCA INVERSA NA POLIA 3X12.10.8
18.        (DROP)\n");
19.    }
20.    public void TreinoPerna() {
21.        System.out.print("AGACHAMENTO NO SMITT 3X12.10.8 - DESCANSO
22.        DE 10 SEG\nLEG 45 graus 3X12 (RESTPAUSE)\nCADEIRA EXTENSORA
23.        3X14.12.10 (DROP)\nCADEIRA ABDUTORA 3X20 - ISOMETRIA DE 30 segs. -
24.        FALHA\nMESA FLEXORA 3X12.10.8 (DROP) + AFUNDO LIVRE 3X12\nHACK
25.        INVERSO 3X12 + CADEIRA FLEXORA 3X15\nPANTURRILHA UNI. NO SMITT

```

```

3X20\nPANTURRILHA BANCO 3X25\n");
10.    }
11.    public void TreinoAbdominal() {
12.        System.out.print("ABS COM ANILHA NO DECLINADO 3X20\nABS
OBLÍQUO 3X15\nABS BICICLETA 3X15\n");
13.    }
14.}
15. ...

```

Descrição: Código de todos os treinos disponíveis.

Na superclasse *Endereço* houve a criação dos atributos protegidos do tipo *String*, CEP, cidade, estado, bairro, rua, número e complemento.

#### Quadro 12

```

1. public class Endereco {
2.     protected String cep;
3.     protected String cidade;
4.     protected String estado;
5.     protected String bairro;
6.     protected String rua;
7.     protected String numero;
8.     protected String complemento;
9. ...

```

Descrição: Código de todos atributos da superclasse *Endereço*.

Cujos construtores são:

#### Quadro 13

```

1. // Construtores
2. public Endereco(String cep, String cidade, String estado,
String bairro, String rua, String numero, String complemento) {
3.     this.cep = cep;
4.     this.cidade = cidade;
5.     this.estado = estado;
6.     this.bairro = bairro;
7.     this.rua = rua;
8.     this.numero = numero;
9.     this.complemento = complemento;
10. }
11. public Endereco(String cep, String cidade, String estado) {
12.     this.cep = cep;
13.     this.cidade = cidade;
14.     this.estado = estado;
15. }
16. public Endereco(String rua, String bairro, String complemento,
String numero) {
17.     this.rua = rua;
18.     this.bairro = bairro;
19.     this.complemento = complemento;

```

```
20.         this.numero = numero;
21.     }
22. ...
```

Descrição: Código de todos os construtores da superclasse Endereço.

E os métodos *get* e *set*.

#### Quadro 14

```
1.  // Métodos Get e Set
2.      public String getCep() {
3.          return cep;
4.      }
5.      public void setCep(String cep) {
6.          this.cep = cep;
7.      }
8.      public String getCidade() {
9.          return cidade;
10.     }
11.     public void setCidade(String cidade) {
12.         this.cidade = cidade;
13.     }
14.     public String getEstado() {
15.         return estado;
16.     }
17.     public void setEstado(String estado) {
18.         this.estado = estado;
19.     }
20.     public String getBairro() {
21.         return bairro;
22.     }
23.     public void setBairro(String bairro) {
24.         this.bairro = bairro;
25.     }
26.     public String getRua() {
27.         return rua;
28.     }
29.     public void setRua(String rua) {
30.         this.rua = rua;
31.     }
32.     public String getNumero() {
33.         return numero;
34.     }
35.     public void setNumero(String numero) {
36.         this.numero = numero;
37.     }
38.     public String getComplemento() {
39.         return complemento;
40.     }
41.     public void setComplemento(String complemento) {
42.         this.complemento = complemento;
43.     }
44. ...
```

Descrição: Código de todos os métodos *get* e *set* da superclasse Endereço.

Outrossim, semelhante à classe pessoa, também é possível observar três diferentes tipos de impressão de dados.

Quadro 15

```
1. // Métodos de retorno
2.     public String TodosDadosEndereco() {
3.         return "CEP: " + this.cep + "\nCidade: " + this.cidade +
4.         "\nEstado: " + this.estado + "\nBairro: " + this.bairro + "\nRua: "
5.         + this.rua + "\nComplemento: " + this.complemento + "\nNúmero: \n"
6.         + this.numero;
7.     }
8.     public String DadosEnderecoNecessarios() {
9.         return "CEP: " + this.cep + "\nCidade: " + this.cidade +
10.        "\nEstado: " + this.estado;
11.    }
12.    ...
```

Descrição: Código de todos os métodos de retorno da superclasse Endereço.

Além destas classes ora apresentadas, de modo semelhante faz-se a classe Academia. Esta, por sua vez, contém apenas um único atributo composto - isto é, um objeto da classe Endereco - homônimo a este último.

Quadro 16

```
1. public class Academia {
2.     private Endereco endereco;
3.     ...
```

Descrição: Criação do atributo composto “endereço” na classe Academia.

Por conseguinte, deu-se início à constituição do construtor desta mesma classe, aos quais foram passados todos valores homônimos em relação aos atributos, cujos tipos são sempre da classe *String*.

Quadro 17

```
1. public Academia(Endereco endereco) {
2.     this.endereco = endereco;
3. }
4. ...
```

Descrição: Codificação do único construtor da classe Academia.

Por fim, é o método de retorno “EnderecoCompletoAcademia” que permite a impressão, quando o programa for executado, do endereço completo da academia “Maromba’s Gym”.

Quadro 18

```
1.      public String EnderecoCompletoAcademia() {
2.          return "CEP: " + endereco.cep + "\nCidade: " +
           endereco.cidade + "\nEstado: " + endereco.estado + "\nBairro: " +
           endereco.bairro + "\nRua: " + endereco.rua + "\nComplemento: " +
           endereco.complemento + "\nNúmero: " + endereco.numero;
3.      }
4.
5.  }
6.  ...
```

Descrição: Implementação do método de retorno “EnderecoCompletoAcademia” na classe Academia

#### 4. Conclusão:

Antes de serem iniciados as considerações no que tange à classe App, é necessário, primordialmente, ser mencionado o tratamento de dados existente em todos os campos de preenchimento necessários ao cadastramento do usuário à academia. Logo, caso o cliente recuse-se a inserir os seus dados em qualquer um dos quesitos solicitados, não será possível o prosseguimento da sua inscrição. Para tanto, fez-se necessário a construção da função com recebimento de parâmetros e retorno denominada “estaVazio()”, a qual será feita exibição do seu código na imagem abaixo (quadro 19).

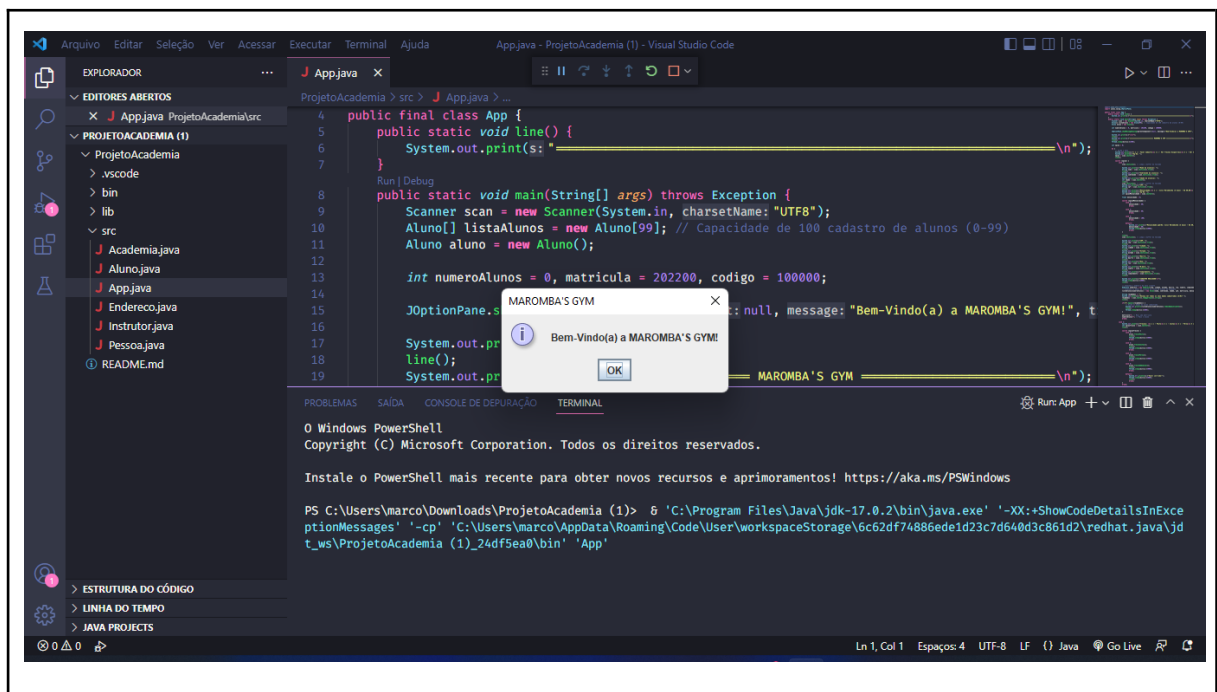
Quadro 19

```
1.      // Verifica se o usuário deixou vazio um campo, ou seja, sem
           preencher.
2.      public static String estaVazio(String texto, String campo)
           throws InterruptedException {
3.          Scanner scan = new Scanner(System.in, "UTF-8");
4.          while(texto.isEmpty()) {
5.              line();
6.              System.err.print("VOCÊ NÃO PREENCHEU O CAMPO!\n");
7.              line();
8.              Thread.sleep(2000);
9.              System.out.print("Digite o " + campo + " de novo: ");
10.             texto = scan.nextLine().trim();
11.         }
12.         return texto;
13.     ...
```

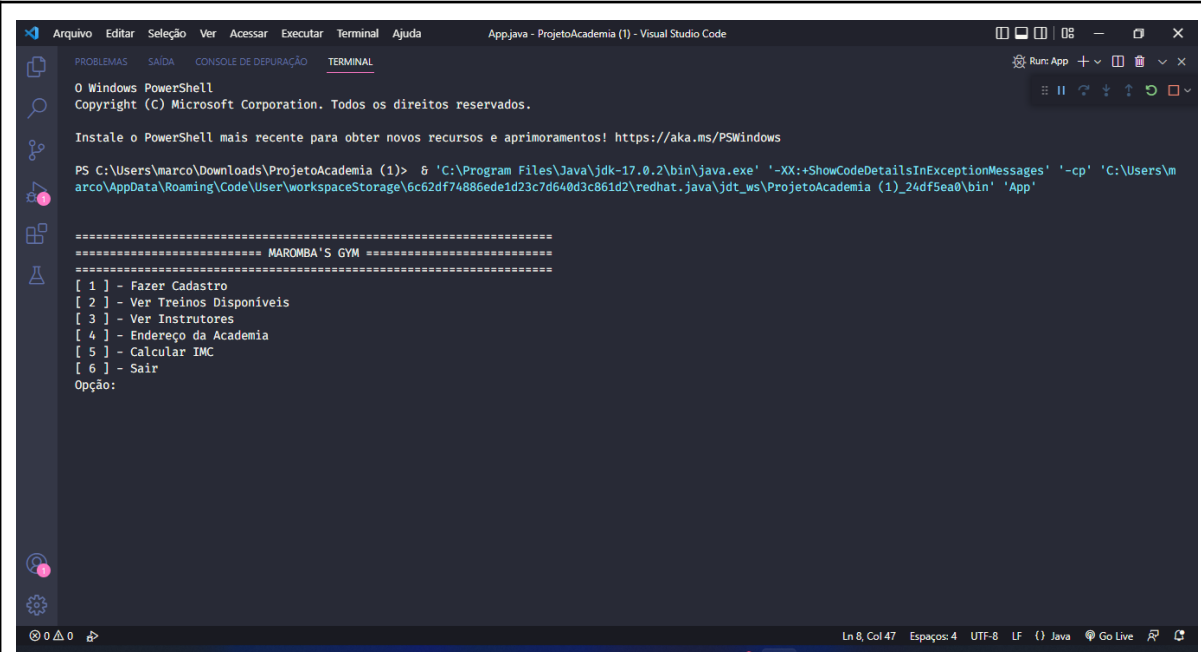
Descrição: Código da função “estaVazio”, da classe App, cuja função é, justamente a verificação de espaços vazios no ato da matrícula na academia.

Ao início da execução da classe App, onde está localizado o programa principal, é apresentado um diálogo de mensagem com as devidas saudações à academia. Subsequente a isto, é posto um menu contendo seis opções à escolha do usuário, como fazer cadastro, ver treinos e instrutores disponíveis, endereço da academia, cálculo do Índice de Massa Corporal (IMC) e saída do programa.

Quadro 19







```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda Appjava - ProjetoAcademia (1) - Visual Studio Code
PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\marco\Downloads\ProjetoAcademia (1)> & 'C:\Program Files\Java\jdk-17.0.2\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\marco\AppData\Roaming\Code\User\workspaceStorage\6c62df74886ede1d23c7d640d3c861d2\redhat.java\jdt_ws\ProjetoAcademia (1)_24df5ea0\bin' 'App'

=====
===== MAROMBA'S GYM =====
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção:
```

Descrição: *PrintScreen* da tela de inicialização da classe App.

Caso opte pela primeira opção, são inseridos os dados pessoais do usuário, tais como nome do aluno, seu sobrenome, idade, CPF, mensalidade, CEP, cidade, estado, bairro, rua, número e complemento.

Quadro 20

The screenshot shows a Java application running in Visual Studio Code. The terminal window displays the following text:

```
[ 6 ] - Sair
Opção: 1
=====
Nome do aluno(a): Willian
=====
Sobrenome do aluno(a): Wallace
=====
Idade do aluno(a): 24
=====
CPF: 12345678901
=====
Mensalidade:
[ 1 ] - Civil Marombinha (3 dias) - R$ 60,00
[ 2 ] - Padawan Maromba (5 dias) - R$ 80,00
[ 3 ] - Mestre Jedi Marombão (7 dias) - R$ 100,00
Opção: 3
=====
CEP: 12345678
=====
Cidade: Kurumin
=====
Estado: Acre
=====
Bairro: Tupinambá
=====
Rua: Oiapoque
=====
Estado: Acre
Bairro: Tupinambá
Rua: Oiapoque
Número: 273
Complemento: Sobrado
=====
```

Descrição: *PrintScreen* da tela de inserção dos dados do usuário.

Não obstante, o menu igualmente dá a opção de ver todos os seus dados cadastrados.

Quadro 21

The screenshot shows a Java application running in Visual Studio Code. The terminal window displays the following text:

```
Bairro: Santa Tereza
=====
Rua: Rio de Contas
=====
Número: 170
=====
Complemento: Sobrado
=====
CADASTRO REALIZADO!
=====
Deseja ver todos os seus dados cadastrados [S/N]? s
=====
Nome do aluno(a): Marco T?lio
Sobrenome: Alves
Idade: 16
CPF: 123456789
Matrícula: 202200
Valor da mensalidade: R$ 100.0
CEP: 46100000
Cidade: Brumado
Estado: Bahia
Bairro: Santa Tereza
Rua: Rio de Contas
Número: 170
Complemento: Sobrado
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 
```

Descrição: *PrintScreen* da tela mostrando todos os dados do usuário cadastrados.

Caso selecione a tecla número dois, verá todos os treinos disponíveis, tais como peito, costas, perna e abdômen, sendo que cada um deles detém um índice próprio, que revela os treinos específicos de cada um.

Quadro 22

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda Appjava - ProjetoAcademia (1) - Visual Studio Code
PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 2
=====
Treinos:
[ 1 ] - Peito
[ 2 ] - Costas
[ 3 ] - Perna
[ 4 ] - Abdômem
Opção: 1
=====
VOADOR PEITORAL 1X12, 1X10, 1X9, 1X8 , 1X6 (AUMENTANDO CARGA)
SUPINO RETO 3X12 - DESCANSA 10SEG. - MAIS 10 REPETIÇÕES (RESTPAUSE)
FLAV INCLINADO 3X12.10.8 (DROP)
CRUCIFIXO COM HALTER 3X15
TRÍCEPS FRANÇÊS - POLIA ALTA 3X15
TRÍCEPS PULLEY 3X12.10.8 (DROP)
TRÍCEPS COICE 3X15
DESENVOLVIMENTO COM BARRA 3X15 (RESTPAUSE)
REMADA ALTA NO CROSS 3X12
ELEVACÃO FRONTAL COM A BARRA H 3X15
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção:

Ln 8, Col 47 Espaços: 4 UTF-8 LF ( ) Java Go Live
```

```
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda Appjava - ProjetoAcademia (1) - Visual Studio Code
PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 2
=====
Treinos:
[ 1 ] - Peito
[ 2 ] - Costas
[ 3 ] - Perna
[ 4 ] - Abdômem
Opção: 2
=====
PUKADA ALTA COM TRIÂNGULO 3X12.10.8 (DROP - REDUZINDO CARGA)
CRUCIFIXO INVERSO COM HALTER 3X15
REMADA CURVADA NO CROSS 3X12 - DESCANSA 10 SEG. - FAZER ATE A FALHA
PUKADA ALTA 3X12 (RESTPAUSE)
ROSCA DIRETA COM A BARRA W 3X12
ROSCA UNILATERAL NO BANCO SCOOT 3X15
ROSCA COMBINADA 3X15 + ENCOLHIMENTO DE OMBROS
ROSCA INVERSA NA POLIA 3X12.10.8 (DROP)
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção:

Ln 8, Col 47 Espaços: 4 UTF-8 LF ( ) Java Go Live
```

```
===== MAROMBA'S GYM =====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 2
=====
Treinos:
[ 1 ] - Peito
[ 2 ] - Costas
[ 3 ] - Perna
[ 4 ] - Abdômem
Opção: 3
=====
AGACHAMENTO NO SMITT 3X12.10.8 - DESCANSO DE 10 SEG
LEG 45 graus 3X12 (RESPAUSE)
CADEIRA EXTENSORA 3X14.12.10 (DROP)
CADEIRA ABDUTORA 3X20 - ISOMETRIA DE 30 segs. - FALHA
MESA FLEXORA 3X12.10.8 (DROP) + AFUNDO LIVRE 3X12
HACK INVERSO 3X12 + CADEIRA FLEXORA 3X15
PANTURRILHA UNI. NO SMITT 3X20
PANTURRILHA BANCO 3X25
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 1
=====
CADEIRA EXTENSORA 3X14.12.10 (DROP)
CADEIRA ABDUTORA 3X20 - ISOMETRIA DE 30 segs. - FALHA
MESA FLEXORA 3X12.10.8 (DROP) + AFUNDO LIVRE 3X12
HACK INVERSO 3X12 + CADEIRA FLEXORA 3X15
PANTURRILHA UNI. NO SMITT 3X20
PANTURRILHA BANCO 3X25
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 2
=====
Treinos:
[ 1 ] - Peito
[ 2 ] - Costas
[ 3 ] - Perna
[ 4 ] - Abdômem
Opção: 4
=====
ABS COM ANILHA NO DECLINADO 3X20
ABS OBLÍQUO 3X15
ABS BICICLETA 3X15
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 1
```

Descrição: *PrintScreen* exibindo todos os treinos disponibilizados pela academia.

Porventura, caso alguém opte pelo terceiro item, poderá visualizar os instrutores já cadastrados no sistema da academia.

Quadro 23

```

[ 6 ] - Sair
Opção: 2
=====
Treinos:
[ 1 ] - Peito
[ 2 ] - Costas
[ 3 ] - Perna
[ 4 ] - Abdômem
Opção: 4
=====
ABS COM ANILHA NO DECLINADO 3X20
ABS OBLÍQUO 3X15
ABS BICICLETA 3X15
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 3
=====
Nome: Godofreudo Marombinha | Código: 100000 | Horário: 5:00 às 11:00
Nome: Jubileusson Maromba | Código: 100001 | Horário: 11:00 às 17:00
Nome: Jhereynt Marombão | Código: 100002 | Horário: 17:00 às 23:00
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 

```

Descrição: Exibição dos nomes dos treinadores cadastrados na academia e seus respectivos códigos e horários disponíveis.

E, na quarta opção, encontra-se o endereço da academia.

Quadro 24

```

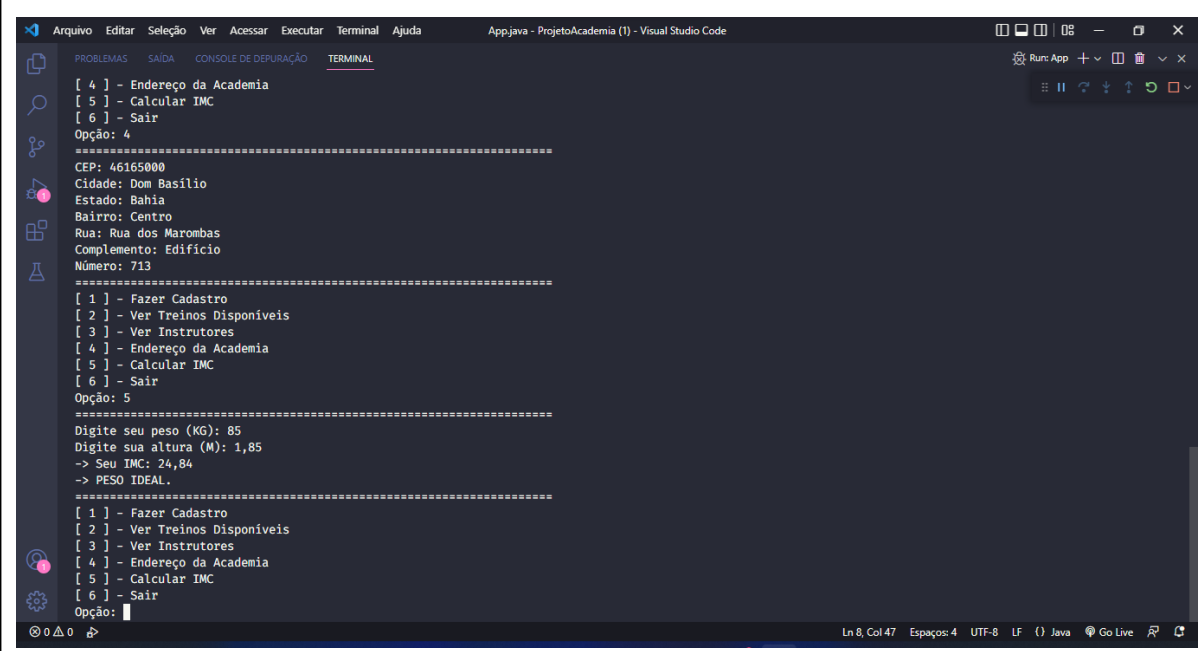
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 3
=====
Nome: Godofreudo Marombinha | Código: 100000 | Horário: 5:00 às 11:00
Nome: Jubileusson Maromba | Código: 100001 | Horário: 11:00 às 17:00
Nome: Jhereynt Marombão | Código: 100002 | Horário: 17:00 às 23:00
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 4
=====
CEP: 46165000
Cidade: Dom Basílio
Estado: Bahia
Bairro: Centro
Rua: Rua dos Marombas
Complemento: Edifício
Número: 713
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 

```

Descrição: Endereço completo da academia.

Por fim, no quinto quesito, existe a possibilidade do cálculo do Índice de Massa Corporal (IMC) do usuário.

Quadro 25

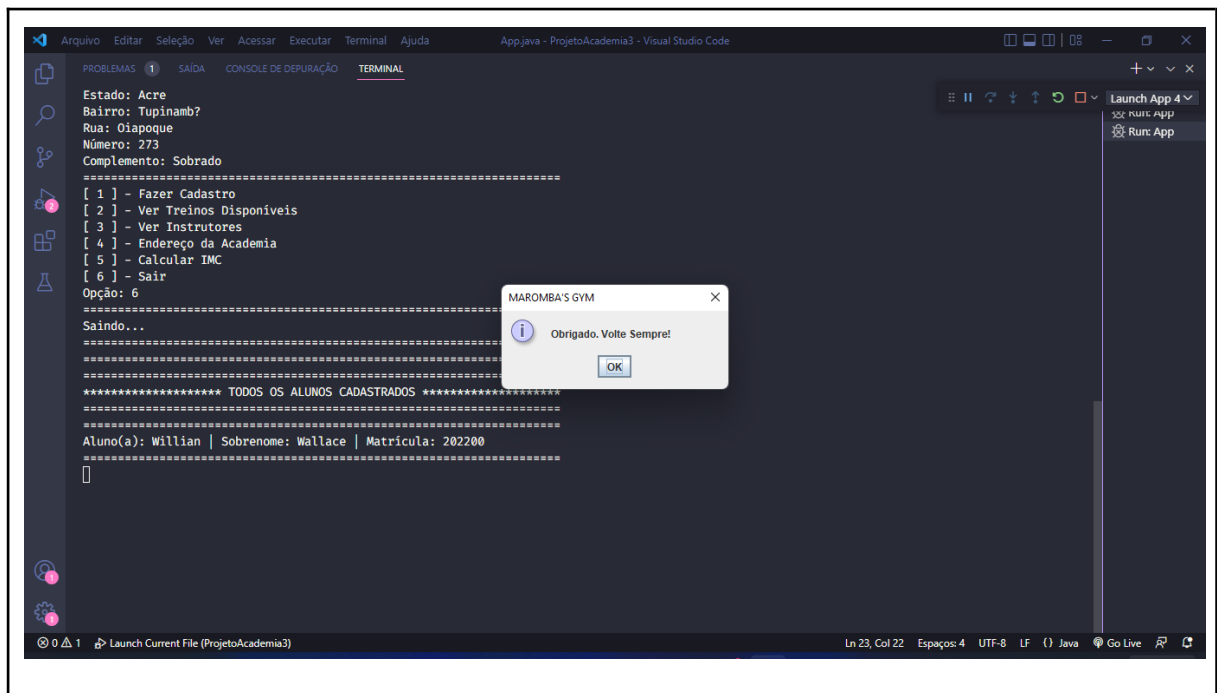


```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  Terminal  Ajuda  App.java - ProjetoAcademia (1) - Visual Studio Code
PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 4
=====
CEP: 46165000
Cidade: Dom Basílio
Estado: Bahia
Bairro: Centro
Rua: Rua dos Marombas
Complemento: Edifício
Número: 713
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 5
=====
Digite seu peso (KG): 85
Digite sua altura (M): 1,85
-> Seu IMC: 24,84
-> PESO IDEAL.
=====
[ 1 ] - Fazer Cadastro
[ 2 ] - Ver Treinos Disponíveis
[ 3 ] - Ver Instrutores
[ 4 ] - Endereço da Academia
[ 5 ] - Calcular IMC
[ 6 ] - Sair
Opção: 
```

Descrição: Demonstração do cálculo do IMC de um usuário.

Ao sair do programa, quando pressionar a tecla seis, aparecerá ao usuário as seguintes mensagens.

Quadro 26



Descrição: *PrintScreen* da mensagem exibida após o término da execução do programa.

## 5. Referências:

1. BLOCH, Joshua. **Effective java (the java series)**. Prentice Hall PTR, 2008.
2. WINSTON, Patrick Henry; NARASIMHAN, Sundar. **On to Java**. Addison Wesley Longman Publishing Co., Inc., 1996.
3. DARWIN, Ian F. **Java cookbook**. " O'Reilly Media, Inc.", 2004.