***Anotações***

Orientação a Objetos → Resume-se a melhorar a estrutura do código e não colocar tudo no mesmo lugar. Serve para organizar, reutilizar código e principalmente para facilitar a manutenção dos sistemas (proteger e colocar regras).

Abstrair o mundo real para o sistema.

Modularizar o código / quebrar em partes.

Direcionar as coisas para o sistema → abstrair

Variáveis (atributos) representam características e comportamentos (métodos) representam ações.

A base da POO são as CLASSES → Representação de um objeto dentro do código que tem REGRAS que estão dentro da classe.   
Classe começa sempre com nome maiúsculo e no singular. EX: Carro

Variavel Membro → Variavel que ajuda a auxiliar no meio do codigo - (m\_variavel).

Instanciar é criar um objeto novo → quando usa o NEW ...(). Cria uma memória quando instancia o objeto (faz o alocamento de espaço com o necessário para o objeto, antigamente fazia isso a mão, agora não mais, pois as linguagens fazem gerenciamento de memória).

Construtor →Obriga a inicializar um objeto com tais argumentos – o java cria um automático com os atributos padrões. Construtor com a mesma assinatura dá erro, pq ele identifica que vc está tentando fazer as mesmas coisas já existentes (NÃO PODE!)

O que é um construtor implícito e um explicito?

Construtores não tem void, nem retorno?

This faz referencia para a própria classe  
  
public Pessoa() {  
This(“Usuario”);  
}

**ENCAPSULAMENTO →** Omite valores para fora da classe (limita acesso de atributos) – delegar a responsabilidade de como os atributos são definidos para a própria classe.

Modificadores de acesso são as camadas de encapsulamento → public, private ou protected

Public pode ser acessado por qualquer classe  
Private so pode ser acessado dentro da classe em que for declarado  
protected são atributos e metodos que podem ser acessados dentro da própria classe e das subclasses

p.nome → acessa diretamente o atributo e altera

acesso public é uma ma pratica de programação

Getters(leitura) e Setters(alterar nome de fora da classe e altera dentro da classe) → são métodos para enviar os dados para fora da classe

Getter sempre retorna o tipo do dado declarado.  
  
Quando é um tipo booleano o get e o set devem ser declarados com isVariavel e não get e set

O set deve ser uma função VOID (sem retorno) e com o parâmetro do valor a ser modificado

É necessário proteger com get e set para fazer verificações dentro dos métodos

Não é obrigado a ter get e set sempre, é so se quiser que o atributo ou método seja acessado

Utilizar private em método serve para usar um método dentro de outro método (cria um método auxiliar)