
V.office
Treinamentos
PROGRAMADOR
PRÁTICAS DE LINGUAGEM
PROGRAMAÇÃO

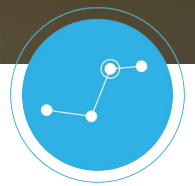
CONTEÚDO DO MÓDULO

1. Instalação e configuração do Java
2. Principais aplicativos para desenvolvimento
3. Comentário
4. Tipos de dados
5. Utilização de variáveis e constantes
6. Estrutura de controle
7. Estrutura de repetição
8. Vetor
9. Matriz
10. Programação estruturada ou modular

V.OFFICE Treinamentos

Conteúdo Aula 1

1. Instalação e configuração do Java
2. Principais aplicativos para desenvolvimento
3. Comentários



Instalação e configuração do Java

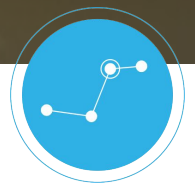
- **Instalar o Java? Para que? Já tenho o Java instalado.**
- **O java que temos instalado em nossas máquinas é a JVM (Java Virtual Machine);**
- **A JVM é o interpretador Java que é chamado sempre que mandamos executar alguma aplicação em Java;**
    - **Módulo de segurança dos bancos;**
- **Sim, precisamos instalar o Java (JDK), sem ele não conseguimos desenvolver em Java;**
- **É através dele que realizamos a compilação do nosso código e a execução;**

Download do Java

- **Faça o download no seguinte link: java;**
- **Selecione a opção: Accept License Agreement;**
- **Verifique a versão do seu windows 32 bits(x86) ou 64 bits (x64);**
- **Selecione a versão desejada e clique no link na coluna Download;**

Instalação do Java

- **A instalação é bem simples, não precisa ser configurado nada, apenas seguir as orientações de instalação.**

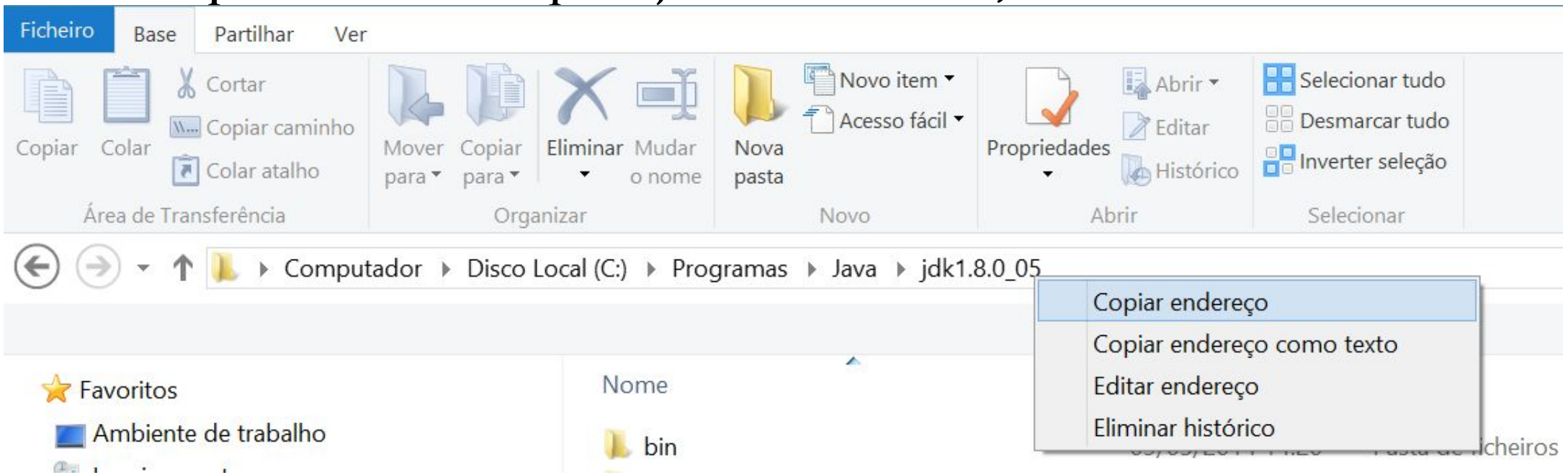


Testando o Java

- **Abra o prompt de comando:**
  - **java -version**
  - **javac -version**

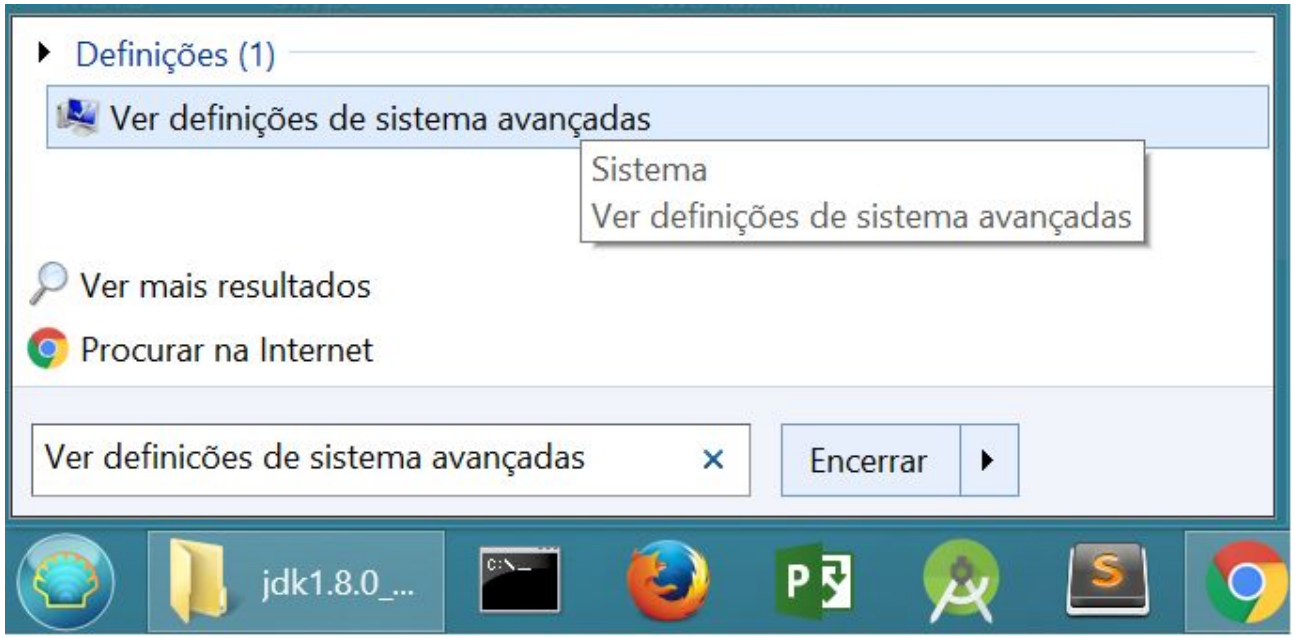
Configurar o Java

- **1º copiem o caminho que o jdk está instalado;**



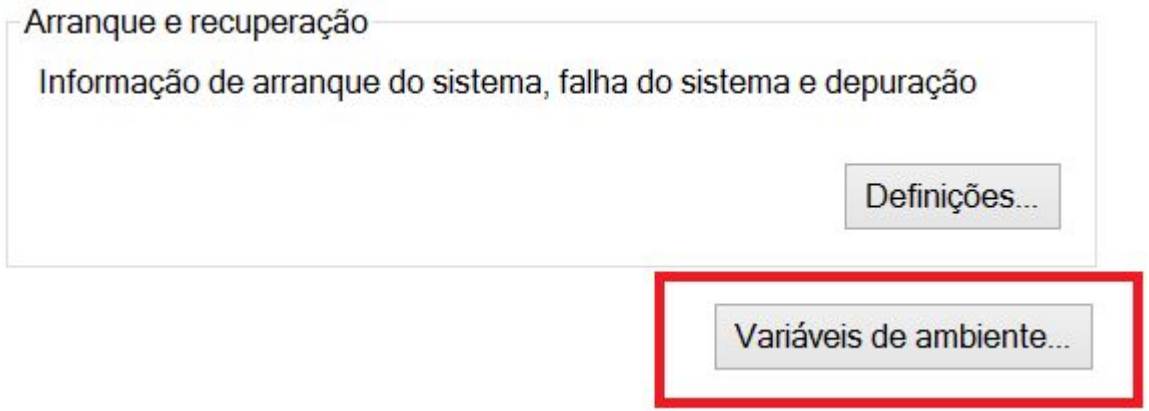
Configurar o Java

- **2º clique no iniciar e busque por: 'Ver definicões de sistema avançadas';**



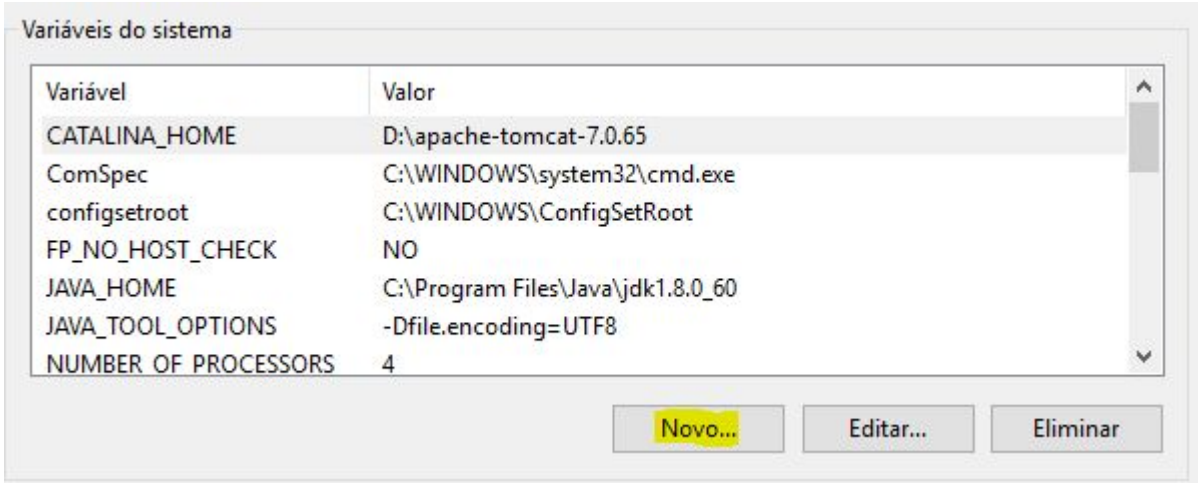
Configurar o Java

- **3º clique no botão Variáveis de ambiente;**



Configurar o Java

- **4º clique no botão Novo nas variáveis do sistema;**

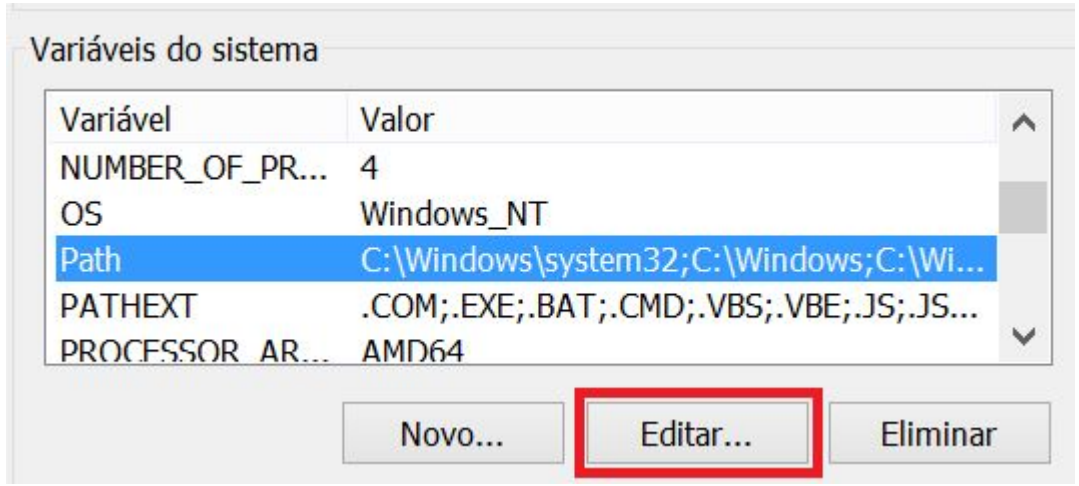


Configurar o Java

- **5º preencha o campo: 'Nome da variável' com o valor: 'JAVA_HOME' e no campo: 'Valor da variável' preencha com o caminho do jdk que copiamos anteriormente;**

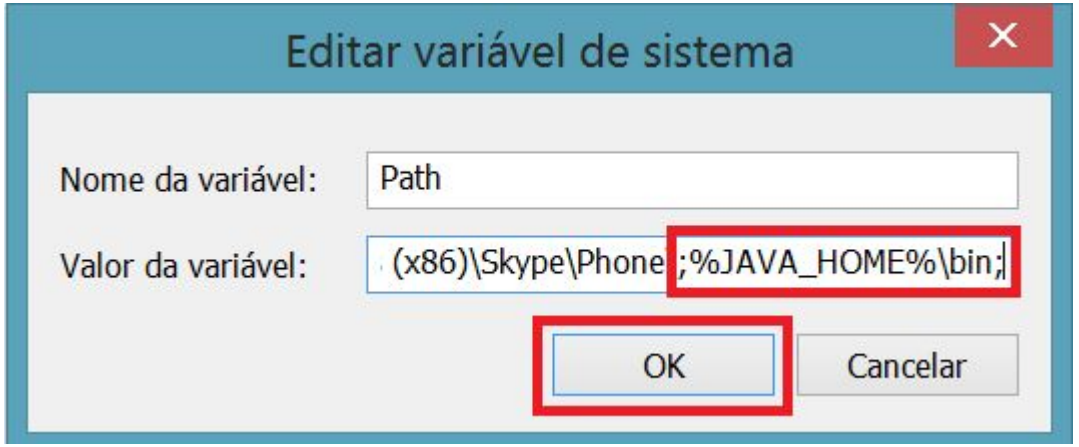
Configurar o Java

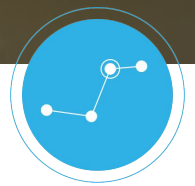
- **6º edite a variável de sistema já existente: ‘Path’;**



Configurar o Java

- 6º deve ser acrescentado no fim do campo: ‘Valor da variável’ que já existe o caracter: ‘;’ caso não exista e após o: ‘;’ acrescente o seguinte texto: ‘%JAVA_HOME%\bin;’;





Testando novamente o Java

- **Abra o prompt de comando:**
  - **java -version**
  - **javac -version**

Principais aplicativos para desenvolvimento

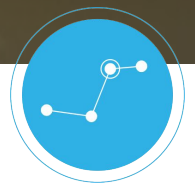
- **JEdit**
- **Netbeans**
- **Eclipse**
- **IntelliJ**
- **Notepad++;**

Principais aplicativos para desenvolvimento

O que nos facilita no momento do desenvolvimento quando utilizamos um editor de código de uma linguagem ao invés de um bloco de notas:

- Identação automática do código;
- Facilidade na observação de erros como falta de '}' em uma linha.
- Diferenciação de cores em palavras reservadas da linguagem.
- Auto-completar palavras, instruções.

Os editores servem apenas para a edição do código mesmo, para as demais tarefas como compilação precisamos de outras ferramentas específicas.

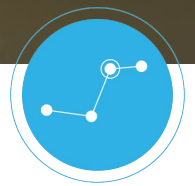


Compiladores

Compilador é um programa que faz a análise do código e transforma esse código em uma outra linguagem mais próxima da linguagem de máquina. Em java a compilação gera um arquivo intermediário chamado de bytecode (.class), ao invés de um arquivo com o código de máquina.

Durante o processo de compilação conseguimos eliminar diversos erros:

- Caracteres inválidos;
- Nomes de variáveis, funções e procedimentos inválidos;
- Sequência de comandos inválida: ausência de delimitadores '{}';
- Tipos e quantidades de parâmetros, retornos de funções, etc.



Interpretadores

Os interpretadores tem a função de traduzir em tempo de execução linha a linha para código de máquina.
Algumas linguagens não possui o compilador, apenas interpretadores tais como: JavaScript, Basic, Perls, etc.
Em linguagens apenas interpretadas, os erros só podem ser visualizados durante a execução do programa.
Apesar de o Java ser uma linguagem compilada, ela também é uma linguagem interpretada, o código .class tem que ser interpretado para a linguagem de máquina.

Escrevendo seu 1º programa em java

Copie o trecho de código abaixo e salve o arquivo com o nome: OlaMundo.java.

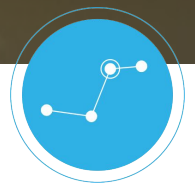
```
public class OlaMundo {
        public static void main(String args[]) {
            System.out.println("Ola turma!");
        }
}
```

Compilação e execução no Java

Utilizando o prompt de comando, acesse o local no qual você salvou seu arquivo .java

Compilação: javac OlaMundo.java

Execução: java OlaMundo



Comentários

Ocasionalmente precisamos documentar instruções do desenvolvimento, para que outros programadores saibam o que está sendo feito em determinados módulos.

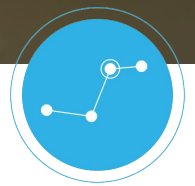
Temos 2 tipos de comentários:

Comentários inline '//':

```
// A linha toda é comentada.
```

Comentários Multiline '/* */':

```
/* Múltiplas linhas para comentário,
   Utilizado para textos longos.
*/
```



Comentários

Especificamente em Java, podemos criar o JavaDoc, que são comentários que depois podemos gerar um arquivo com uma documentação de nossas classes.

```
/**
    Tudo que eu digitar aqui dentro será depois gerado um arquivo.
*/
```

Dos: javadoc src/* -d Docs

Eclipse:

Project → Generate Javadoc... → seleciono o java doc dentro do bin do java e mando exportar.

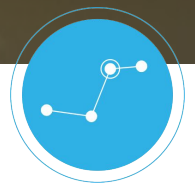
Conteúdo Aula 2

1. Tipos de dados
2. Utilização de variáveis e constantes

Tipos de dados, constantes e variáveis

Para guardar as informações que serão processadas durante a execução do programa, é necessário reservar espaços na memória do computador (variáveis).
Variáveis são espaços de memória alocados para guardar informações obtidas pelo usuário ou pelo próprio computador.
Os valores das variáveis podem ser alterados várias vezes.

Há momentos que essa informação não deve ser alterada, nesse caso chamamos essa variável de constante.



Tipos de dados, constantes e variáveis

```
codigo = 5; //Certo? Errado?
int codigo = 10;
codigo = 20
```

Para realizar a utilização de variáveis ou constantes, precisamos primeiro declarar elas.

Como trabalharemos com informações diferentes, não apenas valores inteiros, reais, literais, etc, necessitados da definição de um **tipo de dado**.

Tipos de dados, constantes e variáveis

Para a definição do tipo de dado, precisa primeiramente saber quais serão os dados armazenados nesta variável.

Uma das razões para utilizarmos o tipo de dado é a otimização do uso de memória.

Tipos de dados

Numérico

Inteiro: 100; 200; 20; 30; 1.

Real: 10,7; 200,0; 0,06.

Literal

Valores alfanuméricos, em forma de texto: “100,5”, “Hoje o dia está lindo.”; “R$10,00”; “10 Reais.”.

Lógico

Armazena valores do tipo: “Verdadeiro” e “Falso”.

Tipos de dados

Tipos inteiros:

Tipo	Memória consumida	Valor Mínimo	Valor Máximo
byte	1 byte	-128	127
short	2 byte	-32.768	32.767
int	4 bytes	-2.147.483.648	2.147.483.647
long	8 bytes	-9.223.372.036.854.775.808	9.223.372.036.854.775.807

Tipos de dados

Tipos reais:

Tipo	Memória consumida	Valor Mínimo	Valor Máximo	Precisão
float	4 bytes	-3,4028E + 38	3,4028E + 38	6 -7 dígitos
double	8 bytes	-1,7976E + 308	1,7976E + 308	15 dígitos

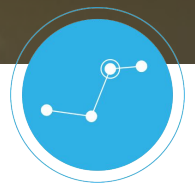
Tipos de dados

Tipos literais

O tipo char ocupa 2 bytes, o que torna o Java ideal para programar em línguas que utilizam caracteres diferentes do padrão ASCII.

O padrão ASCII utiliza apenas um byte que fornece 256 letras diferentes, mas o padrão utilizado em Java (ISO) nos dá a possibilidade de até 65.536 caracteres diferentes.

O char permite que seja armazenado um único caractere (letra, símbolo, etc).



Tipos de dados

Tipos literais

Para a utilização de um conjunto de caracteres (palavras, frases, etc), deve-se utilizar a String.

```
char letra = 'Palavra'; //esta correto isso?
String palavra = "Teste";
char letraCorreta = 'A';
```

Outro detalhe importante, quando utilizamos String, o conteúdo armazenado nessa variável fica entre aspas duplas ("") já quando utilizamos char, o conteúdo fica entre aspas simples ('').

Tipos de dados

Tipo lógico

O tipo de dado boolean nos permite armazenar apenas 2 valores true ou false (Verdadeiro/Falso).
Com esse tipo de dados podemos armazenar valores lógicos sendo possível esses valores serem resultantes de expressões lógicas.

```
boolean isFeminino = true;
boolean isPar = 4%2 == 0;
```

Tipos de dados

Além dos tipos primitivos que vimos anteriormente, podemos criar nossos próprios tipos de dados, são chamadas estruturas de dados, ou tipos de dados compostos.

Constantes e variáveis

Constantes

Constante como o próprio nome já diz, é algo constante, fixo que não pode ser alterado.
Esse valor não poderá ser alterado durante a execução do sistema.
Ex: Maior idade no Brasil hoje é 18 anos, esse valor durante a execução do sistema não deve ser alterado em momento algum.

```
final int MAIOR_IDADE = 18;
final boolean IS_ATIVO = true;
final String PESSOA_FISICA = “PF”;
final char SEXO_FEMININO = ‘F’
```

Constantes e variáveis

Variáveis

Ao contrário das constantes, as variáveis podem ser alteradas durante a execução do programa, isso é normal acontecer.

```
double saldo = 200.00;
int qtd = 10;
boolean isFeminino = true;
char tipoEmpresa = ‘F’;
String nome = “Maria”;
Saldo = 1000.00;
```

Constantes e variáveis

Para guardar as informações que serão processadas durante a execução do programa, é necessário reservar espaços de memória no computador (variáveis).
Variáveis são espaços de memória alocados para guardar informações obtidas pelo próprio computador ou pelo usuário.
Essa variável pode ser alterada várias vezes, seja decorrente a cálculos feitos pelo sistema, ou pelo usuário que alterou o valor.
Há casos em que essa informação não deve ser alterada, nesse caso chamamos essa variável de constante.

Constantes e variáveis

Para utilizar variáveis ou constantes, precisamos primeiro declarar elas.

int codigo = 10;
codigo = 20

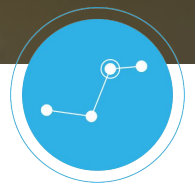
Como iremos trabalhar com dados diferente não apenas valores inteiros, necessitados da definição de um **tipo de dado**.
Para a definição do tipo de dado, precisa primeiramente saber quais dados serão armazenados nesta variável.
Uma das razões para utilizarmos o tipo de dado é a otimização do uso de memória.

Identificadores

Os nomes das variáveis (identificadores) são muito importantes para o entendimento do programa durante o desenvolvimento.
É muito importante que os nomes das variáveis sejam sempre o mais claro possível.

Nome de uma variável que irá armazenar o nome de uma empresa:

```
String nome;
Sring nomeEmpresa;
```



Identificadores

Java é uma linguagem case-sensitive, o que é isso?
Java diferencia as letras maiúsculas e minúsculas.

```
String datadecriacao;
String dataDeCriacao;
String DataDeCricao;
String datadeCriacao;
String dataDecricao;
```

Desta forma devemos ter muito cuidado quando definimos nossos identificadores.

Identificadores

Nos identificadores é permitido a utilização de:

- Letras de A a Z (maiúsculas e minúsculas).
- _ (underline)
- $ (cifrão)
- Números de 0 a 9, somente após o 2º caractere

Ex:

exemploResidencial
contador1
_valor
$salario
dataDeNascimento
STATUS_RESERVA
PF

Identificadores

Os identificadores podem ter 1 ou mais caractere.
Devem iniciar com uma letra.
Não podem ter espaços em branco no meio do nome;
Devem ser o mais claro possível em relação a função que terão na variável ou constante.

Palavras reservadas do Java

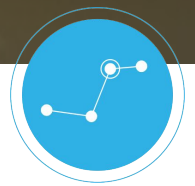
abstract	double	int	super
boolean	else	interface	switch
break	extends	long	synchronized
byte	final	native	this
case	finally	new	throw
catch	float	package	throws
char	for	private	transient
class	goto	protected	try
const	if	public	void
continue	implements	return	volatile
default	import	short	while
do	instanceof	static	

Recebendo dados do usuário

No mundo real grande parte dos valores das variáveis são informados pelos usuários.

Além dos dados vindos dos usuário, obtermos os valores através de:

- **Banco de dados.**
- **Serviços de internet.**
- **Bibliotecas de programas.**

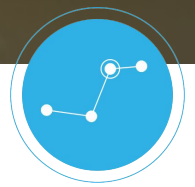


Lendo dados do teclado

Existem 2 forma mais simples de obtermos dados do usuário pelo teclado:

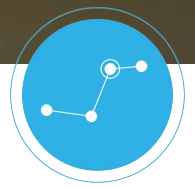
Scanner (Prompt de comando)

JOtionPane (Janelinha com interface gráfica)



Sacanner

```
Import java.util.Scanner;
public class ExemploTeclado {
        public static void main(String args[]) {
                String nome;
                byte idade;
                int numero;
                boolean cadastrado;
                Scanner teclado = new Scanner(System.in);
                System.out.println("Entre com o seu nome: ");
                nome = teclado.nextLine();
                System.out.println("Entre com a sua idade: ");
                idade = teclado.nextByte();
                System.out.println("Entre com o valor do emprestimo: ");
                numero = teclado.nextInt();
                System.out.println("Tem casa própria? ");
                cadastrado = teclado.nextBoolean();
        }
}
```



JOptionPane

```
import javax.swing.JOptionPane;
public class ExemploTecladoJOptionPane {
        public static void main(String args[]) {
                String nome = JOptionPane.showInputDialog("Entre com o seu nome: ");
                char sexo = JOptionPane.showInputDialog("Entre com o seu sexo(F/M):
        ").charAt(0);
                byte idade = Byte.parseByte(JOptionPane.showInputDialog("Entre com a sua
        idade: "));
                int numero = Integer.parseInt(JOptionPane.showInputDialog("Entre com o
        valor do emprestimo: "));
                JOptionPane.showMessageDialog(null, "Nome: " + nome
                                + "\nIdade: " + idade
                                + “\nSexo: ” + sexo
                                + "\nEmpréstimo: " + numero);
            }
}
```

Exercícios

Utilizando sempre o melhor tipo de dado afim de otimizar o uso da memória, façam os seguintes exercícios.

1. **Implemente um programa para calcular a área de um trapézio, onde:**
   a = altura
   b = base menor
   B = base maior
   área = (a . (b + B)) / 2
2. **Faça o programa acima calcular utilizando valores reais e depois imprimir na tela duas informações:**
   Valor exato da área.
   Valor arredondado para inteiro.

Exercícios

Utilizando sempre o melhor tipo de dado a fim de otimizar o uso da memória, façam os seguintes exercícios.

3. **Obter o salário de 5 funcionários e informar a média salarial dos funcionários.**
4. **Faça um programa que receba o valor do produto e o percentual de aumento que esse produto terá.**
5. **Escrever um programa para determinar o consumo médio de um automóvel sendo fornecida a distância total percorrida pelo automóvel e o total de combustível gasto.**

Exercícios

Utilizando sempre o melhor tipo de dado a fim de otimizar o uso da memória, façam os seguintes exercícios.

6. **Escrever um programa que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, informar o seu nome, o salário fixo e salário no final do mês.**
7. **Escrever um programa que leia o nome de um aluno e as notas das três provas que ele obteve no semestre. No final informar o nome do aluno e a sua média.**

Exercícios

Utilizando sempre o melhor tipo de dado a fim de otimizar o uso da memória, façam os seguintes exercícios.

8. **Escrever uma programa em que leia dois valores para as variáveis A e B, e efetuar as trocas dos valores de forma que a variável A passe a possuir o valor da variável B e a variável B passe a possuir o valor da variável A. Apresentar os valores trocados.**
9. **Ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é: F=(9*C+160) / 5, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.**

Conteúdo Aula 3

Estrutura de Controle **1**

Desvio condicional Simples **a.**

Desvio condicional composto **b.**

Desvio condicional simples

Uma estrutura de decisão (condicional ou de seleção) nos permite a execução de determinadas instruções caso uma condição seja satisfatória ou não.

Essas condições são representadas por expressões lógicas:

```
if (nota > 7) {
```

Devemos utilizar estruturas de decisão, sempre que nos depararmos com mais de uma possibilidade de ação.

Veremos daqui para frente que grande partes dos programas necessitam de estruturas condicionais.

Desvio condicional simples

Nessa estrutura condicional, a execução de determinadas instruções dependem de um teste prévio.
Se a condição, que é uma expressão lógica for verdadeira, o conjunto de instruções deve ser executado, caso contrário, o conjunto de instruções é ignorado.
A sintaxe para a estrutura condicional simples em Java é:

```
if (<expressão boolean OU valor booleano>) {
    // instruções do bloco “verdadeiro”
}
```

Desvio condicional simples

```
public class ExemploCondicaoSimples {
    public static void main(String args[ ]) {
        int idade = 18;
        if(idade >= 18) {
            System.out.println(“Maior de idade”);
        }
        System.out.println(“Fim do programa”);
    }
}
```

Desvio condicional composta

Essa estrutura é para os casos em que precisamos executar determinadas instruções caso a condição seja verdadeira, e outras instruções caso seja falsa.

A sintaxe para a estrutura condicional composta é:

```
if (<expressão boolean OU valor booleano>) {
    // instruções do bloco “verdadeiro”
} else {
    // instruções do bloco “falso”
}
```

Desvio condicional composta

```
public class ExemploCondicaoComposta {
    public static void main(String args[ ]) {
        int idade = 18;
        if(idade >= 18) {
            System.out.println(“Maior de idade”);
        } else {
            System.out.println(“Menor de idade”);
        }
        System.out.println(“Fim do programa”);
    }
}
```

Exercícios

1. **Faça um programa que obtenha 3 valores e diga qual o maior valor entre eles e qual o menor valor entre eles.**
2. **Faça um programa que obtenha um ano e diga se esse ano é bissexto ou não. Sabe-se que a fórmula para saber se um ano é bissexto é a seguinte: `ano%4 == 0 && ano%100 != 0 || ano%400 == 0`.**
3. **Faça um programa que obtenha uma letra: F (Feminino) ou M (Masculino). Após obter a letra, escreva na tela Feminino se a pessoa digitou a letra F, e Masculino e a pessoa digitou a letra M.**
4. **Faça um programa que obtenha uma letra e informe na tela se essa letra é uma vogal ou uma consoante.**
5. **Faça um programa que leia o preço de 1 produto de 3 lojas diferentes e mostre na tela qual das lojas você deveria comprar o produto.**

Exercícios

6. **Faça um programa que leia 3 números e apresente-os em ordem crescente.**
7. **Faça um programa que leia 3 notas, e calcule a média e apresente: Aprovado, caso a média seja maior do que 7 e Reprovado, caso a nota seja menor do que 7.**
8. **Faça um programa que receba a idade de uma pessoa e mostre na saída em qual categoria ela se encontra:**

10-14 - infantil
15-17 - juvenil
18-25 - adulto

Exercícios

9. **Faça um programa que pergunte o preço de um produto e em quantas vezes irá ser pago o produto.**

   1 = 20% de desconto

   2 até 5 = 5% de acréscimo

   6 até 10 = 15% de acréscimo

Conteúdo Aula 4

Estrutura de Controle **1**

Desvio condicional encadeado **a.**

Tomada de decisão por seleção **b.**

Desvio condicional encadeado

As estruturas condicionais podem ser encadeadas, ou seja, podemos colocar outras estruturas condicionais dentro de uma estrutura “Se” ou “Senão”.

A sintaxe para a estrutura condicional encadeada em Java é:

```
if (<expressão boolean OU valor booleano>) {
    // instruções do bloco “verdadeiro”
    if(<expressão boolean OU valor booleano>) {
        // instruções do bloco “verdadeiro dentro de outro bloco verdadeiro”
    }
} else {
    // instruções do bloco “falso”
}
```

Desvio condicional encadeado

```
public class ExemploCondicoesAninhadas {
    public static void main(String args[ ]) {
        int x = 7;
        if(x + 5 > 0) {
            if(x > 0) {
                System.out.println(“positivo”);
            } else {
                System.out.println(“negativo”);
            }
            System.out.println(“Soma maior”);
        } else {
            System.out.println(“Soma menor”);
        }
        System.out.println(“Fim do programa”);
    }
}
```

Tomada de decisão por seleção

Quando temos várias condições para uma mesma variável do programa, ao invés de encadear estruturas condicionais, podemos utilizar uma outra estrutura para fazer múltiplas verificações.

Tomada de decisão por seleção

A sintaxe para a tomada de decisão por seleção é:

```
switch (<variável>) {
    case <valor> :
        // instruções
        break;
    case <valor> :
        // instruções
        break;
    default:
    // instruções
}
```

Tomada de decisão por seleção

```
public class ExemploSelecaoMultipla {
    public static void main(String args[ ]) {
        int x = 2;
        switch (x) {
            case 0:
                System.out.println(“Opção 0”);
                break;
            case 1:
                System.out.println(“Opção 1”);
                break;
            case 2:
                System.out.println(“Opção 2”);
                break;
            default:
                System.out.println(“Opção inválida”);
        }
    }
}
```

Exercícios

1. **Um funcionário irá receber um aumento de acordo com o seu plano de trabalho, de acordo com a tabela abaixo:**

Plano	Aumento
A	10%
B	15%
C	20%

Faça um programa que leia o plano de trabalho e o salário atual de um funcionário e calcula e imprime o seu novo salário. Use o comando switch.

2. **Faça um programa que leia um número entre 0 e 10, e escreva este número por extenso. Use o comando switch.**

Exercícios

3. **Crie uma calculadora usando a instrução SWITCH, que pergunte qual das operações básicas quer fazer (+, -, * e /), em seguida peça os dois números e mostre o resultado da operação matemática entre eles.**
4. **Faça um programa que obtenha um número de 0 até 5 e escreva esse número por extenso.**
5. **Faça um programa que obtenha um número de 1 até 12 e retorne o mês e a quantidade de dias que esse número representa. Lembre-se dos anos bissextos (Fevereiro possui 29 dias).**

Conteúdo Aula 5

Estrutura de Repetição **1**

- Repetição com interrupção no início **a.**
- Repetição com interrupção no fim **b.**

Estrutura de repetição

```
public class Listagem10Alunos {
    public static void main(String[] args) {
        System.out.println("Nome 1 - Nota 1");
        System.out.println("Nome 2 - Nota 2");
        System.out.println("Nome 3 - Nota 3");
        System.out.println("Nome 4 - Nota 4");
        System.out.println("Nome 5 - Nota 5");
        System.out.println("Nome 6 - Nota 6");
        System.out.println("Nome 7 - Nota 7");
        System.out.println("Nome 8 - Nota 8");
        System.out.println("Nome 9 - Nota 9");
        System.out.println("Nome 10 - Nota 10");
    }
}
```

Estrutura de repetição

Durante o desenvolvimento de um programa, é comum nos depararmos com a repetição de um bloco de instruções.

Ex:

Apresentar a lista de alunos: nomes e notas.

Para não repetirmos o mesmo bloco de instruções 10 vezes no código, utilizamos a estrutura de repetição.

Uma estrutura de repetição permite que um bloco de instruções seja executado repetidamente até que uma condição de interrupção seja satisfeita.

Essa condição de interrupção é representada por uma expressão lógica.

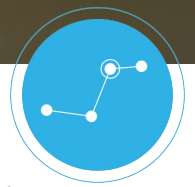
Repetição com interrupção no início

Sintaxe no java:

```
while (teste) {
    //instruções
}
```

Neste tipo de repetição um trecho do código é repetido enquanto a condição utilizada for verdadeira.

Neste caso o trecho do código pode nunca ser executada, ser executada uma única vez, ou várias vezes.



Repetição com interrupção no início

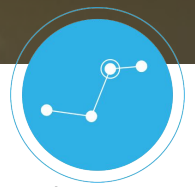
```
public class Listagem10Alunos {
    public static void main(String[] args) {
        int count = 1;
        while (count <= 10) {
            System.out.println("Nome " + count + " - Nota " + count);
            count++;
        }
    }
}
```

Repetição com interrupção no fim

Do mesmo modo que na repetição com interrupção no início, um trecho de código é executado enquanto a condição for verdadeira.

A diferença está na condição que sempre será verificada após a execução dos trechos de códigos que serão repetidos.

Neste caso o trecho do código será obrigatoriamente executado ao menos uma única vez, ou várias vezes.



Repetição com interrupção no fim

```
public class Listagem10Alunos {
	public static void main(String[] args) {
		int count = 1;
		do {
			System.out.println("Nome " + count + " - Nota " + count);
			count++;
		} while (count <= 10);
	}
}
```

Exercícios

1. **Uma empresa possui 10 funcionário. Ela resolveu realizar uma pesquisa entre seus funcionários, coletando dados sobre o salário e número de filhos. A empresa deseja saber:**

   a) média do salário da população;
   b) média do número de filhos;
   c) maior salário;
   d) menor salário;
   e) percentual de pessoas com salário até R$1000,00.

Exercícios

2. **Construir um algoritmo que calcule a média aritmética de vários valores inteiros positivos, lidos externamente. O final da leitura acontecerá quando for lido um valor negativo.**

3. **João tem 1,50 metro e cresce 2 centímetros por ano, enquanto Manoel tem 1,10 metro e cresce 3 centímetros por ano. Construa um algoritmo que calcule e imprima quantos anos serão necessários para que Manoel seja maior que João.**

Exercícios

4. **Em uma eleição presidencial existem quatro candidatos. Os votos são informados através de códigos. Os dados utilizados para a contagem dos votos obedecem à seguinte codificação:**

   1, 2, 3, 4 = voto para os respectivos candidatos;

   5 = voto nulo;

   6 = voto em branco;

Elabore um algoritmo que leia o código do candidato em um voto. Calcule e escreva:

a) total de votos para cada candidato;

b) total de votos nulos;

c) total de votos em branco;

Como finalizador do conjunto de votos, tem-se o valor 0.

Exercícios

5. **Escreva um algoritmo que calcule a média dos números digitados pelo usuário, se eles forem pares. Termine a leitura se o usuário digitar zero (0).**

6. **Escrever um algoritmo que leia um número n que indica quantos valores devem ser lidos a seguir. Para cada número lido, mostre uma tabela contendo o valor lido e o fatorial deste valor.**

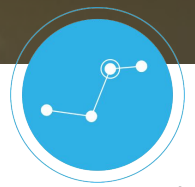
Exercícios

7. **Faça um jogo para descobrir o número correto, inicie gerando um número aleatório conforme comando que será apresentado abaixo, depois solicite para o usuário um número até que ele acerte o valor gerado aleatório, após cada erro, apresentar uma mensagem informando se o valor correto é maior ou menor ao valor que foi digitado. Ao fim, quando for acertado o valor, apresentar ao usuário em quantas tentativas ele conseguiu acertar o número.**

```
import java.util.Random;
public class ValorAleatorio {
	public static void main(String[] args) {
		int teste = new Random().nextInt();
		System.out.println(teste);
	}
}
```

Conteúdo Aula 6

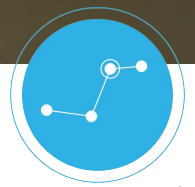
1. Repetição com variável de controle
2. Vetor



Repetição com variável de controle

Sintaxe no java:

```
for ([inicialização] ; [teste] ; [incremento]) {
    //instruções
}
```



Repetição com variável de controle

```
public class Listagem10Alunos {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println("Nome " + i + " - Nota " + i);
        }
    }
}
```

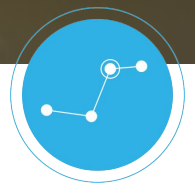
Repetição com variável de controle

Do mesmo modo que na repetição com interrupção no início ou fim, um trecho de código é executado enquanto a condição for verdadeira.

É escolhido este tipo de repetição quando já sabemos previamente o número de vezes que deverá ser repetida as instruções.

Esta estrutura de repetição possui um mecanismo para contar o número de vezes que os trechos de código são repetidos.

Podendo assim controlar o número de repetições.



Vetor

Vamos aprender agora estrutura de dados, na qual uma única variável pode armazenar inúmeros valores.

Utilizada quando precisamos armazenar uma coleção de valores.

Ex: armazenar 5 notas de um determinado aluno.

Essas estruturas são chamadas de tipos de dados compostos que se dividem em homogêneos(vetores, matrizes) e heterogêneos(registros).

Vetores são estruturas de dados lineares e estáticas que possuem um número fixo de elementos de um determinado tipo de dados.

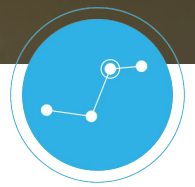
Vetor

Podemos acessar cada posição do vetor, indicando qual o índice que se deseja acessar. Lembrando que sempre se inicia da posição 0 um vetor.

Sempre declararmos um vetor devemos informar o tamanho do nosso vetor, pois é através desse tamanho que é reservado o espaço da memória.

Após a definição do tamanho do vetor, seu tamanho não poderá mais ser alterado.

Seus valores armazenados em cada posição do vetor podem ser alteradas a qualquer momento.

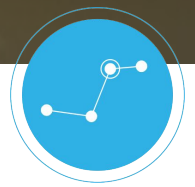


Vetor

```
public class ExemploMedia {
    public static void main(String args[]) {
        int[] idades = new int[3];
        double soma = 0;
        idades[0] = 32;
        idades[1] = 28;
        idades[2] = 15;
        for(int i = 0; i < idades.length; i++) {
            soma = soma + idades[i];
        }
        System.out.println(“Media = ” + soma / 3);
    }
}
```

Exercícios

1. **Escrever um algoritmo que leia um conjunto de 5 informações contendo, cada uma delas, a altura e o sexo de uma pessoa (código=1 - masculino, código=2 - feminino), calcule e mostre o seguinte:**
   a) a maior e a menor altura da turma
   b) a média da altura das mulheres
   c) a média da altura da turma.
2. **Faça um algoritmo que lê um valor N inteiro e positivo e que calcula e escreve o fatorial de N (N!).**
3. **Faça um algoritmo que leia 2 valores inteiros e positivos: X e Y. O algoritmo deve calcular e escrever a função potência X Y**

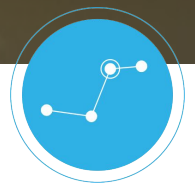


Exercícios

4. **Leia um conjunto de notas, cuja quantidade seja determinada pelo usuário. Calcule a média de todas elas. Exiba o conjunto das notas maiores do que a média calculada. Em seguida, de forma agrupada, exiba o outro conjunto de notas (menores do que a média).**
5. **Crie um array de inteiros “a” e um valor inteiro “x” e apresente na tela a quantidade de vezes que “x” aparece no array “a”.**
6. **Escreva um programa que recebe um array de números e devolve a posição onde se encontra o maior valor do array. Se houver mais de um valor maior, devolver a posição da primeira ocorrência.**

Conteúdo Aula 7

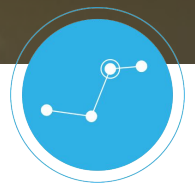
1. Matriz
2. Procedimentos e funções



Matriz

Matrizes são vetores bidimensionais que representam um conjunto de valores do mesmo tipo, referenciados pelo mesmo nome.

Na declaração de matrizes deverão ser fornecidas, principalmente o nome, o número de linhas e o número de colunas, além do tipo de dados da matriz.



Matriz

Existem algumas forma de declararmos uma matriz.

Com expressões de criação de vetores:

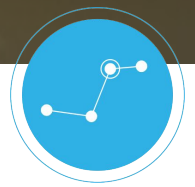
```
int matriz[][] = new int[2][3];
```

Declarando e inicializando:

```
int matriz[][] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

Com linhas de diferentes tamanhos:

```
int matriz[][] = new int[2][];
matriz[0] = new int[5];
matriz[1] = new int[3];
```



Matriz

Existem algumas forma de declararmos uma matriz.

Declarando e inicializando linhas de diferentes tamanhos:

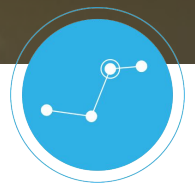
```
int matriz[][] = {{1, 2}, {3, 4, 5, 6, 7}, {8, 9, 10}};
```

Equivale as seguintes atribuições:

```
matriz[0][0] = 1;
matriz[0][1] = 2;

matriz[1][0] = 3;
matriz[1][1] = 4;
matriz[1][2] = 5;
matriz[1][3] = 6;
matriz[1][4] = 7;

matriz[2][0] = 8;
matriz[2][1] = 9;
matriz[2][2] = 10;
```



Matriz

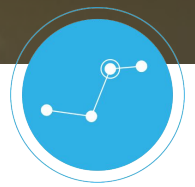
Para saber os tamanhos da matriz utilizamos a propriedade length:

```
int matriz[][] = {{1, 2}, {3, 4, 5, 6, 7}};
```

m.length -> determina o número de linhas.

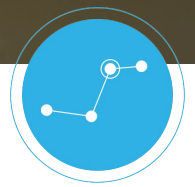
m[i].length -> determina o número de colunas da linha i.

```
for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[i].length; j++) {
        System.out.println(matriz[i][j]);
    }
}
```



Matriz

```
public class ExemploMatriz {
    public static void main(String args[]) {
        int[][] notas = new int[1][1];
        double soma = 0;
        notas[0][0] = 32;
        notas[1][0] = 15;
        for(int i = 0; i < notas.length; i++) {
            for(int j = 0; j < notas[i].length; j++) {
                soma = soma + notas[i];
            }
            System.out.println(“Soma notas = ” + soma);
        }
    }
}
```



Procedimentos e Funções

Procedimentos e Funções são sub-rotinas que executam uma tarefa específica.

Além das sub-rotinas que podemos criar, existem várias já criadas pelo Java que podemos utilizar: funções matemáticas, funções de texto que podem pegar partes de um texto(substring), contar a quantidade de caracteres, transformar tudo em maiúsculo, entre outras.

Procedimentos e Funções

Qual a utilidade?

Procedimentos e Funções

São muito utilizadas para organizar os programas, criando sub-rotinas que poderão ser reutilizadas em várias partes dos programas.

Dessa forma, ao invés de termos vários trechos de códigos idênticos, teremos esse trecho de código centralizado, ou seja, se possuir erros corrigiremos em um único local.

Procedimentos e Funções

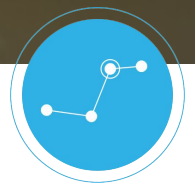
Qual a diferença entre função e procedimento?

Procedimentos e Funções

A principal diferença se deve ao fato de uma função obrigatoriamente retornar um valor, enquanto um procedimento não retorna valor algum, apenas tem o intuito de executar uma ação.

Procedimento

```
public class ExemploPocedimento {
    public static void main(String args[]) {
        for(int i = 0; i < 5; i++) {
            calcularSoma();
        }
    }
    public static void calcularSoma() {
        int numero1 = Integer.parseInt(JOptionPane.showInputDialog(“Número 1"));
        int numero2 = Integer.parseInt(JOptionPane.showInputDialog(“Número 2"));
        int soma = numero1 + numero2;
        JOptionPane.showMessageDialog(null, “Soma: " + soma);
    }
}
```



Função

```
public class ExemploFuncao {
    public static void main(String args[]) {
        for(int i = 0; i < 5; i++) {
            int soma = calcularSoma();
            JOptionPane.showMessageDialog(null, “Soma: " + soma);
        }
    }
    public static int calcularSoma() {
        int numero1 = Integer.parseInt(JOptionPane.showInputDialog(“Número 1"));
        int numero2 = Integer.parseInt(JOptionPane.showInputDialog(“Número 2"));
        //int soma = numero1 + numero2;
        //return soma;
        return numero1 + numero2;
     }
}
```

Exercícios

1. **Construa uma matriz 3 x 3, preencha ela com valores inteiros e após isso verifique se essa matriz é uma matriz identidade.**
2. **Escreva um programa para que mostre o menor e o maior valor de uma matriz 4 x 4.**
3. **Construa uma agenda médica, com os horários marcados para cada paciente. Será uma matriz 7 x 24, ou seja, contendo 7 dias e 24 horas no dia.**
4. **Escreva um programa que contenha 2 matriz 3 x 3 e uma matriz resultante da multiplicação das 2 matrizes anteriores.**
5. **Faça uma função que leia 10 valores positivos e retorna a média aritmética dos mesmos.**
6. **Faça um procedimento que leia 5 valores inteiros e retorne o maior e o menor deles.**

Conteúdo Aula 8

Procedimentos e funções **1**

Escopo de variáveis **a.**

Utilização de parâmetros **b.**

Escopo de variáveis

O escopo de uma variável vai de acordo com o bloco que ela foi declarada.

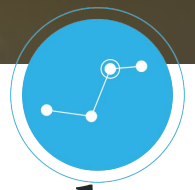
A variável é criada no primeiro acesso e é destruída após sair do bloco de execução.

Portanto quando uma variável é criada dentro de um método, ou limitador de blocos, essa variável será destruída assim que o interpretador sair do método ou bloco delimitador.

Porém existem as variáveis globais, que são variáveis que podem ser acessadas por toda a classe, essas variáveis devem ser evitadas ao máximo, pois são variáveis que podem estar expostas a alteração por qualquer parte do sistema.

Escopo de variáveis

```
import javax.swing.JOptionPane;
public class VariavelGlobal {
	static int global;
	public static void main(String[] args) {
		global = 10;
		JOptionPane.showMessageDialog(null, global);
		testeVariavelGlobal();
	}
	static void testeVariavelGlobal() {
		JOptionPane.showMessageDialog(null, global);
	}
}
```



Escopo de variáveis

```
import javax.swing.JOptionPane;
public class VariavelLocal {
	public static void main(String[] args) {
		int global = 10;
		JOptionPane.showMessageDialog(null, global);
		testeVariavelGlobal();
	}
	static void testeVariavelGlobal() {
		int global = 20;
		JOptionPane.showMessageDialog(null, global);
	}
}
```

Utilização de parâmetros

Os parâmetros são passados por valores, ou seja, a variável que foi passada como parâmetro quando foi invocado o procedimento ou função, não é a mesma variável que declaramos quando escrevemos o método.

Utilização de parâmetros

```
import javax.swing.JOptionPane;
public class Quadrado {
	public static void main(String[] args) {
		int numero = 10;
		int numero_quadrado = quadrado(numero);
		JOptionPane.showMessageDialog(null, numero + " elevado ao
 quadrado é " + numero_quadrado);
	}
	public static int quadrado(int num) {
		int quadrado;
		quadrado = num * num;
		return quadrado;
	}
}
```

Exercícios

1. **Faça uma função que recebe um valor inteiro e verifica se o valor é par ou ímpar. A função deve retornar um valor booleano.**
2. **Faça um procedimento que recebe a idade de um nadador por parâmetro e retorna a categoria desse nadador de acordo com a tabela abaixo:**

Idade	Categoria
5 a 7 anos	Infantil A
8 a 10 anos	Infantil B
11-13 anos	Juvenil A
14-17 anos	Juvenil B
Maiores de 18 anos (inclusive)	Adulto

Exercícios

3. **Escreva um procedimento que recebes 3 valores reais X, Y e Z e que verifique se esses valores podem ser os comprimentos dos lados de um triângulo e, neste caso, retornar qual o tipo de triângulo formado.**

Para que X, Y e Z forme um triângulo é necessário que a seguinte propriedade seja satisfeita:

O comprimento de cada lado de um triângulo é menor do que a soma do comprimento dos outros dois lados.

O procedimento deve identificar o tipo de triângulo formado observando as seguintes definições:

Triângulo Equilátero: os comprimentos dos 3 lados são iguais.
Triângulo Isósceles: os comprimentos de 2 lados são iguais.
Triângulo Escaleno: os comprimentos dos 3 lados são diferentes.

Exercícios

4. **Faça uma função que recebe a média final de um aluno por parâmetro e retorna o seu conceito, conforme a tabela abaixo:**

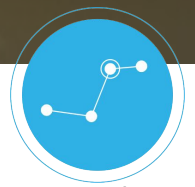
Nota	Conceito
de 0,0 a 4,9	D
de 5,0 a 6,9	C
de 7,0 a 8,9	B
de 9,0 a 10,0	A

Conteúdo Aula 9

Procedimentos e funções **1**

Recursividade **a.**

Refinamento sucessivo **b.**



Recursividade

A recursividade trabalha de forma similar a uma repetição, na verdade tudo que fazemos em laço pode ser feito em recursividade.

A recursividade nada mais é do que uma função dentro da outra, formando assim uma pilha.

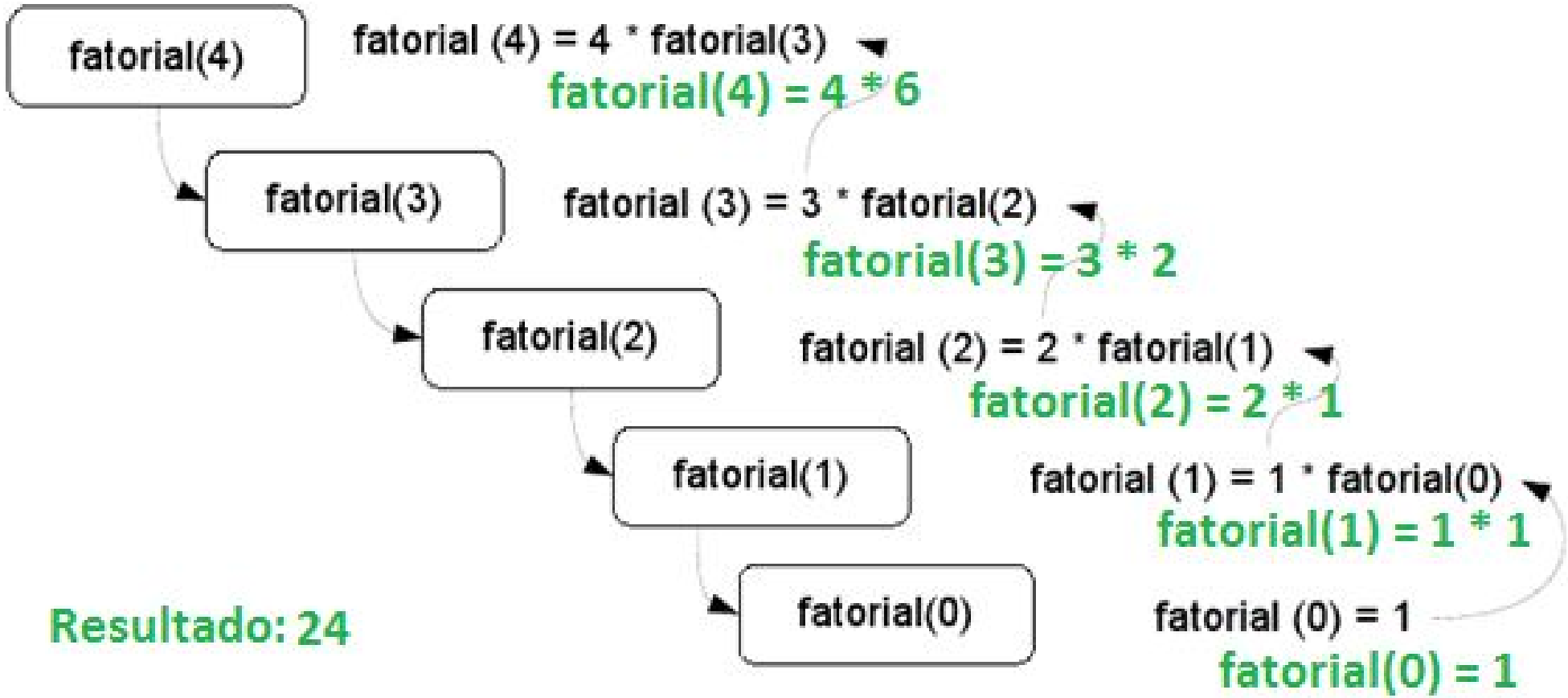
A estrutura de pilha consiste em realizar as chamadas até a base (caso base) e trazer os cálculos ou rotinas de cada instrução até o topo.

Recursividade

Sempre que trabalhar com fatorial temos que definir um caso base, para que assim o sistema não fique em loop infinito.

No caso a seguir que veremos fatorial, o caso base é o fatorial de 0, sempre que o número passado por parâmetro for 0, deve retornar o número 1.

Recursividade



Recursividade

```
import javax.swing.JOptionPane;
public class FatorialRecursividade {
	public static void main(String[] args) {
		int numero = 4;
		int fatorial = fatorial(numero);
		JOptionPane.showMessageDialog(null, "A fatorial de: " + numero + " é: " +
fatorial);
	}
	public static int fatorial(int num){
		if (num == 0) {
			return 1;
		} else {
			return num * fatorial(num - 1);
		}
	}
}
```

Refinamento sucessivo

A ideia de refinamento sucessivo é partir de um problema maior, e dividi-lo em problemas menores, de solução mais simples. Seria o termo dividir para conquistar.

Dessa forma o processo de divisão em problemas menores é repetido até conseguir dividir ao máximo o problema original.

Refinamento sucessivo

Principais vantagens do refinamento:

Estímulo a divisão em problemas menores, para um maior detalhamento em cada problema.

Incentiva a reutilização de códigos já prontos e facilitando o processo de detecção de eventuais erros de programação.

Refinamento sucessivo

Como conseguimos dividir o seguinte problema:

Multiplicar 2 matrizes, armazenar o resultado em uma terceira matriz e apresentar o resultado na tela.

Exercícios

1. **Faça um programa (utilizando recursividade) que peça para o usuário digitar um número, em seguida faça a soma de todos os algarismos do número.**
   Exemplos:
   2090 = 2 + 0 + 9 + 0 = 11
   1111 = 1 + 1 + 1 + 1 = 4

2. **Desenvolva um algoritmo que obtenha um número e calcule a soma de 0 até o número digitado.**