Jonatha Salles Menezes 2211312125

Departamento de Computação.

**Disciplina: Programação Orientada a Objetos I.**

Rio de Janeiro

Maio de 2023

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO – UERJ - ZO

Jonatha Salles Menezes

**Programação Orientada a Objetos.**

Texto apresentado como pré-requisito da disciplina de Programação Orientada a Objetos I dos Cursos de Computação do DEPCOMP / FCEE / UERJ-ZO.

Professor Engo. Carlos A. Sicsú A. do Nascimento - DSc.

Rio de Janeiro

Maio de 2023

**SUMÁRIO**

Introdução.............................................................. 4

1 O problema...................................................................................... pag

2 O Desenvolvimento do Projeto....................................................... pag

1.1 Objetivo................................................................................. pag

1.2 Desenvolvimento.................................................................. pag

....

....

3 Conclusões..................................................................................... pag

3.1 Conclusões........................................................................... pag

3.2 Melhorias Futuras................................................................. pag

4 Bibliografia..................................................................................... pag

**INTRODUÇÃO**

Através desse trabalho, será desenvolvido um sistema capaz de armazenar diversas informações de três tipos distintos de veículos aquáticos.

**CAPÍTULO 1 – O Problema.**

**Trabalho obrigatório para prova AV1 (1.0 pt).**

**Pode ser desenvolvido em duplas ou individualmente.**

**Prazo de entrega:**

* **19/05/2023**

**Modo de entrega: Entregar este arquivo devidamente preenchido com as suas soluções na tarefa da equipe no TEAMS dentro do prazo (se for feito em dupla é obrigatório que os todos os componentes entreguem o mesmo arquivo individualmente e dentro do prazo).**

***Sejam as seguintes classes:***

|  |  |  |
| --- | --- | --- |
| **Classes** | **Atributos** | **Métodos** |
| **Lancha** | . Marca: Texto  . Modelo: Texto  . Preço: real  . NumeroPassageiros: inteiro  . Banheiro : lógico | . Métodos de acesso (Setters & Getters para todos os atributos)  . Cadastrar (...) - Todos os atributos do objeto  . Imprimir ( ) - Exibir todos os atributos do objeto  . Entrada( ) – Efetua a entrada de dados via teclado. |
| **Iate** | . Marca: Texto  . Modelo: Texto  . Preço: real  . NumeroPassageiros: inteiro  . NumeroTripulantes : inteiro | . Métodos de acesso (Setters & Getters para todos os atributos)  . Cadastrar (...) - Todos os atributos do objeto  . Imprimir ( ) - Exibir todos os atributos do objeto  . Entrada( ) – Efetua a entrada de dados via teclado. |
| **JetSki** | . Marca: Texto  . Modelo: Texto  . Preço: real  . TipoCasco: Texto | . Métodos de acesso (Setters & Getters para todos os atributos)  . Cadastrar (...) - Todos os atributos do objeto  . Imprimir ( ) - Exibir todos os atributos do objeto  . Entrada( ) – Efetua a entrada de dados via teclado. |

**Baseado nas classes apresentadas, pede-se:**

1. Crie uma superclasse que seja capaz de atender às necessidades das subclasses **Lancha**, **JetSki** e **Iate**, com pelo menos 8 métodos construtores;
2. Criar as subclasses **Lancha**, **Iate** e **JetSki** e outras classes caso necessário, utilizando os conceitos de herança a partir da superclasse criada e com o mínimo 8 métodos construtores (cada);
3. Cada subclasse deverá reaproveitar no mínimo 3 métodos construtores da superclasse;
4. Não se esqueça de reaproveitar, sempre que possível, métodos da superclasse estendendo suas ações (imprimir, cadastrar, etc);
5. Crie uma aplicação, que crie pelo menos 3 objetos **Lanchas**, 3 objetos **JetSkis** e 3 objetos **Iates**, sendo que estes objetos deverão ser criados com o uso dos diferentes construtores disponibilizados por cada objeto.
6. Aplique as regras da boa prática de programação em java.

**CAPÍTULO 2 – O Desenvolvimento do Projeto.**

**2.1 Objetivo.**

O objetivo deste projeto é armazenar e exibir diversos dados distintos pertencentes a um de três tipos de veículos aquáticos.

**2.2 Descrição.**

Para o desenvolvimento do projeto foram usados os conceitos de programação orientada a objetos herança, métodos *getters* e *setters e* métodos construtores, desenvolvido na linguagem Java, na IDE NetBeans 17.

**2.3 Código principal (Controle - Aplicação):**

public class ProjetoAquatico

{

public static void main(String[] args)

{

JetSki jet1 = new JetSki();

JetSki jet2 = new JetSki(2000);

JetSki jet3 = new JetSki("Honda", "Crock", 25000.30, "Tipo 1");

jet1.cadastrar("Hugo", "Apache", 30000, "Tipo 2");

jet2.setMarca("JetEsqui");

jet2.setModelo("Iron");

jet2.setTipoCasco("Tipo 3");

Lancha lan1 = new Lancha();

Lancha lan2 = new Lancha(true);

Lancha lan3 = new Lancha("Lego", "Mod", 2320.5, 20, true);

lan1.cadastrar("Raluca", "Print", 100000, 4, false);

lan2.setMarca("Mark");

lan2.setModelo("Muerto");

lan2.setNumeroPassageiros(11);

lan2.setPreco(200000);

Iate iat1 = new Iate();

Iate iat2 = new Iate(21000.4, 3);

Iate iat3 = new Iate("EA", "Ravenclaw", 350000, 10, 5);

iat1.cadastrar("Lenovo", "PR0", 14000, 13, 4);

iat2.setMarca("Nova");

iat2.setModelo("MKIII");

iat2.setNumeroPassageiros(9);

System.out.println("\n=== JetSkis ===");

jet1.imprimir();

System.out.print("\n");

jet2.imprimir();

System.out.print("\n");

jet3.imprimir();

System.out.println("\n=== Lanchas ===");

lan1.imprimir();

System.out.print("\n");

lan2.imprimir();

System.out.print("\n");

lan3.imprimir();

System.out.println("\n=== Iates ===");

iat1.imprimir();

System.out.print("\n");

iat2.imprimir();

System.out.print("\n");

iat3.imprimir();

}

}

**2.4 Códigos das classes:**

**2.4.1 – Classe VeiculoAquatico.**

public class VeiculoAquatico

{

String marca, modelo;

double preco;

public VeiculoAquatico() { }

public VeiculoAquatico(String marca)

{

setMarca(marca);

}

public VeiculoAquatico(double preco)

{

setPreco(preco);

}

public VeiculoAquatico(String marca, double preco)

{

setMarca(marca);

setPreco(preco);

}

public VeiculoAquatico(String marca, String modelo)

{

setMarca(marca);

setModelo(modelo);

}

public VeiculoAquatico(double preco, String modelo)

{

setPreco(preco);

setModelo(modelo);

}

public VeiculoAquatico(double preco, String modelo, String marca)

{

setPreco(preco);

setModelo(modelo);

setMarca(marca);

}

public VeiculoAquatico(String marca, String modelo, double preco)

{

setMarca(marca);

setModelo(modelo);

setPreco(preco);

}

public void setMarca(String marca)

{

if (!marca.isEmpty())

{

this.marca = marca;

}

}

public void setModelo(String modelo)

{

if (!modelo.isEmpty())

{

this.modelo = modelo;

}

}

public void setPreco(double preco)

{

if (preco > 0.0)

{

this.preco = preco;

}

}

public String getMarca()

{

return marca;

}

public String getModelo()

{

return modelo;

}

public double getPreco()

{

return preco;

}

public void imprimir()

{

System.out.println("Marca -> " + getMarca());

System.out.println("Modelo -> " + getModelo());

System.out.println("Preço -> " + getPreco());

}

public void cadastrar(String marca, String modelo, double preco)

{

setMarca(marca);

setModelo(modelo);

setPreco(preco);

}

public void entrada()

{

Scanner ent = new Scanner(System.in);

System.out.print("Marca -> ");

setMarca(ent.nextLine());

System.out.print("Modelo -> ");

setModelo(ent.nextLine());

System.out.print("Preço -> ");

setPreco(Double.parseDouble(ent.nextLine()));

}

}

**2.4.2 – Classe JetSki.**

public class JetSki extends VeiculoAquatico

{

String tipoCasco;

public JetSki() { }

public JetSki(String marca)

{

super(marca);

}

public JetSki(String marca, double preco)

{

super(marca, preco);

}

public JetSki(String marca, double preco, String tipoCasco)

{

super(marca, preco);

setTipoCasco(tipoCasco);

}

public JetSki(double preco)

{

super(preco);

}

public JetSki(String marca, String tipoCasco)

{

super(marca);

setTipoCasco(tipoCasco);

}

public JetSki(String marca, String modelo, String tipoCasco)

{

super(marca, modelo);

setTipoCasco(tipoCasco);

}

public JetSki(String marca, String modelo, double preco, String tipoCasco)

{

super(marca, modelo, preco);

setTipoCasco(tipoCasco);

}

public void setTipoCasco(String tipoCasco)

{

if (!tipoCasco.isEmpty())

{

this.tipoCasco = tipoCasco;

}

}

public String getTipoCasco()

{

return tipoCasco;

}

public void imprimir()

{

super.imprimir();

System.out.println("Tipo do casco -> " + getTipoCasco());

}

public void cadastrar(String marca, String modelo, double preco, String tipoCasco)

{

super.cadastrar(marca, modelo, preco);

setTipoCasco(tipoCasco);

}

public void entrada()

{

Scanner ent = new Scanner(System.in);

super.entrada();

System.out.print("Tipo do casco -> ");

setTipoCasco(ent.nextLine());

}

}

**2.4.3 – Classe VeiculoAquaticoColetivo.**

public class VeiculoAquaticoColetivo extends VeiculoAquatico

{

int numeroPassageiros;

public VeiculoAquaticoColetivo() { }

public VeiculoAquaticoColetivo(int numeroPassageiros)

{

setNumeroPassageiros(numeroPassageiros);

}

public VeiculoAquaticoColetivo(double preco)

{

super(preco);

}

public VeiculoAquaticoColetivo(double preco, int numeroPassageiros)

{

super(preco);

setNumeroPassageiros(numeroPassageiros);

}

public VeiculoAquaticoColetivo(double preco, String modelo, int numeroPassageiros)

{

super(preco, modelo);

setNumeroPassageiros(numeroPassageiros);

}

public VeiculoAquaticoColetivo(String marca, String modelo, int numeroPassageiros)

{

super(marca, modelo);

setNumeroPassageiros(numeroPassageiros);

}

public VeiculoAquaticoColetivo(String marca, double preco, int numeroPassageiros)

{

super(marca, preco);

setNumeroPassageiros(numeroPassageiros);

}

public VeiculoAquaticoColetivo(String marca, String modelo, double preco, int numeroPassageiros)

{

super(marca, modelo, preco);

setNumeroPassageiros(numeroPassageiros);

}

public void setNumeroPassageiros(int numeroPassageiros)

{

if (numeroPassageiros > 0)

{

this.numeroPassageiros = numeroPassageiros;

}

}

public int getNumeroPassageiros()

{

return numeroPassageiros;

}

public void imprimir()

{

super.imprimir();

System.out.println("Número de passageiros -> " + getNumeroPassageiros());

}

public void cadastrar(String marca, String modelo, double preco, int numeroPassageiros)

{

super.cadastrar(marca, modelo, preco);

setNumeroPassageiros(numeroPassageiros);

}

public void entrada()

{

Scanner ent = new Scanner(System.in);

super.entrada();

System.out.print("Número de passageiros -> ");

setNumeroPassageiros(Integer.parseInt(ent.nextLine()));

}

}

**2.4.4 – Classe Lancha.**

public class Lancha extends VeiculoAquaticoColetivo

{

boolean banheiro;

public Lancha() { }

public Lancha(int numeroPassageiros)

{

super(numeroPassageiros);

}

public Lancha(double preco)

{

super(preco);

}

public Lancha(boolean banheiro)

{

setBanheiro(banheiro);

}

public Lancha(double preco, int numeroPassageiros)

{

super(preco, numeroPassageiros);

}

public Lancha(double preco, int numeroPassageiros, boolean banheiro)

{

super(preco, numeroPassageiros);

setBanheiro(banheiro);

}

public Lancha(boolean banheiro, String marca, String modelo, int numeroPassageiros)

{

super(marca, modelo, numeroPassageiros);

setBanheiro(banheiro);

}

public Lancha(String marca, String modelo, double preco, int numeroPassageiros, boolean banheiro)

{

super(marca, modelo, preco, numeroPassageiros);

setBanheiro(banheiro);

}

public void setBanheiro(boolean banheiro)

{

this.banheiro = banheiro;

}

public boolean getBanheiro()

{

return banheiro;

}

public void imprimir()

{

super.imprimir();

String banheiro = this.banheiro? "sim" : "não";

System.out.println("Tem banheiro? -> " + banheiro);

}

public void cadastrar(String marca, String modelo, double preco, int numeroPassageiros, boolean banheiro)

{

super.cadastrar(marca, modelo, preco, numeroPassageiros);

setBanheiro(banheiro);

}

public void entrada()

{

Scanner ent = new Scanner(System.in);

super.entrada();

System.out.println("Tem banheiro? 1 - Sim, 2 - Não ");

setBanheiro(Integer.parseInt(ent.nextLine()) == 1);

}

}

**2.4.5 – Classe Iate.**

public class Iate extends VeiculoAquaticoColetivo

{

int numeroTripulantes;

public Iate() { }

public Iate(int numeroTripulantes)

{

setNumeroTripulantes(numeroTripulantes);

}

public Iate(double preco, int numeroTripulantes)

{

super(preco);

setNumeroTripulantes(numeroTripulantes);

}

public Iate(double preco, int numeroPassageiros, int numeroTripulantes)

{

super(preco, numeroPassageiros);

setNumeroTripulantes(numeroTripulantes);

}

public Iate(String marca, double preco, int numeroPassageiros)

{

super(marca, preco, numeroPassageiros);

}

public Iate(String marca, double preco, int numeroPassageiros, int numeroTripulantes)

{

super(marca, preco, numeroPassageiros);

setNumeroTripulantes(numeroTripulantes);

}

public Iate(String marca, String modelo, int numeroPassageiros, int numeroTripulantes)

{

super(marca, modelo, numeroPassageiros);

setNumeroTripulantes(numeroTripulantes);

}

public Iate(String marca, String modelo, double preco, int numeroPassageiros, int numeroTripulantes)

{

super(marca, modelo, preco, numeroPassageiros);

setNumeroTripulantes(numeroTripulantes);

}

public void setNumeroTripulantes(int numeroTripulantes)

{

if (numeroTripulantes > 0)

{

this.numeroTripulantes = numeroTripulantes;

}

}

public int getNumeroTripulantes()

{

return numeroTripulantes;

}

public void imprimir()

{

super.imprimir();

System.out.println("Número de tripulantes -> " + getNumeroTripulantes());

}

public void cadastrar(String marca, String modelo, double preco, int numeroPassageiros, int numeroTripulantes)

{

super.cadastrar(marca, modelo, preco, numeroPassageiros);

setNumeroTripulantes(numeroTripulantes);

}

public void entrada()

{

Scanner ent = new Scanner(System.in);

super.entrada();

System.out.print("Número de tripulantes -> ");

setNumeroTripulantes(Integer.parseInt(ent.nextLine()));

}

}

**CAPÍTULO 3 – Conclusões.**

**3.1 Conclusões.**

Através deste trabalho, foi possível fixar os conceitos de herança, bem como os métodos construtores, *setters* e *getters*. Uma pequena dificudade encontrada foi alinhar os métodos construtores das subclasses aos das superclasses.

**3.2 Melhorias Futuras.**

O desenvolvimento deste trabalho poderia receber melhorias futuras tais como o armazenamento dos dados de outros veículos aquáticos e veículos com outros meios de locomoção

**Capítulo 4 - Bibliografia.**

* Local (Danki Code cursos) - disponível em:

< https://cursos.dankicode.com/> Acesso em 24 de fevereiro de 2022.

* Apresentações de slide em aula do professor.
* Notas de aula do aluno.