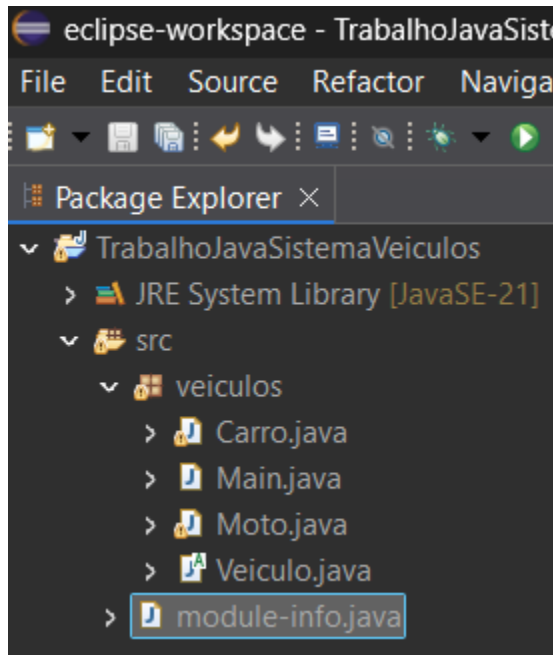


Nome: Vivian Garcia
RA: 219320
Prof: Alexandre

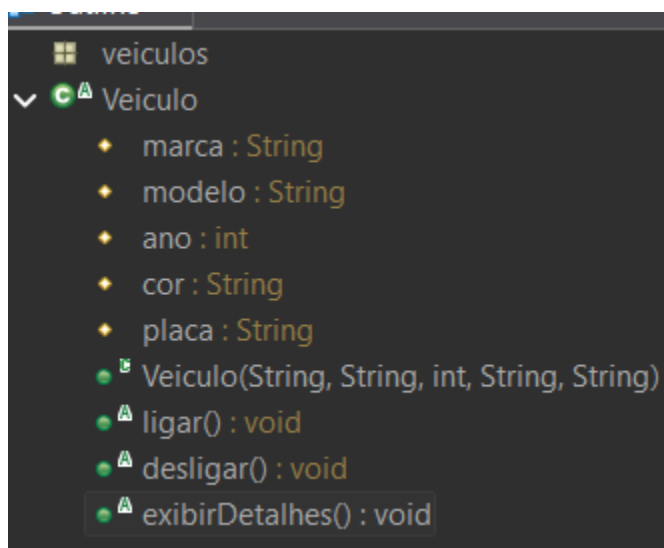
Projeto Herança: Java

Relatório – Descrição das Classes do Projeto Sistema de Veículos:

Arquivos:



Classes/Atributos:



Veiculo.java :

```
1 package veiculos;
2
3 public abstract class Veiculo {
4     protected String marca;
5     protected String modelo;
6     protected int ano;
7     protected String cor;
8     protected String placa;
9
10    public Veiculo(String marca, String modelo, int ano, String cor, String placa) {
11        this.marca = marca;
12        this.modelo = modelo;
13        this.ano = ano;
14        this.cor = cor;
15        this.placa = placa;
16    }
17
18    public abstract void ligar();
19    public abstract void desligar();
20    public abstract void exibirDetalhes();
21 }
22
```

Superclasse abstrata:

Essa é a classe principal abstrata, usada como base para outras classes. Nela eu coloquei os atributos que são comuns a qualquer veículo, como marca, modelo, ano, cor e placa. Também criei três métodos abstratos: `ligar()`, `desligar()` e `exibirDetalhes()`. Eles não têm código aqui, porque cada tipo de veículo vai implementar do seu jeito.

Usei essa classe pra organizar melhor o código e evitar repetir as mesmas coisas em Carro e Moto.

Carro.java :

```

1 package veiculos;
2
3 public class Carro extends Veiculo {
4     private int portas;
5     private boolean temAr;
6     private boolean ligado;
7
8     public Carro(String marca, String modelo, int ano, String cor, String placa, int portas, boolean temAr) {
9         super(marca, modelo, ano, cor, placa);
10        this.portas = portas;
11        this.temAr = temAr;
12        this.ligado = false;
13    }
14
15    public void ligar() {
16        ligado = true;
17        System.out.println(modelo + " está ligado.");
18    }
19
20    public void desligar() {
21        ligado = false;
22        System.out.println(modelo + " foi desligado.");
23    }
24
25    public void exibirDetalhes() {
26        System.out.println("Carro: " + marca + " " + modelo + " (" + ano + ") - " + cor +
27            " - " + portas + " portas - Ar-condicionado: " + (temAr ? "Sim" : "Não"));
28    }
29 }

```

Classe Carro.java (subclasse):

Essa classe representa um carro de verdade. Ela herda tudo da classe Veiculo, mas também tem atributos próprios, como:

- número de portas, se tem ar-condicionado, e um booleano para saber se o carro está ligado ou não.

Implementei os métodos obrigatórios aqui. O “ligar()” e “desligar()” mudam o estado do carro, e o “exibirDetalhes()” mostra as informações completas do carro. Essa classe serve para criar objetos específicos de carro com comportamento realista.

Moto.java :

```

1 package veiculos;
2
3 public class Moto extends Veiculo {
4     private int cilindradas;
5     private String tipoGuidon;
6     private boolean ligado;
7
8     public Moto(String marca, String modelo, int ano, String cor, String placa, int cilindradas,
9         String tipoGuidon) {
10         super(marca, modelo, ano, cor, placa);
11         this.cilindradas = cilindradas;
12         this.tipoGuidon = tipoGuidon;
13         this.ligado = false;
14     }
15
16     public void ligar() {
17         ligado = true;
18         System.out.println(modelo + " está ligada.");
19     }
20
21     public void desligar() {
22         ligado = false;
23         System.out.println(modelo + " foi desligada.");
24     }
25
26     public void exibirDetalhes() {
27         System.out.println("Moto: " + marca + " " + modelo + " (" + ano + ") - " + cor +
28             " - " + cilindradas + "cc - Guidon: " + tipoGuidon);
29     }
30 }

```

Classe Moto.java (subclasse):

Aqui fiz a mesma ideia do Carro, só que agora para uma moto. Também herda de "Veiculo", mas com atributos diferentes, como:

- quantidade de cilindradas, tipo do guidão, e o estado (ligada ou desligada).

Os métodos também foram implementados para funcionar como uma moto real. Assim, mesmo herdando da mesma superclasse, Carro e Moto funcionam de formas diferentes.

Main.java : (Classe Principal)

```

1 package veiculos;
2
3 public class Main {
4     public static void main(String[] args) {
5         Carro carro1 = new Carro("Toyota", "Corolla", 2020, "Preto", "VBC-1234", 4, true);
6         Moto moto1 = new Moto("Honda", "CB 500", 2022, "Vermelha", "YOD-5678", 500, "Esportivo");
7
8         System.out.println("----Carro:");
9
10        carro1.ligar();
11        carro1.exibirDetalhes();
12        carro1.desligar();
13
14        System.out.println("----Moto:");
15
16        moto1.ligar();
17        moto1.exibirDetalhes();
18        moto1.desligar();
19    }
20 }

```

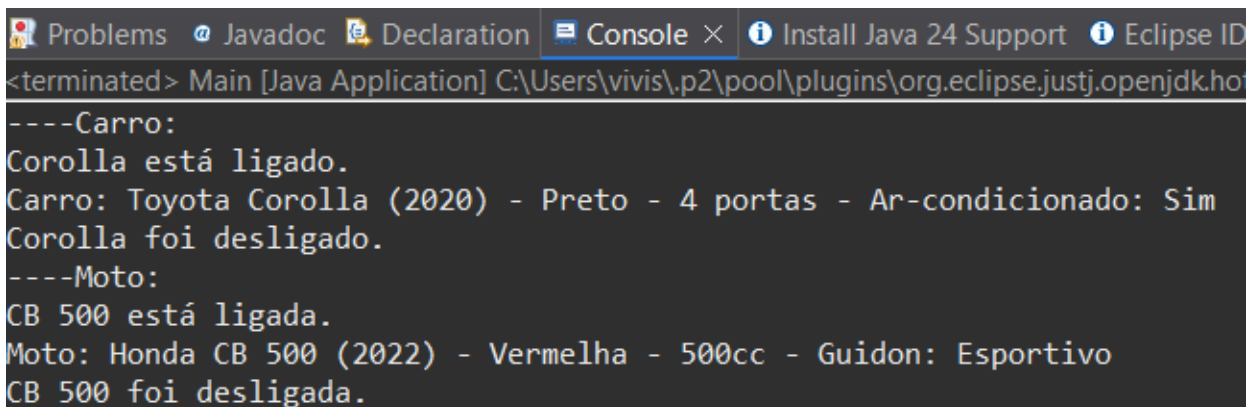
Classe Main.java (classe de testes)

Essa é a classe principal do sistema, onde testei tudo. Aqui eu:

- Criei um carro e uma moto, liguei os dois, mostrei os detalhes na tela, e depois desliguei os dois.

Ela serve para mostrar o funcionamento prático das classes que criei, e também como a herança e o "polimorfismo" funcionam no código.

Saída:



```

<terminated> Main [Java Application] C:\Users\vivis\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
----Carro:
Corolla está ligado.
Carro: Toyota Corolla (2020) - Preto - 4 portas - Ar-condicionado: Sim
Corolla foi desligado.
----Moto:
CB 500 está ligada.
Moto: Honda CB 500 (2022) - Vermelha - 500cc - Guidon: Esportivo
CB 500 foi desligada.

```

Conceitos que usei no projeto:

- **Abstração:** usei uma classe abstrata para servir de modelo geral.
- **Herança:** Carro e Moto herdaram de "Veiculo".
- **Polimorfismo:** mesmo os métodos tendo o mesmo nome, cada classe implementa de um jeito.
- **Encapsulamento:** deixei alguns atributos protegidos ou privados para manter o controle das informações.

Resumo do Projeto:

Este sistema foi desenvolvido com o objetivo de **simular o cadastro e controle de veículos**. Esse tipo de sistema poderia ser o **início de um sistema de gerenciamento de frotas**, onde empresas controlam veículos cadastrados, consultam dados, verificam status e executam ações.

No mundo real, ele poderia ser expandido com:

- Interface gráfica ou web
- Banco de dados para salvar os veículos
- Cadastro de condutores, manutenções, etc.

Requisitos do trabalho:

1. Uma superclasse abstrata

- ✓ Veiculo é uma classe abstrata

2. Duas subclasses criadas a partir da superclasse

- ✓ Carro herda de Veiculo
- ✓ Moto também herda de Veiculo

3. Cada classe deve ter pelo menos 5 atributos

- ♦ Veiculo (superclasse):
 - marca, modelo, ano, cor, placa → 5 atributos ✓
 - ♦ Carro (subclasse):
 - Herdados: marca, modelo, ano, cor, placa
 - Próprios: portas, temAr, ligado → total: 8 atributos ✓
 - ♦ Moto (subclasse):
 - Herdados: marca, modelo, ano, cor, placa
 - Próprios: cilindradas, tipoGuidon, ligado → total: 8 atributos ✓
-

4. Cada classe deve ter pelo menos 3 métodos (sem contar construtor, getters e setters)

- ♦ Veiculo:
 - Métodos abstratos:
 - ligar()
 - desligar()
 - exibirDetalhes()
- ♦ Carro e Moto:
 - Implementam os três métodos obrigatórios:
 - ligar()
 - desligar()

- `exibirDetalhes()`