**Programação Orientada a Objetos em Java - Conceitos e Fundamentos**

**1. Classes e Objetos**

A **classe** é a estrutura que define os atributos e métodos dos objetos. No código, a classe Carro define um modelo para criar instâncias de carros com atributos e comportamentos específicos.

public class Carro {

public int ano;

public float velocidade;

public String marca, modelo;

}

A partir dessa definição, objetos podem ser criados e manipulados, como feito na Main.java:

Carro obj1 = new Carro();

Carro obj2 = new Carro(2023, 30, "Fiat", "Argo");

**2. Encapsulamento**

O código permite acesso direto aos atributos (public int ano;), o que não é recomendado na prática, pois viola o princípio de **encapsulamento**. O ideal seria utilizar **modificadores de acesso** private e fornecer métodos get e set para manipulação.

Exemplo de encapsulamento correto:

public class Carro {

private int ano;

public int getAno() {

return ano;

}

public void setAno(int ano) {

this.ano = ano;

}

}

**3. Construtores**

Construtores são usados para inicializar objetos. No código, há dois construtores:

// Construtor padrão

public Carro(){

this.marca = "marca não definida";

this.modelo = "modelo não definido";

}

// Construtor com parâmetros

public Carro(int ano, float velocidade, String marca, String modelo){

this.ano = ano;

this.velocidade = velocidade;

this.marca = marca;

this.modelo = modelo;

}

Na Main.java, ambos os construtores são utilizados:

Carro obj1 = new Carro(); // Usa o construtor padrão

Carro obj2 = new Carro(2023, 30, "Fiat", "Argo"); // Usa o construtor com parâmetros

**4. Métodos de Instância**

A classe Carro implementa métodos que alteram o estado do objeto:

* exibirDetalhes()
* acelerar(float x)
* frear(float x)

public void exibirDetalhes(){

JOptionPane.showMessageDialog(null, "Marca: " + this.marca +

"\n Modelo: " + this.modelo +

"\n Ano: " + this.ano +

"\n Velocidade: " + this.velocidade);

}

public void acelerar(float x){

this.velocidade += x;

}

public void frear(float x){

if (this.velocidade >= x){

this.velocidade -= x;

} else {

JOptionPane.showMessageDialog(null, "Velocidade não pode ser negativa");

}

}

Esses métodos permitem manipular e visualizar os atributos do carro.

**5. Manipulação de Entrada e Saída com JOptionPane**

O programa utiliza JOptionPane para interagir com o usuário, coletando e exibindo informações.

Exemplo:

obj1.ano = Integer.parseInt(JOptionPane.showInputDialog("Ano"));

Isso permite ao usuário digitar valores em **janelas de diálogo**, tornando o programa mais interativo.

**6. Tratamento de Condições**

O método frear(float x) garante que a velocidade nunca fique negativa:

public void frear(float x){

if (this.velocidade >= x){

this.velocidade -= x;

} else {

JOptionPane.showMessageDialog(null, "Velocidade não pode ser negativa");

}

}

Esse controle evita erros lógicos no comportamento do objeto.

**Conclusão**

O código implementa **princípios básicos da POO**, como:

✅ Classes e Objetos  
✅ Construtores  
✅ Métodos de Instância  
✅ Manipulação de Entrada/Saída  
✅ Controle de Condições

Para melhorias, recomenda-se: ✔️ Aplicar **encapsulamento** corretamente (tornar atributos private)  
✔️ Criar **métodos get e set** para manipular atributos  
✔️ Adicionar **tratamento de exceções** para entradas inválidas