***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)

Associação → vinculo de diversos objetos\relação entre eles | ex: uma disciplina faz interação com um professor | Professor faz referencia a Disciplina  
Uma associação é uma relação de posse, um objeto possui um objeto, ou seja, uma disciplina tem sempre um único professor

O objeto Professor (ou o objeto que foi criado dentro de outro) como a Disciplina, pode ter os atributos e métodos públicos acessados dentro da Classe no qual o objeto foi criado.  
resumo: quando vc cria um objeto dentro do outro, esse objeto criado vira um “atributo”

Coleção de Objetos → estrutura que permite que armazene vários objetos (coleção é um objeto dentro de java)

Array → não é uma coleção mesmo dando para armazenar objetos  
Coleções podem adicionar, remover, pesquisar etc. Um array estático não te da funcionalidades, só da os índices

Array são estruturas estáticas, onde o tamanho não pode ser alterado (tem que indicar o tamanho na hora que criar), mas pode alterar/manipular os dados que estão dentro. Não tem como ver o índice e nem quantas posições estão ocupadas.  
  
Listas → estrutura dinâmica que o tamanho pode ser alterado, tem como acessar os índices diretamente, tem como saber quantas posições foram preenchidas e tem funcionalidades que ajudam a manipular.

A única forma de aumentar o tamanho de um array estático, é copiar os dados de um array anterior e criar um novo com + índices.

Java Collections

LinkedList eh mais rápido do que ArrayList

ArrayList → ArrayList<TIPO DE OBJETO> não pode ser tipo primitivo como int por exemplo, o nome deve ser no plural

Hierarquia de classes  
modificador de acesso usa o protected pq fica visível para toda a família (protected na superclasse para serem acessados pelas subclasses)  
  
Quando um construtor eh implementado na classe pai, todos os filhos precisam obrigatoriamente chamar o construtor pai no seu próprio construtor. Para isso, utilizamos a palavra super e passamos os parâmetros que o construtor do pai exige.

Classes abstratas, métodos abstratos e interfaces

Classes abstratas → tipo de classe que não pode ser instanciada, uma referencia nunca vai armazenar um objeto da mesma classe.   
Serve para reutilizar atributos, métodos e usar o polimorfismo. A escabilidade continua sendo uma vantagem.

Classe abstrata pode estar em qualquer nível de hierarquia

Métodos Abstratos → métodos que não possuem implementação e só podem ser implementadas em classes abstratas

Um método abstrato serve para obrigar as classes filhas a herdarem os métodos abstratos

Método abstrato é ( public abstract void latir(); ) e na sobrescrita não tem abstract, só o public void latir() {}

Interface é equivalente a uma classe abstrata, mas não tem atributo, apenas métodos abstratos public. Não possui variáveis mas pode ter constantes.

Interfaces são estruturas sem atributos e somente com métodos.

Interface é implementada em outras classes

Uma classe abstrata TEM QUE TER FILHOS, se não esta errado