Progrramação Orientada a Objetos

Média dos exercícios ( runcodes) correspondem a 50% da nota final e 50% vai ser um projeto ou prova (não usa gpt nem copia)

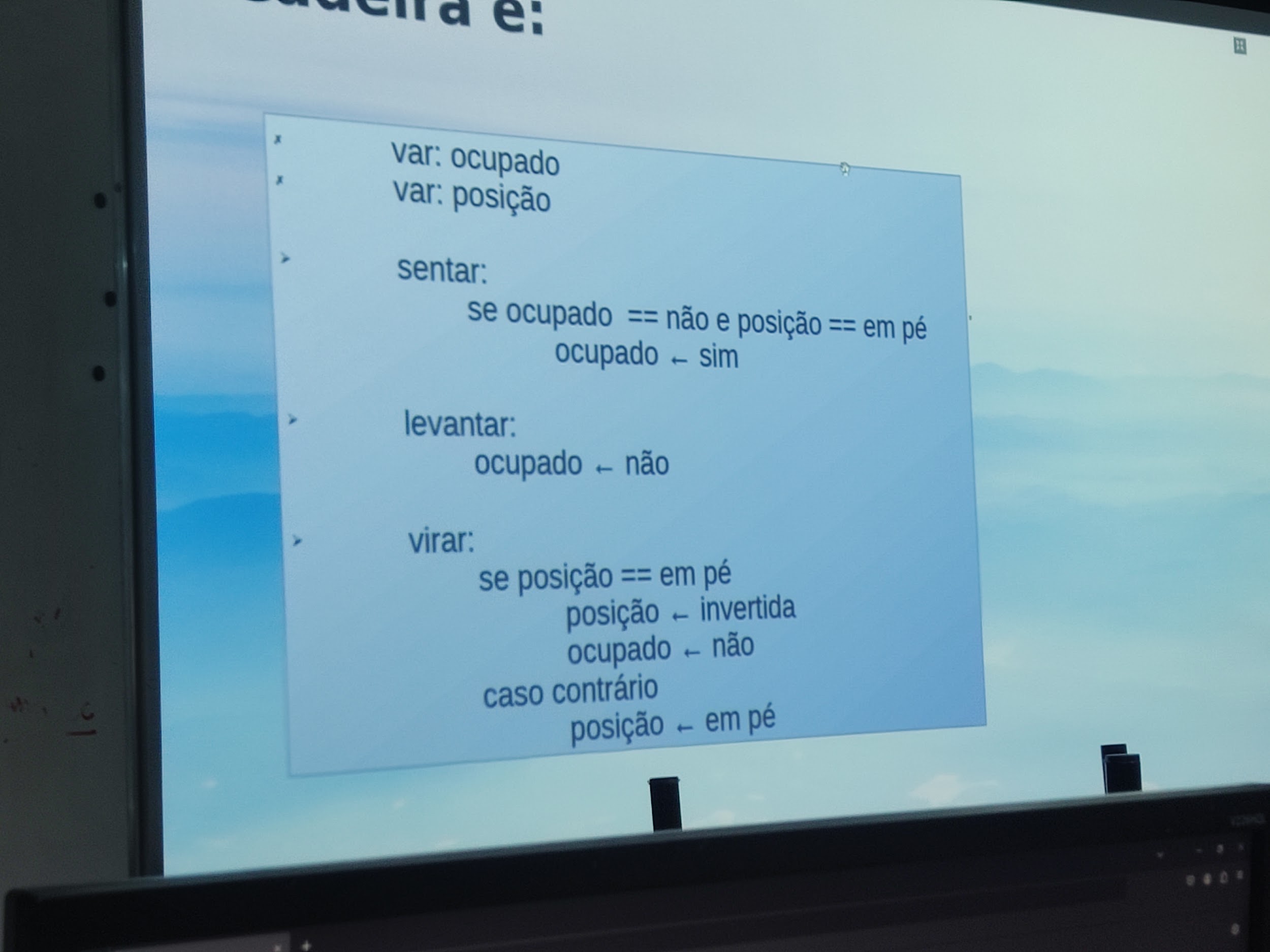
Paradigmas de programação -> modo de se pensar um programa: imperativo(POO, procedural - C), declarativo (Prolog, Funcional)

Linguagens de baixo nível: 1) Linguagem de máquina 2) Assembly

Linguagens de alto nível: 3) Linguagem imperativa 4) POO 5) Linguagem Declarativa

Funções são métodos e variáveis são atributos

Tornar as coisas do dia dia um código. Ex: cadeira



Classe são os atributos e métodos de um objeto. Instanciar um objeto é colocar uma classe a um objeto. Ex: cadeira1 <- nova cadeira

Objeto é uma instância de uma classe.

Objeto: comida feita. Receita: classe

Construtor é chamado quando se cria um objeto => estado inicial do objeto. Método criar (\_\_init\_\_). Getter pega o valor de algum atributo -> só serve pra mostrar o estado do objeto, mais nada

Atributo ou método = membro da classe ou do objeto

Composição: objetos que juntos compõem outros objetos

Var estático: tipo variável (atributo) (ou função - método) global para todos os objetos. Membro não estático pode modificar uma estática (o contrário não pode)

Método estático eu só pode mudar as variáveis estáticas

Uma interface em Java é uma estrutura que pode incluir métodos abstratos (sem corpo) e constantes (valores públicos, estáticos e finais). Interfaces não fornecem implementações de métodos, mas definem um "contrato" que as classes devem seguir. Ou seja, uma interface estabelece o que uma classe deve fazer, mas não como deve fazer. Isso permite que classes, mesmo não estando na mesma hierarquia de herança, possam ser tratadas de maneira uniforme através dos métodos definidos pela interface.

### **Exemplos no Slide**

1. Classe AnimalComparavel Implementando Comparable: Este exemplo mostra uma classe que implementa a interface Comparable. A palavra-chave implements é usada para indicar que a classe AnimalComparavel deve implementar os métodos definidos na interface Comparable. Isso é útil para permitir comparações entre instâncias de AnimalComparavel, como em algoritmos de ordenação.
2. Interfaces e Genéricos: O slide também discute o uso de genéricos em Java, que são uma forma de permitir que classes, interfaces e métodos trabalhem com tipos de dados que são especificados apenas durante a instanciação (como em Pilha<Object>). Isso aumenta a reutilização do código, permitindo, por exemplo, a criação de uma estrutura de dados como uma pilha que pode trabalhar com qualquer tipo de objeto.

### **Benefícios das Interfaces**

* Polimorfismo: Interfaces permitem polimorfismo, que é a capacidade de tratar objetos de diferentes classes que implementam a mesma interface de forma uniforme.
* Desacoplamento de código: Como as interfaces definem apenas "o que" deve ser feito, mas não "como", isso permite que detalhes de implementação sejam mudados sem afetar o código que usa a interface.

### **Exemplo de Erro**

O slide também mostra um erro (ClassCastException) que ocorre quando se tenta tratar um objeto que não implementa uma interface esperada (neste caso, Comparable). Isso destaca a importância de garantir que objetos sejam do tipo correto antes de usá-los em contextos que exigem interfaces específicas.

Esses conceitos são fundamentais para desenvolver aplicações robustas e flexíveis em Java, facilitando o gerenciamento e a extensão do código.