**Conteúdo**

1 - Conceitos básicos de POO

2 - Modificadores de acesso

3 - Relacionamento entre classes

HAS-A

IS-A (herança)

4 - Sobrecarga e sobrescrita

5 - Abstração

6 - Polimorfismo de subtipos

7 - WEB

-Templates

-Request/Response

-Rotas

-Conexão com Banco de dados

-Deploy

**Conceitos básicos de POO**

**Classe:** É uma estrutura que possui as especificações de comportamento de características dos objetos.

**Objeto:** É uma instância de uma classe em memória.

**Atributo:** São as características definidas na classe (campos)

**Método:** São as ações definidas em uma classe

**Estado:** São os valores dos atributos em um instante de tempo

Inicialização de variaveis

**int** é um tipo primitivo, todo tipo primitivo possui um valor inicial default (por padrão)

Em uma classe, o valor padrão é nulo

**Exercicio 1:** Implemente a classe Revolver (p/um jogo).

Todo revolver possui uma quantidade atual de munição e uma quantidade máxima. Suas funcionalidades são:

-Atirar: Só atira se houver munição

-Recarregar: Só recarrega até o máximo possível

-Mostrar: Mostra a quantidade atual de munição

**Exercicio 2:** Implemente a classe Quadrado, esta deve prover funcionalidades de calculo de área e perímetro.

**Construtores:** São estruturas para usar no instancia do objeto. Um construtor é executado apenas uma vez e define o estado inicial. O construtor possui o mesmo nome da classe

**Operadores aritméticos:**

Os operadores aritméticos são usados para realizar operações matemáticas básicas. São eles:

* "+" (adição)
* "-" (subtração)
* "\*" (multiplicação)
* "/" (divisão)
* "%" (resto da divisão)

**Operadores relacionais:**

* "==" (igual a)
* "!=" (diferente de)
* ">" (maior que)
* ">=" (maior ou igual a)
* "<" (menor que)
* "<=" (menor ou igual a)

**Operadores lógicos:**

Esses operadores são usados quando queremos verificar duas ou mais condições e/ou expressões na aplicação. Eles fazem a comparação de valores booleanos e retornam também um resultado booleano.

São três operadores: **AND (&&), OR (||) e NOT (!)**.

O operador AND (&&), que traduzindo para o português seria o E, é usado para verificar se duas condições são verdadeiras. Se ambas as condições forem verdadeiras, o resultado será verdadeiro. Caso contrário, o resultado será falso. Aqui está um exemplo:

boolean a =true;

boolean b =false;

if (a && b) {

// Este código não será executado, já que a é verdadeiro e b é falso.

}

**O operador OR (||)**, que traduzindo para o português seria o OU, é usado para verificar se pelo menos uma das condições é verdadeira. Se pelo menos uma das condições for verdadeira, o resultado será verdadeiro. Caso contrário, o resultado será falso.

**O operador NOT (!)** é usado para negar uma condição. Se a condição for verdadeira, o resultado será falso. Se a condição for falsa, o resultado será verdadeiro. Aqui está um exemplo:

**Operadores de incremento:**

Além dos operadores citados anteriormente, o operador de incremento é usado para aumentar o valor de uma variável em 1. Existem dois tipos de operadores de incremento: o operador de pré-incremento (++variavel) e o operador de pós-incremento (variavel++).

O operador de pré-incremento (++variavel) aumenta o valor da variável em 1 antes de usar a variável em uma expressão. Aqui está um exemplo:

int num = 5;

int resultado = ++num; //num é incrementado para 6 e depois atribuído a resultado

System.out.println(num); // imprime 6

System.out.println(resultado); // imprime 6

Já o operador de pós-incremento (variavel++) aumenta o valor da variável em 1 depois de usar a variável em uma expressão. Aqui está um exemplo:

int num = 5;

int resultado = num++; //num é atribuído primeiramente à variável resultado e depois incrementado para 6

System.out.println(num); // imprime 6

System.out.println(resultado); // imprime 5

Aqui estão algumas das principais convenções de código do Java:

* Nomes de classes devem começar com letra maiúscula e usar a convenção PascalCase (também conhecida como Upper CamelCase).
  + Exemplo: MinhaClasse.
* Nomes de métodos devem começar com letra minúscula e usar a convenção camelCase.
  + Exemplo: meuMetodo().
* Nomes de constantes devem ser totalmente em letras maiúsculas, separadas por underline.
  + Exemplo: MINHA\_CONSTANTE.
* Nomes de variáveis devem começar com letra minúscula e usar a convenção camelCase.
  + Exemplo: minhaVariavel.
* Todas as linhas de código devem ter no máximo 80 caracteres de largura para facilitar a leitura.
* Recomenda-se usar espaços em branco para separar operadores, palavras-chave e elementos de controle de fluxo.
  + Exemplo: if (condicao) {.
* Use comentários para documentar seu código, explicando o que ele faz e por que ele faz isso. Comentários devem ser claros e concisos.

<https://www.hackerrank.com/>

Aula 26/02

file > new > Java project

new>package

projeto e package > minusculo

new > class

exemplo: implemente a classe contador que vai expor 4 funcionalidades

-incrementar

-decrementar

-mostrar

-zerar

Exemplo: Implemente a classe quadrado que possua as funcionalidades : Calcular perimetro e calcular área

perimetro =4\*lado

area= lado^2

o lado deve ser sempre positivo

Aula 04/03

Construtores

Definição: São rotinas que definem o estado inicial dos objetos. Eles são chamados no “momento do new”. O construtor é chamado apenas uma vez e possui o mesmo nome da classe

Gerar automaticamente:

Source →Generate

Constructor

using

fields

Exercicio: Implemente uma classe p/ controlar dois operandos de uma calculadora. A calculadora deverá possuir funcionalidade p/ as 4 operações. Use construtores.

ENUMERAÇÕES

Definição: Uma enumeração é um tipo que possui uma quantidade finita e fixa de valores

Exercicio: Implemente uma classe chamada Moeda. Possui valor e corrência. A classe deveria

1)Mostrar as informações

1. Converter:

BRL para USD

USD para BRL

BRL para EUR

EUR para BRL

BRL para BTC

BTC para BRL

obs: usar cotações de hoje

EX2: Implemente a classe Produto que possua os atributos nome, tipo e peço. A classe possuirá funcionalidades para:

1. Mostrar informações
2. Calcular desconto e mostra-lo

15% para livro

20% para escritório

10% para tecnologia

**ENUMERAÇÃO**

São tipos de dados que restringem seus valores a um número finito de opções.

OBS: No Switch/Case não fazemos referência ao nome da ENUM

**ACESSO A MEBROS (ATRIBUTO/METODO)**

**-Atributos:**

a) Leitura: É quando usamos um atributo em uma expressão.

int x=**obj.atributo**+5

System.out.println(**obj.atributo**);

b)Escrita: É quando usamos o membro do atributo do lado esquerdo da atribuição.

**obj.atributo** = 7;

Leitura + Escrita:

**obj.atributo**=**obj.atributo**+1

**-Método:**

c) Chamada: É quando invocamos um metodo.

obj.metodo( );

Acoplamento baixo, coesão alta!

Aula 11/03

**Modificadores de acesso**

**Encapsulamento: É o ato de** esconder um objeto

**Modificadores**

Public →acesso liberado para todas as classes/objetos

Private →acesso liberado para a própria classe

Protected →pacote + herança

Dedault (package, no modificar) →acesso liberado dentro do pacote

Acoplamento de classes → é o quanto uma classe A mexe no estado de uma classe B

Atributo público aumenta o acoplamento

**Modificadores de acesso**

**Definição de acesso:**

1. atributo

a) leitura: É quando obtemos o valor de um atributo System.out.println(obj.atributo);

b)Escrita: É quando fazemos uma atribuição obj.atributo=5;

Obj.atributo=obj.atributo+1

**Métodos getter** → libera acesso de leitura

**Métodos setter** → libera acesso de escrita(evitar!)

**Record (Java 14→)**

**Definição** → É uma “classe” com os atributos privados e imutáveis

**Quando usar?**

1- Quando perceberes que o estado não está sendo mudado na classe

2- Tipos mais detalhado

aula 25/03

Modificador Static

definição: O modificador Static desacopla um membro de sua instancia. um membro static pode ser acessivel pela classe

Relacionamento entre classes (HAS-MANY)

Definição: É quando uma classe A possui N referências a uma classe B

**MÉTODOS GETTERS E SETTER**

Getter: Libera acesso de leitura

Setter: Libera acesso de escrita(na maioria dos casos evitar)

Exercicio: Crie a classe Cartao, ao qual a mesma possui valor de fatura e um tipo de cliente (platinum, gold ou silver). Crie um construtor, seus gets e um metodo para calcular o desconto da fatura.

Gold → 10%

Silver →5%

Platinum →15%

**Relacionamento entre classes**

1. HAS-A “1-1”: É quando uma classe A possui referência a uma classe B.

Dizemos então que A “HAS-A” B, ou A “tem um B”.

public class Aluno{

private String nome, ra;

private Endereco endereco;

public void exemplo() {

endereco.mostrar();

}

}

RECORDS: São classes aos quais atributos não variam, usamos os records para incrementar a segurança de tipos (Type-drivew development).

2 has-many (tem vários): É uma relação ao qual uma classe A possui várias referencias

A

-B[} bs

B

* metodob()

30/09

**Herança (is-A)**

É uma relação hierarquica entre classes

A classe superior (superclasse) provê acesso privilegiado para as suas subclasses.

obs: Não existe herança multipla em Java

07/10

**Sobrecarga(overload)**

Def: É uma maneira de resolver ambiguidades de nomes de método de forma estática( Em tempo de compilação). A sobrecarga ocorre quando um mesmo nome de metodo (ambiguidade) é usado com parâmetros diferentes, ou seja, assinaturas distintas.

Exemplos:

1. public void algo(){}

Assinatura: Nome = algo

lista de parâmetros =[]

b. public void algo(int x){}

Assinatura: Nome = algo

lista de parâmetros = [int]

c. public void algo(String z){}

Assinatura: Nome = algo

Lista de parâmetros =[String]

“SQL”

insert into Cliente(nome, cpf)

values(”teste”, “123456789-00”);

1. update Cliente

set nome = “teste2”

Sobrescrita(override)

É uma maneira de resolução de ambiguidade através de despacho dinamico (dinamic dispatch). Para obter uma sobrescrita válida devemos ter a mesma assinatura em uma relação de herança.

OBS: Não pode ter redução de visibilidade na sobrescrita.

OBS: O retorno não pode ser mudado na sobrescrita.

21/10

Classes Abstratas e interfaces

Classe Abstrata: É uma classe não pode ser instanciada e possui metodos abstratos.

Uma classe abstrata espera sempre ter subclasses(Não-abstratas).

Método abstrato: Método sem corpo de cóodigo que usado como um “template” / algoritmos.

Tal metodo deve ser sobrescrito em uma classe não abstrata (obrigatório)

Interface

Uma interface possui o mesmo intuito que uma classe abstrata. Com algumas diferenças.

-Classe abstrata exemplos extends uma classe, ao passo que interface implementa n outras classes.

-Todo método em uma interface é automaticamente public e abstract a não ser que você diga ao contrario.

-Uma interface possui métodos concretos chamado default.

-Uma interface pode possuir métodos estáticos.

-Uma interface possui apenas constantes(final). Não possui atributo