

Classes e Objetos

Exercícios

Vinicius Takeo Friedrich Kuwaki

Universidade do Estado de Santa Catarina

Exercício 1

Resolução Exercício 1

Exercício 2

Resolução Exercício 2

Exercícios

1. Uma pessoa possui nome, idade (em anos), altura (em metros) e massa (em kilogramas). Implemente em Java duas classes, uma que representa a Pessoa, com seus devidos atributos e outra que representa um Grupo de pessoas. A classe Grupo possui um número fixo de pessoas (fica a seu cargo escolher). Além do mais, a classe Pessoa deve possuir um método para calcular o IMC (índice de massa corporal), que é calculado pela seguinte fórmula:

$$IMC = \frac{massa}{altura^2} \quad (1)$$

A classe Grupo também deve possuir um método que exibe as pessoas em ordem decrescente de IMC.

Exercício 1

Resolução Exercício 1

Exercício 2

Resolução Exercício 2

Resolução Exercício 1

- Primeiro vamos definir a classe Pessoa;
- Esta terá quatro atributos como descrito no enunciado:

```
public class Pessoa {  
    private String nome;  
    private int idade;  
    private float altura;  
    private float massa;
```

- Utilizaremos um construtor vazio:

```
public Pessoa() {  
    }  
}
```

- Todos os atributos devem possuir seus métodos getters e setters (boas práticas de programação orientada a objetos):

Resolução Exercício 1

```
public String getNome() {  
    return this.nome;  
}  
  
public void setNome(String nome) {  
    this.nome = nome;  
}  
  
public int getIdade() {  
    return this.idade;  
}  
  
public void setIdade(int idade) {  
    this.idade = idade;  
}  
  
public float getAltura() {  
    return this.altura;  
}  
  
public void setAltura(float altura) {  
    this.altura = altura;  
}
```

Resolução Exercício 1

```
public float getMassa() {  
    return this.massa;  
}  
  
public void setMassa(float massa) {  
    this.massa = massa;  
}
```

- E também é necessário implementar o método para calcular o IMC, no qual retorna um resultado de ponto flutuante, dado pela divisão entre a massa e o quadrado da altura da pessoa:

```
public float calculaImc() {  
    return this.massa / (this.altura * this.altura);  
}
```

- Agora iremos definir a classe Grupo;
- Esta irá possuir um número fixo de pessoas;
- Eu escolhi o número 5;

Resolução Exercício 1

- Logo, teremos um atributo que será um vetor de objetos do tipo Pessoa. De tamanho 5.
- Haverá também um atributo do tipo inteiro para controlar a quantidade de pessoas no array.
- A classe Grupo também possuirá um construtor vazio:

```
public class Grupo {  
    private Pessoa[] pessoas = new Pessoa[5];  
    private int numeroPessoas = 0;  
  
    public Grupo() {  
  
    }  
}
```

- Como o atributo pessoas é um array, seu set recebe um objeto a ser adicionado e a cada inserção incrementa em um a variável que controla o número de objetos do array.

Resolução Exercício 1

```
public void setPessoa(Pessoa p) {  
    if (this.numeroPessoas < 5) {  
        pessoas[this.numeroPessoas] = p;  
        this.numeroPessoas++;  
    }  
}
```

- Também foi implementado um método ordena, no qual realiza um bubble sort utilizando do método **calculaImc()** da classe Pessoa:

Resolução Exercício 1

```
public void ordena() {  
    for (int i = 0; i < 5; i++) {  
        for (int j = i + 1; j < 5; j++) {  
            if (this.pessoas[j].calculaImc() > this.pessoas[i].calculaImc()) {  
                Pessoa aux = this.pessoas[j];  
                this.pessoas[j] = this.pessoas[i];  
                this.pessoas[i] = aux;  
            }  
        }  
    }  
}
```

- Observe que o método não retorna nada, ele apenas altera as posições do vetor.
- Agora para testar, criaremos uma classe Main, e instanciaremos um objeto do tipo Grupo:

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Grupo g = new Grupo();  
  
    }  
}
```

Resolução Exercício 1

- Criaremos cinco instâncias da classe Pessoa, para adicionar no vetor da Grupo:

```
Pessoa p1 = new Pessoa();  
p1.setNome("Joao");  
p1.setAltura(1.70 f);  
p1.setIdade(19);  
p1.setMassa(70.0 f);
```

```
Pessoa p2 = new Pessoa();  
p2.setNome("Julia");  
p2.setAltura(1.65 f);  
p2.setIdade(19);  
p2.setMassa(62.5 f);
```

```
Pessoa p3 = new Pessoa();  
p3.setNome("Marcos");  
p3.setAltura(1.79 f);  
p3.setIdade(20);  
p3.setMassa(75);
```

Resolução Exercício 1

```
Pessoa p4 = new Pessoa();  
p4.setNome("Luiza");  
p4.setAltura(1.68f);  
p4.setIdade(20);  
p4.setMassa(65);
```

```
Pessoa p5 = new Pessoa();  
p5.setNome("Leticia");  
p5.setAltura(1.66f);  
p5.setIdade(20);  
p5.setMassa(69);
```

- Para definir valores do tipo float é necessário incluir o "f" após o número!
- Agora é necessário adicionar todas as pessoas a Grupo "t":

```
g.setPessoa(p1);  
g.setPessoa(p2);  
g.setPessoa(p3);  
g.setPessoa(p4);  
g.setPessoa(p5);
```

Resolução Exercício 1

- Realizar a ordenação delas com o método **ordena()** da classe Grupo:

```
g.ordena();
```

- E por fim, exibiremos no console os atributos e valores após a ordenação:

```
for (int i = 0; i < 5; i++) {  
    System.out.println("Nome: " + g.getPessoas()[i].getNome());  
    System.out.println("Altura: " + g.getPessoas()[i].getAltura());  
    System.out.println("Idade: " + g.getPessoas()[i].getIdade());  
    System.out.println("Massa: " + g.getPessoas()[i].getMassa());  
    System.out.println("IMC: " + g.getPessoas()[i].calculaImc() + "\n");  
}  
}
```

Exercício 1

Resolução Exercício 1

Exercício 2

Resolução Exercício 2

Exercicio 2

2. Faça um programa em Java, orientado a objetos que gerencie as informações de uma pet-shop. Esse programa deve ser capaz de listar os animais atendidos pelos veterinários com todas as informações relacionadas a eles, além de permitir o cadastro de novos animais e donos. O aluno deve desenvolver esse programa como uma aplicação em duas camadas, contendo uma camada com as classes de dados e outra camada contendo a interface com o usuário. Todos os dados referentes ao pet-shop deverão ser fornecidos por meio da interface com o usuário. Segue abaixo a lista de classes que o programa deve conter, com seus respectivos atributos e métodos:

Exercicio 2

Classe Endereço	
Atributos	Métodos
rua numero bairro cidade estado cep	gets()/sets()

Tabela 1: Classe Endereço

Exercicio 2

Classe Veterinário	
Atributos	Métodos
nome salario Endereço Animais[] quantidadeAnimais	gets()/sets()

Tabela 2: Classe Veterinário: Essa classe possui uma lista (array) de Animais, que representam os animais atendidos pelo veterinario, a quantidade de itens do array é controlada pelo atributo *quantidadeAnimais*. Além de possuir um objeto do tipo Endereço.

Exercicio 2

Classe Dono	
Atributos	Métodos
nome Endereço cpf	gets()/sets()

Tabela 3: Classe Dono: essa classe representa o dono de um animal. Todo dono possui um objeto do tipo Endereço.

Classe Animal	
Atributos	Métodos
nome Dono especie descricao	gets()/sets()

Exercicio 2

Tabela 4: Classe Animal: todo animal possui um dono, uma especie (String) e uma descrição sobre o animal (também uma String)

Classe SistemaPetShop	
Atributos	Métodos
Veterinarios[50]	cadastrar Veterinario() mostrar Veterinarios() cadastrar Endereço do Veterinario()
quantidadeVeterinarios	cadastrar Animal() mostrar Animais() cadastrarDono() cadastrar Endereço do Dono()

Tabela 5: Classe SistemaPetShop: abaixo está a descrição completa dos métodos:

Exercicio 2

- **void cadastrarVeterinario():** esse método requisita ao usuário as informações relacionadas ao veterinario, sendo estas: nome e salário. O sistema então adiciona ao array de veterinarios e incrementa o atributo quantidadeVeterinarios
- **mostrarVeterinarios():** esse método deve exibir todos os veterinarios cadastrados no sistema, cada um contendo um número que o identifique (a posição no vetor).
- **void cadastrarEnderecoVeterinario():** esse método deve exibir os veterinarios já cadastrados e o usuário deve escolher qual veterinario ele quer cadastrar o endereço. Após escolhido o veterinário, é requisitado ao usuário as informações referentes ao endereço: rua, numero, bairro, cidade, estado e cep. Após o usuário digitar essas informações, o endereço é cadastrado ao veterinário escolhido.

Exercicio 2

- **cadastrarAnimal():** esse método deve exibir os veterinarios ja cadastrados e o usuário deve escolher qual veterinario ele quer cadastrar um novo animal. Após escolhido o veterinário, é requisitado ao usuário as informações referentes ao animal: nome, especie e descrição. Após o usuário digita essas informações, o animal é cadastrado ao funcionário escolhido e o atributo quantidadeAnimais é incrementado.
- **mostrarAnimais():** esse método deve exibir os veterinarios ja cadastrados e o usuário deve escolher qual veterinario ele deseja visualizar os animais atendidos por ele, cada um contendo um número que o identifique (a posição no vetor).
- **cadastrarDono():** esse método deve exibir os animais já cadastrados e o usuário deve escolher qual animal ele deseja cadastrar um dono. Após escolhido o animal, é requisitado ao usuário as informações referentes ao dono: nome e cpf. Após digitadas essas informações, o dono é cadastrado ao animal escolhido.

Exercicio 2

- **cadastrarEnderecoDono():** esse método deve exibir os animais cadastrados e o usuário deve escolher qual animal ele deseja cadastrar o endereço do dono. Após escolhido o animal, é exibido as informações referentes ao dono: nome e cpf, e requisitado as informações referentes ao endereço: rua, numero, bairro, cidade, estado e cep. Após o usuário digitar essas informações, o endereço é cadastrado ao dono do animal escolhido.

Exercício 1


Resolução Exercício 1

Exercício 2

Resolução Exercício 2

Resolução Exercício 2

- A resolução do exercício 2 estará disponível ao final da aula.

 KUWAKI, V. T. F. Modelo de slides udesc lattex. In: . [S.l.]: Disponível em: <<https://github.com/takeofriedrich/slidesUdescLattex>>. Acesso em: 24 jan. 2020.

Duvidas:
Vinicius Takeo Friedrich Kuwaki
vinicius.kuwaki@edu.udesc.br
github.com/takeofriedrich