Java operações aritméticas

Int i = x>=y?2:4

se x maior igual a y então i = 2 senao i = 4

== igual igual

!= diferente

&& = e

Operador e

|| = ou

Operador ou

^ = ou exclusivo

Operador ou exclusivo sera falso se todas opções forem verdadeiras, so sera verdadeiro se apenas uma for verdadeira

! = não

Operador não, nega a afirmação, ou seja inverte o valor

Quando comparar string d objetos:

String nome1 = “Marcos”

String nome2 = new String “Marcos”

Usar o

String res = nome1.equals(nome2)?”Igual”:”Diferente”

Pela forma de ser criada a estrutura ser diferente com == ele não dará o resultado esperado mesmo com o mesmo conteudo

Para ter entrada de valores usar

Scanner teclado = new Scanner(System.in);

Cria o objeto teclado que busca o valor digitado

String nome = teclado.nextLine();

O final muda de acordo com o tipo de variável, como float, double

Exemplo de Swing

int nasc = Integer.parseInt(tAno.getText());

Cria uma variável inteira nasc que vai ser puxada do campo tAno

int idade = 2024 - nasc;

rIdade.setText(Integer.toString(idade));

Muda o label (campo) rIdade para o resultado da variável idade (o integer.toString e para transformar era numa String)

String situ = idade>=18?"Maior":"Menor";

Coloca na String situ a seguinte afirmação - se idade maior ou igual a 18 escrever maior senão escrever menor

rSitu.setText(situ);

Muda o label (campo) rSitu para o resultado da variável situ

Neste caso o objetivo do projeto foi descobrir a idade de uma pessoa a partir de seu ano de nascimento e dizer se é menor de idade ou maior

Quebrar linha

“<html> alguma coisa /<html>”

Vai quebrar o texto caso ele seja muito grande

Numero aleatório

Double n = 1 + Math.random() \* (5 – 1);

Neste caso o numero aleatório gerado é um double, por isto esta classe

O numero 5 no começo e o 1 no final estão ai pois é o intervalo desse numero aleatório, nesse caso entre 1 e 5

Caso fosse de 3 até 10 ficaria

Double n = 3 + Math.random() \* (10 – 3);

Pois primeiro o math.random pega um numero aleatório como 0.4 multiplica por 7 que resulta em 2.8 e soma com 3, dando 5.8 este numero aleatório

Obs. Nestes casos nunca resultar no valor 10, no máximo 9.99... então colocar um valo um pouco maior, como 10.001

Se você quiser apenas números inteiros devera fazer, como por exemplo de 1 a 5:

double n = (int) 1 + Math.random() \* (6-1);

Observe que coloquei o valor 6 no lugar do 5 para ser possível obter o valor 5

int v = (int)((int) n);

Esses dois (int) após o sinal, servem pa converter o numero n que é um double em inteiro, assim o resultado sera apenas números inteiros

**Funçoes para formatar uma String**

Colocar o nome da variável string antes ponto a função desejada

toLowerCase()

Coloca tudo em minúsculo

toUpperCase()

Coloca tudo em maiúsculo

trim()

Retira o espaçamento dos cantos da String, não os do meio

**Funçoes para recortar uma String**

substring(inicio)

Ele pega e monta a String a partir do inicio desejado e apaga as letras anteriores, como por exemplo na frase “Minha cachorra se chama lua”, usando essa função com qualquer inicio como por exemplo 2, ele vai retirar as duas primeiras letras da fase, pois buscara do index 2 adiante, então o index 0 e 1, ficara de fora e ficara “nha cachorra se chama lua”, observa se, que esta forma so determina o inicio então ela continuara igual até o final

substring(inico, fim)

é a mesma coisa que a função acima mas você define o fim tambem

**Funçoes para substituir uma String**

replace(char, char)

esta função serve para substituir um caractere por outro, como por exemplo uma String nome = Marcos, usando a função nome.replace(‘r’, ‘z’), ele substituirá toda vez que tiver o caractere r pelo caractere z, nesse caso, nome ficaria Mazcos

Observação neste caso ele ira substituir apenas os r minusculos

replace(string, string)

esta função serve para substituir uma String por outra, como por exemplo uma String nome = Marcos Augusto Barcos, usando a função nome.replace(“os”, “xy”), ele substituirá toda vez que tiver a String os pela String xy, observa-se que ele é igual o anterior so modifica o fato de poder definir mais que um caractere por vez, nesse caso, nome ficaria Marcxy Augusto Barcxy

**Funçoes para buscar uma String**

indexOf

Ele serve para achar o index inicial de uma string desejada, no caso do nome: Marcos Augusto Barcos, fazendo int i =nome.indexOf(“os”), ele buscara a posição do primeiro os e colocara no int i. neste caso 4, pois ele começa no index 4

lastIndexOf

Ele serve para achar o index final de uma string desejada, no caso do nome: Marcos Augusto Barcos, fazendo int i =nome.lastIndexOf(“os”), ele buscara a posição do ultimo e colocara no int i. neste caso 19, pois o ultimo os começa no index 19

Funçoes para dividir uma String

str.Split(“”)

Esta função é feira para dividir uma String em varias outras, ele gera um vetor com as partes cortadas, como por exemplo string s = “potato apple lemon”, usando a função String[] vect = s.split(“ “), o programa, vai dividir a string toda vez que sbarra com um espaço, como determinado dentro do parêntese e criara tres vetores, um com potato outro com apple outro com lemon

Estruturas condicionais

If or Else

String v;

If m > 8 {

v = “Verdadeiro”:

}

Else {

v = “Falso”:

}

O primeiro comando é o se senão, no caso o if que sempre vem como if condição e abertura de chaves, quer dizer se m for maior que 8 entao v é verdadeiro, fecha chaves

Já o else que também tem que ter chaves aberta após seu uso , quer dizer senão, no caso senão dor aquilo v é igual a falso, fecha chaves

Exemplo condicional composta

if ((idade >= 16 && idade <18) || (idade>70)) {

Se (idade maior igual a 16 e idade menor que 18) ou idade maior que 70 faca

situ = "Voto opcional";

} else {

Senão faca

situ = "Voto obrigatorio";

}

Tambem podemos usar o else if { } como a estrutura senão se

Switch

switch (p) {

case 1:

tipo = "Saci";

break;

case 2:

tipo = "Bipede";

break;

case 4:

tipo = "Quadrupede";

break;

case 6, 8:

tipo = "Aracnideo";

break;

default:

tipo = "ET";

break;

}

O comando switch seria o comando escolha onde ele ira te retornar dependendo do valor de entrada escolhendo qual caso se encaixa, o default no final é o “outro”, ele é o caso escolhido quando a entrada não corresponde a nenhum outro caso

Obs.: No default não há a necessidade de colocar o break, mas também não tem problema ele ser colocado

Obs.: Ele só serve para números inteiros!

Estruturas de repetições

for (int i = 1; i < 10; i++){

System.out.println(i);

}

Estrutura for, nesse caso simplesmente o programa diz que um inteiro i tem valor igual a 1 e para i menor que 10 faça os seguintes comandos e soma mais um a i a cada repetição (caso do i++)

Exemplo tabuada

for (int tabuada = 1; tabuada <= 10; tabuada++){

for(int valor = 1; valor <= 10; valor++){

System.out.println(tabuada + " x " + valor + " = " + (tabuada \* valor));

Neste exemplo utilizamos um laço for dentro do outro para fazer uma tabuada

int cont = 1;

while (cont <= 10){

System.out.println(cont);

cont++;

função while, ou seja enquanto cont menor igual a 10 for verdadeiro então imprima cont

obs: todas as estruturas de repetição são abertas com chaves e tem que ser fechadas

public static void main(String[] args) {

int cont = 11;

do{

System.out.println(cont);

cont--;

}while (cont> 0 && cont <= 10);

}

O método Do faz a mesma função do while e e utilizado com ele, mas tem uma pequena diferença, usando do antes, você faz o programa rodar a primeira vez aquele laço, mesmo o while sendo falso, ou seja, vai checar se o while e verdadeiro a partir da segunda repetição, como nesse caso que mesmo o while so acontecendo sendo menor igual a 10, e vai imprimir o 11 e continuar o laço, pois o próximo numero com o decremento já é 10, se no lugar do 11 fosse 12, iria imprimir so o 12, pois ia rodar a primeira vez e após isso a condição ainda não seria verdadeira e encerrar

**Programação orientada a objetos**

**Nova classe(Carros)**

public class Carros { //cria a classe Carros, você pode adicionar vários objetos nessa classe, seguindo a estrutura dela direto do programa principal

int ano;

String cor;

void ligar(){

System.out.println("engine On......");

}

void desligar(){

System.out.println("engine off .");

}

void acelerar(){ //Cria o método acelerar, ele pode ser puxado no programa principal

System.out.println("vruummmmm");

}

**Programa Principal**

package carros;

public class Ferrari {

public static void main(String args[]) {

Carros ferrari = new Carros(); //Cria o objeto Ferrari na classe Carros

ferrari.ano = 2025; //na parte ano da estrutura Carros poe 2025 no objeto ferrari

ferrari.cor = "Vermelha"; //na parte cor da estrutura Carros poe vermelho no objeto ferrari

System.out.println("Carros: Ferrari");

System.out.println("Ano: " + ferrari.ano); // imprimi o ano salvo naquele objeto

System.out.println("Cor: " + ferrari.cor); // imprimi a cor salva naquela objeto

ferrari.ligar(); // convoca o método presente na estrutura carros chamado ligar

ferrari.acelerar();// convoca o método presente na estrutura carros chamado acelerar

}

}

Exemplo 2

**Classe carros**

package carros;

import java.util.Random;

public class Carros {

int ano;

String cor;

public Carros(){ // Isto é um construtor, ele vai dar o retorno do Chassi sem precisar pedir

Random gerador = new Random();

int chassi = gerador.nextInt(2000);

System.out.println("Chassi: " + chassi);

}

public Carros(int ano, String cor){ // isto também é um construtor, e serve para ao invez de vc lançar o valor de cada variável uma por uma, você coloca as duas no parêntese e ela já substitui os valores

this.ano = ano; // lança o valor do parêntese no ano

this.cor = cor; //lança o valor da cor na String cor

}

void ligar(){

System.out.println("engine On......");

}

void desligar(){

System.out.println("engine off .");

}

void acelerar(){

System.out.println("vruummmmm");

}

}

**Programa Principal**

package carros;

public class Ferrari {

public static void main(String args[]) {

Carros ferrari = new Carros(); //Usando esse método o programa vai utilizar o construtor com o chassi, e ira mostrar um numero para o chassi

ferrari.ano = 2025;

ferrari.cor = "Vermelha";

System.out.println("Carros: Ferrari");

System.out.println("Ano: " + ferrari.ano);

System.out.println("Cor: " + ferrari.cor);

ferrari.ligar();

ferrari.acelerar();

Carros fusca = new Carros(1975, "Bege"); //Desta forma ira o outro construtor, onde não precisa declarar o valor das variáveis uma por uma, as dentro do parêntese já vão tomar os devidos lugares

System.out.println("Carros: Fusca");

System.out.println("Ano: " + fusca.ano);

System.out.println("Cor: " + fusca.cor);

}

}

Obs: O java vai utilizar apenas um dos métodos construtores nesse caso

**Herança**

Herança é basicamente uma classe que herda coisa de uma outra, no caso a seguir vou criar uma classe avião, que herda características da classe carro

package Carros;

public class Aviao extends Carros { //O comando extends faz a nova classe avião herdar os atributos da classe Carros, ou seja os comando, variáveis e métodos criados na classe Carros, também estão presentes na classe avião

double envergadura;

void aterrizar(){

System.out.println("-------\_\_\_\_\_\_\_\_");

}

void acelerar(){ //Isto é chamado de polimorfismo, o comando acelerar já existe na classe carro, lança lo denovo na classe avião serve para reescrever esse comando toda vez que se utilizar a classe avião, então ao invés de ser “vrummmmmmmm”, o comando executado vai ser o abaixo

System.out.println("\_\_\_\_\_\_\_--------");

}

}

**Modificadores de acesso**

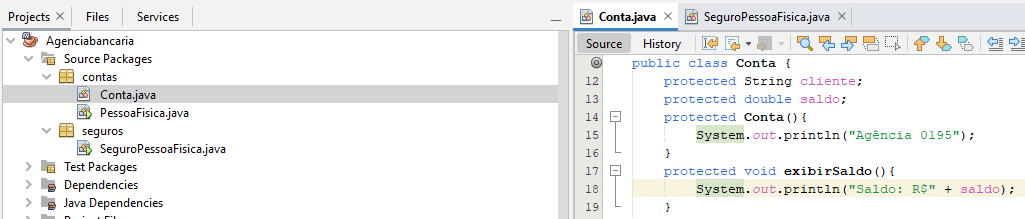
Ao criar uma nova classe fora do pacote das suas outras classes, para utilizar atributos das classe presentes no outro pacote, você primeiro precisar importar ela, utilizando o método “import pacote.classe” substituindo pelo nome do pacote e a classe desejada, mas mesmo cumprindo esse requisito, a única forma de utilizar os atributos lá presentes é transformando eles em públicos, ou seja, as variáveis métodos ou comando tem que ter um “public” antes de sua declaração na classe fonte, caso contrário você só poderá a utilizar dentro do mesmo pacote

public – Deixa as variáveis métodos ou comando públicos para que possa ser utilizado em qualquer lugar dentro do programa

private – Deixa as variáveis métodos ou comando privadas para que possa ser utilizada apenas dentro daquela classe

protected – Deixa as variáveis métodos ou comando protegidos para que possa ser utilizado apenas dentro do mesmo pacote

Como contornar esse problema caso o seu acesso seja protected



public class SeguroPessoaFisica extends Conta { //Transformando a classe fora do pacote em uma herança da classe que você quer, você consegue utilizar os atributos mesmo sendo protected

public static void main(String[] args){

SeguroPessoaFisica cc3 = new SeguroPessoaFisica();

cc3.cliente = “Robson Valverde”;

cc3.saldo = 9000;

System.out.println(“Cliente: “ + cc3.cliente);

cc3.exibirSaldo();

}

}

**Métodos**

Apesar de já termos criados métodos, vamos mostrar mais alguns exemplos e como criar um método que retorne um valor

**Classe Conta**

package contas;

public class Conta { //Esta é a classe principal do nosso programa

protected String cliente; //Variavel protegida, onde so os membros do mesmo pacote ou de heranças conseguem utilizar

protected double saldo;

protected Conta(){

System.out.println(“Agência 0195”); //Cria um comando, sempre que um novo objeto em Conta for criado, vai executar esse comando

}

protected void exibirSaldo(){ //Método exibirSaldo, toda vez que for chamado vai executar os passos a seguir, dentro do seu colchetes

System.out.println(“Saldo: R$” + saldo);

}

void sacar(double valor){ //Método sacar, toda vez que for chamado vai executar os passos a seguir, dentro do seu colchetes

saldo -= valor;

System.out.println(“Debito: R$” + valor);

}

void depositar(double valor){ Método depositar, toda vez que for chamado vai executar os passos a seguir, dentro do seu colchetes

saldo += valor;

System.out.println(“Credito: R$” + valor);

}

void transferir(Conta destino, double valor){ //Método transferir Metodo sacar, toda vez que for chamado vai executar os passos a seguir, dentro do seu colchetes, mas para ele funcionar os valores conta destino e double valor, tem que ser escritos no programa

this.sacar(valor);

destino.depositar(valor);

System.out.println(“Transferência: R$” + valor);

}

double soma(double cc1, double cc2){ //Método soma, percebe-se que ele foi criado sem a utilização do void, isso porque quando você quer que um valor retorne, você não pode utilizar o void

double total = cc1 + cc2;

return total; //O valor de total vai ser retornado

}

}

Observação a diferença entre métodos e comandos, são que os métodos sempre precisam ser chamados dentro do programa para serem executados, já os comando sempre que uma classe for chamada, ou chamada de uma forma especifica ele vai executar sem precisar de comando

**Programa principal**

package contas;

public class PessoaFisica {

public static void main(String[] args){

Conta cc1 = new Conta(); //Cria um novo objeto cc1 com os comandos da classe conta

cc1.cliente = "Leandro Ramos";

System.out.println("Cliente: " + cc1.cliente);

cc1.saldo = 10000;

cc1.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

cc1.sacar(1000); //Aqui estamos chamando o método sacar

cc1.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

System.out.println("-------------------------");

Conta cc2 = new Conta();

cc2.cliente = "Cirlene Santos";

System.out.println("Cliente: " + cc2.cliente);

cc2.saldo = 8500;

cc2.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

cc2.depositar(500); //Aqui estamos chamando o método depositar

cc2.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

System.out.println("-------------------------");

System.out.println("Trasnferencia");

System.out.println("Cliente: " + cc1.cliente);

System.out.println("Favorecido: " + cc2.cliente);

cc1.transferir(cc2, 2000); //Aqui estamos chamando o método transferir da conta cc1 (conseguimos ver isso com o cc1.transferir), e informando para ele qual o cliente de destino e o valor

System.out.println("");

System.out.println("Cliente: " + cc1.cliente);

cc1.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

System.out.println("Cliente: " + cc2.cliente);

cc2.exibirSaldo(); //Aqui estamos chamando o método exibirSaldo

System.out.println("-------------------------");

System.out.println("Relatorio Gerencial");

Conta gerente = new Conta();

double relatorio = gerente.soma(cc1.saldo, cc2.saldo); //Desta forma estamos atribuindo a variável relatório ao valor que vai retornar do método soma utilizando os parâmetros dentro dos parênteses

System.out.println("Saldo total nas contas: R$" + relatorio);

}

}

Observa-se que os métodos podem ser chamados varias vezes dentro de um programa

**Encapsulamento**

Quando se trata de dados quanto mais segurança melhor, e um método de dar mais segurança ao seu código é com o encapsulamento, nele você não consegue modificar diretamente variáveis ou coisas do tipo, assim dando mais segurança, a seguir mostra um método de se fazer esse encapsulamento

public class Conta {

private String cliente; //observa-se que você transformou a variável em private, então do programa principal você não consegue acessa-la, somente da classe onde ela esta

public String getCliente(){ //Este método foi criado para retornar o valor do cliente para o programa principal

return cliente;

}

public void setCliente(String cliente){ //Esse método foi feito para atribuir o valor de cliente ao objeto, observa-se que não é atribuído nem retornado diretamente

this.cliente = cliente;

}

private double saldo; //observa-se que você transformou a variável em private, então do programa principal você não consegue acessa-la, somente da classe onde ela esta

public double getSaldo(){//Este método foi criado para retornar o valor do saldo para o programa principal

return saldo;

}

public void setSaldo(double saldo){ //Esse método foi feito para atribuir o valor do saldo ao objeto, observa-se que não é atribuído nem retornado diretamente

this.saldo = saldo;

}

**Programa principal**

package contas;

public class PessoaFisica {

public static void main(String[] args){

Conta cc1 = new Conta();

cc1.setCliente("Leandro Ramos"); //Usa o método set para atribuir um valor ao cliente

System.out.println("Cliente: " + cc1.getCliente()); //Usa o método get para retornar o valor atribuído ao cliente

cc1.setSaldo(10000); //Usa o método set para atribuir um valor ao saldo

cc1.exibirSaldo();

cc1.sacar(1000);

cc1.exibirSaldo();

System.out.println("-------------------------");

Conta cc2 = new Conta();

cc2.setCliente("Cirlene Santos"); //Usa o método set para atribuir um valor ao cliente

System.out.println("Cliente: " + cc2.getCliente()); //Usa o método get para retornar o valor atribuído ao cliente

cc2.setSaldo(8500); //Usa o método set para atribuir um valor ao saldo

cc2.exibirSaldo();

cc2.depositar(500);

cc2.exibirSaldo();

{

{

**Formatando numero**

package fahrenheit;

import java.text.DecimalFormat; //Importa a biblioteca java.text.DecimalFormat

import java.util.Scanner; //Importa a biblioteca java.util.Scanner

public class Fahrenheit {

public static void main(String[] args) {

Scanner teclado = new Scanner(System.in); //Usado para criar uma forma de entrada pelo usuário

System.out.println("Digite a temperatura que deseja converter em celcius: ");

double temperatura = teclado.nextDouble(); //Usa o método teclado para trazer um valor tipo double

double celcius = (5\*(temperatura-32))/9;

DecimalFormat formatador = new DecimalFormat("#0.00"); //Usado para criar uma forma de formatar um numero, no caso com duas casas decimais como o numero no parentese

System.out.println("A temperatura em celcius é de " + formatador.format(celcius) + "º celcius"); //Invoca o formatador para formatar o celcius

}

}

**Vetores e Matrizes**

Vetores e matrizes em java são chamados de array

**Array simples**

public static void main(String[] args) {

String[] carros = {"Ferrari", "Fusca", "Bugati", "Lamborguine"}; //Criando um vetor do tipo String armazenando todos esses dados

System.out.println("Tamanho do array: " + carros.length); //Função para ver o numero de itens dentro de um vetor

System.out.println("Carro: " + carros[0]);

carros[0] = "Porsche"; //Dessa forma modificamos um elemnto dentro de um vetor

System.out.println("Carro: " + carros[0]);

System.out.println("Carros: ");

for (int i = 0; i < carros.length; i++) { //Forma para percorrer toda a lista

System.out.println(carros[i]); Imprimi os carros um por um

//Outra forma de percorrer toda lista seria:

For (String carro : carros){ //Esta é a forma mais eficaz de imprimir o vetor, você utiliza o laço for, dai você informa o tipo de variáveis que esta nesse vetor, nesse caso String e coloca um nome qualquer para criar uma Variavel dentro daquele laço e atribuir o valor do seu vetor, neste caso vai ser atribuído o valor d carros a variável carro, de toda a sua lista do vetor

System.out.println(carro);

}

}

}

}

Usando a combinação ctrl espaço após um for, aparece dicas de continuação para sua expressão

**Array Dinamico**

A diferença entre os arrays simples e arrays dinâmicos, são que o array simples já tem um tamanho pré definido, já os arrays dinâmicos pode ter seu tamanho diferente ao longo do programa

import java.util.ArrayList; //Esse método terá que ser importado

public class arrayList {

public static void main(String args[]) {

// TODO code application logic here

ArrayList<String> contatos = new ArrayList<>(); //Método de criar um array dinamico

contatos.add("José de Assis"); //Observa se que o array começa vazio, e com esse método você vai adicionando novos intens a ele

contatos.add("12 98576-5432");

contatos.add("ze@gmail.com");

contatos.add("Mauricio");

contatos.add("12 98456-2362");

contatos.add("maur@gmail.com");

System.out.println("Tamanho do Array: " + contatos.size()); //Método para descobrir o tamanho do seu Array dinâmico no momento

System.out.println(contatos);

System.out.println("Email Jose de Assis: " + contatos.get(2)); //Neste tipo para você puxar o item no index desejado você utiliza esse método .get, como no caso que vai puxar o item de contatos index 2

System.out.println("----------------------------------");

for (int i = 0; i < contatos.size(); i++) { //Forma de imprimir os itens dentro de um array dinâmico um por vez

System.out.println(contatos.get(i));

}

}

}

**Matrizes**

public static void main(String args[]) {

String[][] agenda = {{"José Assis", "12 99854-4786", "ze@gmail.com"}, {"Joao Barbosa", "12 95845-5446", "jobdf@gmail.com"}, {"Marta Inacio", "12 88652-4842", "marti@gmail.com"}}; //Forma de criar uma matriz com os [][], e depois para atribuir valor coloca um colchetes pa fechar tudo e um pa cada linha da matriz

System.out.println(agenda[1][0]);

for (int i = 0; i < agenda.length; i++) {

for (int j = 0; j < agenda.length; j++) {

System.out.println(agenda[i][j]); //Forma de imprimir uma matriz, cada item por vez

}

}

}

}