

**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



**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 – 2° 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**

1. **Introdução……………………………………………………………………………04**
2. **Projeto..………………………………………………………………………………04**
3. **Desenvolvimento….………………………………………………………………….04**
4. **Conclusão…………………………………………………………………………….13**
5. **Referências…………………………………………………………………………...19**
6. **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.

1. **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.

Quadro 01

|  |
| --- |

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.

1. **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 "endereço", 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 “matrícula” 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 , 1X6 (AUMENTANDO CARGA)\nSUPINO RETO 3X12 - DESCANSA 10SEG. - MAIS 10 REPETIÇOES (RESTPAUSE)\nFLAY INCLINADO 3X12.10.8 (DROP)\nCRUCIFIXO COM HALTER 3X15\nTRÍCEPS FRANÇÊS - POLIA ALTA 3X15\nTRÍCEPS PULLEY 3X12.10.8 (DROP)\nTRÍCEPS COICE 3X15\nDESENVOLVIMENTO COM BARRA 3X15 (RESTPAUSE)\nREMADA ALTA NO CROSS 3X12\nELEVAÇÃO FRONTAL COM A BARRA H 3X15\n");* 4. *}* 5. *public void TreinoCostas() {* 6. *System.out.print("PUXADA ALTA COM TRI NGULO 3X12.10.8 (DROP - REDUZINDO CARGA)\nCRUCIFIXO INVERSO COM HALTER 3X15\nREMADA CURVADA NO CROSS 3X12 - DESCANSA 10 SEG. - FAZER ATE A FALHA\nPUXADA ALTA 3X12 (RESTPAUSE)\nROSCA DIRETA COM A BARRA W 3X12\nROSCA UNILATERAL NO BANCO SCOOT 3X15\nROSCA COMBINADA 3X15 + ENCOLHIMENTO DE OMBROS\nROSCA INVERSA NA POLIA 3X12.10.8 (DROP)\n");* 7. *}* 8. *public void TreinoPerna() {* 9. *System.out.print("AGACHAMENTO NO SMITT 3X12.10.8 - DESCANSO DE 10 SEG\nLEG 45 graus 3X12 (RESTPAUSE)\nCADEIRA EXTENSORA 3X14.12.10 (DROP)\nCADEIRA ABDUTORA 3X20 - ISOMETRIA DE 30 segs. - FALHA\nMESA FLEXORA 3X12.10.8 (DROP) + AFUNDO LIVRE 3X12\nHACK 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 + "\nEstado: " + this.estado + "\nBairro: " + this.bairro + "\nRua: " + this.rua + "\nComplemento: " + this.complemento + "\nNúmero: \n" + this.numero;* 4. *}* 5. *public String DadosEnderecoNecessarios() {* 6. *return "CEP: " + this.cep + "\nCidade: " + this.cidade + "\nEstado: " + this.estado;* 7. *}* 8. *public String DadosEnderecoAdcionais() {* 9. *return "Bairro: " + this.bairro + "\nRua: " + this.rua + "\nComplemento: " + this.complemento + "\nNúmero: " + this.numero;* 10. *}* 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. *...* |
| --- |

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

1. **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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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.

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