**PORQUE DEVEMOS PROGRAMAR EM POO?**

* Mais realista
* Focado na modelação de entidades e nas interações com elas
* Foca “no que fazer” e não “como fazer”
* Maior coesão
* Acoplamentos flexíveis
* Gap semântico = aproxima o real ao computacional.
* Coletor de lixo = limpa automaticamente coisas que não estão sendo usadas, como objetos, deixando a memória mais limpa.
* Orientação a objetos = analise, projeto e programação

**PILARES DA POO**

* Abstração: Se isolar de características acidentais e focar somente nas principais.
* Reuso: Criar novos códigos a partir de outros já existentes.
* Encapsulamento: esconder complexidades e proteger dados.

**CLASSES**

* Serve de molde para o nosso conceito, para traduzir o mundo real.
* Substantivos.

**ATRIBUTO**

* Carrega as características para definir os conceitos que serão utilizados na classe.
* Diferença de atributo e variavel: Atributo = peculiar a uma classe ou uma coisa. Variavel = sujeito a variações e mudanças.
* Substantivos e adjetivos.

**MÉTODO**

* Porção de código disponibilizado pela classe encarregada de realizar alguma função, serviço, ação ou comportamento no programa.
* Para criar: visibilidade, retorno, nome e parâmetros.
* Verbos.
* Métodos especiais: Construtor e destrutor.
* Construtor = Mesmo nome da classe. Prover valores iniciais como parametro.
* Destrutor = auxilia a “limpar o lixo” do programa, excluindo o objeto.

**OBJETO**

* Representação de algo do mundo real, podendo ser concreto (bola, árvore) ou abstrato (viagem, compra). Para instanciarmos um objeto, precisamos de uma classe com atributos bem definidos.
* Nossos atributos são definidos na classe, porém ela é estática e não faz nada. É na criação do objeto que manipulamos nossa classe.

**MENSAGEM**

* Processo de ativação de um método do objeto. Acontece quando há uma requisição é efetuada, realizando algum comportamento descrito na classe.

**HERANÇA**

* Quando ocorre uma relação de uma classe chamada subclasse com a sua extensão, chamada de superclasse, conseguindo reaproveitar os atributos dela.
* As subclasses podem definir seus próprios atributos e métodos.
* A finalidade da herança é criar subtipos específicos e não reuso.
* Nem tudo que esta na superclasse pode ser visto na subclasse.
* Simples = apenas uma classe mãe, Multiplica = mais de uma classe mãe.
* Java não possui herança múltipla.
* Upcast = subir na minha hierarquia. Pego a filha e a transformo na mãe.
* Downcast = descer na minha hierarquia. Pego a mãe e a transformo na filha.

**POLIMORFISMO**

* Mesmo método se comportando diferente.
* Sobrescrita = mesmo método PODENDO se comportar diferente

**ASSOCIAÇÃO**

* Possibilita às classes objetos se ajudarem.
* Estrutural-Composição = EX: Pessoa e endereço. Toda pessoa tem um endereço, então, se a pessoa deixa de existir o endereço também deixa. “Com parte todo”.
* Estrutural-Agregação = EX: Disciplina e Aluno. O aluno possui uma disciplina, mas se essa disciplina deixar de existir o aluno nao deve deixar de existir, pois pode estar matriculado em outras disciplinas. “sem parte todo”
* Comportamental-Dependencia: um metodo depende do outro objeto.

**INTERFACE**

* Você mostra seus métodos e a interface usa da maneira que for

necessario.

**PACOTES**

* Organização logica ou distinta para separar classes com responsabilidades diferentes.

**VISIBILIDADE**

* Determinar até que ponto uma classe, metodo, objeto pode ser usado.
* Private = visivel somente na classe.
* Protected = visivel dentro do mesmo pacote, classe e subclasse.
* Public = qualquer lugar