

Agenda 03: A lógica aplicada em Java

Conceitos trabalhados:

Sumário

- Momento de Reflexão
- Por que Aprender?
- Para Começar o Assunto
- Mergulhando no Tema
- Ampliando Horizontes
- Resumindo o Estudo

Momento de Reflexão

Olá, estudante ;)

Boas-vindas à terceira agenda do Módulo 1, que aprofundará mais aspectos da linguagem de programação, **tratando da linguagem Java**. Para começar, vamos retomar uma situação que você já viu na agenda 1:



Você já sabe que os computadores não pensam como nós, nem entendem nossa língua. Por isso, para que possam executar uma ação, é preciso explicar tudo nos mínimos detalhes e na **língua deles**. O problema é que a linguagem dos computadores é uma grande sequência de **números binários**, ou seja, zeros e uns. Por exemplo:

```
101110111011011001110110011000101 0
```

Isto traz muita dificuldade para nós, seres humanos. Já imaginou ler ou escrever instruções longas? Você precisaria decorar centenas de códigos binários! Será que você encararia um curso como o nosso, se precisasse programar desse modo?

Mas não se assuste! Para facilitar a comunicação entre homem e máquina, foram desenvolvidas as **linguagens de programação** que dispõem de um **compilador** que interpreta os comandos da linguagem e **transforma em binários**, ou seja, em instruções que serão entendidas pelo processador do computador. Uma das linguagens de programação é o **Java**, tema desta agenda!

Por que Aprender?

Você se lembra de como aprendeu o seu **idioma** nativo? Primeiro você se apropriou das palavras essenciais para satisfazer suas necessidades e desejos, depois aprendeu as letras e como fazer combinações entre elas. Quando viu, já estava lendo e escrevendo. Não foi assim?

Aprender a programar não é muito diferente disso! O primeiro passo é aprender a lógica de programação. Sabe por quê? Porque ela consiste em aprender **técnicas para escrever códigos** que possam ser interpretados por computadores. **A lógica não muda**, independente da linguagem que você escolher.

Em uma linguagem de programação, também é preciso aprender seus **comandos básicos**, que são derivados da lógica de programação. Sendo assim, se você compreender qual o momento certo para exibir uma mensagem na tela do usuário, pode executar esse processo com o auxílio de **qualquer linguagem**, basta “traduzir” o comando.

Para os nossos estudos, utilizaremos a linguagem de programação **Java**, por ser gratuita e uma das mais utilizadas no mercado, como você verá a seguir.

Para Começar o Assunto

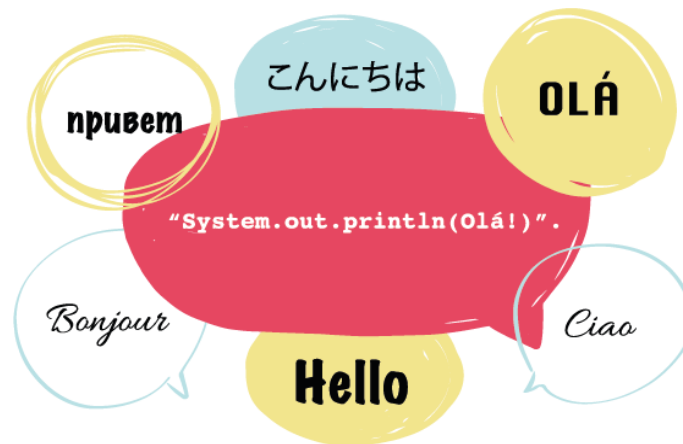
Já pensou em quantas vezes por dia as pessoas acessam sites diversos para estudar, fazer compras, ouvir músicas, ver vídeos e notícias? Você acredita que todas essas aplicações, inclusive aquelas envolvidas no processo de atualização das notícias que aparecem a cada minuto na tela do seu computador, podem ser desenvolvidas a partir do Java?



A partir de agora vamos aplicar a lógica de programação na linguagem Java, **fazendo um paralelo entre fluxogramas e pseudocódigos com a linguagem Java.** Preparado?

Mergulhando no Tema

Linguagem de programação Java



Você já viu que para escrevermos um programa desenvolvido em Pseudocódigo, de forma que o computador possa **compilá-lo** e executá-lo, precisamos utilizar uma **linguagem de programação**, que nada mais é do que o "**idioma**" necessário para conversar com o computador. No desenvolvimento de nossos estudos em lógica de programação utilizaremos a linguagem de programação Java.

O Java atualmente é uma das linguagens de programação mais utilizadas no mercado de trabalho, sendo capaz de fornecer uma portabilidade muito grande. Além disso, pode ser utilizada para desenvolver páginas da Internet, e aplicativos para celulares que utilizam o Sistema Operacional Android.

O que é Java?



Java é uma **linguagem de programação orientada a objetos** que foi desenvolvida nos anos de 1990 pela Sun Microsystems, projetada para ser pequena, simples e portátil a todas as plataformas e Sistemas Operacionais.

É utilizada para desenvolver aplicativos corporativos, páginas web com conteúdo dinâmico e interativo, aprimorar a funcionalidade de servidores www e está cada vez mais sendo utilizada para desenvolvimento de aplicativos móveis para telefones celulares, pagers e PDAs.

Java é uma Linguagem de Programação Orientada a Objetos, porém para que você compreenda melhor a Lógica utilizando essa Linguagem, iremos usar o Console, uma forma estruturada, para exercitar os comandos.

Você sabia que o Java não é a única linguagem de programação que funciona em todos os principais Sistemas Operacionais do mercado? Pesquise outras linguagens existentes no mercado e reflita sobre a eficiência delas em relação à linguagem Java.



Conhecendo as ferramentas Java

Para iniciar o estudo, primeiramente é necessário conhecer as diferentes ferramentas para desenvolvimento que o Java oferece: JRE, JVM, JSE, JEE, JME, JDK... parece uma sopa de letrinha, não é mesmo?

Em seguida, **com base na aplicação que pretende desenvolver**, deve selecionar as tecnologias e ferramentas necessárias para este fim.

É mais ou menos assim: imagine que você precisa fazer um brigadeiro! Com a receita em mãos, você vai até o supermercado, compra os ingredientes: leite condensado, chocolate em pó, manteiga, chocolate granulado e ainda providencia as ferramentas necessárias para a confecção do doce: panela e colher de pau.



A mesma coisa acontece com o desenvolvimento de um aplicativo em Java! Conhecendo o tipo de aplicação que deseja desenvolver, você seleciona as ferramentas necessárias para tal finalidade.

Como em breve seremos desenvolvedores e não usuários, precisaremos instalar em nosso computador o **kit para desenvolvimento de programas** feitos em Java e a **Máquina Virtual (JVM)**.

Java Virtual Machine – JVM

O que faz com que a portabilidade da linguagem Java seja eficiente é uma aplicação responsável por executar programas desenvolvidos na linguagem. Sua função é **simular um computador** permitindo a execução do código fonte, por isso, recebe o nome de Máquina Virtual. Na prática, basta instalar em seu computador a JVM desenvolvida para o Sistema Operacional correspondente ao que estiver instalado em seu computador e você estará pronto para executar programas desenvolvidos em Java.

O Java Development Kit (JDK) e a Integrated Development Environment(IDE) Eclipse



O **kit para desenvolvimento de programas (JDK)** feitos em Java inclui o **compilador** da linguagem de programação. Sem ele não é possível finalizar um programa desenvolvido em Java, mesmo que você escreva o código fonte completo.

Apesar de já ser possível criar programas apenas com o JDK, utilizaremos uma **interface de desenvolvimento integrada (IDE)** para nos auxiliar na escrita, compilação e testes dos nossos programas. O Java tem como principais IDEs de desenvolvimento o **NetBeans** e o **Eclipse**. Independente da IDE escolhida, os comandos sempre serão os mesmos, logo, se você aprende a programar em Java, conseguirá utilizar qualquer uma das IDEs sem maiores problemas.

Durante o desenvolvimento das atividades utilizaremos a **IDE Eclipse**, porém você pode utilizar qualquer outra, desde que siga fielmente as estruturas listadas neste material.

Colocando a mão na massa

Para começar seu primeiro software, é necessário baixar e instalar as ferramentas que você acabou de conhecer.

Neste material, utilizamos a distribuição do Java SE (Standard Edition), que é voltada para o Desenvolvimento de Sistemas Desktop. Existem também as Distribuições Java ME (Micro Edition) que é voltada para dispositivos de pequeno porte e Java EE (Enterprise Edition) que é direcionado a aplicações corporativas e Web.

Você pode efetuar o download e instalação do JDK e do Eclipse nestes links:

<https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html>

<https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-ee-developers>

Primeiro, faça a instalação do JDK. Depois, baixe a IDE Eclipse.

- Para fazer o download do JDK, você precisará fazer um pequeno cadastro no site da Oracle.
- Se você não souber qual a versão do sistema operacional do seu computador, as orientações [desse link](#) podem te ajudar.

Criando um projeto utilizando a IDE Eclipse

Na estrutura de organização do Eclipse Oxygen, cada programa desenvolvido é caracterizado como um Projeto. E cada código fonte, será tratado como uma Classe.

Assista aos vídeos abaixo, sobre como realizar a configuração do seu ambiente de trabalho para o uso do JAVA. Veja como configurar as variáveis de ambiente PATH e JAVA_HOME:

A Lógica aplicada em Java #01 Introdução (Ag03)



A Lógica aplicada em Java #02 Primeiro Progra...



No vídeo tutorial abaixo, você terá informações detalhadas e precisas sobre como iniciar seu projeto:

Novotec | Criando um projeto com a IDE Eclipse



Como utilizaremos a **Linguagem de Programação Estruturada Java**, não entraremos fundo no significado de Projeto, Pacote e Classe, pois essas definições são utilizadas em programas **Orientados a Objetos**.

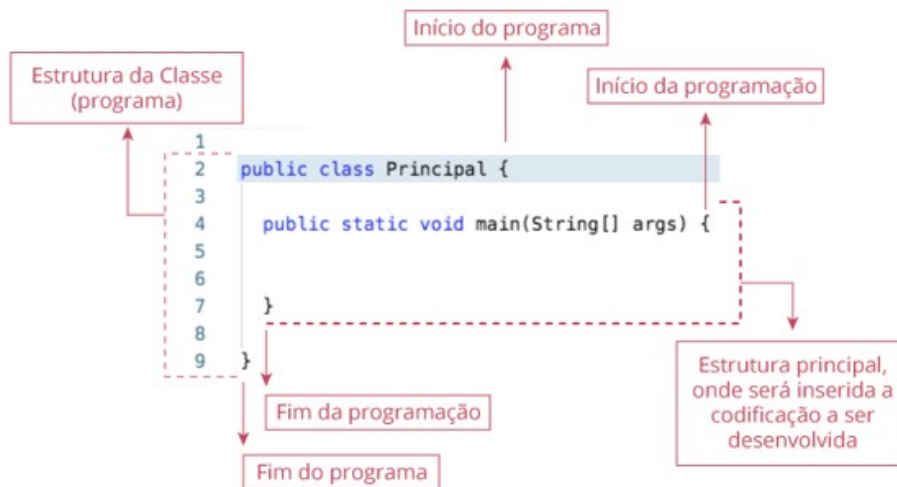
Conhecendo a estrutura e os comandos de programação em JAVA

Até este momento você teve contato com a programação utilizando instruções que serviram para que você construísse os primeiros modelos de lógica de programação, distribuídos no formato de algoritmos. A partir de agora, você já tem o JDK instalado e configurado em seu computador, além da IDE de desenvolvimento ECLIPSE.

Vamos nos aprofundar na estrutura de programação em JAVA, seus comandos, suas referências de programação estruturada e como iremos construir os primeiros modelos de programas que poderão ser executados em seu computador.

1. 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:



Você pode ver na imagem que o programa em si está dentro de uma classe. Essa estrutura é necessária para permitir que no futuro o seu programa em Java se torne um programa Orientado 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 apresentados a você sempre serão apresentados em Java.

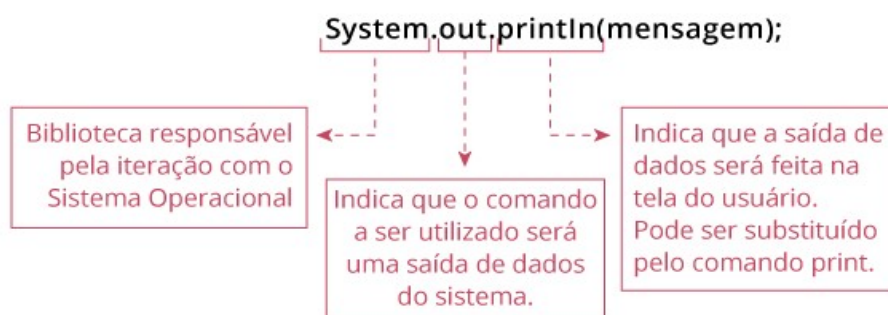
2. O comando ESCRIVA

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 esse comando para mostrar tudo o que você deseja que o usuário veja.

Em Java, esse comando é escrito na forma abaixo. 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:



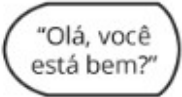
Agora, reflita: você conhece o comando print e println? Qual dos dois comandos é o mais eficiente? Existe alguma situação específica para a utilização de cada um dos comandos?

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, antes de iniciar

a 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**.

Por exemplo: desejamos que o usuário veja a seguinte mensagem: Olá, você está bem?

Fluxograma	Pseudocódigo	Java
	escreva "Olá, você está bem?"	<pre>System.out.println("Olá, você está bem?");</pre>

3. Declarando um 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, indicado em **laranja**:

int numeroMatricula;

Agora, conheça os tipos de variáveis e suas representações em pseudocódigo e em java:

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.	0.134
	double	Números reais entre -10 ³⁰⁸ até 10.	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

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.

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

Pseudocódigo	JAVA
declare nome da variavel como tipodado	declare tipodado nome da variavel





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

	Fluxograma	Pseudocódigo	Java
Exemplo	nota 1 como real nota 2 como real nota 3 como real media como real	declare nome como caractere idade como inteiro preco como real	String nome; Int idade; Double preco;

3.1. 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:

a) As variáveis nunca podem conter um espaço em seu nome





nome do aluno como caractere 	nomeAluno como caractere 
String data nascimento; 	String data_nascimento; 

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 (data nascimento > datanascimento ou data_nascimento).

Remova as preposições do nome da variável, assim o nome dela ficará mais objetivo e fácil de entender (nome do aluno > nomeAluno).

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


b) As variáveis nunca podem conter caracteres especiais em seu nome

masculino/feminino como caractere 	sexo como caractere 
int di@sem@n@; 	int diasemana; 

Em uma linguagem de programação, os caracteres especiais são palavras reservadas utilizadas pela linguagem para trabalhar comandos específicos, 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 isso, tentar colocar um nome desses resultará em erro de compilação.





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

c) Nomes de variáveis não podem receber acentuação

string preço; 	string preco; 
double média; 	double media; 

Como as linguagens de programação são todas escritas em inglês, na qual não há acentuação nas palavras, não podemos utilizá-los em declarações de variáveis. O cê-cedilha (Ç) apesar de ser um caractere, também entra nessa regra.

d) Nomes de variáveis não podem ser iniciados por números

1nota como real 	nota1 como real 
String 10convidado; 	String convidado10; 

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, gerando, assim, um erro de compilação.

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

4. O comando LEIA

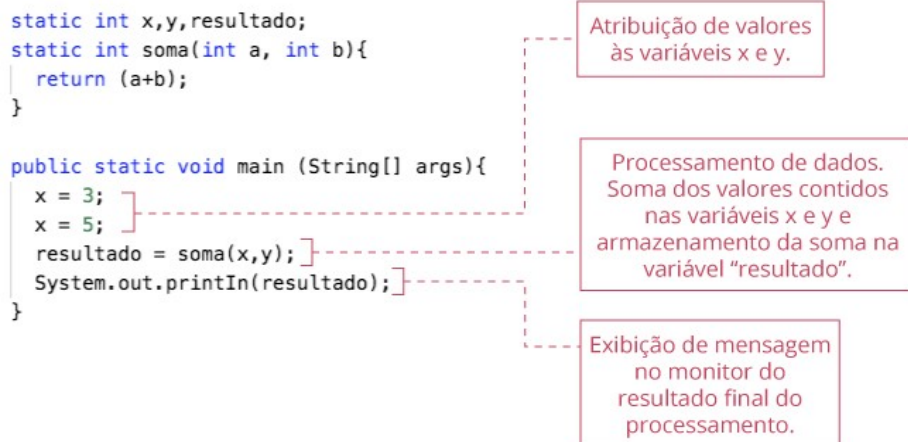
O comando Leia é o responsável por receber dados inseridos pelo usuário. Em geral, esses dados são inseridos por meio do teclado, podendo ser numérico ou caractere, dependendo do tipo de dados para que a variável - que receberá o valor - estiver configurada.

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, em pseudocódigo, é:

leia(variável)

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.

Exemplo:

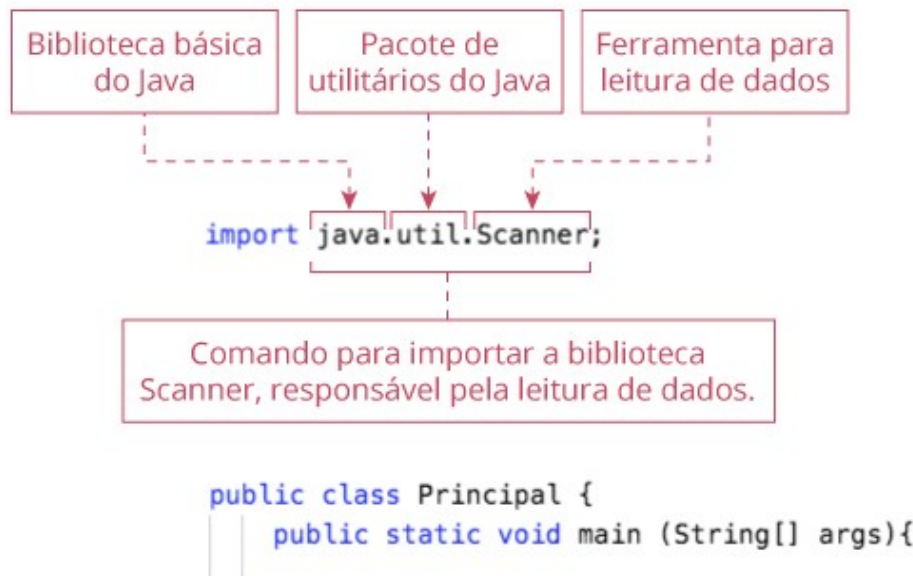


5. O comando Importar

Para que seja possível a utilização de recursos diferentes, é necessário realizar uma importação de uma biblioteca de classes para o seu projeto. Essas 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: isso poderá acarretar perda de desempenho desnecessária em sua aplicação.

Para importar uma biblioteca, basta seguir o seguinte comando:



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

5.1. Entendendo melhor a importação de bibliotecas

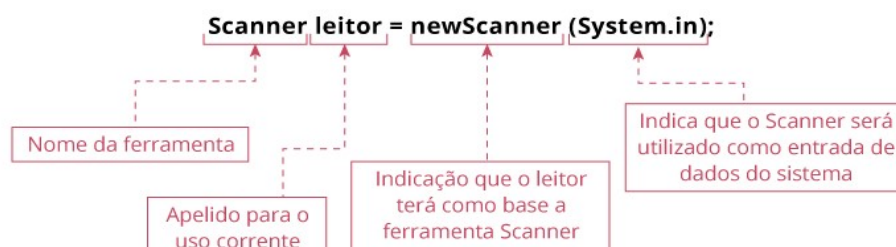
Imagine que você está trabalhando na construção de um objeto sobre uma mesa. Na mesa ficarão apenas as ferramentas mais utilizadas, como chave de fenda e alicate. Caso você necessite de uma chave inglesa, você deverá ir até a sua mala de ferramentas, no compartimento de chaves, e trazer a chave inglesa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

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

5.2. Mas existe apenas a Biblioteca JAVA para ser importada?

Não! O Java possui inúmeras bibliotecas que poderão ser importadas sempre que necessário. Além disso, 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:



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 desse

momento, o leitor será carregado na memória do computador junto com a sua aplicação.

6. Leitura de dados utilizando a ferramenta Scanner

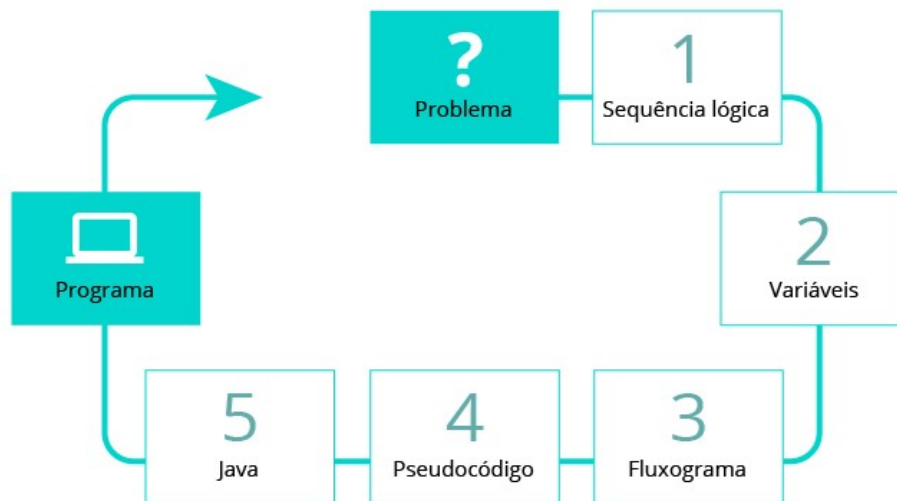
Agora que já temos o nosso leitor, estamos prontos para ler uma entrada de dados feita pelo usuário por meio do teclado e armazená-la em uma variável. Para que essa leitura seja feita de forma adequada pelo Java, devemos adotar uma leitura específica para cada tipo de variável, conforme a tabela a seguir:

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

7. 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.



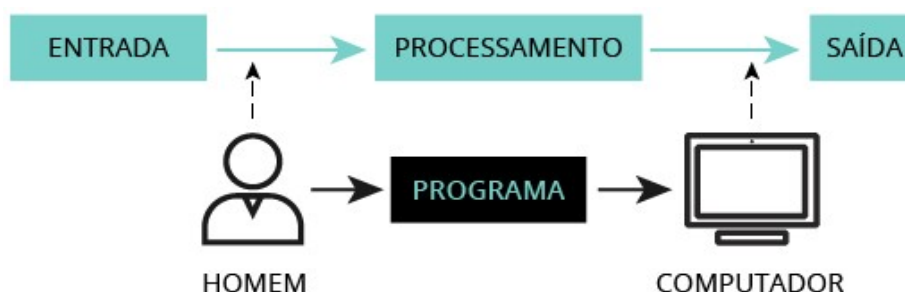
Passo 1: Definir a sequência lógica do programa

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 desses números.

Saída: o programa exibirá a soma desses números.



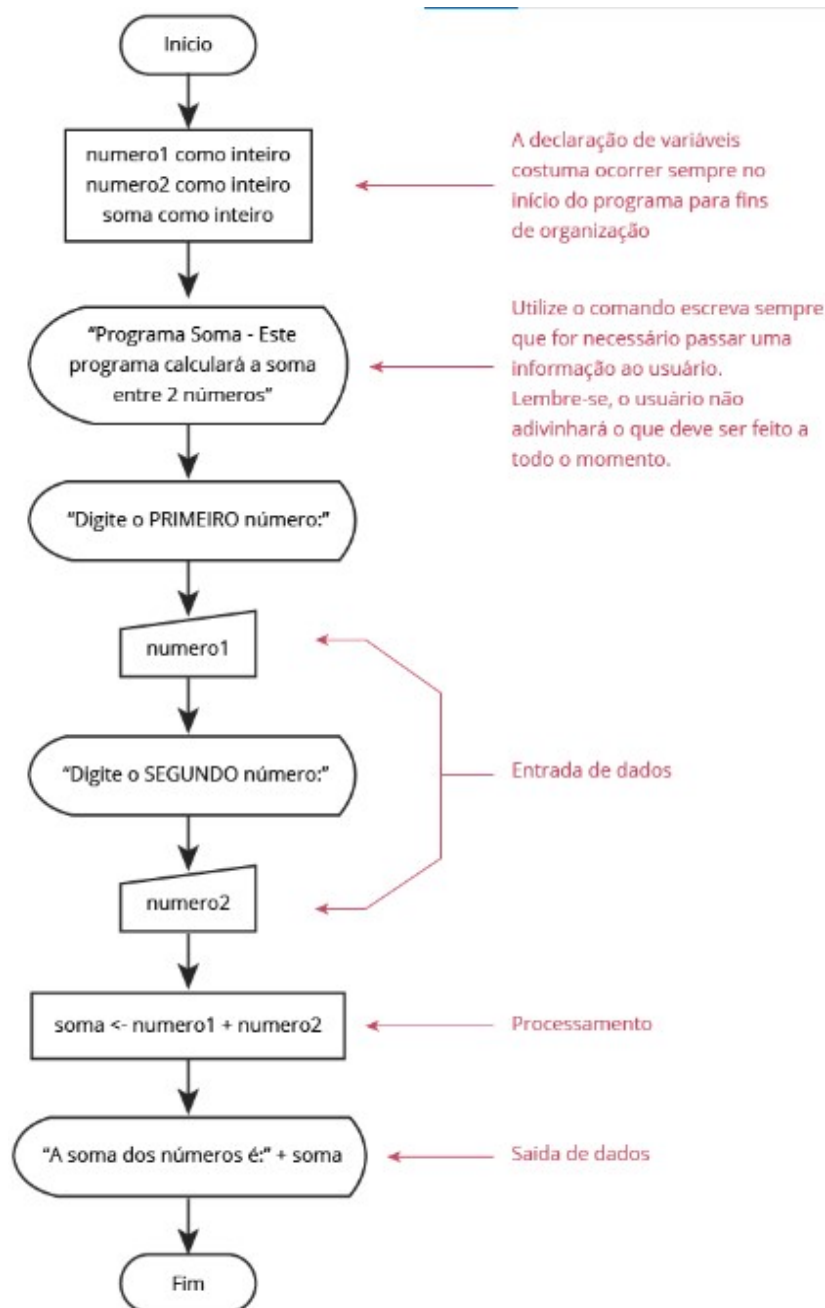
Passo 2: Definir quais serão as variáveis necessárias para o programa.

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 modificados cada vez que o usuário utilizar o programa.

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

Passo 3: Construir o Fluxograma

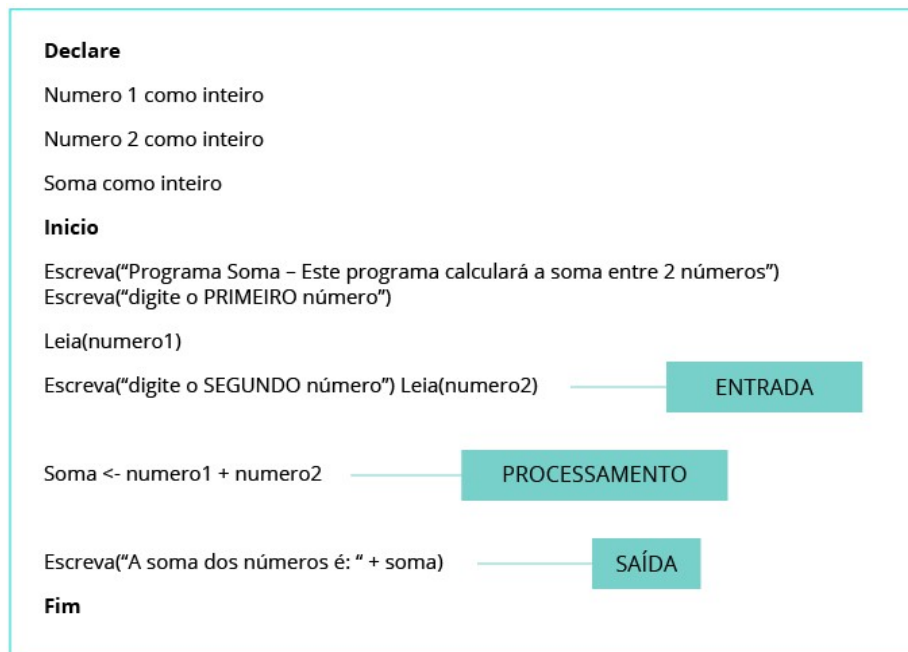
Agora que já conhecemos as variáveis e a sequência lógica do nosso programa, podemos construir o fluxograma conforme a simbologia apresentada anteriormente:



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:



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 novo projeto para este exemplo, da mesma forma que você viu 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);

// inicio 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);

}

}
```

Abaixo, uma representação de como o código ficará no editor da sua IDE Eclipse:

```
1 import java.util.Scanner;
2
3 public class SomaValores {
4     public static void main(String args[]) {
5         /* O código desenvolvido por você deverá estar aqui.
6            As chaves marcam o início do código.
7            Diferente do pseudocódigo, as variáveis em java podem ser declaradas diretamente no
8            corpo do programa, mas a recomendação continua: devemos declarar as avariáveis logo
9            no início do código fonte.
10
11         */
12         // declaração das variáveis
13         int numero1;
14         int numero2;
15         int soma;
16         //para facilitar o entendimento, também habilitamos o leitor no início do código
17         Scanner leitor = new Scanner(System.in);
18         // início do programa
19         System.out.println("Programa Soma- Este programa calculará a soma entre dois números");
20         System.out.println("Digite o PRIMEIRO valor");
21         //leitura do primeiro valor
22         numero1 = leitor.nextInt();
23         System.out.println("Digite o SEGUNDO valor");
24         //leitura do segundo valor
25         numero2 = leitor.nextInt();
26         //processamento
27         Soma = numero1 + numero2;
28         //saída de dados
29         System.out.println("O resultado da soma é" + soma);
30     }
31 }
32
```

Algumas considerações importantes:

1. Você percebeu que há explicações precedidas por duas barras // ou /* ? Esses sinais indicam comentários do Java, com eles você poderá escrever qualquer mensagem para o programador ou então anotar informações importantes sobre o seu código. Esses 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 esse 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(), em que o S deve ser sempre digitado em letra maiúscula.

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 Eclipse fique sublinhado em vermelho, significa que ele está com erro. Se isso ocorrer, primeiro você deve sempre corrigi-lo, caso contrário seu programa não

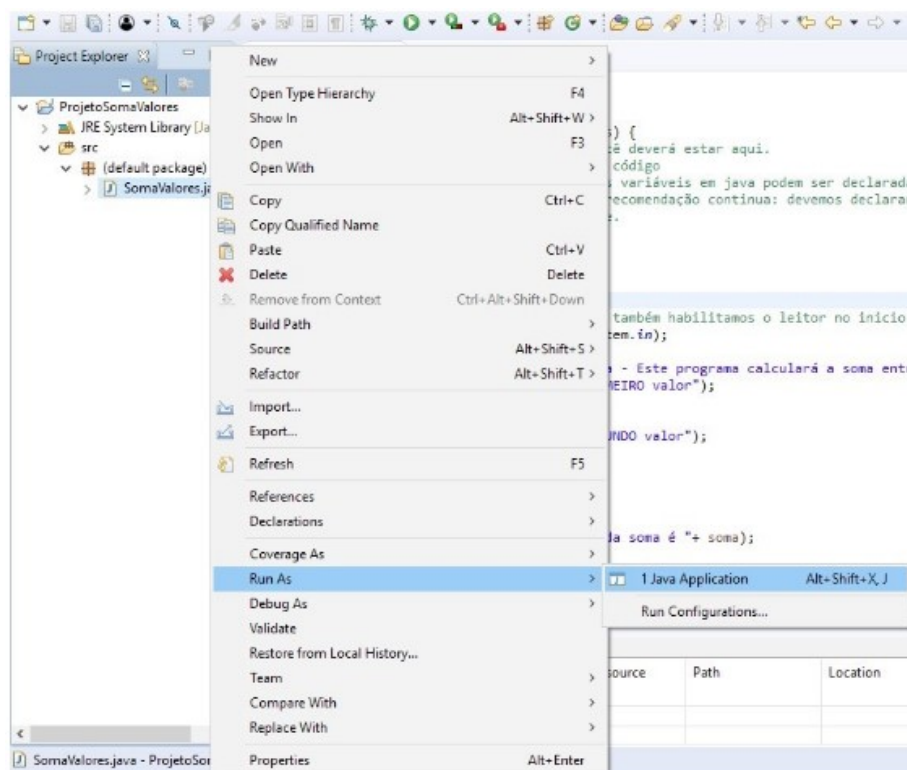
funcionará. Geralmente esses erros são causados por “erro de sintaxe”, em outras palavras, o comando foi digitado incorretamente.

5. Comandos sublinhados em amarelo não são erros. Geralmente são recomendações ou aviso de variáveis declaradas, mas não utilizadas.

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 Run > Java Application

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



A saída da execução ficará como apresentado abaixo:

```
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 haja erros em seu código fonte, o Eclipse retornará uma mensagem de erro, pois não é possível executar o seu código enquanto os erros (sublinhados em vermelho) não sejam corrigidos.

8. Aprimorando a comunicação com o usuário

Antes de mais nada, os programas devem ser fáceis de se utilizar e os usuários devem fazer isso intuitivamente. Para tanto, todo programador deve atentar-se à interface com o usuário. Vamos aprender uma maneira alternativa de entrada e de saída de dados mais elegante para utilizarmos em nossos futuros programas em Java.

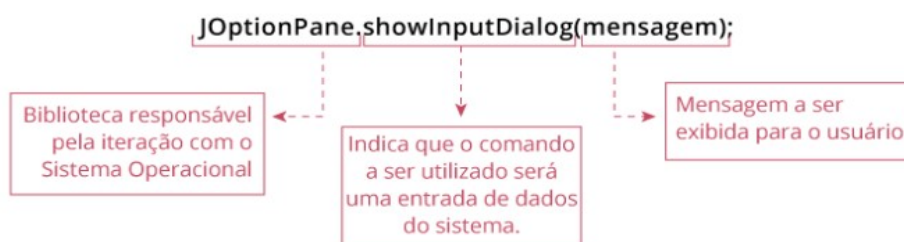
Até aqui utilizamos o que chamamos de modo console, ou seja, uma tela de terminal que não aceitava elementos gráficos para interagirmos com o programa.

```
Entre com um nome:
José
O nome é: José
```

Atualmente, vivemos em um mundo onde os elementos gráficos predominam em todos os aspectos de comunicação. Portanto, será apresentado um modo gráfico utilizando as janelas do sistema operacional para realizarmos a entrada e a saída de dados.

8.1. Entrada de Dados

Para realizarmos a entrada de dados para um programa, utilizaremos o comando **JOptionPane** do Java. A sintaxe do comando é a seguinte:

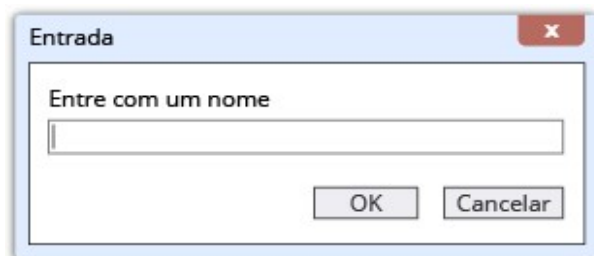


Exemplo de aplicação em um programa:

```
1  import javax.swing.JOptionPane;
2
3  public class JoptionPane {
4
5      public static void main(String args[]) {
6
7          string entrada; //variável de entrada
8          entrada = JOptionPane.showInputDialog("Entre com um nome");
9
10     }
11 }
12 }
13 }
```

Na **linha 7**, realizamos a declaração da variável entrada do tipo String que armazenará o conteúdo inserido pelo usuário.

Na **linha 8**, a variável entrada recebe o conteúdo do comando `JOptionPane.showInputDialog("Entre com um nome")`. Vamos analisar como essa linha se comporta. Esse comando solicita ao usuário que digite algum dado para o sistema. Mas qual dado é esse? No nosso exemplo, o dado solicitado é o nome – Entre com um nome. O resultado para o usuário será:



8.2. Saída de Dados

Um resultado muito mais bonito esteticamente que o modo console. Para realizarmos a saída dos dados de um programa também utilizaremos o comando **JOptionPane**, mas com a seguinte sintaxe:



Exemplo de aplicação em um programa:

```
1 import javax.swing.JOptionPane;
2
3 public class JOptionPaneShow {
4
5     public static void main(String args[]) {
6         JOptionPane.showMessageDialog(null, "Saída de dados");
7     }
8 }
9
10
11
```

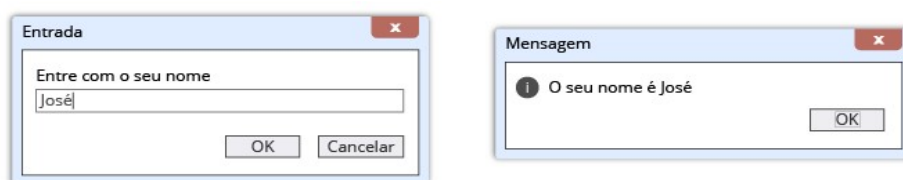
Na linha 6, o comando `JOptionPane.showMessageDialog (null, "saída de dados");` faz a saída da mensagem para o usuário. O primeiro argumento sempre será null, seguido pela mensagem que queremos exibir para o usuário – Saída de dados. O resultado será:



Agora, sabendo como realizar a entrada e a saída de dados de forma gráfica, conseguimos fazer um programa que leia e exiba dados para o usuário em modo gráfico, como no exemplo a seguir:

```
1 import javax.swing.JOptionPane;
2
3 public class JOptionPane {
4
5     public static void main(String args[]) {
6
7         String nome; //variável nome do tipo String
8
9         //entrada de dados
10        nome = JOptionPane.showInputDialog("Entre com o seu nome");
11
12        //saída de dados
13        JOptionPane.showMessageDialog(null, "O seu nome é" + nome);
14    }
15
16
```

Neste exemplo apresentado, o programa exibe uma janela pedindo o nome do usuário na linha 10 e posteriormente exibe o nome digitado na linha 13. Resultado:



9. Conversão de Tipos

O comando `JOptionPane.showInputDialog` sempre gera uma saída de dados do tipo `String` (sequência de caracteres alfanuméricos). Portanto, se utilizarmos tipos de variáveis diferentes como, por exemplo, inteiro, devemos fazer a conversão de tipos antes de armazenarmos na sua variável correspondente conforme o exemplo a seguir:

```
1  import javax.swing.JOptionPane;
2
3  public class JOptionPaneCast {
4
5      public static void main(String args[]) {
6          String auxiliar; //variável auxiliar
7          int numeroInteiro;
8          double numerodouble;
9          float numerofloat;
10
11         // entrada de dados salvando na variável auxiliar (string)
12         auxiliar = JOptionPane.showInputDialog("Entre com um número inteiro");
13
14         //conversão do tipo string para inteiro - Integer.parseInt
15         numeroInteiro = Integer.parseInt(auxiliar);
16         numerodouble = Double.parseDouble(auxiliar);
17         numerofloat = Float.parseFloat(auxiliar);
18
19         //mensagem de saída
20         JOptionPane.showMessageDialog(null, "O número inteiro é" + numeroInteiro);
21         JOptionPane.showMessageDialog(null, "O número double é" + numerodouble);
22         JOptionPane.showMessageDialog(null, "O número float é" + numerofloat);
23
24     }
25
26 }
```

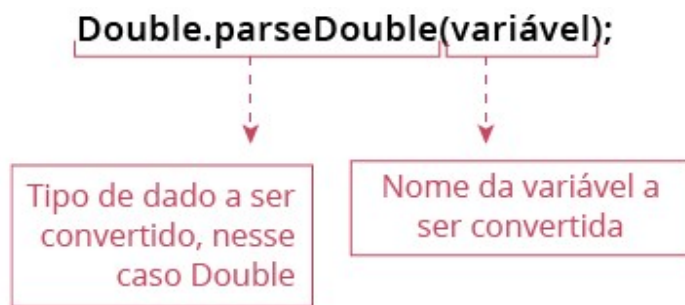
Um tipo nada mais é do que o conteúdo que uma variável consegue armazenar. Um tipo `"String"` pode armazenar caracteres e números. Um tipo `"int"`, números inteiros e os tipos `"double"` e `"float"`, números reais.

No exemplo da imagem acima, a entrada de dados feita na **linha 12** é salva na variável `auxiliar` que é do tipo `String`. Porém, nesse exemplo, estamos trabalhando com números e um dado do tipo `String` para armazenar textos. Para isso, devemos fazer a conversão de tipo (cast em inglês), ou seja, temos que realizar a conversão de um tipo de dado para outro. Ex.: de **String para int**, de **String para double** etc. O Java possui comandos específicos para executar o cast.

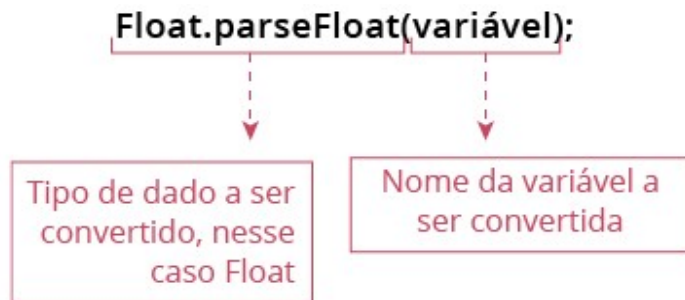
A sintaxe para a conversão de um tipo **String para Inteiro** é:



A sintaxe para a conversão de um tipo **String para Double** é:



A sintaxe para a conversão de um tipo **String** para **Float** é:



Conhecendo a sintaxe do comando; na **linha 15**, a variável `numeroInteiro` recebe o resultado da conversão do valor da variável auxiliar de **String** para **Inteiro**, assim como as **linhas 16 e 17** realizam o cast para **double** e **float**, respectivamente.

Um aspecto interessante na linguagem Java é que no comando `JOptionPane.showMessageDialog`, não precisamos realizar a conversão de tipos para o `String`. O Java, automaticamente, converte tipos numéricos para `String` antes de exibir a mensagem para o usuário.

Uma observação: existem outras formas de realizarmos a conversão de tipos e para mais tipos de dados, porém não serão apresentadas aqui neste momento.

Para finalizar, convidamos você a pensar: o que poderia acontecer se fizéssemos uma adição entre os números 10 e 15, sendo esses números declarados como variáveis do tipo `String`, sem que tivéssemos feito uma conversão para o tipo numérico antes de realizar a operação? Qual seria o resultado apresentado pelo Java?

Traga a sua resposta no Fórum de Colaboração e apoio, da Agenda 03.

Ampliando Horizontes

Onde encontro mais informações sobre linguagem Java?

Para aprofundamento dos temas discutidos nesta agenda, seguem abaixo algumas dicas de vídeos e livros que se relacionam com o conteúdo estudado. Essas dicas são muito importantes para você!

Videoaula de Informática, abordando os comandos ESCREVA e LEIA



Nesta videoaula, preparada para o curso Técnico de Informática, Módulo I - Agenda 11, o professor Sandro Valérius explica os comandos Escreva e Leia com detalhes.

Caso você tenha interesse em testar o nosso código com o pseudocódigo, execute-o com o auxílio da ferramenta **Visualg**. Na videoaula a seguir, o professor Sandro detalha o uso desse software.

Informática - Módulo I - Agenda 11 - Aprenda a u...



Observação: O Visualg é um software gratuito que poderá ser baixado facilmente pela internet. É um programa que edita, interpreta e executa algoritmos em português e é muito utilizado no ensino de programação de computadores em todo o mundo.

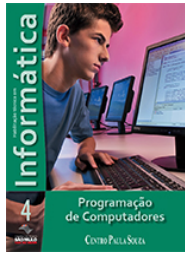
Curso de Java para iniciantes

Curso de Java #01 - História do Java - Gustavo G...



Essa playlist oferece um curso grátis para iniciantes em Java, assista às videoaulas 1 a 8.

[Informática, programação de computadores | Vol. 4 \(Links to an external site.\)](#). Centro Paula Souza. (2010). São Paulo: Fundação Padre Anchieta.



Java para iniciantes. SCHILDT, Hebert. Editora Bookman, 2015.



Lógica de programação e estruturas de dados com aplicações em Java. PUGA, Sandra; RISSETTI, Gerson. Editora Pearson, 2009.



Java - Como Programar. DEITEL, P; DEITEL, H. São Paulo: Pearson Prentice Hall, 2010.

Resumindo o Estudo

Esta foi uma agenda extensa! Você teve seu primeiro contato com a linguagem Java, e conheceu algumas das ferramentas que utilizará daqui para frente: diversos comandos fundamentais para programar nessa linguagem. Você também entendeu como acontece o desenvolvimento de um programa simples.



Após este trajeto desafiador, você está preparado para compreender as estruturas de decisão!

Até a próxima agenda!

