

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

PROGRAMA NOVOTEC VIRTUAL
GOVERNO DO ESTADO DE SÃO PAULO
EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO
QUALIFICAÇÃO PROFISSIONAL DE ASSISTENTE DE DESENVOLVIMENTO DE SISTEMAS
LÓGICA DE PROGRAMAÇÃO

PÚBLICO ALVO: ALUNOS DA 3ª SÉRIE DO ENSINO MÉDIO
TEMPO DE INTEGRALIZAÇÃO: 34 SEMANAS

Autores:

Eliana Cristina Nogueira Barion

Marcelo Fernando Iguchi

Paulo Henrique Mendes Carvalho

Rute Akie Utida

Revisão Técnica: Sandra Maria Leandro

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação: Flávio Biazim

AGENDA 3

A LÓGICA APLICADA EM JAVA





MERGULHANDO NO TEMA...

Java é uma linguagem orientada a objetos¹ 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.



Muito provavelmente você já deve ter ouvido falar sobre a linguagem Java. Basta ler uma revista de informática ou matérias sobre desenvolvimento de *softwares* que logo encontra alguma informação sobre Java, dado o sucesso que esta linguagem tem tido no mercado, principalmente pelo fato dos programas escritos em Java poderem ser executados virtualmente em qualquer plataforma e aceitos em qualquer tipo de computador ou outros aparelhos, uma característica marcante da Internet. Por isso, a linguagem Java tem se destacado muito no mercado e aprendê-la é importante para você e para a sua profissão!



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

Para escrevermos um programa desenvolvido em Pseudocódigo de forma que o computador possa compilá-lo² 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. É também capaz de fornecer uma portabilidade muito grande, sendo compatível com a maioria dos Sistemas Operacionais disponíveis para usuários gerais.

O mesmo programa escrito com a Linguagem de Programação Java, poderá ser utilizado em um computador com os Sistemas Operacionais Windows, Linux ou Mac OS sem que nenhuma linha do código fonte³ necessite ser alterada, ganhando tempo no desenvolvimento de novas aplicações.

1 - Linguagem de Programação Orientada a Objetos (P.O.O) significa que esta Linguagem de Programação utilizará os conceitos de Orientação a Objetos (O.O.) o qual veremos no módulo seguinte com mais profundidade.

2 - Compilação é a ação de transformar um código amigável escrito com uma Linguagem de Programação em um programa executável baseado em Código de Máquina.

3 - Código Fonte é o conjunto de instruções criadas pelo programador utilizando uma Linguagem de Programação.

Além disto, Java é uma linguagem de programação que pode ser utilizada para desenvolver páginas da Internet, por meio de um Servidor Web configurado para executar páginas do tipo Java Server Pages (jsp). Com ele também podemos desenvolver aplicativos para celulares que utilizam o Sistema Operacional Android.

Pela sua vasta lista de plataformas suportada, utilizaremos o Java como Linguagem de Programação de apoio no desenvolvimento da Lógica de Programação.



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 e chocolate granulado e ainda providencia as ferramentas necessárias para a confecção do doce: panela e colher de pau.



Imagem 03: Freepik – Doce de brigadeiro

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.

Vamos lá:

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 a um computador permitindo a execução do código fonte, por isto recebe o nome de Máquina Virtual.

Na prática, basta instalar em seu computador a JVM desenvolvida para o Sistema Operacional e você estará pronto para executar programas desenvolvidos em Java.

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



Como em breve seremos desenvolvedores e não usuários, também precisaremos instalar em nosso computador o kit para desenvolvimento de programas feitos em Java, o que inclui o compilador⁴ da linguagem de programação e a Máquina Virtual (JVM). 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 auxiliar nos 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 IDE desde que siga fielmente as estruturas listadas neste material.

Durante os nossos estudos, utilizaremos 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.

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:

Antes de continuar, você pode efetuar o download e instalação do JDK e do Eclipse nestes links:

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



Java SE
Soporte para Java SE
Java Embedded
Java EE
Java ME
Java FX
Java DB
Web Tier
Comunidade

Resumo Downloads Documentação Comunidade Tecnologia Formação

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day hands-on workshops \(free\) and other events](#)
- [Java Magazine](#)

JDK 8u201 [checksum](#)
JDK 8u202 [checksum](#)

Java SE Development Kit 8u201		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.98 MB	jdk-8u201-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u201-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.98 MB	jdk-8u201-linux-i586.rpm
Linux x86	185.77 MB	jdk-8u201-linux-i586.tar.gz

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources

- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

Imagem 04: Site da Oracle para download e instalação do JDK

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

Criando um projeto utilizando a IDE Eclipse



Eclipse IDE for Java EE Developers

Package Description

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

This package includes:

- Data Tools Platform
- Git integration for Eclipse
- Eclipse Java Development Tools
- Eclipse Java EE Developer Tools
- JavaScript Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Eclipse Plug-in Development Environment

Download Links

Windows 32-bit
Windows 64-bit
Mac OS X (Cocoa) 64-bit
Linux 32-bit
Linux 64-bit

Downloaded 1,098,739 Times

► [Checksums...](#)

[Bugzilla](#)

Imagem 05: Site para download e instalação do 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.

Como utilizaremos a Linguagem de Programação estruturada⁵ Java, não entraremos fundo no significado de Projeto, Pacote e Classe, pois estas definições são utilizadas em programas Orientados a Objetos⁶.



VOCÊ NO COMANDO

ATIVIDADE: FÓRUM DE DISCUSSÃO

Discuta com seus colegas e com o professor-tutor quais são as outras APIs existentes no Mercado que suportam a Linguagem de Programação Java. Todas elas são gratuitas? Quais são suas vantagens e desvantagens em relação a API Eclipse? Pesquise e reflita antes de prosseguir com a leitura.

Esse vídeo contém explicações do tutorial a seguir. Você pode optar em assisti-lo ou acompanhar os passos a passos do tutorial descritivo, logo a seguir.



Disponível em https://www.youtube.com/watch?time_continue=177&v=1MKD4nEGGTA&feature=emb_logo

Tutorial passo a passo...

Para criarmos um novo programa, após abrir a janela principal do Eclipse, selecionamos o menu File, New, Java Project, conforme a imagem a seguir:

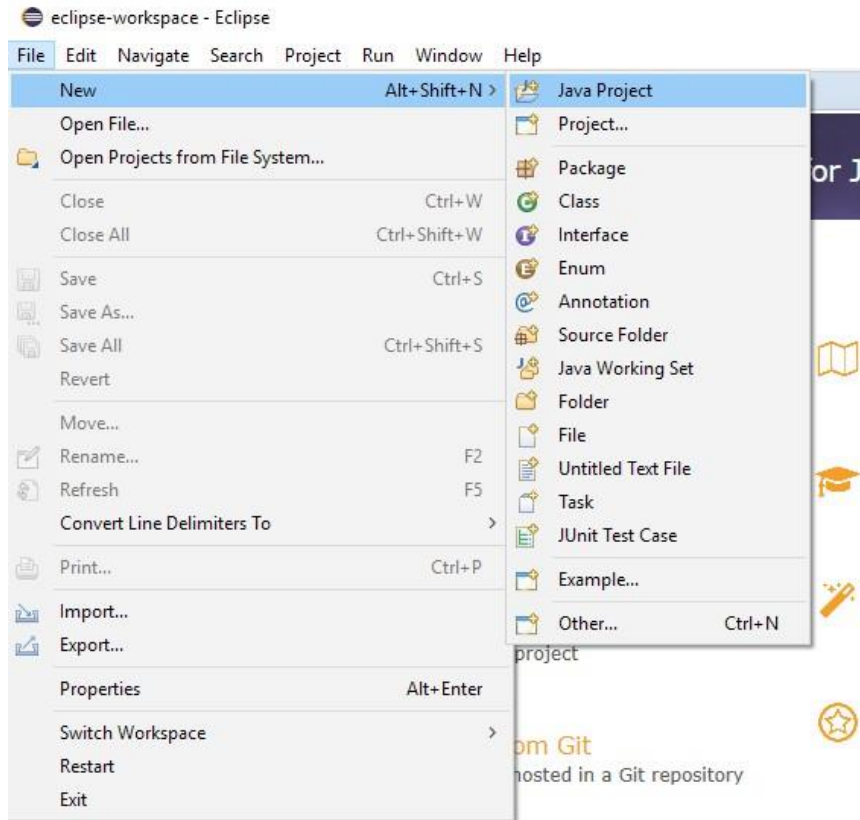


Imagem 06: Janela principal do Eclipse para seleção do menu File. Apresenta a seleção de opções para abrir um novo projeto em Java. Menu File – Opção New – Java Project.

5 - A Programação Estruturada é uma forma de desenvolver programas baseando-se no conceito de início – meio – fim de forma linear e em um único código fonte.

6 - A Orientação a Objetos é uma forma de desenvolver programas baseada em Classes e Objetos, afim de aproximar a forma na qual o código fonte é desenvolvido das nossas ações do dia a dia.

A seguinte janela será exibida:

As opções fornecidas são:

1. Nome do Projeto: nesta opção você deve indicar um nome para o seu programa. O nome do programa deve ser simples e objetivo. Não é recomendado que sejam utilizados espaços para separar palavras. Caso seja necessário, utilize o *underline* (traço baixo). Este campo é obrigatório.

2. Selecionar onde ficará salvo o projeto: é altamente recomendável que todos os projetos desenvolvidos com o Eclipse sejam salvos em um único lugar, preferencialmente no local indicado ao iniciar o Eclipse. desmarque a opção “*Use default location*” apenas se a sua intenção for salvar o seu projeto em um local diferente do padrão.

3. JRE: você só deve alterar esta opção caso você tenha o JDK instalado em seu computador em mais de uma versão e deseje testar seus programas em uma versão mais antiga do JDK.

4. Project Layout: nesta opção você

indicará como o Eclipse deve organizar os arquivos de seu programa. O ideal é manter a opção padrão selecionada, pois ela manterá seus códigos fontes sempre em uma pasta separada do restante do projeto.

5. Working Sets: apenas para usuários avançados. Com esta opção você poderá mesclar dados do seu projeto com o de outros projetos localizados em diferentes *workspaces*.

Após inserir um nome para o seu projeto e alterar as demais opções, caso seja necessário, clique em Finish. Você voltará para a tela inicial, porém desta vez haverá um pequeno botão ao lado esquerdo ou direito do Eclipse, dependendo da versão utilizada. A imagem do botão aparece na imagem a seguir:

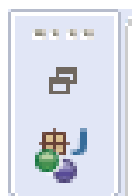


Imagem 08: Botão do Eclipse para voltar à tela inicial

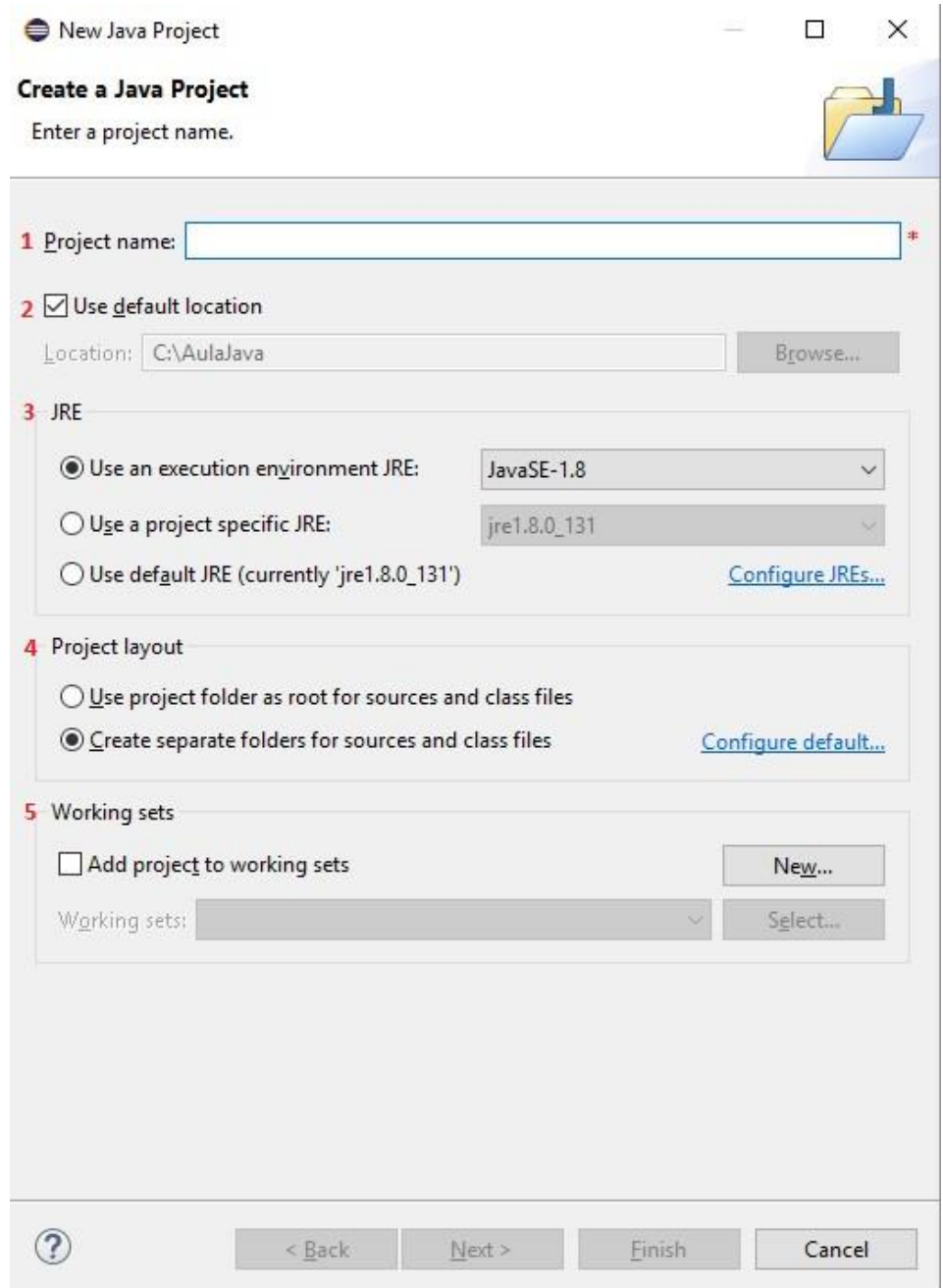


Imagem 07: Janela de criação de novo projeto em Java

Note que ao lado direito, marcado com o número 1, temos a janela Project Explorer. Será por esta Janela que localizaremos os arquivos contidos no nosso projeto.

Importante: Caso você feche acidentalmente a janela Project Explorer, poderá acessá-la novamente por meio do menu Window > Show View > Project Explorer.

Agora que temos o projeto, vamos criar um arquivo para o nosso código fonte: em Project Explorer, entre na pasta referente ao seu projeto, localize a pasta chamada `src`⁷, clique com o botão direito nela e selecione New > Class:

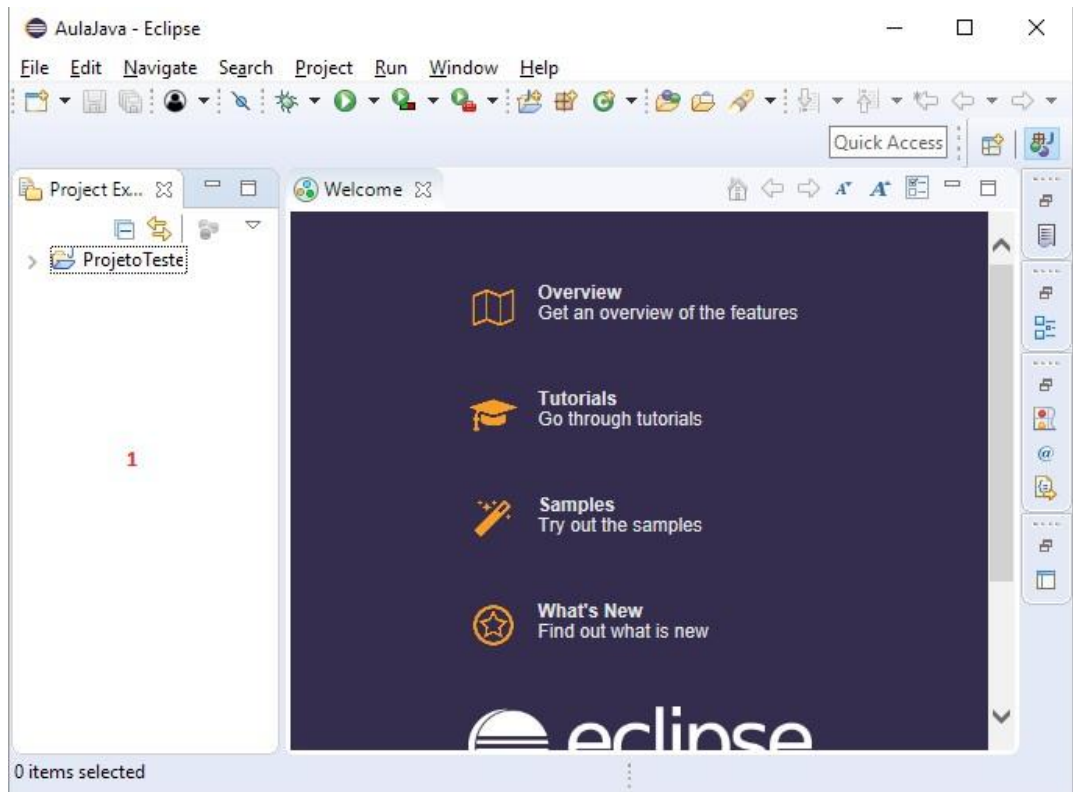


Imagem 09: Tela inicial do Eclipse

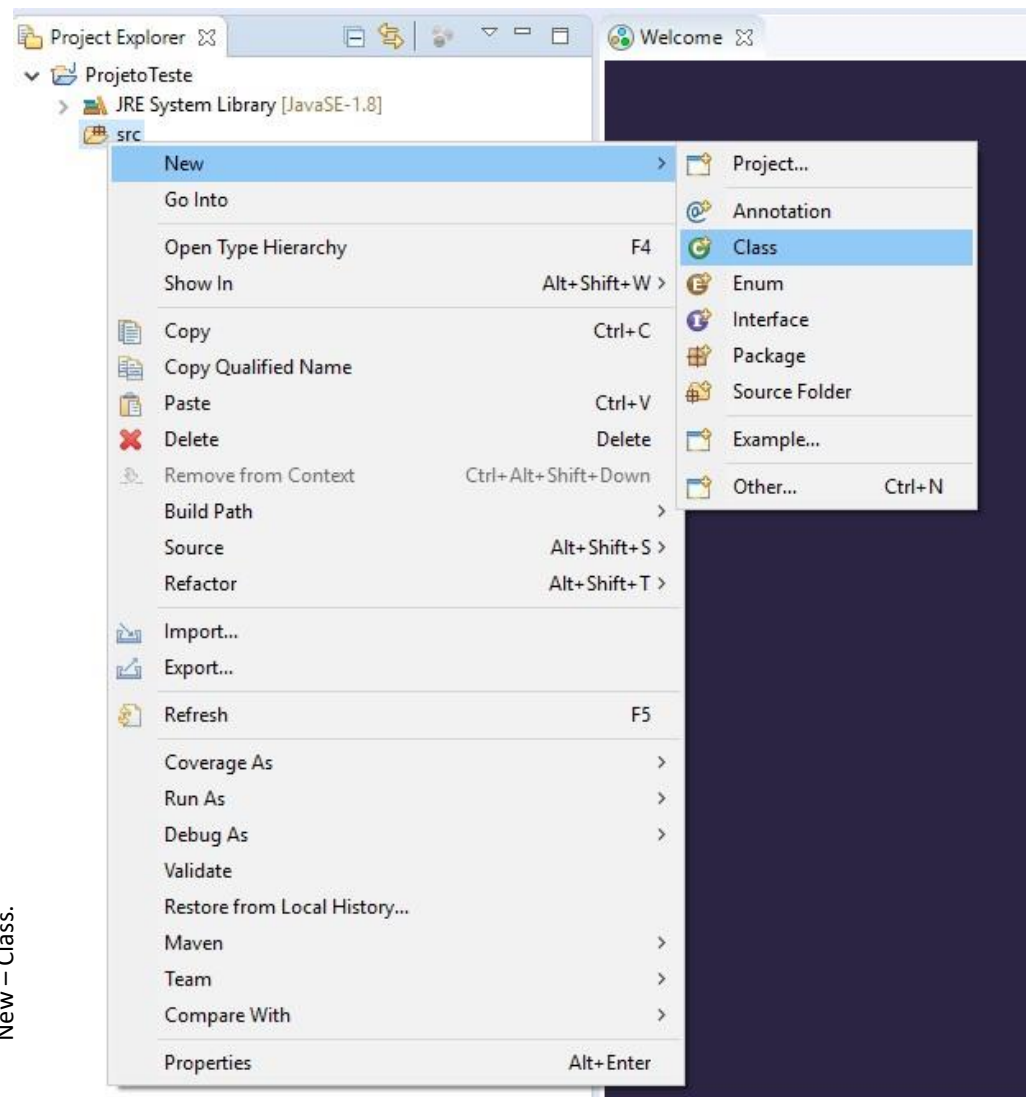
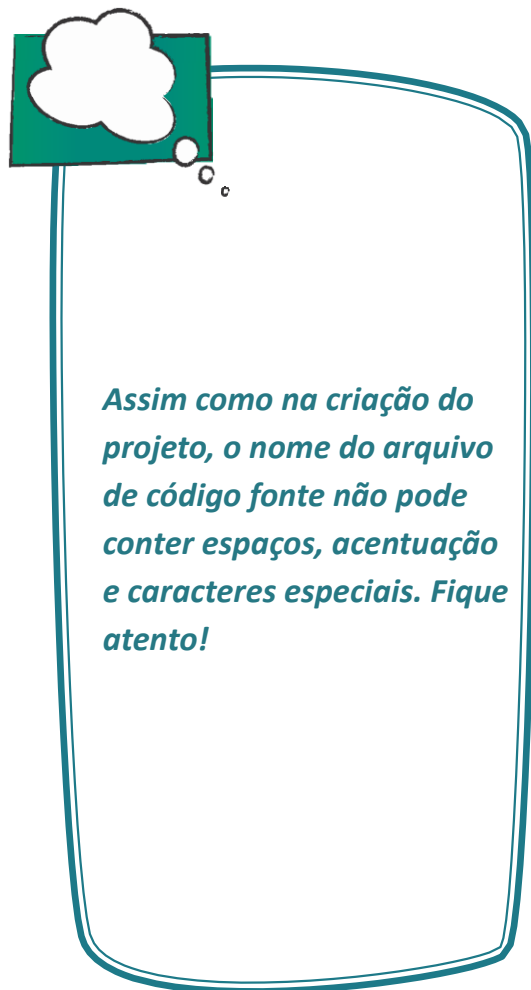


Imagem 10: Tela do Project Explorer de Eclipse para criação de Classe. Passos para a seleção de Classe: Project Explorer – ProjetoTeste – SRC – New – Class.

7 - A pasta `src` (abreviação para source Code) é a pasta padrão do eclipse para receber códigos fontes

Por não utilizarmos a Orientação a Objetos na programação Java, precisaremos apenas definir um nome para o arquivo (1) e selecionar a opção “public static void main(String[] args)” (2) :



New Java Class

Java Class

⚠ The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

Name: **1**

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

2 ☒ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Imagem 11: Tela de criação de nova classe em Java

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:

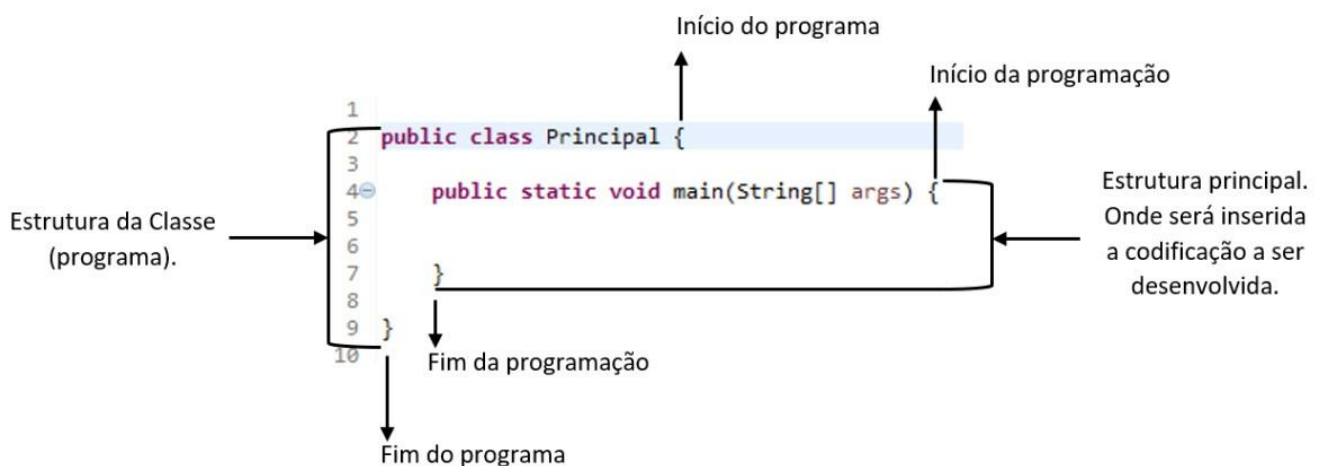


Imagem 12: A imagem representa a estrutura básica de uma classe o símbolo “abre e fecha chaves” significa início e fim de programa, respectivamente. A estrutura de um programa é composta por: public class “nome do Programa”, abre chave, public static void main (String[] args, fecha a chave { da estrutura 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. Esta 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.

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:

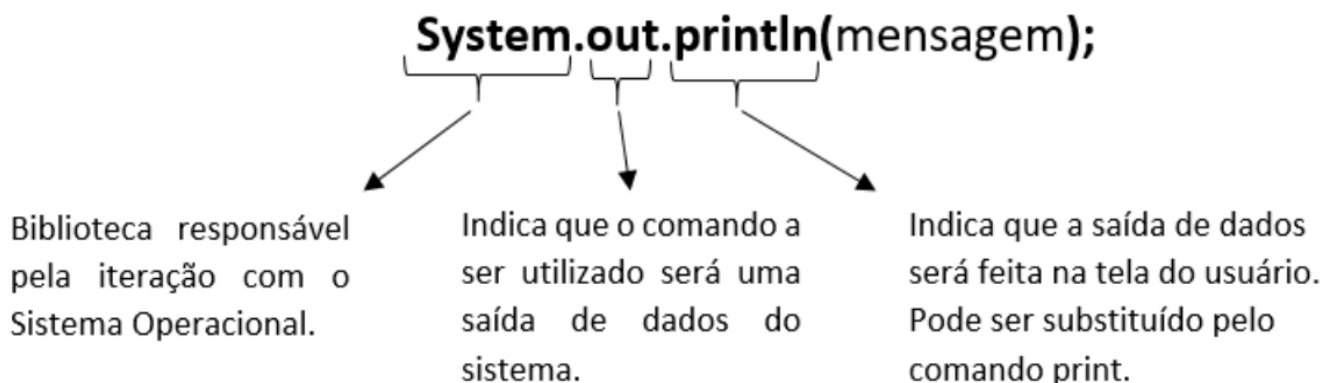


Imagem 13: Representação da estrutura do comando System.out.println. Descrição do comando: System: Biblioteca responsável pela interação com o Sistema Operacional. Out: Indica que o comando a ser utilizado será uma saída de dados do sistema. Println: Indica que a saída de dados será feita na tela do 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, 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?

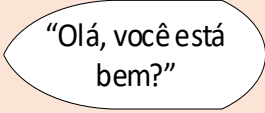
Fluxograma	Pseudocódigo	Java
	escreva “Olá, você está bem?”	System.out.println(“Olá, você está bem?”);

Imagem 14: no fluxograma é apresentado o símbolo de saída de dados com a mensagem “Olá, você está bem?”

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 15: 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.

Voltando ao caso da Agenda Telefônica, qual seria o tipo de dado a ser aplicado para indicar o número do telefone informado por Juvenal?

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-la⁸ em seu programa, seguindo o padrão para cada linguagem de programação, sendo:

PSEUDOCÓDIGO

declare
nome da variavel como tipodedado

JAVA

declare
tipodedado 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;

Imagem 16: 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

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

Imagem 17: 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 especiais, 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.

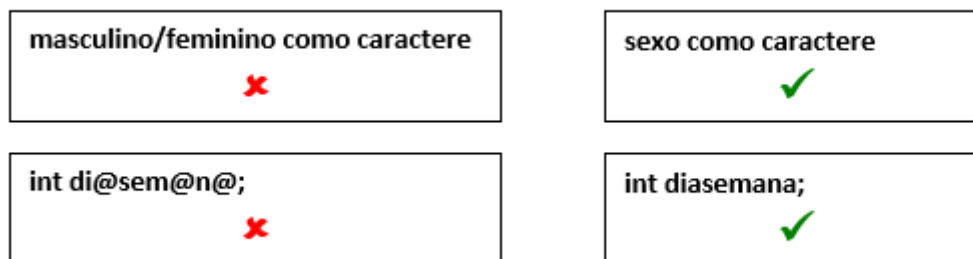


Imagem 18: Nomeação de variáveis – exemplos de variáveis corretas e incorretas, não é permitido o uso de caracteres especiais. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “masculino/feminino como caractere”, 2º quadro: `int di@sem@n@;`. Esses dois quadros, posicionados à esquerda, 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 as linguagens de programação são todas escritas em Inglês, onde 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.

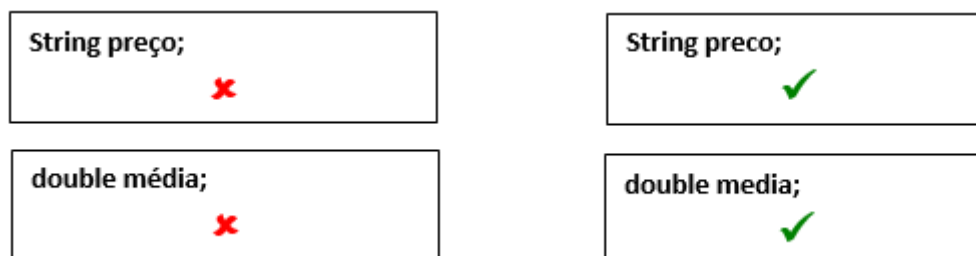


Imagem 19: Nomeação de variáveis – exemplos de variáveis corretas e incorretas, não é permitido o uso de acentuação nos nomes. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “String preço”, 2º quadro: Double média. Esses dois quadros, posicionados à esquerda, estão com um xis vermelho 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

9- Erros de compilação são configurados quando há erros na estrutura do código fonte. Caso este tipo de erro ocorra, não será possível a execução do programa.

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.

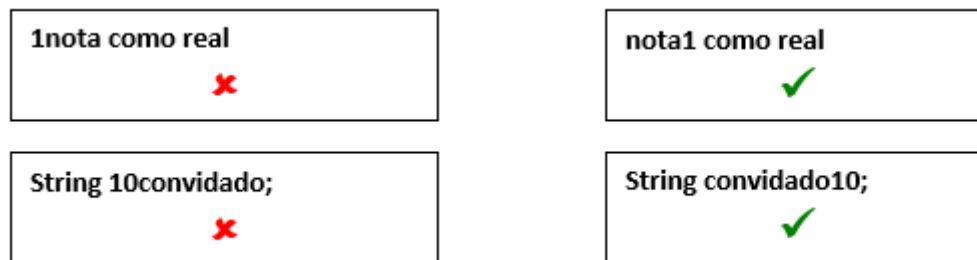


Imagem 20: Nomeação de variáveis- exemplos de variáveis corretas e incorretas, não é permitido iniciar o nome com um número. Na imagem temos 4 quadros com declaração de variáveis. 1º quadro: “1nota como real”, 2º quadro: String 10convidado. Esses dois quadros, posicionados à esquerda, estão com um xis 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:

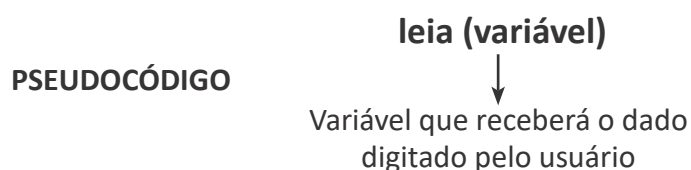


Imagem 21: Comando “leia” – recebe um valor digitado pelo usuário

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, consequentemente, 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:

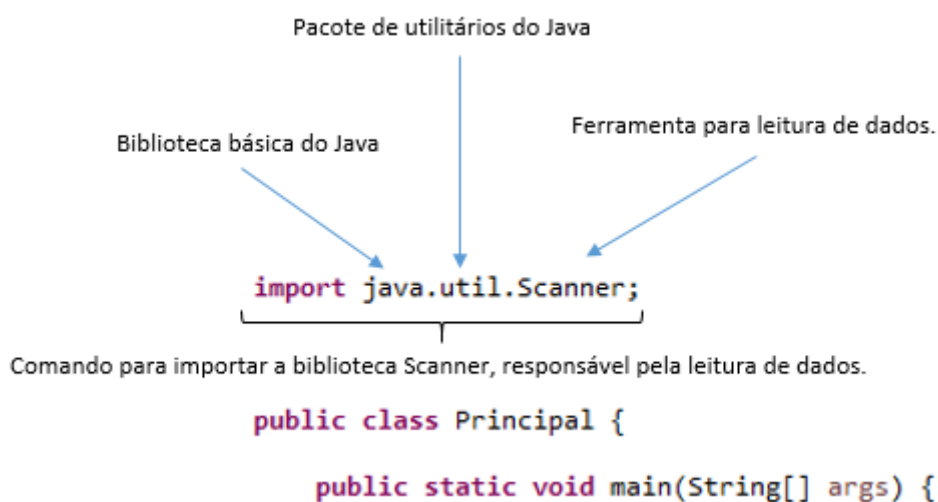


Imagem 22: comando para importação da biblioteca Java. A imagem explica cada comando de importação da biblioteca import java.util.Scanner; sendo: import java é a biblioteca básica do Java, útil o pacote de utilitários do Java e Scanner a ferramenta para leitura de dados. Este é o comando para importar 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 Eclipse.

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 necessita 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 23: 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 24: 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; numero = leitor.nextByte();</code>
short	<code>leitor.nextShort();</code>	<code>short numero; numero = leitor.nextShort();</code>
int	<code>leitor.nextInt();</code>	<code>Int numero; numero = leitor.nextInt();</code>
long	<code>leitor.nextLong();</code>	<code>long numero; numero = leitor.nextLong();</code>
float	<code>leitor.nextFloat();</code>	<code>float numero; numero = leitor.nextFloat();</code>
double	<code>leitor.nextDouble();</code>	<code>double numero; numero = leitor.nextDouble();</code>
char	<code>leitor.next().charAt(0);</code>	<code>char letra; letra = leitor.next().charAt(0);</code>
String	<code>leitor.next();</code>	<code>String palavra; palavra = leitor.next();</code>
boolean	<code>leitor.nextBoolean();</code>	<code>Boolean teste; teste = leitor.nextBoolean();</code>

Imagem 25: 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.

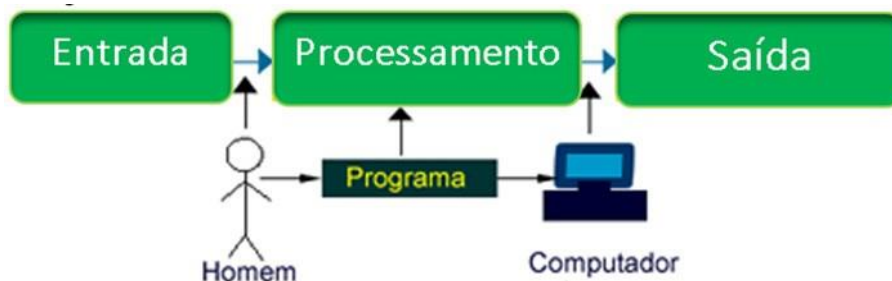


Imagem 26: A imagem mostra três quadrados verdes escrito Entrada, Processamento e Saída de Dados. Na seta que vai da entrada para o processamento, temos a representação de um usuário. Um rótulo escrito Programa que aponta para o processamento e uma seta com um computador aponta para a saída de dados.

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:

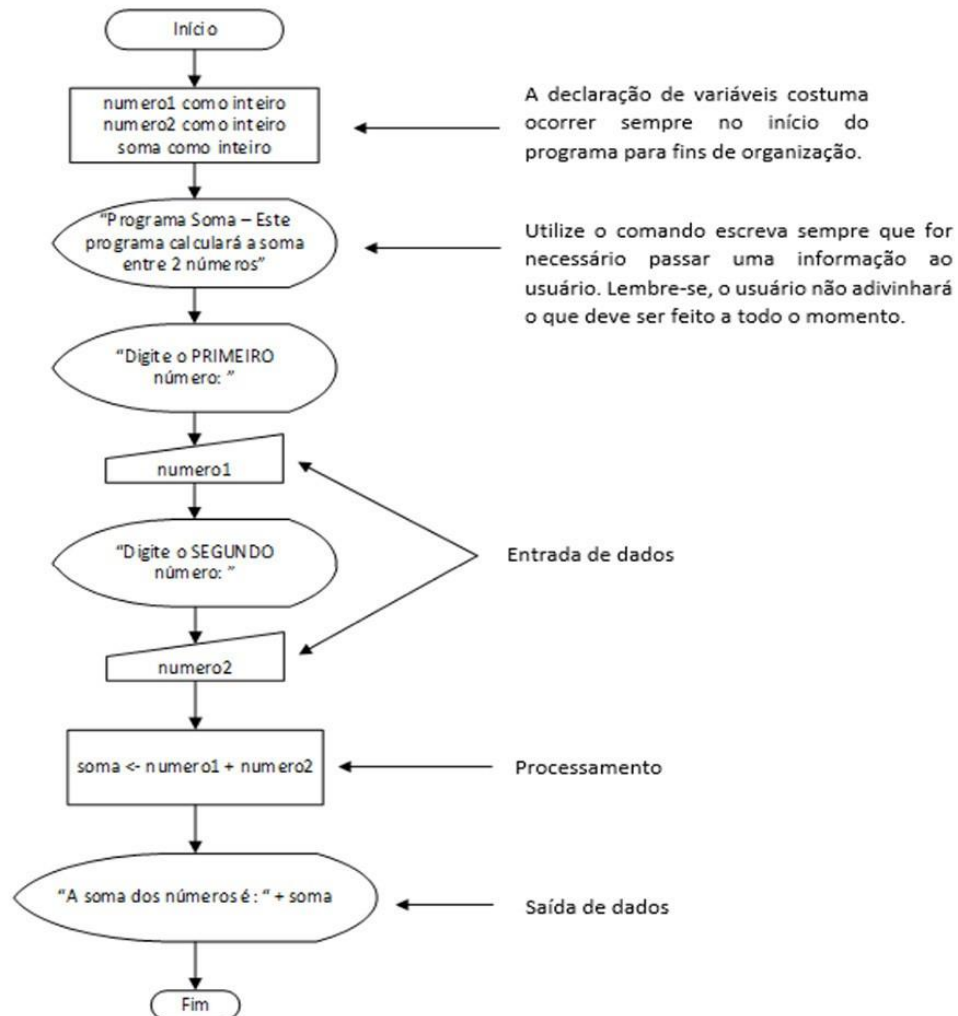


Imagem 27: GEEaD: Representação de um Fluxograma – o exemplo usado é para ler dois números, calcular a soma destes e apresentar o resultado.

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

Numero 1 como inteiro

Numero 2 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 28: 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 novo 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 29: 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.
 - a. Os comandos para iniciar um comentário em Java são:
 - b. `//` comentário de uma única linha
 - c. `/*`
 - i. Comentário de múltiplas linhas
 - d. `*/`
 - e. 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.
-  **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. Caso isto ocorra, primeiro você deve sempre corrigi-lo, 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 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:

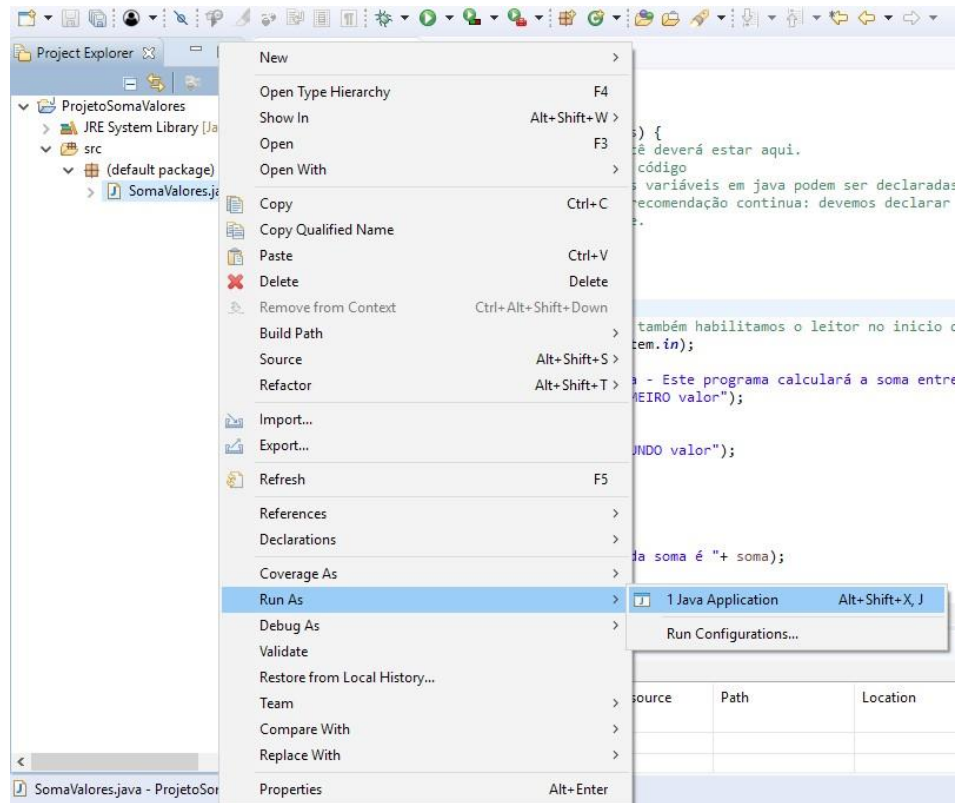


Imagem 30: Seleção da opção “Run As” de uma aplicação em Java- para executar a aplicação é necessário usar esta opção.

```
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
```

Imagem 31: Janela console – nesta janela será visualizada a execução da aplicação criada.

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.

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 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
```

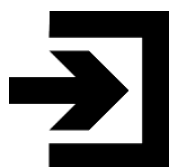
Imagem 32: Terminal de saída de dados – janela onde será exibida a execução da aplicação.

VOCÊ NO COMANDO

Um programador deve sempre tentar desenvolver uma interface com o usuário que seja simples de ser utilizada. Contudo, essa interface deve agradar ao cliente que está contratando os seus serviços. Pensando como programador, antes de prosseguir a leitura, reflita como isso pode interferir na prática de programação.

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.

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:



Imagem 33: Sintaxe do comando JOptionPane para entrada de dados. Composição da sintaxe

`JOptionPane.showInputDialog(mensagem);`

`JOptionPane` – é a biblioteca responsável pela interação com o Sistema Operacional. `showInputDialog` – indica que o comando a ser utilizado será uma entrada de dados do sistema. `(mensagem)` – é a mensagem a ser exibida para o usuário.

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- }

```

Imagem 34: Mostra um exemplo de aplicação de um programa em Java.

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 esta linha se comporta. Este comando solicita ao usuário que digite algum dado para o sistema. Mas qual dado é este? No nosso exemplo, o dado solicitado é o nome – Entre com um nome. O resultado para o usuário será:

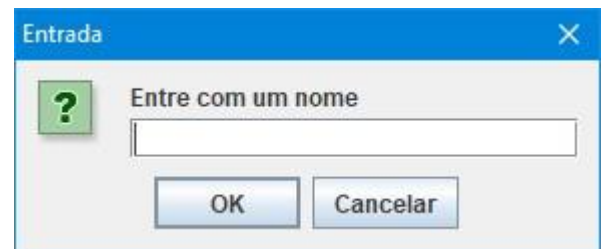


Imagem 35: Tela de entrada de dados – com uso do modo gráfico. Essa imagem mostra uma caixa de entrada de dados com uma mensagem: “Entre com um nome”, e dois botões: “Ok” à esquerda e “Cancelar” à direita.

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:

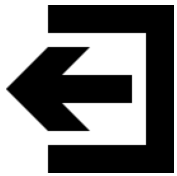


Imagem 36: Sintaxe do comando JOptionPane para saída de dados. Composição da sintaxe `JOptionPane.showMessageDialog(null, mensagem);`;

`JOptionPane` – é a biblioteca responsável pela interação com o Sistema Operacional. `showMessageDialog` – indica que o comando a ser utilizado será uma saída de dados do sistema. `(null, mensagem)` – é a mensagem a ser exibida para o usuário.

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- }
```

Imagem 37: Tela de exemplo de codificação do comando JOptionPane para saída de dados

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:

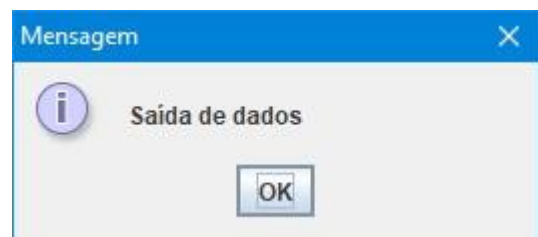


Imagem 38: Caixa de mensagem de informação, escrito "Saída de dados" e um botão de Ok centralizado.

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- }

```

Imagem 39: Codificação para leitura e exibição de dados em modo gráfico

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:

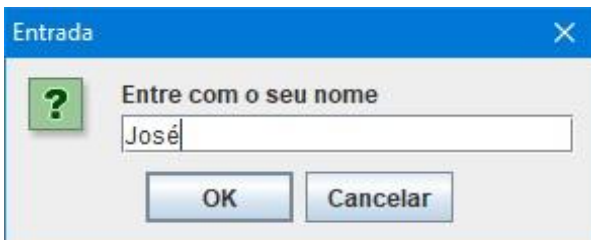


Imagem 40: Caixa de Entrada de Dados do tipo "Question" com a mensagem "Entre com o seu nome". Na caixa de entrada de dados foi digitado o texto "José". À esquerda tem o botão "OK" e à direita o botão "Cancelar".

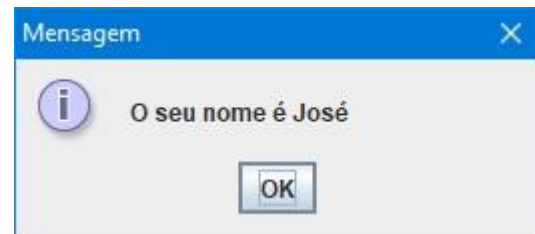


Imagem 41: Caixa de Saída de Dados informando que o nome digitado foi "José" e um botão de "Ok".

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- }

```

Imagem 42: Codificação de conversão de dados – exemplo de uso do comando `JOptionPane.showInputDialog`.

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 isto, 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:

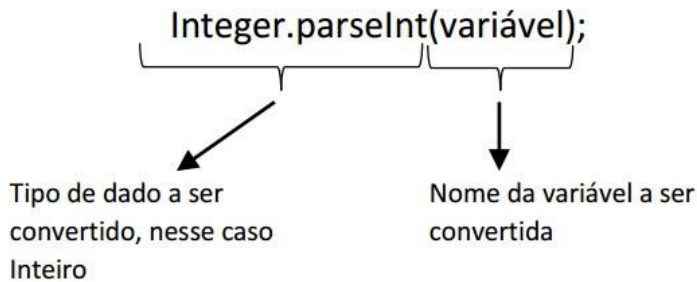


Imagem 43 sintaxe para a conversão de um tipo String para Inteiro. Composição da sintaxe `Integer.parseInt(variável);`
`Integer.parseInt` – é o tipo de dado a ser convertido, nesse caso, inteiro.
`(variável);` - é o nome da variável a ser convertida.

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

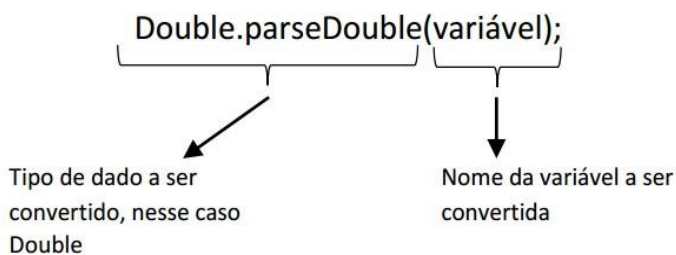


Imagem 44: sintaxe para a conversão de um tipo String para Double.
 Composição da sintaxe `Double.parseDouble(variável);`
`Double.parseDouble` – é o tipo de dado a ser convertido, nesse caso, Double.
`(variável);` - é o nome da variável a ser convertida.

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

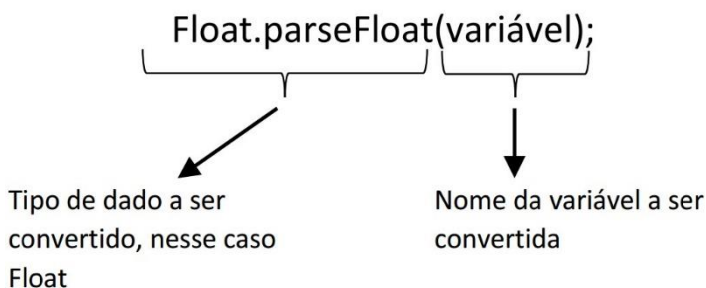


Imagem 45: sintaxe para a conversão de um tipo String para Float.
 Composição da sintaxe `Float.parseFloat(variável);`
`Float.parseFloat` – é o tipo de dado a ser convertido, nesse caso, Float.
`(variável);` - é o nome da variável a ser convertida.

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 nesse momento.

VOCÊ NO COMANDO

Pense no 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.

Pensou?

Agora veja as explicações abaixo para ver se você acertou a resposta:

O resultado apresentado pelo Java seria **"1015"**.

Quando há uma soma entre valores do tipo `String`, o Java apenas os junta, formando um novo valor com a junção dos conteúdos que estavam nas variáveis. Esta operação chama-se **Concatenação** e é muito utilizada em casos onde é necessária a junção de palavras, como por exemplo: O seu nome com o sobrenome.

Por isto é importante sempre realizar a conversão de dados quando for necessária a execução de operações matemáticas para não concatenarmos os valores acidentalmente.

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).

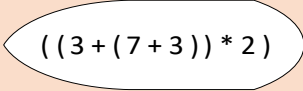
b. Leia dois números inteiros, realize a soma deste e exiba o resultado final.

Agora confira se você acertou:

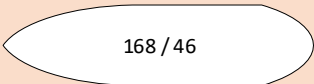
Seguindo as regras de precedência matemática, de operadores aritméticos e do comando escreva, temos:

1.

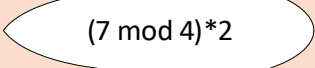
a) Multiplicar por 2 o resultado de $(7 + 3) + 3$.

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

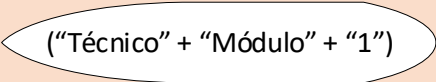
b) dividir 168 por 46

Fluxograma	Pseudocódigo	Java
	escreva (168/46)	System.out.println(168/46);

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

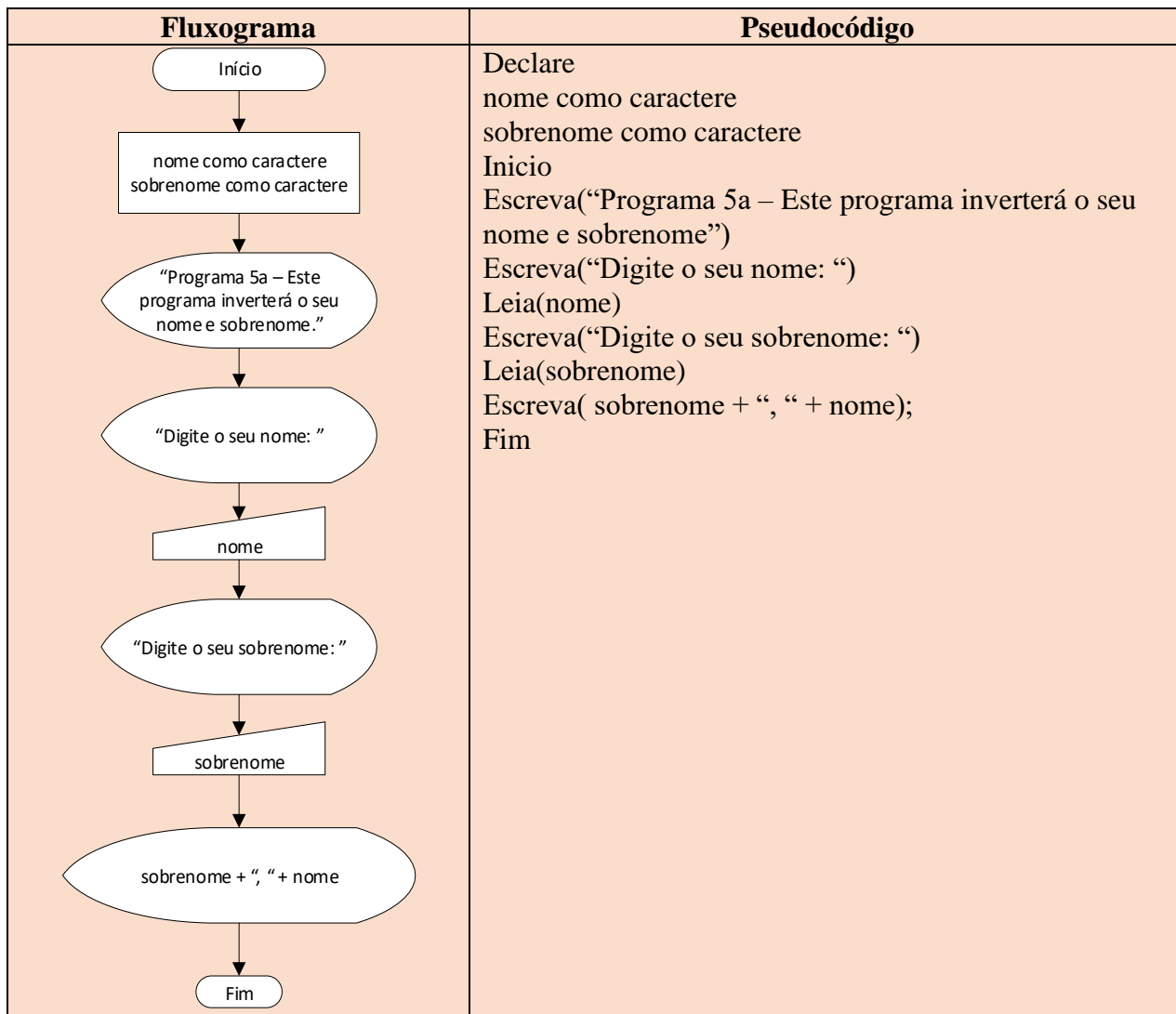
Fluxograma	Pseudocódigo	Java
	escreva (7 mod 4)*2	System.out.println((7 % 4)*2);

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

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

2.

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



Codificação em Java:

```

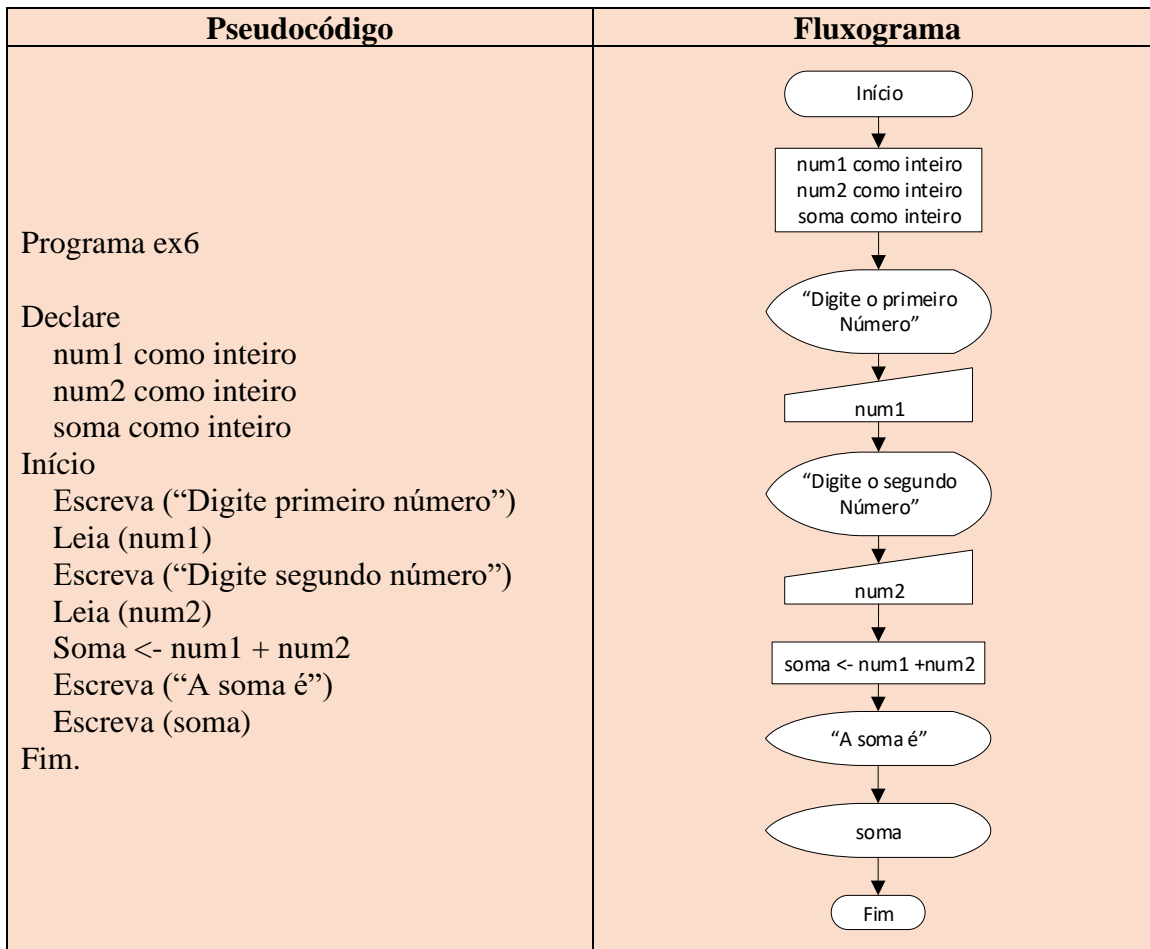
import java.util.Scanner;

public class Programa5a {

    public static void main(String args[]) {
        // declaração de variáveis e inicialização do leitor
        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 soma deste e exiba o resultado final.



Codificação em Java:

```

1- import javax.swing.JOptionPane;
2-
3- public class JoptionPane_ex4 {
4-
5- public static void main(String args[]) {
6- //exercício 4
7-
8- //declaração de variáveis
9-     int num1, num2, soma;
10-     String aux; //variável aux é auxiliar;
11-
12-     // entrada de dados
13-     aux = JOptionPane.showInputDialog("Digite o primeiro
número");
14-     num1 = Integer.parseInt (aux); //faz a conversão de tipo
para inteiro
15-
16-     Num2 = Integer.parseInt(JOptionPane.showInputDialog("Digite
o segundo número");
17-
18-     //processamento
19-     Soma = num1 + num2
20-
21-     //saída de dados
22-     JOptionPane.showMessageDialog(null,"A soma é " + soma);
23-
24- }
25-
26- }

```