**PRIMEIRAS ANOTAÇÕES**

**Documentação:**

https://docs.oracle.com/javase/8/docs/api/

**Comandos:**

Println= Imprime na tela

Var= Armazene na memória

= é uma atribuição

* Soma de string

**Variáveis:**

Int: Inteiros

Strings: textos

Double: decimais

Boolean: Verdadeiro/falso

**Tipos de Dados:**

**Tipo** /  **Referência** / **Conteúdo**

String / “ login “ / “ola”

Int / “number 1 “ / “10”

Double / “ Percent “ / “ 24.30”

Boolean / ” isDriver” / “true”

Boolean / ” Prove” / “2+10==12” automático true

**Funções:**

Armazenamento de função/ referência / tipo de variável e conteúdo ( parâmetros)

Static void sum ( int x, int y)

sum(10 ,5 ) retorno, e definição

**Condição**

If( ) {

} else or else if ( as vezes não é necessário parênteses no else ) {

}

**MENTORAMA ANOTAÇÕES**

**Array:** uma variável com uma sequência de valores

**String:** declara uma variável que vai armazenar caractere, também é possível pedir para o usuário colocar um valor de entrada

**String next()** - retorna uma cadeia de caracteres simples, ou seja, que não usa o caractere espaço em branco;

* **double nextDouble()** - retorna um número em notação de ponto flutuante normalizada em precisão dupla de 64 bits (usado para receber valores reais ou monetários);
* **boolean hasNextDouble()** - retorna true se o próximo dado de entrada pode ser interpretado como um valor double;
* **int nextInt()** - retorna um número inteiro de 32 bits;
* **boolean hasNextInt()** - retorna true se o próximo dado de entrada pode ser interpretado como um valor int;
* **String nextLine()** - retorna uma cadeia de caracteres, por exemplo: “DevMedia - Canal Java”;
* **long nextLong()** - retorna um número inteiro de 64 bits.

**System.out.print:** a informação é mostrada na mesma linha

**System.out.Println:** a informação é mostrada na linha abaixo, funciona como um quebra-linha

**Class & Arquivo:** nome da class e do arquivo devem ser sempre iguais

Nome de variável e nome de método sempre começa com letra minúscula,  
caso sua variável tenha mais duas palavras use os métodos snakcase ou smallcase

**Variáveis:** existem 8 tipos de variáveis no java  
  
**Byte:**

**Short:**

**Char:**

**Int:**

**Long:**

**Float:**

**Double:**

**Boolean:**

**String:** nota não é bem uma variável, é uma classe que serve para armazenar funções e textos

**Operadores:** Sinais de operação matemática

**Aritméticos:** +adição, -subtração, \*multiplicação, /divisão, % resto de divisão

**Bônus função.** Math.pow(x, y); representa a operação de potência matemática assim como a de raiz se você souber bem as propriedades, X é a base, Y é a potência

**Atribuição: (os exemplo estão relacionados, sempre pegando o anterior como referência**

**= (atribui um valor a variável no momento em que é usado)**

exemplo : valor = 2

**+=(esse operador atribui a variável o novo valor somado ao que ele já é ou era) considere que valor é igual a 2 ( valor é 2, 2 é atribuído ao valor )**  
 exemplo: valor +=1 // ou seja valor aqui vai ser 3

isso é o mesmo de: valor = 2 + 1 // que também será 3

Valor = 3

**-=( essa operação segue a mesma ideia do +=, a diferença que aqui está subtraindo)**

Valor = 3

exemplo: valor -=1 // valor aqui volta a ser 2

é similar a valor = valor - 1 ou valor = 3-1 // também volta a ser 2

Valor = 2

**\*=( mesma ideia)**

Valor = 2

Valor \*= 2 ou Valor = Valor\*2 ou Valor = 2\*2

Valor = 4

**/=(mesma ideia)**

Valor = 4

Valor /= 2 ou Valor= Valor/2 ou Valor = 4/2

Valor = 2

**%=(mesma ideia)**

Valor = 2

Valor %= 1 ou Valor = Valor%1 ou Valor = 2%1

Valor = 0 ( resto de 2 dividido por 1 é 0)

**/=**

Valor = 4

Valo

**Operadores relacionais: sempre retorna verdadeiro ou falso**

int valor = 2;  
boolean t = false;

**== : sinal matemático de equivalente**

ex: t=(valor == 2); // t=true

**!= : sinal matemático de diferente**  
 ex: t=(valor != 2);// t=false

**<: sinal matemático de menor**

ex: t=(valor < 2);// t=false

**<=** : **sinal de matemático de menor ou igual**

ex: t=( valor <= 2);// t=true

**> :** **sinal matemático de maior**

ex: t=(valor > 1); // t=true

**>=: sinal matemático de maior ou igual**

ex: t=(valor>= 1); // t=true

**Operadores lógicos:**

**&&:** **é equivalente a “e” nele ambas afirmações vinda antes e depois tem que estarem como corretas para que o resultado seja verdadeiro, apenas uma falsa, o resultado final será falso**

ex:

int valor = 30;  
boolean teste = false;

teste = valor < 40 && valor >20; // teste = true

teste = valor < 40 && valor>30; // teste = false

**||: é equivalente a “ou” nele se qualquer afirmação for verdadeira o resultado será verdadeiro, só será falso se todas forem falsas:**

ex

int valor = 30;  
boolean teste = false;

teste = valor > 30 || valor > 20; // teste = true  
teste = valor > 30 || valor < 20; // teste = false

**If Then Else**

**ex:**

if (preco<0) {

System.out.println(“ O preço do produto não pode ser negativo”);

}

else {

System.out.println(“Produto cadastrado com sucesso”);

}

**Loops**

**primeira forma:**

int contador = 0;

while (contrador<100) {

System.out.println(“E ai truta?”);  
 contador ++;

}

**segunda forma:**

for (int contador = 0; contador<100; contador ++) {  
 System.out.println(“E aí truta?”);  
}

**switch case break:** uma cadeia de condições, muito bom para menus

exemplo:

switch (opcao) {

case 1:

System.out.println("Opcao 1 selecionada");

break;

case 2:

System.out.println("Opcao 2 selecionada");

break;

}

Scanner scanner = new Scanner(System.in); este comando é para que aquele processo possa receber valores de entrada do usuário

**public**: é um modificador que é possível acessar de qualquer lugar, qualquer momento que criar a classe instância é permitido usar ele

**private:** só é possível usar esse método/modificador de dentro da própria classe

**protect:** pode usar a classe dentro do mesmo pacote, ou caso você vá herdar também pode

**default:** parecido com o protect porém só acessa via herança

**extend:** você pode herdar a classe em outro lugar, ou seja continuar a desenvolver a classe em outra classe, aquela determina classe recebe acesso a outra classe, sendo as duas como uma só

exemplo: para verificar os exemplo acesse link GitHub e acesse a página classe e objetos:

public class Quarta extends Primeiro

obs: class quarta recebe acesso a class primeira

public integer:

public é o modificador, integer é o retorno

void: é um retorno vazio

caso você utilize outra coisa ao em vez de void é bom usar um return

ex:

public int metodoInt() {

return 0;

}

**métodos estáticos:** São métodos que não estão dentro de uma instância de uma classe, ele terá uma palavra reservada para ser utilizada, é parecido com um método porém você precisa de uma palavra reservada. e você não pode usar instanciando uma classe. Um método estático não consegue acessar um método não estático. um método estático só pode acessar outro método estático

private static Integer variavel =1;

public static Integer metodoEstatico() {

return variavel;

}

classe é uma nomenclatura para algo que representa várias instâncias, e instância é uma única coisa específica.

**Linha de raciocínio para POO em java :**

Vamos começar pensando classe como um conceito que é tudo que representa várias coisas, por exemplo profissão, profissão contém um conjunto de profissionais, cada profissional pode ser um atributo mas também pode ser uma classe já que a profissão médico por exemplo contém vários médicos, agora a pessoa em si não pode ser uma classe, pois aqui a pessoa é vista como valor único, indivisível, é uma coisa só, um valor bruto. Claro que se mudarmos a perspectiva uma pessoa pode ser uma classe de células, ou seja um organismo que contém várias células, e por sua vez essas células podem ser atributos brutos ou um conjunto de átomos, e esse átomos de novo únicos ou conjunto de partículas positivas, neutras, negativas ou até em seguida quânticas. Perceba que aqui tudo depende do contexto ou seja do problema, e o que vai definir o que são classes, e atributos é a lógica ou seja o algoritmos que você tá usando para resolver os problemas em um passo a passo, o POO nada mais é que uma ferramenta, uma maneira de facilitar seu algoritmo. Outra coisa sempre que temos um conceito alongado de classe, nessas situações que temos classes dentro de uma classe podemos chamar de extensão aquela classe que se origina de outra classe e assim por em diante a extensão da extensão.

Outra coisa pra se ressaltar é o conceito e associação entre instância, atributo e objeto , de novo aqui tudo depende do contexto mas o padrão a ser aplicado é sempre o mesmo, a instância é um valor único daquela classe porém pode também ser visto como um atributo.

exemplo: informações da tabela profissão , podemos dividir profissão como uma classe que tem várias instâncias como médico, engenheiro, todas essas funções possuem um conceito só, porém aqui se quisermos dar um zoom na nossa base conceitual, médico também pode ser uma classe com instância Eduardo por exemplo. Então note que aqui médico era apenas uma instância que por sua vez se tornou uma classe com outra instância . Agora o atributo obrigatoriamente deve levar a um valor único de alguma coisa sempre, por exemplo médico é classe, o nome é um atributo, pois existem vários nomes de várias pessoas que são médicas porém o nome “eduardo” em si é a instância ou objeto, então note que nome é o atributo dado a classe médico, e a instância/objeto é “eduardo”. Também podemos usar outro exemplo, se abaixarmos o zoom, profissão pode ser visto como um conjunto de conceitos onde cada conceito corresponde a função, nesse caso profissão seria a classe, o atributo seria a função, e por sua vez objeto seria o nome daquela função exemplo “médico”. perceba que tudo depende do contexto e da profundidade da coisa, mas sempre associe atributos como uma categorias para uma classe, a classe é como uma entidade e possui várias atributos como visto em SQL, e por sua vez cada valor de um atributo é uma instância objeto, classe=funcionário, salário=atributo, 1200 reais= instância, objeto.

**final:** a constante não pode ser alterada com esse comando, é bom quando vc usar um parâmetro que nunca é alterado

getters e setters:

**get:** adiciona um método a aquela variável

**set:** adiciona um método de uma variável a outra variável