

Qualificação Profissional de Assistente de
Desenvolvimento de Sistemas

LÓGICA DE PROGRAMAÇÃO

GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

São Paulo – SP, 2019

Expediente

GOVERNO DO ESTADO DE SÃO PAULO
EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO
TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS
LÓGICA DE PROGRAMAÇÃO

Autores:

Eliana Cristina Nogueira Barion
Marcelo Fernando Iguchi
Paulo Henrique Mendes Carvalho
Rute Akie Utida

Revisão Técnica: Kelly Dal Pozzo

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação: Flávio Biazim

AGENDA 4

A LÓGICA APLICADA EM JAVA

Comandos
de Entrada
e Saída de
dados





Agora que você já instalou a IDE Java, seja o Netbeans, Eclipse, IntelliJ IDEA ou outro tipo de IDE, vamos estudar a estrutura de um programa Java e colocar em prática as estruturas lógicas de programação. Vamos lá?

A estrutura de um programa feito em Java

Agora que você já criou um código fonte em Java, você poderá aprender a estrutura básica de um programa. A seguir, temos uma imagem representando um código fonte em Java:

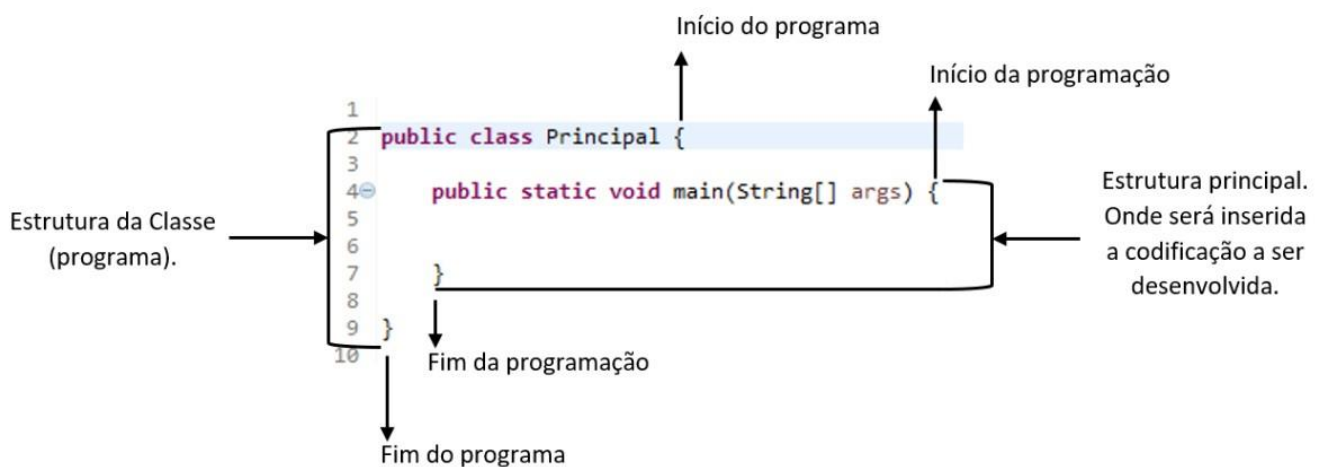


Imagem 21: A imagem representa a estrutura básica de uma classe o símbolo “abre e fecha chaves” significa início e fim de programa, respectivamente. A estrutura de um programa é composta por: public class “nome do Programa”, abre chave, public static void main (String[] args, fecha a chave { da estrutura principal, onde será inserida a codificação a ser desenvolvida e fecha a chave da classe principal.

Você pode ver na imagem que o programa em si está dentro de uma classe, essa estrutura é necessária para seu funcionamento, porém, para estudos de lógica de programação não abordaremos os conceitos de Orientação a objetos.

Os comandos Início e Fim contidos no Pseudocódigo são substituídos por chaves { } em Java, sendo a abertura de Chave { o início e, o fechamento de chave } o final do seu programa. Em algumas estruturas que veremos posteriormente, também utilizaremos marcações de início e fim por meio de chaves.

A partir de agora, além da sua forma utilizada no fluxograma e no pseudocódigo, os novos comandos também serão apresentados em Java.

O comando Escreva

Quando desejamos exibir alguma mensagem na tela do usuário utilizando o pseudocódigo, utilizamos o comando:

escreva
mensagem

Com o auxílio dele é possível enviar uma simples mensagem ao usuário do programa, como também, mostrar o resultado de uma conta. Na prática, você deverá utilizar este comando para mostrar tudo o que você deseja que o usuário veja.

Em Java, por ser uma linguagem de programação, devemos indicar o caminho completo da operação de exibição de mensagem para o usuário, que é feita da seguinte forma:

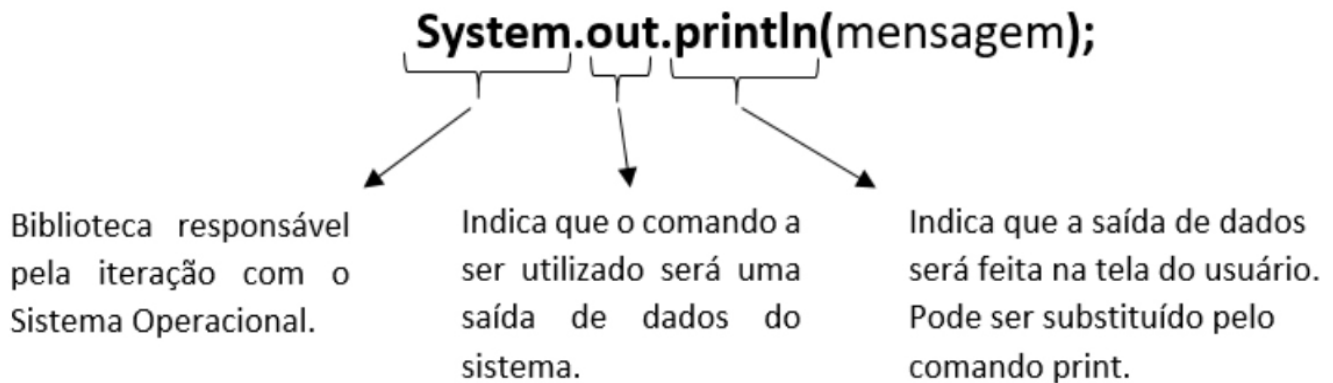


Imagem 22: Representação da estrutura do comando System.out.println. Descrição do comando: System: Biblioteca responsável pela interação com o Sistema Operacional. Out: Indica que o comando a ser utilizado será uma saída de dados do sistema. Println: Indica que a saída de dados será feita na tela do usuário. Pode ser substituído pelo comando print.

VOCÊ NO COMANDO

Refleta e responda: Agora que vimos o comando print e println, reflita e responda: Qual dos dois comandos você acha que é o mais eficiente? Existe alguma situação específica para a utilização de cada um dos comandos?

Veja se você acertou:

A resposta é muito simples: depende do caso! O comando print escreverá a mensagem na tela sem efetuar nenhuma modificação na mensagem. O comando println pulará uma linha na tela, ao final da escrita da mensagem.

Portanto, se a sua intenção é exibir a mensagem grudada na mensagem exibida anteriormente, utilize o **print**, caso você deseje que a mensagem seja exibida em uma nova linha, utilize o **println**.

Declarando uma variável

Para utilizar uma variável, devemos primeiramente declará-la no texto do programa. Mas o que significa declarar uma variável?

Com a declaração de variável avisamos o computador que ele deve criar um espaço na memória para receber um valor posteriormente. Todo o espaço na memória declarado deve ter um nome e ser vinculado a um tipo de variável. Informamos qual é o seu tipo (inteiro, texto ou real, por exemplo) e, além disso, qual é o nome que usaremos para referenciá-la no texto do programa. Por exemplo, para declarar uma variável do tipo inteiro que representa o número de matrícula de um aluno, utilizamos o seguinte código:


```
int numeroMatricula;
```

Tipos de variáveis Pseudocódigo	Tipos de variáveis Java	Valores compreendidos dentro do tipo	Exemplo de valores em Java.
Inteiro	byte	Números entre -128 e 127.	10
	short	Números entre -32768 e 32767.	13320
	Int	Números entre -2147483648 e 2147483647.	-170000000
	long	Números entre -9223372036854775808 e 9223372036854775807.	14000222999333
Real	float	Números reais entre -10^{38} até 10^{38} .	0.134
	double	Números reais entre -10^{308} até 10^{308} .	143.4938293019283
Caractere	char	Um único caractere (letra), entre apóstrofes. Exemplo: 'a'.	'd'
	String	Mais de um caractere, entre aspas. Exemplo: "Técnico em Informática".	"Informática"
Lógico	boolean	Verdadeiro ou Falso.	Falso

Imagem 23: Tabela de tipos de variáveis e suas representações em pseudocódigo e Java – no pseudocódigo temos as variáveis inteiro, real, caractere e lógico em Java o inteiro é representado pelos tipos byte, short, int, long, o real por float e double, o tipo caractere por char e String e o lógico como boolean, lembrando que cada tipo de variável numérica é representada pela sua faixa de valores.

Note que em Java, o único tipo de dado que se inicia com letra maiúscula é o String. Vale lembrar também que os números decimais são separados por ponto (.) ao invés de vírgula.

No caso de armazenamento de número de telefone, qual seria o tipo de dado a ser aplicado?

O número de telefone seria uma variável do tipo String, pois apesar de ser um número de telefone, não efetuamos nenhum tipo de cálculo com este número, sendo assim, não é necessário definir este tipo de dado como inteiro.

Após identificar o tipo da variável que você utilizará, basta declará-la em seu programa, seguindo o padrão para cada linguagem de programação, sendo:

PSEUDOCÓDIGO

declare

nome da variavel como tipodedado

JAVA

tipodedado nome da variavel;

A seguir, temos um exemplo de sua aplicação utilizando o pseudocódigo e o Java:

	Pseudocódigo	Java
Exemplo	declare nome como caractere idade como inteiro preco como real	String nome; int idade; double preco;

Imagem 24: Tabela de representação de declaração de variáveis em fluxograma, pseudocódigo e codificação em Java – deve-se seguir o padrão de cada linguagem, ambos fluxograma e pseudocódigo as variáveis são declaradas da mesma forma, nome da variável e o seu tipo, em Java primeiro declara-se o tipo de dado e depois o nome da variável.

Como nomear uma variável

Para garantir um rápido entendimento e continuidade do desenvolvimento do software, as variáveis devem ser nomeadas de forma objetiva, para esclarecer facilmente o programador sobre qual é a sua função.

Quando nomeamos as variáveis, é imprescindível também seguir algumas regras, que são:

1. As variáveis nunca podem conter um espaço em seu nome

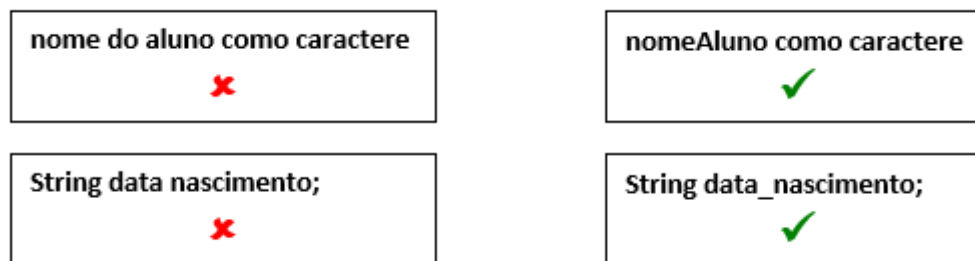


Imagem 25: Nomeação de variáveis – exemplos de nomes de variáveis corretas e incorretas, variáveis corretas sem espaço em branco, variáveis incorretas com espaço em branco. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “nome do aluno como caractere”, 2º quadro: string data de nascimento. Esses dois quadros, posicionados à esquerda, estão com um xis vermelho indicando erro. Do lado direito temos mais dois quadros. 3º quadro: nomeAluno como caractere e o 4º quadro: string data_nascimento. Esses dois últimos quadros estão com um ticket verde à esquerda indicando que estão corretos.

Caso você necessite de mais de uma palavra para definir o nome de uma variável, junte as palavras ou então separe-as apenas com um underline.

Remova as preposições do nome da variável, assim o nome dela ficará mais objetivo e fácil de entender.

Não inclua mais do que duas palavras em um nome de variável, seja sempre objetivo.

2. As variáveis nunca podem conter caracteres especiais em seu nome

Em uma linguagem de programação, os caracteres especiais são palavras reservadas que são utilizadas pela linguagem para trabalhar comandos, cálculos etc. Se você utilizar caracteres especiais em nome de variáveis, o computador não entenderá se ele deverá demarcar o espaço da variável na memória ou se iniciará algum procedimento especial do computador, por isto tentar colocar um nome destes resultará em erro de compilação⁹.

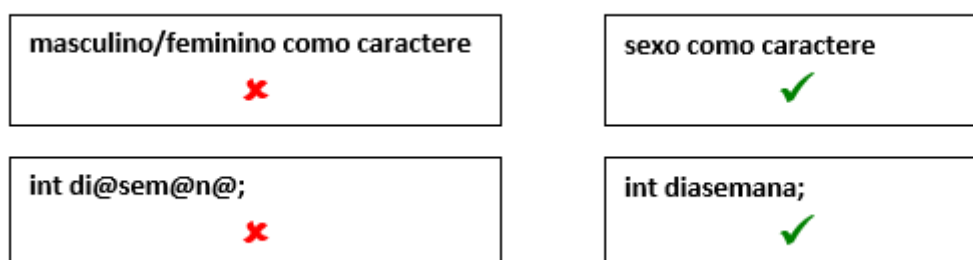


Imagem 26: Nomeação de variáveis – exemplos de variáveis corretas e incorretas, não é permitido o uso de caracteres especiais. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “masculino/feminino como caractere”, 2º quadro: int di@sem@n@. Esses dois quadros, posicionados à esquerda, estão com um xis vermelho indicando erro. Do lado direito temos mais dois quadros. 3º quadro: sexo como caractere e o 4º quadro: int diasemana. Esses dois últimos quadros estão com um ticket verde à esquerda indicando que estão corretos.

Entende-se por caracteres especiais os seguintes sinais: !, @, #, \$, %, \, /,], [, (,), {, }, e todos os caracteres não alfanuméricos.

3. Nomes de variáveis não podem receber acentuação

Como essa linguagem de programação é escrita em Inglês, em que não há acentuação nas palavras, não podemos utilizá-los em declarações de variáveis. O cê-cedilha (ç) apesar de ser um caractere, também entra nesta regra.

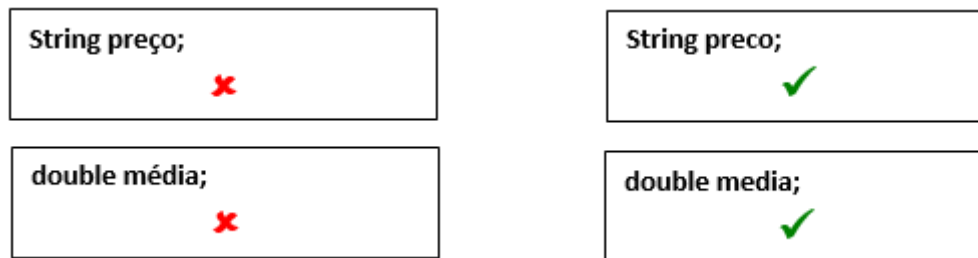


Imagem 27: Nomeação de variáveis – exemplos de variáveis corretas e incorretas, não é permitido o uso de acentuação nos nomes. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “String preço”, 2º quadro: Double média. Esses dois quadros, posicionados à esquerda, estão com um xis vermelho indicando erro. Do lado direito temos mais dois quadros. 3º quadro: String preco e o 4º quadro: Double media. Esses dois últimos quadros estão com um ticket verde à esquerda indicando que estão corretos.

4. Nomes de variáveis não podem ser iniciados por números

Quando a linguagem de programação encontra um número em uma codificação, logo ela entende que deverá ser feito um cálculo. Caso esse número venha junto com um caractere em seguida, o computador não identificará o caractere como sendo um número válido para efetuar um cálculo, assim gerando um erro de compilação.

Você poderá utilizar, normalmente, um número no nome da sua variável, desde que este número não seja o primeiro caractere de seu nome.

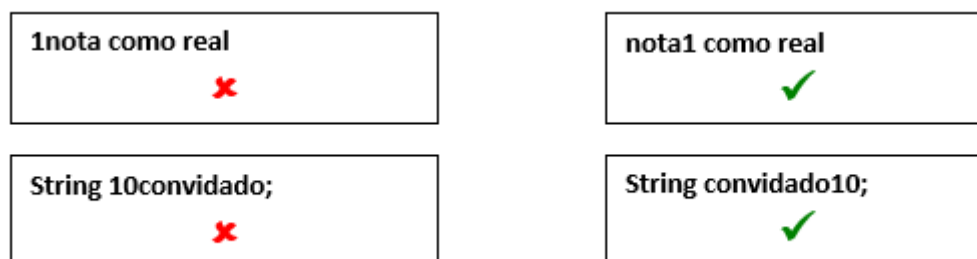


Imagem 28: Nomeação de variáveis- exemplos de variáveis corretas e incorretas, não é permitido iniciar o nome com um número. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “1nota como real”, 2º quadro: String 10convidado. Esses dois quadros, posicionados à esquerda, estão com um xis vermelho indicando erro. Do lado direito temos mais dois quadros. 3º quadro: nota1 como real e o 4º quadro: String convidado10. Esses dois últimos quadros estão com um ticket verde à esquerda indicando que estão corretos.

O comando Leia

O comando Leia é o comando responsável por receber dados inseridos pelo usuário. Em geral, estes dados são inseridos por meio do teclado, podendo ser numérico ou caractere dependendo do tipo de dados para que a variável - que receberá o valor - estiver configurada.

Como o computador não saberá qual será o valor que o usuário digitará, sempre teremos que utilizar uma variável para armazenar o valor obtido por meio do comando Leia. A sintaxe do comando Leia é:

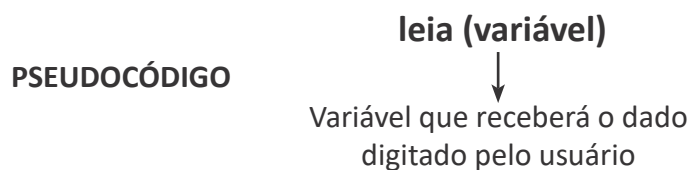


Imagem 29: Comando “leia” – recebe um valor digitado pelo usuário

Em Java, um programa inicial contém a exibição de mensagens no monitor, processamentos básicos de dados e utilização de variáveis com o objetivo de garantir a otimização do espaço de seu programa em disco e, conseqüentemente, o peso da sua aplicação.

Para que seja possível a utilização de recursos diferentes, é necessário realizar uma importação de uma biblioteca de classes para o seu Projeto.

Estas bibliotecas contêm as instruções necessárias para que o Java consiga trabalhar com novas funções conforme a necessidade do programador. Vale lembrar que não é recomendado que você faça uma importação de biblioteca em seu programa, caso não tenha intenção de utilizá-la, pois isto poderá acarretar perda de desempenho desnecessária em sua aplicação.

Para importar uma biblioteca, basta seguir o seguinte comando:

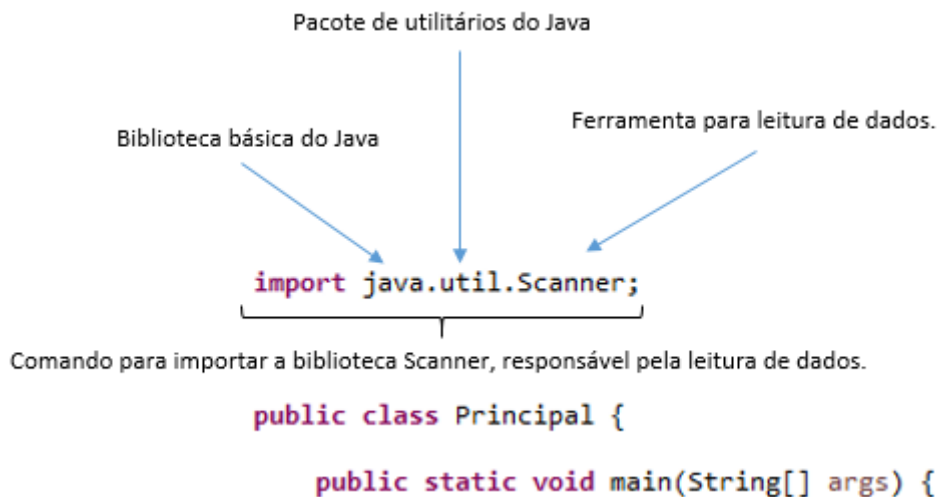


Imagem 30: comando para importação da biblioteca Java. A imagem explica cada comando de importação da biblioteca `import java.util.Scanner;` sendo: `import` – palavra chave para importar bibliotecas, `java` é a biblioteca básica do Java, `util` o pacote de utilitários do Java e `Scanner` a ferramenta para leitura de dados. Este é o comando para importar a biblioteca `Scanner`, responsável pela leitura de dados.

Note que o comando `import` deve ser inserido na primeira linha do seu código, antes mesmo de todos os códigos gerados automaticamente pela IDE Netbeans.

Para entender melhor o comando:

Imagine que você está trabalhando na construção de um objeto sobre uma mesa. Na mesa ficarão apenas as ferramentas mais utilizadas, como chave de fenda e alicate. Caso você necessite de uma chave inglesa, você deverá ir até a sua mala de ferramentas, no compartimento de chaves e trazer a chave inglesa para a sua mesa, não é mesmo?

Na prática, caso a ferramenta de que você precise não estiver disponível, provavelmente você a encontrará na sua mala de ferramentas (biblioteca Java), e dentro do compartimento de chaves (util).

Mas existe apenas a biblioteca Java para ser importada?



Imagem 31: Freepik – Caixa de Ferramentas

Não! O Java possui inúmeras bibliotecas que poderão ser importadas sempre que necessário. Além disto, você também poderá importar bibliotecas feitas por outras pessoas, como objetivo de poupar muito trabalho no desenvolvimento de uma nova ferramenta a partir da estaca zero.

Voltando a leitura de dados. Após a importação da ferramenta Scanner, precisamos “criá-la” dentro do nosso programa, utilizando o seguinte comando:

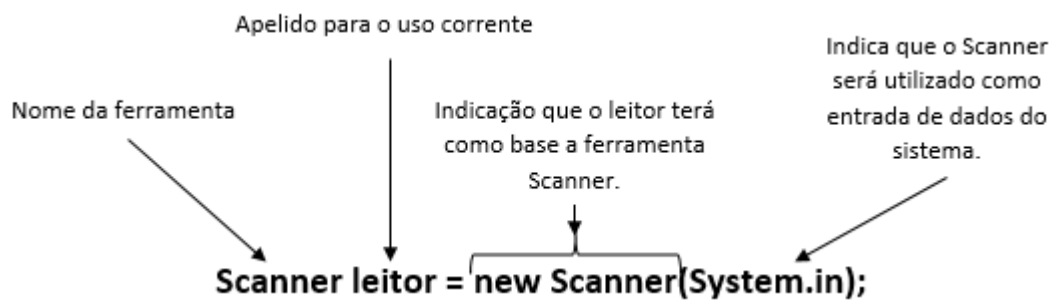


Imagem 32: sintaxe de criação do comando scanner: `Scanner leitor = new Scanner(System.in);` Sendo: Scanner o nome da ferramenta. Leitor é o apelido para o uso corrente. New Scanner é a indicação de que o leitor terá como base a ferramenta Scanner e System.in que indica que o Scanner será utilizado como entrada de dados do sistema.

O comando para criar o leitor dentro do nosso programa é chamado de instância. Na instância, a sua ferramenta importada cria vida, tornando-se funcional e utilizável na sua aplicação. A partir deste momento, o leitor será carregado na memória do computador junto com a sua aplicação.

Leitura de dados utilizando a ferramenta Scanner

Agora que já temos o nosso leitor, estamos prontos para ler uma entrada de dados feita pelo usuário por meio do teclado e armazená-la em uma variável. Para que esta leitura seja feita de forma adequada pelo Java, devemos adotar uma leitura específica para cada tipo de variável, conforme a tabela a seguir:

Tipo da variável que receberá o dado (Java)	Comando utilizado pelo leitor	Exemplo
byte	<code>leitor.nextByte();</code>	<code>byte numero;</code> <code>numero = leitor.nextByte();</code>
short	<code>leitor.nextShort();</code>	<code>short numero;</code> <code>numero = leitor.nextShort();</code>
int	<code>leitor.nextInt();</code>	<code>Int numero;</code> <code>numero = leitor.nextInt();</code>
long	<code>leitor.nextLong();</code>	<code>long numero;</code> <code>numero = leitor.nextLong();</code>
float	<code>leitor.nextFloat();</code>	<code>float numero;</code> <code>numero = leitor.nextFloat();</code>
double	<code>leitor.nextDouble();</code>	<code>double numero;</code> <code>numero = leitor.nextDouble();</code>
char	<code>leitor.next().charAt(0);</code>	<code>char letra;</code> <code>letra = leitor.next().charAt(0);</code>
String	<code>leitor.next();</code>	<code>String palavra;</code> <code>palavra = leitor.next();</code>
boolean	<code>leitor.nextBoolean();</code>	<code>Boolean teste;</code> <code>teste = leitor.nextBoolean();</code>

Imagem 33: Tabela de comandos utilizados pelo leitor – o leitor será associado a um tipo de variáveis na sua sintaxe, como byte, short, int, long entre outros, por exemplo, `leitor.nextByte();`.

Exemplo prático de um programa

Agora que já vimos individualmente as principais ferramentas de uma linguagem de programação, chegou a hora de praticarmos, produzindo um programa completo. Para tanto, vamos criar um programa que calcule a soma de dois números digitados pelo usuário:

Passo 1: Definir a sequência lógica do programa.

Por ser o nosso primeiro programa, antes de construir o fluxograma, precisamos identificar a entrada, o processamento e a saída dos dados dentro do nosso programa:

Entrada: O programa deverá solicitar que o usuário digite dois números, do tipo numérico.

Processamento: O programa calculará a soma destes números.

Saída: O programa exibirá a soma destes números.

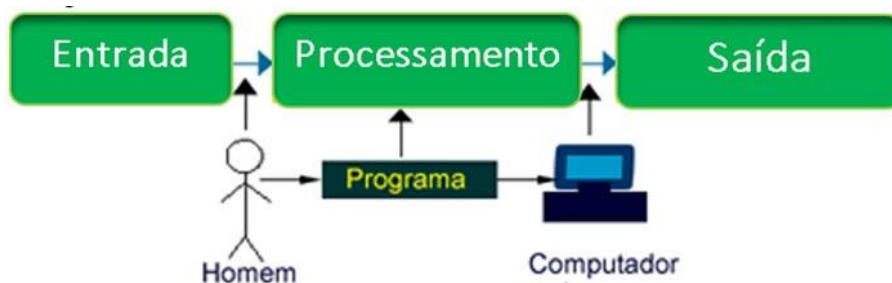


Imagem 34: A imagem mostra três quadrados verdes escrito Entrada, Processamento e Saída de Dados. Na seta que vai da entrada para o processamento, temos a representação de um usuário. Um rótulo escrito Programa que aponta para o processamento e uma seta com um computador aponta para a saída de dados.

Passo 2: Definir quais serão as variáveis necessárias para o programa.

Seguindo a definição de variáveis, precisamos identificar quais dados não serão fixos no nosso programa. Verificando as informações construídas no passo 1, nota-se que os dois números que o usuário digitar e a soma deles não são fixos, podendo ser modificado cada vez que o usuário utilizar o programa.

Logo, as variáveis serão: numero1 como inteiro numero2 como inteiro Soma como inteiro

Passo 3: Construir o Fluxograma.

Agora que já conhecemos as variáveis e a sequência lógica do nosso programa, podemos construir o fluxograma conforme a simbologia apresentada anteriormente:

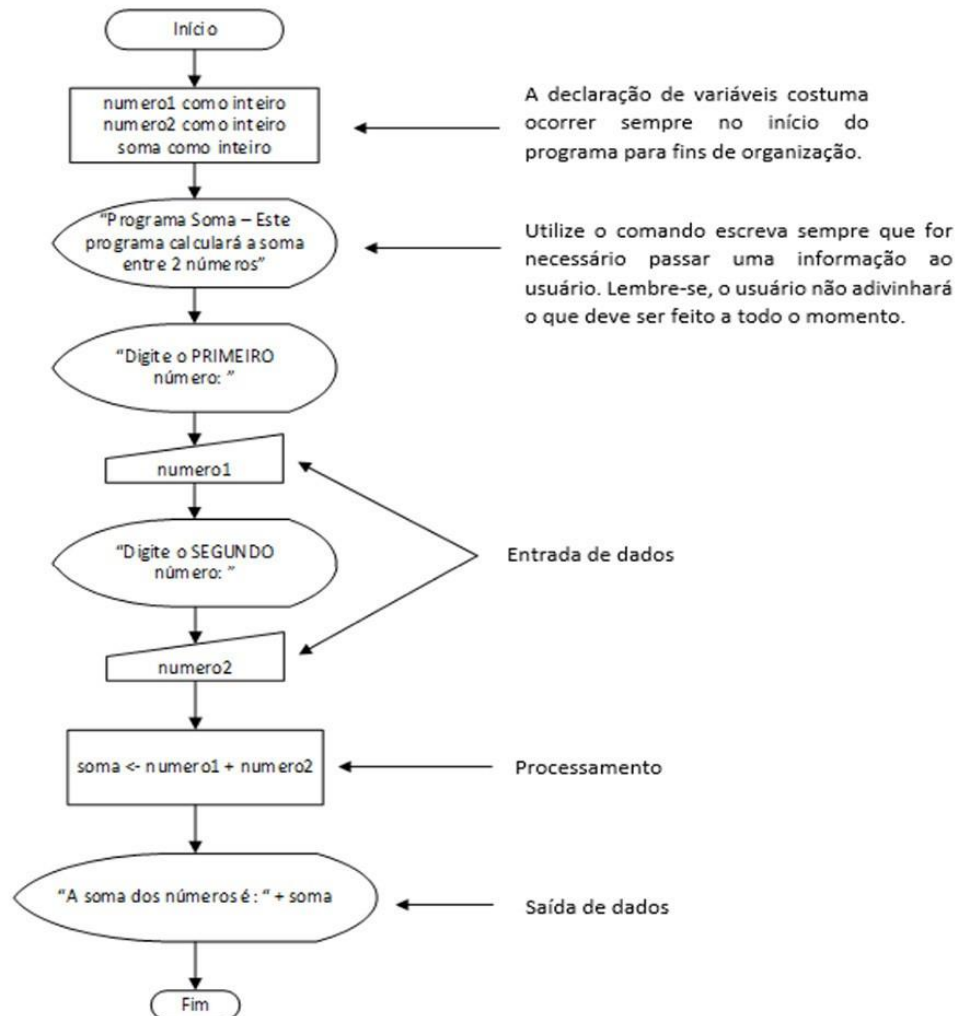


Imagem 35: GEEaD: Representação de um Fluxograma – o exemplo usado é para ler dois números, calcular a soma destes e apresentar o resultado.

Sempre procure identificar a entrada, o processamento e a saída dos seus dados em qualquer fluxograma, assim a chance de conter erros no fluxograma cairá consideravelmente.

Passo 4: Construir o Pseudocódigo

Após construir o fluxograma, ficará muito fácil criar o Pseudocódigo, basta “traduzir” a simbologia do Fluxograma para o Pseudocódigo:

Declare

numero1 como inteiro

numero2 como inteiro

soma como inteiro

Início

Escreva("Programa Soma – Este programa calculará a soma entre 2 números")

Escreva("digite o PRIMEIRO número")

Leia(numero1)

Escreva("digite o SEGUNDO número")

Leia(numero2)

soma <- numero1 + numero2

Escreva("A soma dos números é: " + soma)

Fim

Entrada

Processamento

Saída

Imagem 36: Nesse pseudocódigo existe uma caixa à direita indicando que os comandos "Leia número 1 e leia número 2 representam a entrada de dados, soma = numero1 + numero2 representa o processamento dos dados e o comando Escreva("A soma dos números é: " + soma) representa a saída de dados.

Passo 5: Construir o programa em Java.

Assim como fizemos com o Pseudocódigo, basta aplicar as regras básicas da linguagem Java.

Em primeiro lugar, criaremos um projeto para este exemplo, da mesma forma que vimos anteriormente.

A seguir, aplicamos o Pseudocódigo, adaptando-o para a linguagem Java. Não podemos esquecer de importar a biblioteca responsável pela leitura de dados:


```


import java.util.Scanner;

public class SomaValores {
    public static void main(String args[]) {
        /* O código desenvolvido por você deverá estar aqui.
        As chaves marcam o início do código.
        Diferente do pseudocódigo, as variáveis em java podem ser
        declaradas diretamente no corpo do programa, mas a
        recomendação continua: devemos declarar as avariáveis logo
        no início do código fonte.
        */
        // declaração das variáveis
        int numero1;
        int numero2;
        int soma;
        //para facilitar o entendimento, também habilitamos o leitor no
        início do código
        Scanner leitor = new Scanner(System.in);
        // início do programa
        System.out.println("Programa Soma - Este programa calculará a
        soma entre dois números");
        System.out.println("Digite o PRIMEIRO valor");
        //leitura do primeiro valor
        numero1 = leitor.nextInt();
        System.out.println("Digite o SEGUNDO valor");
        //leitura do segundo valor
        numero2 = leitor.nextInt();
        //processamento
        soma = numero1 + numero2;
        //saída de dados
        System.out.println("O resultado da soma é" + soma);
    }
}

```

Imagem 37: Codificação em Java da Classe SomaValores - o exemplo de soma de dois valores foi representado usando código da linguagem Java.

Algumas considerações:

-  **1.** Você percebeu que há explicações precedidas por duas barras // ou /* ? Estes sinais indicam comentários do Java, com eles você poderá escrever qualquer mensagem para o programador ou então anotar informações importantes sobre o seu código. Estes códigos não serão processados pelo computador.

Os comandos para iniciar um comentário em Java são:

// comentário de uma única linha

/*Comentário de múltiplas linhas */

A codificação indicada como comentário ficará por padrão na cor verde.

- 2. O Java é uma linguagem Case Sensitive, ou seja, diferencia as letras maiúsculas de minúsculas. Para o Java a letra “a” minúscula é diferente da letra “A” maiúscula. Por este motivo, os comandos da linguagem devem ser reproduzidos, fielmente, de acordo com a apresentação, caso contrário o comando não será reconhecido. É o caso do `System.out.println()`, onde o S deve ser sempre digitado em letra maiúscula. Essa regra também é válida para a declaração e utilização de variáveis.
- 3. Todo o comando feito em Java deve ser finalizado por um ponto e vírgula (;). É com o ponto e vírgula que o Java entende que o comando foi finalizado, executando-o e preparando-se para receber o próximo comando. Caso você esqueça do ponto e vírgula (que por sinal é o erro mais comum), o seu programa não funcionará.
- 4. Caso o seu texto digitado com auxílio do Netbeans fique sublinhado em vermelho, significa que ele está com erro. Caso isto ocorra, primeiro você deve sempre o corrigir, caso contrário seu programa não funcionará. Geralmente estes erros são causados por “erro de sintaxe”, em outras palavras, o comando foi digitado incorretamente.
- 5. Comandos sublinhados em amarelo não são erros. Geralmente são recomendações ou aviso de variáveis declaradas, mas não utilizadas.
- 6. Se você passar o mouse em cima de palavras sublinhadas, a IdE apontará as recomendações necessárias para corrigir o problema. Mas atenção: nem sempre a IdE acertará o palpite, lembre-se que o computador não é perfeito, portanto o ideal é sempre rever o código e entender o que está errado.

Passo 6: Executando o programa

A última etapa do nosso programa é executá-lo e testá-lo para ver se tudo ocorreu bem. Para testá-lo, basta clicar no nome da sua classe com o botão direito (em Project Explorer) e selecionar a opção Executar Arquivo, ou utilize as teclas de atalho Shift + F6.

Abaixo do seu código fonte, você verá uma janela chamada Console, onde você poderá inserir os dados para a utilização de seu programa:

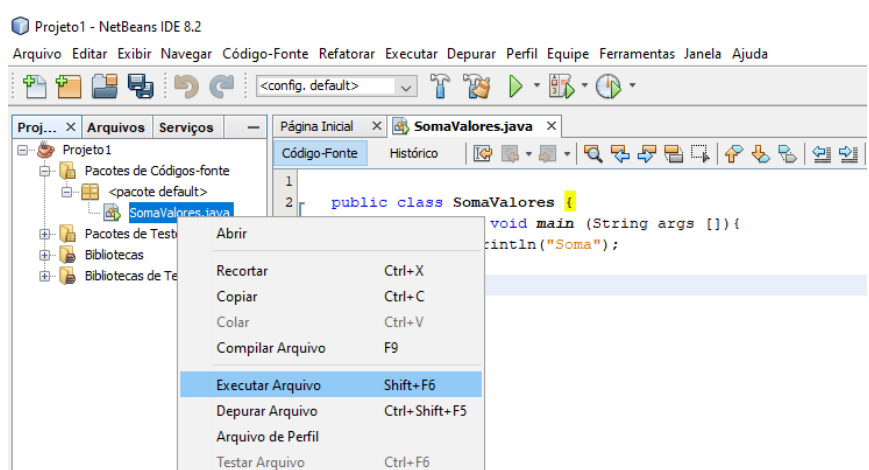


Imagem 38: Seleção da opção “Run As” de uma aplicação em Java- para executar a aplicação é necessário usar esta opção.

```

Programa Soma - Este programa calculará a soma entre dois números
Digite o PRIMEIRO valor
3
Digite o SEGUNDO valor
2
O resultado da soma é 5

```

Caso a janela console não apareça, você pode abri-la acessando o menu Window > Show View > Console



Vale lembrar que caso tenha erros em seu código fonte, a IDE (Netbeans) retornará uma mensagem de erro, pois não é possível executar o seu código enquanto os erros (sublinhados em vermelho) não sejam corrigidos.

Agora tente resolver as questões a seguir e depois confira se você acertou!

1. Considerando o seu aprendizado sobre o comando, escreva os operadores aritméticos, desenvolva o fluxograma, as codificações em Java e o pseudocódigo para exibir as mensagens que satisfaçam as seguintes situações:



Mas lembre-se... as regras de precedência matemática funcionam da mesma forma na programação, com exceção dos parênteses (), colchetes [] e chaves { }, que são todos representados por parênteses.

a) Multiplicar por 2 o resultado de $(7 + 3) + 3$.

Fluxograma	Pseudocódigo	Java

b) dividir 168 por 46.

Fluxograma	Pseudocódigo	Java

c) Multiplicar o resto da divisão entre 7 e 4 por 2.

Fluxograma	Pseudocódigo	Java

d) Juntar as palavras “Técnico” com “Módulo” com “1”.

Fluxograma	Pseudocódigo	Java

2. Crie um Fluxograma, um Pseudocódigo e uma codificação Java que satisfaça às seguintes situações:

a. Leia o seu nome e sobrenome, e exiba-os na sua forma inversa (sobrenome, nome).

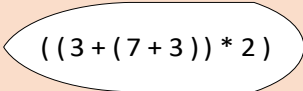
b. Leia dois números inteiros, realize a multiplicação do primeiro pelo segundo e exiba o resultado final.

Agora confira se você acertou:


Seguindo as regras de precedência matemática, de operadores aritméticos e do comando escreva, temos:

1.

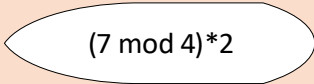
a) Multiplicar por 2 o resultado de $(7 + 3) + 3$.

Fluxograma	Pseudocódigo	Java
	Escreva $((3 + (7 + 3)) * 2)$	<code>System.out.println((3 + (7 + 3)) * 2);</code>

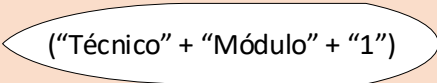
b) dividir 168 por 46

Fluxograma	Pseudocódigo	Java
	escreva $(168 / 46)$	<code>System.out.println(168 / 46);</code>

c) Multiplicar o resto da divisão entre 7 e 4 por 2.

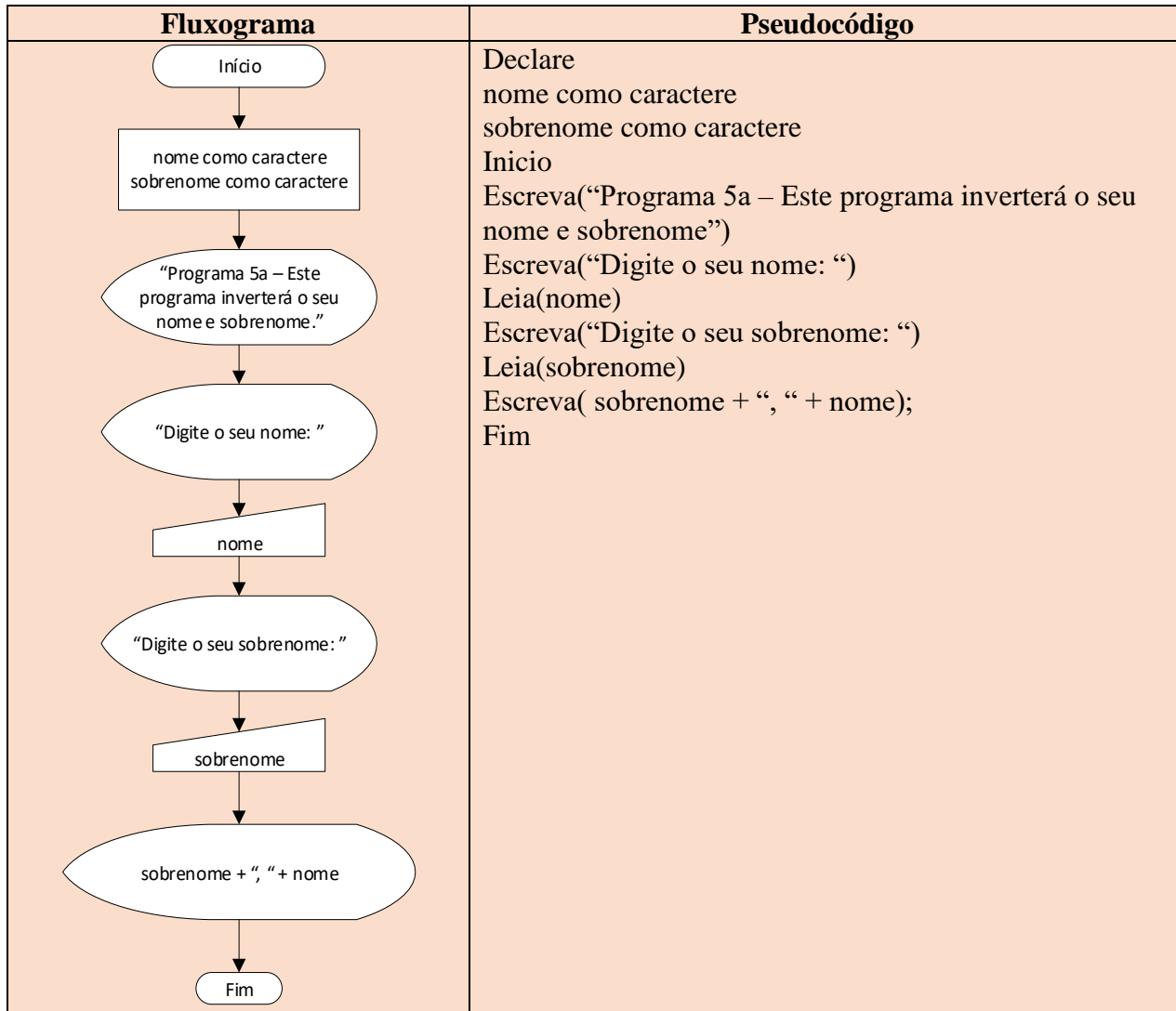
Fluxograma	Pseudocódigo	Java
	escreva $(7 \bmod 4) * 2$	<code>System.out.println((7 % 4) * 2);</code>

d) Juntar as palavras “Técnico” com “Módulo” com “1”.

Fluxograma	Pseudocódigo	Java
	escreva (“Técnico” + “Módulo” + “1”)	<code>System.out.println(“Técnico” + “Módulo” + “1”);</code>

2.

a. Leia o seu nome e sobrenome, e exiba-os na sua forma inversa (sobrenome, nome).



Codificação em Java:

```

import java.util.Scanner;

public class Programa5a {

    public static void main(String args[]) {
        // declaração de variáveis e inicialização do leitores
        String nome;
        String sobrenome;
        Scanner leitor = new Scanner(System.in);
        // início do programa
        System.out.println("Programa 5a – Este programa inverterá o seu nome e sobrenome");
        System.out.println("Digite o seu nome");
        nome = leitor.next();
        System.out.println("Digite o seu sobrenome");
        sobrenome = leitor.next();
        //saída de dados
        System.out.println(sobrenome + ", " + nome);
        //fim do programa.
    }
}

```

b. Leia dois números inteiros, realize a multiplicação e exiba o resultado.

Pseudocódigo	Fluxograma
<p>Programa ex6</p> <p>Declare num1 como inteiro num2 como inteiro multi como inteiro</p> <p>Início Escreva ("Digite primeiro número") Leia (num1) Escreva ("Digite segundo número") Leia (num2) $multi \leftarrow num1 * num2$ Escreva ("A multiplicação é") Escreva (multi)</p> <p>Fim.</p>	<pre> graph TD Inicio([Início]) --> Declara[num 1 como inteiro
num 2 como inteiro
soma como inteiro] Declara --> Digite1{{Digite o primeiro
"número"}} Digite1 --> Leia1[/num 1/] Leia1 --> Digite2{{Digite o segundo
"número"}} Digite2 --> Leia2[/num 2/] Leia2 --> Calcula[$multi \leftarrow num 1 * num 2$] Calcula --> Mostra1{{"A multiplicação é"}} Mostra1 --> Mostra2[/multi/] Mostra2 --> Fim([fim]) </pre>

Codificação em Java:

```
1- import java.util.Scanner;
2-
3- public class Programa_ex6 {
4-     public static void main (String args[]){
5-         //Declaração das variáveis
6-         int numero1;
7-         int numero2;
8-         int multi;
9-         Scanner leitor = new Scanner(System.in);
10-        // início do programa
11-        System.out.println("Programa 6a - Soma de valores");
12-        System.out.println("Digite o PRIMEIRO valor");
13-        //leitura do primeiro valor
14-        numero1 = leitor.nextInt();
15-        System.out.println("Digite o SEGUNDO valor");
16-        //leitura do primeiro valor
17-        numero2 = leitor.nextInt();
18-        //Processamento
19-        multi=numero1*numero2;
20-
21-        //saída
22-        System.out.println("Resultado" + multi);
23-    }
24- }
25- }
```