**UNIVERSIDADE VEIGA DE ALMEIDA (UVA)**

**CURSO DE SISTEMAS DE INFORMAÇÃO**

**THIAGO CARDOSO DOS SANTOS RAMOS**

**PROJETO JAVA - CLASSES, SUPERCLASSES E HERANÇA**

Rio de Janeiro – Rio de Janeiro

2025

**UNIVERSIDADE VEIGA DE ALMEIDA (UVA)**

**THIAGO CARDOSO DOS SANTOS RAMOS**

**PROJETO JAVA - CLASSES, SUPERCLASSES E HERANÇA**

**Projeto Java apresentado como parte da avaliação da disciplina de Programação Orientada a Objetos ministrada pelo professor Luciano de Pinna Vieira**

Rio de Janeiro – Rio de Janeiro

2025

**SUMÁRIO**

1 **INTRODUÇÃO**...............................................................................................................3

2 **CONTEÚDO**...................................................................................................................3

3 **GRÁFICO HIERÁRQUICO DAS CLASSES**.............................................................4

4 **CODIFICAÇÃO**.............................................................................................................4

4.1 **Código da Superclasse Veiculo.java**........................................................................4

4.2 C**ódigo da Classe Aviao.java**....................................................................................7

4.3 C**ódigo da Classe Navio.java**..................................................................................10

4.4 C**ódigo da Classe Main.java**...................................................................................14

4.5 **Código da Run.bat**..................................................................................................19

5 **TESTES REALIZADOS**..............................................................................................20

6 **CONCLUSÃO**...............................................................................................................26

7 **REFERÊNCIAS**............................................................................................................27

**1 INTRODUÇÃO**

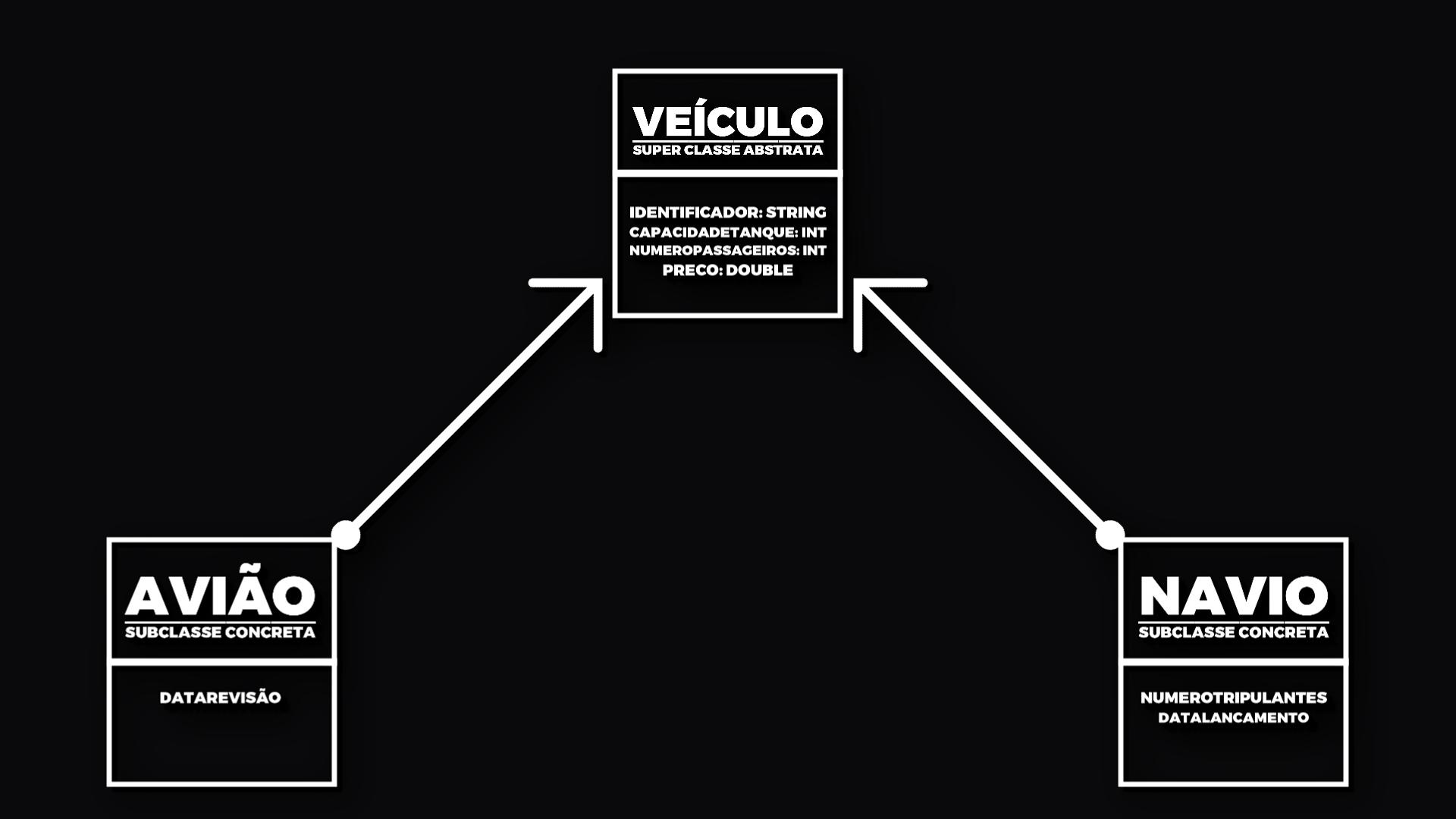
Este trabalho tem como objetivo o desenvolvimento de um sistema em linguagem Java utilizando os conceitos de Programação Orientada a Objetos (POO). O projeto simula um sistema de cadastro de veículos, com ênfase em herança, encapsulamento, uso de construtores, vetores de objetos e tratamento de exceções. A proposta visa reforçar a aplicação prática dos fundamentos da POO por meio da criação de uma superclasse genérica (Veiculo) e duas subclasses específicas (Aviao e Navio), cada uma com atributos próprios e funcionalidades distintas.

**2 CONTEÚDO**

O sistema foi desenvolvido utilizando a linguagem Java, empregando os conceitos fundamentais da Programação Orientada a Objetos. A classe abstrata Veiculo representa a superclasse genérica com atributos comuns a todos os veículos. As classes Aviao e Navio estendem essa superclasse e adicionam atributos e comportamentos específicos, representando um modelo hierárquico de herança.

A classe Main é responsável pela interação com o usuário, permitindo o cadastro de até 10 veículos, listagem, reajuste de preço de aviões com base em percentual e cálculo da razão entre passageiros e tripulantes de navios. Todas as entradas são tratadas com estruturas de exceção (try-catch) para assim, evitar falhas de execução.

**3 GRÁFICO HIERÁRQUICO DAS CLASSES**



**4 CODIFICAÇÃO**

A seguir estão listados os códigos em Java do programa proposto, além do arquivo .bat, que tem como objetivo automatizar a execução do projeto.

**4.1 Código da Superclasse Veiculo.java**

public abstract class Veiculo {

protected String identificador;

protected int capacidadeTanque;

protected int numeroPassageiros;

protected double preco;

public Veiculo() {}

public Veiculo(String identificador, int capacidadeTanque, int numeroPassageiros, double preco) {

this.identificador = identificador;

this.capacidadeTanque = capacidadeTanque;

this.numeroPassageiros = numeroPassageiros;

this.preco = preco;

}

public String getIdentificador() {

return identificador;

}

public void setIdentificador(String identificador) {

this.identificador = identificador;

}

public int getCapacidadeTanque() {

return capacidadeTanque;

}

public void setCapacidadeTanque(int capacidadeTanque) {

this.capacidadeTanque = capacidadeTanque;

}

public int getNumeroPassageiros() {

return numeroPassageiros;

}

public void setNumeroPassageiros(int numeroPassageiros) {

this.numeroPassageiros = numeroPassageiros;

}

public double getPreco() {

return preco;

}

public void setPreco(double preco) {

this.preco = preco;

}

public abstract void entrada();

public abstract void imprimir();

}

**4.2 Código da Classe Aviao.Java**

import java.util.Scanner;

public class Aviao extends Veiculo {

private String dataRevisao;

public Aviao() {

super();

}

public Aviao(String identificador, int capacidadeTanque, int numeroPassageiros, double preco, String dataRevisao) {

super(identificador, capacidadeTanque, numeroPassageiros, preco);

this.dataRevisao = dataRevisao;

}

public String getDataRevisao() {

return dataRevisao;

}

public void setDataRevisao(String dataRevisao) {

this.dataRevisao = dataRevisao;

}

@Override

public void entrada() {

Scanner sc = new Scanner(System.in);

try {

System.out.print("\n--== Adicionando Avião ==--\n");

System.out.print("\nPrefixo: ");

identificador = sc.nextLine();

System.out.print("Capacidade do tanque: ");

capacidadeTanque = Integer.parseInt(sc.nextLine());

System.out.print("Número de passageiros: ");

numeroPassageiros = Integer.parseInt(sc.nextLine());

System.out.print("Preço: R$");

preco = Double.parseDouble(sc.nextLine());

System.out.print("Data de revisão: ");

dataRevisao = sc.nextLine();

} catch (Exception e) {

System.out.println("\n==== AVISO: Erro na entrada de dados. Tente novamente ====\n");

}

}

@Override

public void imprimir() {

System.out.println("\n-----= Dados do Avião =-----\n");

System.out.println("Prefixo: " + this.identificador);

System.out.println("Capacidade do tanque: " + this.capacidadeTanque);

System.out.println("Número de passageiros: " + this.numeroPassageiros);

System.out.println("Preço: R$ " + this.preco);

System.out.println("Data de revisão: " + this.dataRevisao);

}

public void reajustarPreco(double percentual) {

preco += preco \* (percentual / 100);

}

}

**4.3 Código da Classe Navio.Java**

import java.util.Scanner;

public class Navio extends Veiculo {

private int numeroTripulantes;

private String dataLancamento;

public Navio() {

super();

}

public Navio(String identificador, int capacidadeTanque, int numeroPassageiros, double preco, int numeroTripulantes, String dataLancamento) {

super(identificador, capacidadeTanque, numeroPassageiros, preco);

this.numeroTripulantes = numeroTripulantes;

this.dataLancamento = dataLancamento;

}

public int getNumeroTripulantes() {

return numeroTripulantes;

}

public void setNumeroTripulantes(int numeroTripulantes) {

this.numeroTripulantes = numeroTripulantes;

}

public String getDataLancamento() {

return dataLancamento;

}

public void setDataLancamento(String dataLancamento) {

this.dataLancamento = dataLancamento;

}

@Override

public void entrada() {

Scanner sc = new Scanner(System.in);

try {

System.out.print("\n--== Adicionando Navio ==--\n");

System.out.print("\nNome: ");

identificador = sc.nextLine();

System.out.print("Capacidade do tanque: ");

capacidadeTanque = Integer.parseInt(sc.nextLine());

System.out.print("Número de passageiros: ");

numeroPassageiros = Integer.parseInt(sc.nextLine());

System.out.print("Número de tripulantes: ");

numeroTripulantes = Integer.parseInt(sc.nextLine());

System.out.print("Preço: R$");

preco = Double.parseDouble(sc.nextLine());

System.out.print("Data de lançamento: ");

dataLancamento = sc.nextLine();

} catch (Exception e) {

System.out.println("\n==== AVISO: Erro na entrada de dados ====\n");

}

}

@Override

public void imprimir() {

System.out.println("\n-----= Dados do Navio =-----\n");

System.out.println("Nome: " + identificador);

System.out.println("Capacidade do tanque: " + capacidadeTanque);

System.out.println("Número de passageiros: " + numeroPassageiros);

System.out.println("Número de tripulantes: " + numeroTripulantes);

System.out.println("Preço: R$ " + preco);

System.out.println("Data de lançamento: " + dataLancamento);

}

public double passageirosPorTripulante() {

return (double) numeroPassageiros / numeroTripulantes;

}

}

**4.4 Código da Classe Main.Java**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Veiculo[] veiculos = new Veiculo[10];

int opcao;

int contador = 0;

do {

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

System.out.println("1 - Adicionar Avião");

System.out.println("2 - Adicionar Navio");

System.out.println("3 - Listar Veículos");

System.out.println("4 - Reajustar Preço de Avião");

System.out.println("5 - Calcular Quantidade de Passageiros por Tripulante (Navio)");

System.out.println("0 - Sair");

System.out.print("\nSeleção: ");

try {

opcao = Integer.parseInt(sc.nextLine());

switch (opcao) {

case 1:

if (contador < 10) {

Aviao aviao = new Aviao();

aviao.entrada();

veiculos[contador++] = aviao;

System.out.println("\n---- Avião adicionado com sucesso ----\n");

} else {

System.out.println("\n==== AVISO: Limite de veículos atingido ====\n");

}

break;

case 2:

if (contador < 10) {

Navio navio = new Navio();

navio.entrada();

veiculos[contador++] = navio;

System.out.println("\n---- Navio adicionado com sucesso ----\n");

} else {

System.out.println("\n==== AVISO: Limite de veículos atingido ====\n");

}

break;

case 3:

for (int i = 0; i < contador; i++) {

veiculos[i].imprimir();

}

break;

case 4:

System.out.print("\nÍndice do Avião para reajustar (0 a " + (contador - 1) + "): ");

int idx = Integer.parseInt(sc.nextLine());

if (idx >= 0 && idx < contador && veiculos[idx] instanceof Aviao) {

System.out.print("Percentual de reajuste: ");

double p = Double.parseDouble(sc.nextLine());

((Aviao) veiculos[idx]).reajustarPreco(p);

System.out.println("\nPreço reajustado com sucesso.\n");

} else {

System.out.println("\n==== AVISO: Índice inválido ou veículo não é um avião ====\n");

}

break;

case 5:

System.out.print("\nÍndice do Navio para cálculo (0 a " + (contador - 1) + "): ");

idx = Integer.parseInt(sc.nextLine());

if (idx >= 0 && idx < contador && veiculos[idx] instanceof Navio) {

double result = ((Navio) veiculos[idx]).passageirosPorTripulante();

System.out.printf("Passageiros por tripulante: %.2f\n", result);

} else {

System.out.println("\n==== AVISO: Índice inválido ou veículo não é um navio ====\n");

}

break;

case 0:

System.out.println("\n...Encerrando Sistema...");

break;

default:

System.out.println("\n==== Opção inválida ====\n");

}

} catch (Exception e) {

System.out.println("\n==== Erro de entrada. Por Favor, tente novamente ====\n");

opcao = -1;

}

} while (opcao != 0);

sc.close();

}

}

**4.5 Código do run.bat**

@echo off

echo Limpando arquivos antigos...

del \*.class

echo Compilando...

javac \*.java

if errorlevel 1 (

echo Houve erros na compilação. Corrija os erros e tente novamente.

pause

exit /b

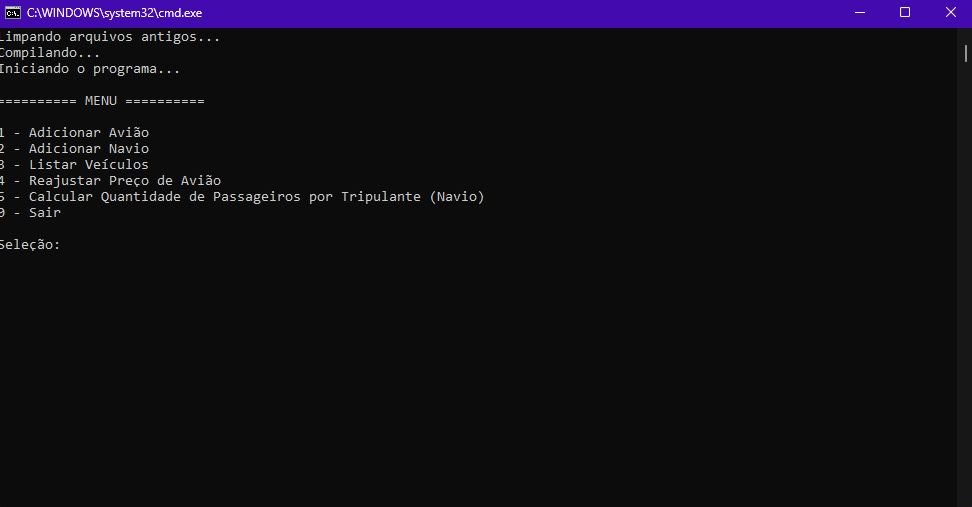
)

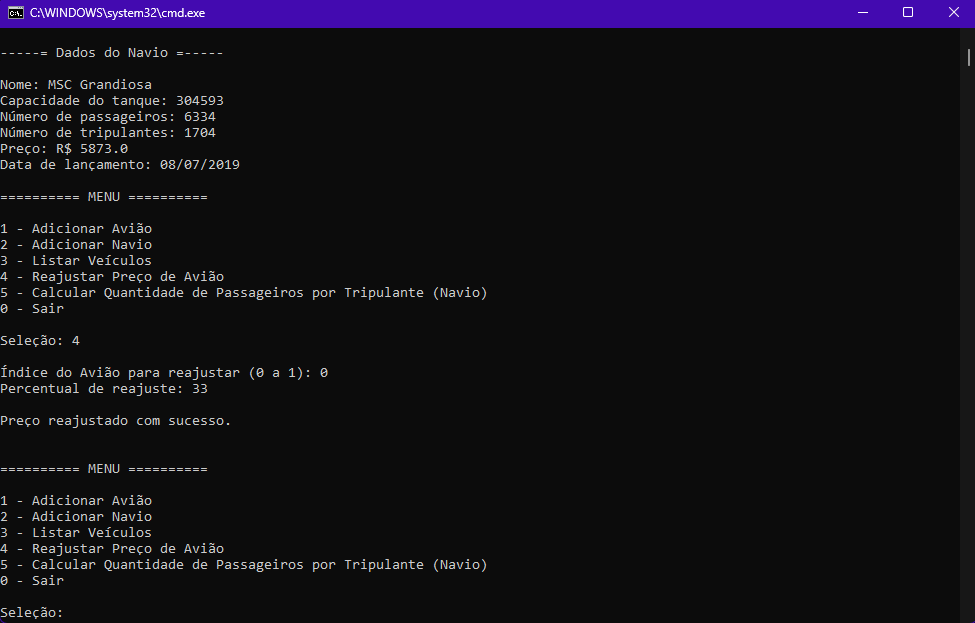
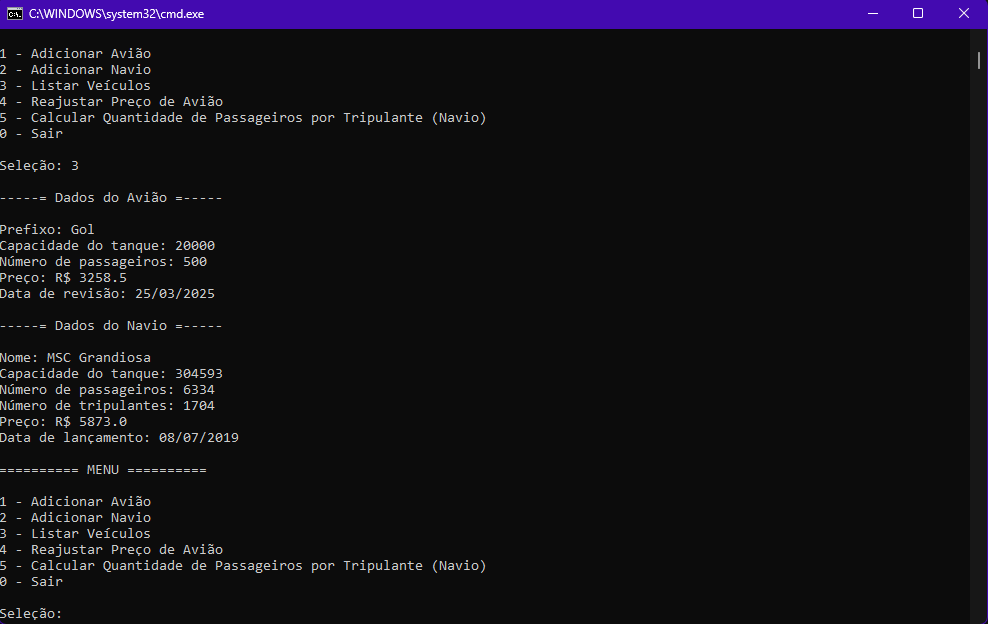
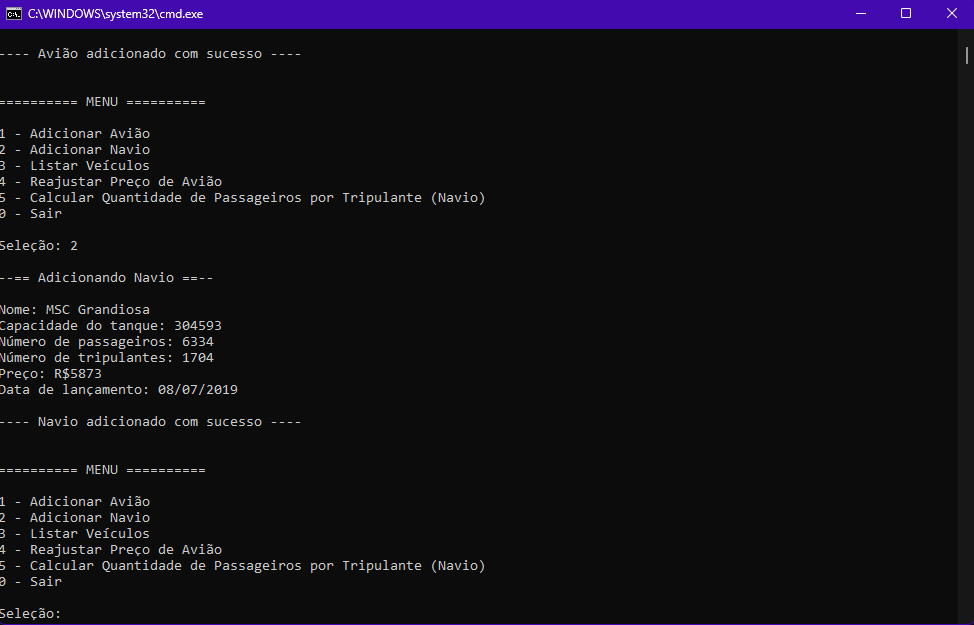
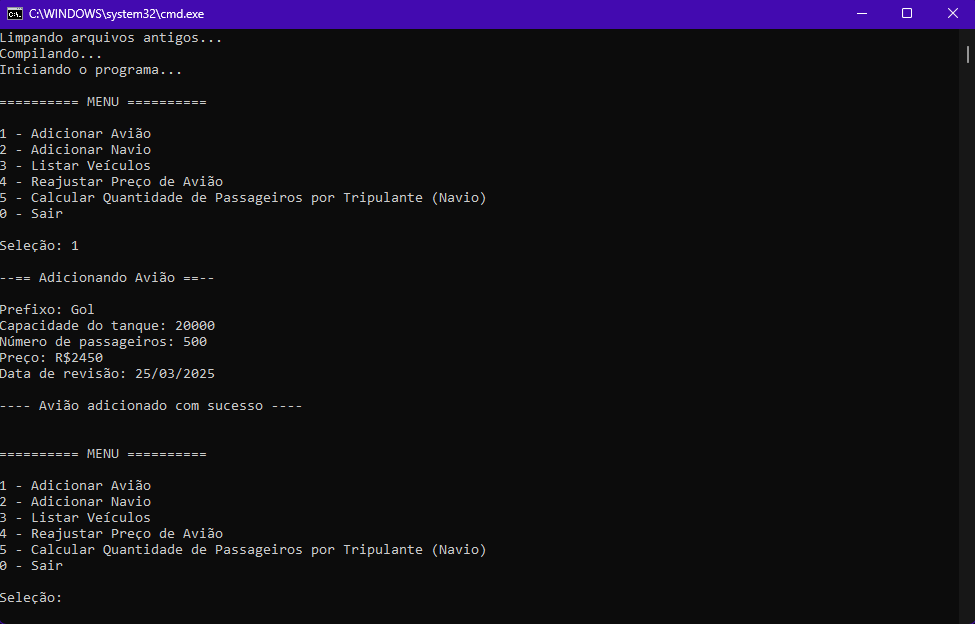
echo Iniciando o programa...

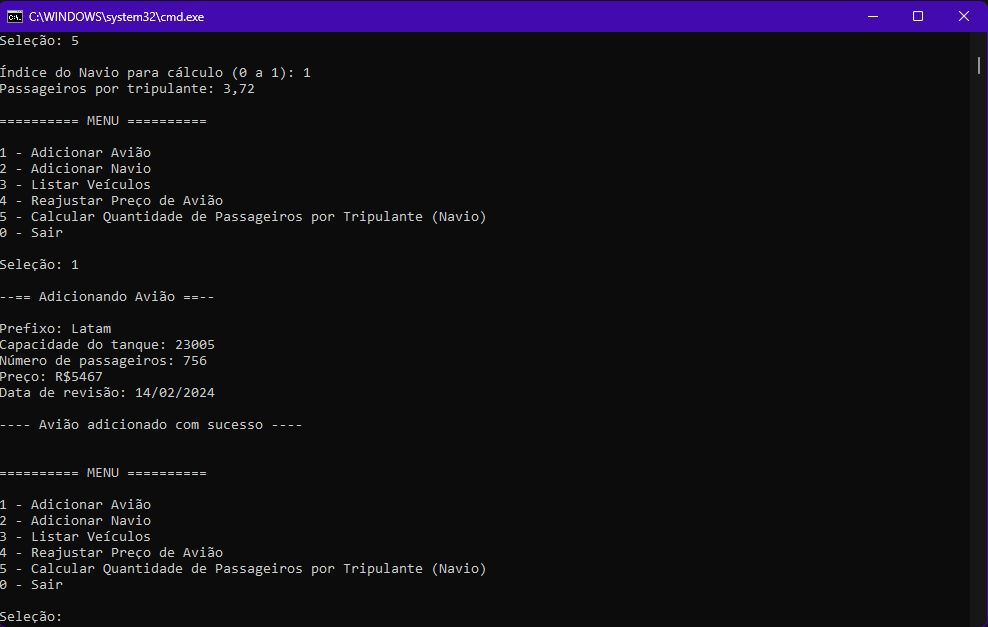
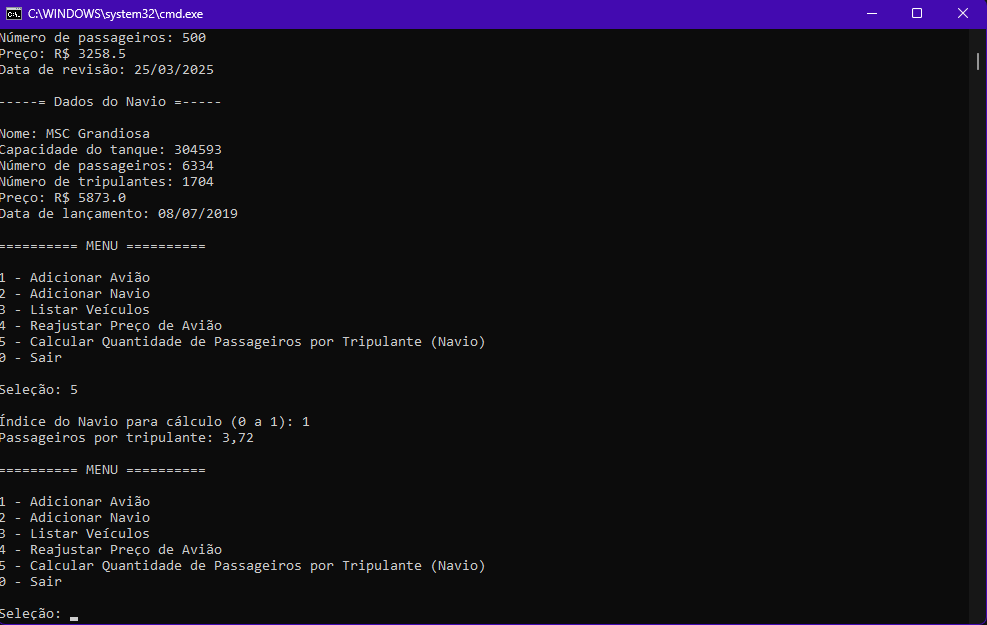
java Main

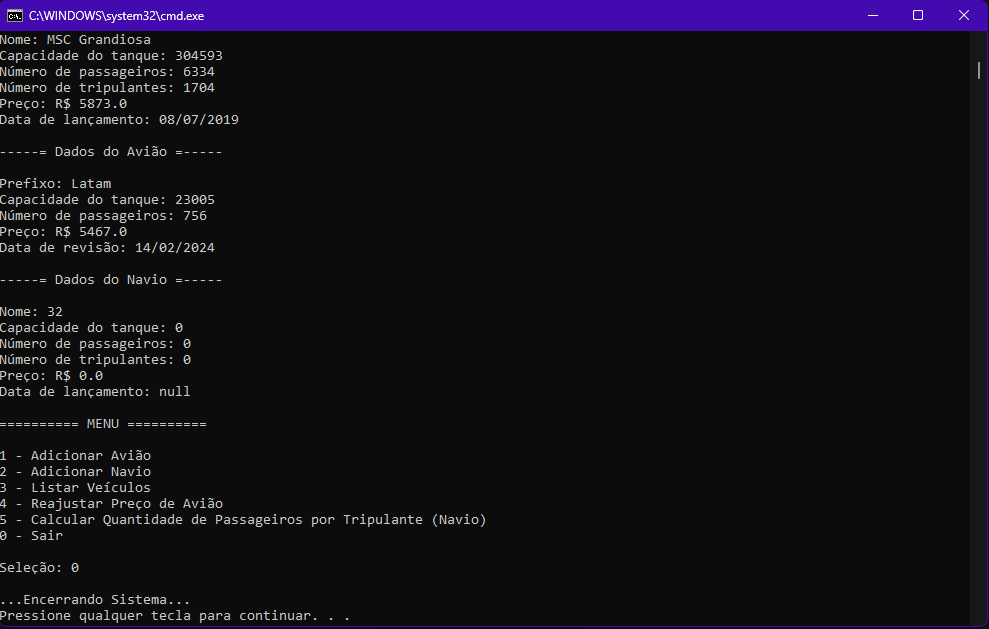
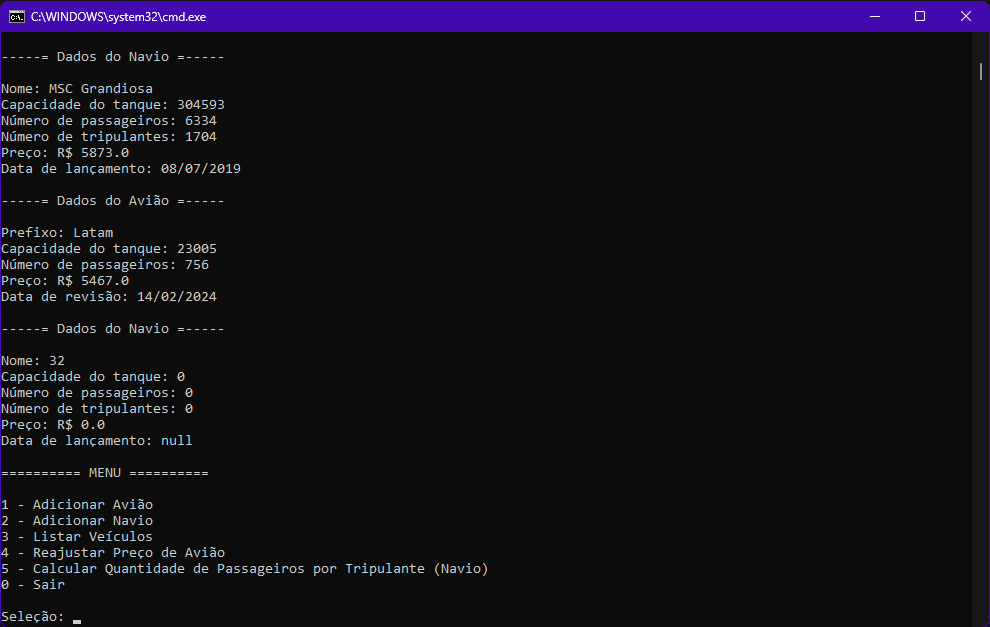
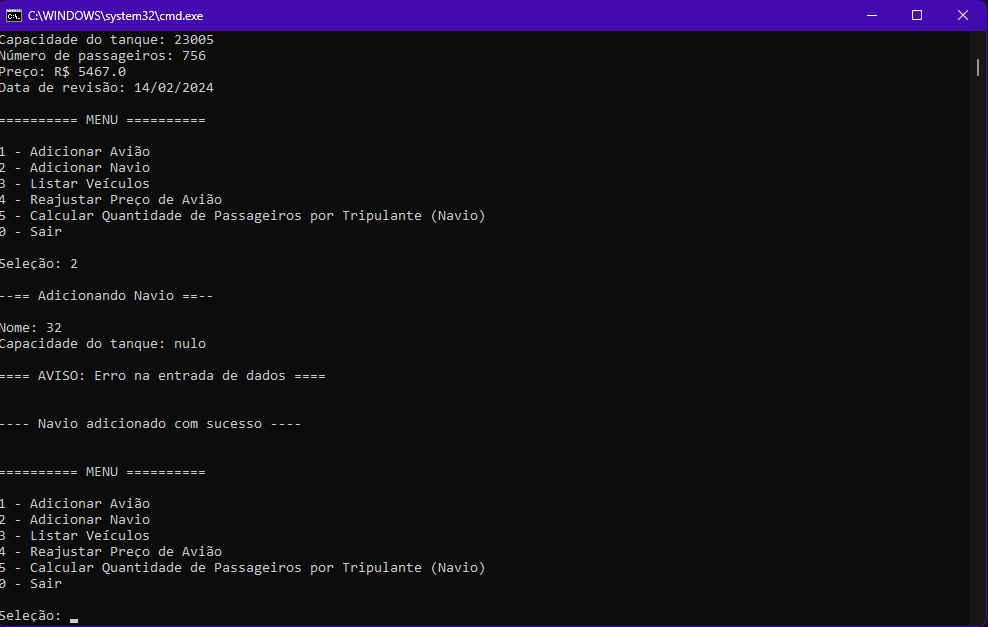
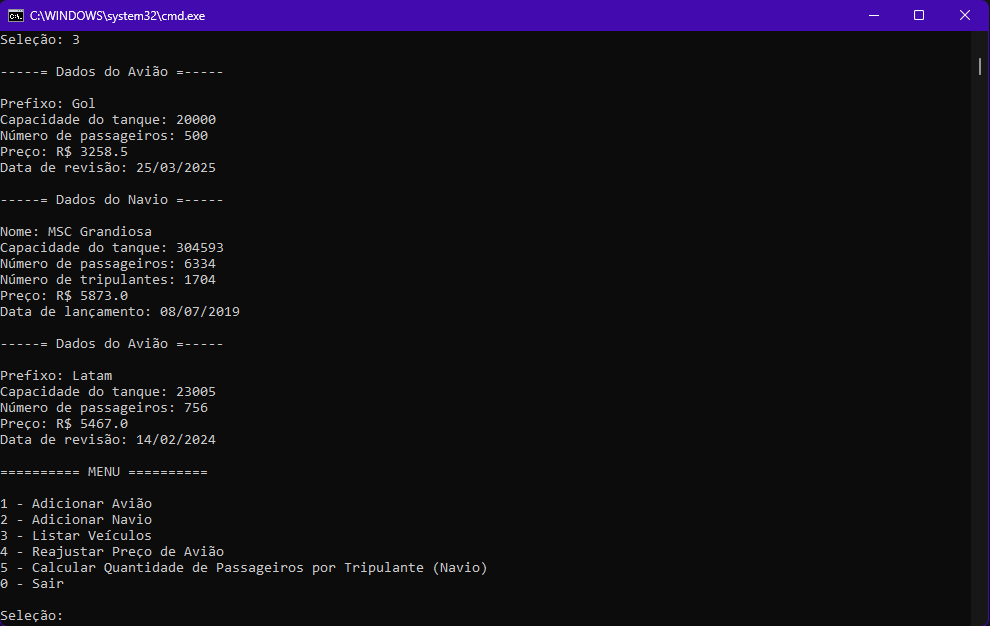
pause

**5 TESTES REALIZADOS**

Nesta seção, são apresentados os testes realizados com a aplicação Java, demonstrando o funcionamento das funcionalidades implementadas, como cadastro de veículos, listagem, reajuste de preços e cálculo de passageiros por tripulante.







**6 CONCLUSÃO**

O desenvolvimento deste projeto possibilitou a aplicação dos conceitos de Programação Orientada a Objetos na prática, por meio da criação de classes com herança, encapsulamento, uso de vetores de objetos e tratamento de exceções. A implementação de uma estrutura hierárquica permitiu organizar o código de forma clara e eficiente, tornando o sistema fácil de manter e expandir. Além disso, os aprendizados aplicados na realização do trabalho anterior permitiram a aplicação de um código mais limpo e otimizado, aproveitando do sistema de getters e setters para criar uma superclasse e suas subclasses.

**7 REFERÊNCIAS**

DEITEL, H. M.; DEITEL, P. J. Java: como programar. 10. ed. São Paulo: Pearson, 2016.

HORSTMANN, C. S.; CORNELL, G. Core Java Volume I - Fundamentos. São Paulo: Pearson, 2015.

Java OOP (Object Oriented Programming) Concepts. Disponível em: <<https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/>>. Acesso em: 28/04/2025.

Classes and Objects in Java. Disponível em: <<https://www.geeksforgeeks.org/classes-objects-java/?ref=lbp>>. Acesso em: 28/04/2025.

Object Class in Java. Disponível em: <<https://www.geeksforgeeks.org/object-class-in-java/?ref=lbp>>. Acesso em: 28/04/2025.

ORACLE. Documentação oficial do Java. Disponível em: <<https://docs.oracle.com/en/java/javase/24/>>. Acesso em: 20/03/2025.

ORACLE. Java Language Changes by Release. Disponível em: <<https://docs.oracle.com/en/java/javase/24/language/java-language-changes-release.html#GUID-6459681C-6881-45D8-B0DB-395D1BD6DB9B>>. Acesso em: 20/03/2025.

ORACLE. Java Platform, Standard Edition & Java Development Kit. Disponível em: <<https://docs.oracle.com/en/java/javase/24/docs/api/index.html>>. Acesso em: 20/03/2025.

ORACLE. JAR File Specification. Disponível em: <<https://docs.oracle.com/en/java/javase/24/docs/specs/jar/jar.html>>. Acesso em: 19/03/2025.

ORACLE. The Java Tutorials. Disponível em: <<https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html>>. Acesso em: 19/03/2025.

Material didático da disciplina.