UNIVERSIDADE FEDERAL DO PARÁ - UFPA

CAMPUS UNIVERSITÁRIO DE TUCURUÍ

FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

Apostila Java para público Juvenil

Autores:

Edson Martins Cavalcante

Erick Vinícius Damasceno da Silva

Fellipe Augusto Santana Queiroz

Natã Ferreira Lobato

Sobre esta apostila

Esta apostila foi elaborada por discentes do curso de engenharia da computação, da UFPA - campus Tucuruí, visando ensinar de maneira sucinta, mostrando apenas o que realmente é necessário para a iniciação da programação na linguagem de Java, buscando transmitir ao leitor uma experiência de leitura mais rápida e dinâmica.

Sumário

1. Introdução a Linguagem Java ---------- 4

1.1. História do JAVA ---------- 4

1.2. Explicar JAVA ---------- 5

1.3. Começando com JAVA ---------- 7

1.3.1. Instalando JDK ---------- 7

1.3.1.1. Instalação JDK no Windows ---------- 7

1.3.1.2. Instalação JDK no Linux ---------- 11

1.3.2. Preparando ambiente de programação (eclipse) ---------- 12

1.3.2.1. Instalação Eclipse IDE no Windows ---------- 12

1.3.2.2. Instalação Eclipse IDE no linux ---------- 14

1.3.2.3. Abertura do Eclipse IDE ---------- 15

1.3.3. Compilando primeiro Código (hello World) ---------- 16

2. Variáveis ---------- 20

2.1. Int ---------- 20

2.2. Long ---------- 21

2.3. Outros tipos para valores inteiros ---------- 21

2.4. Float ---------- 21

2.5. Double ---------- 22

2.6. Char ---------- 22

2.7. String ---------- 23

2.8. Boolean ---------- 23

2.9. Outros tipos de Variáveis ---------- 24

3. Operações ----------26

3.1. Manipulando variáveis com operadores ---------- 26

4. Estruturas Condicionais ---------- 28

4.1. Bloco If, Else ---------- 28

4.2. Bloco Switch Case ---------- 30

4.3. Outro tipo de Condicional ---------- 30

4.4. Operadores lógicos ---------- 31

5. Laços de repetição ---------- 33

5.1. For ---------- 33

5.2. While ---------- 34

5.3. Do-While ---------- 35
6. Arrays e listas ---------- 36
6.1. Array ---------- 36
6.2. Lista ---------- 38
6.3. ArrayList ---------- 38
7. Funções ---------- 40
7.1. Criação de funções ---------- 40
8. Referências ---------- 43

Capítulo 1

Introdução a Linguagem Java

1.1 História do JAVA

Java é uma linguagem de programação que começou a ser criada em 1991 pela antiga Sun Microsystems e mantida através de um comitê. A Sun criou um time, o Green Team, para desenvolver inovações tecnológicas. E tinham como mentores Patrick Naughton, Mike Sheridan e James Gosling. Gosling é considerado o pai do Java.

Esse time não tinha intenção criar uma linguagem de programação, mas sim um interpretador para empresas de eletrônicos de consumo, facilitando a reescrita de software para aparelhos eletrônicos, como vídeo cassete, televisão e aparelhos de TV a cabo, foi aí então que surgiu a linguagem Oak. Mas a ideia não deu certo, Tentaram fechar diversos contratos com grandes fabricantes de eletrônicos, como Panasonic, mas perceberam que a disparidade de hardwares inviabilizaria o seu uso, pois teriam que desenvolver um software para cada tipo de dispositivo.

Com o crescente número de usuários da internet a empresa percebeu que seria possível utilizar a ideia criada em 1992 para aplicações dentro do browser. Na internet havia uma grande quantidade de sistemas operacionais e browsers, e seria vantajoso poder programar numa única linguagem, independente da plataforma. Então o Java 1.0 foi lançado em 1995, uma adaptação e atualização do Oak para a internet, focado em mudar os sites de apenas html estático para aplicações com funções e operações avançadas.

O nome da linguagem desenvolvida é uma homenagem à ilha indonésia Java, de onde os norte-americanos da equipe de James Gosling importavam o café que consumiam.

Até 1994, não haviam aplicações para o Java. Então Jonathan Payne e Patrick Naughton criaram um navegador para Web que executava programas com a linguagem, com o nome de Web Runner.

E em 1996, a Sun Microsystem disponibilizou gratuitamente um kit de desenvolvimento, que ficou conhecido como Java developer 's Kit (JDK). Foi então que começou a aceitação da tecnologia pelos desenvolvedores e empresas.

A tecnologia teve uma enorme aceitação, e grandes empresas como a IBM e Oracle anunciaram que estariam dando suporte ao Java, isso significava que os produtos destas empresas iriam rodar aplicativos feitos em Java. Estimativas mostram que a tecnologia Java foi a mais rapidamente incorporada na história. Em 2003 o Java já tinha mais de 4 milhões de desenvolvedores. Foi nesse momento que a ideia inicial começou a se tornar realidade. A linguagem passou a ser utilizada em dezenas de produtos diferentes. Computadores, celulares, palmtops, e a maioria dos produtos da Apple.

E em abril de 2009, a Oracle fez uma proposta de US $7,4 Bilhões na compra da Sun Microsystem, que foi aceita. Essa compra deu à Oracle direitos dos produtos da Sun, incluindo java e o sistema operacional Solaris. Muito se foi dito sobre a aquisição e o futuro do Java, mas essa mudança de direitos contribuiu muito para que a linguagem tivesse um salto qualitativo.

1.2 Explicando JAVA

A Plataforma é um hardware ou um ambiente de software onde um programa é executado. Boa parte das plataformas pode ser descrita como uma combinação de um sistema operacional e um hardware que o suporta. Como exemplos de plataformas, temos: Microsoft Windows, Linux e Mac OS.

A Plataforma Java é um ambiente de software onde os programas escritos na linguagem Java são executados. Essa plataforma é composta por dois componentes: JVM(Java Virtual Machine) e API (Application Programming Interface).

JVM é uma máquina virtual desenvolvida em código nativo, pois ela conversa diretamente com o sistema operacional para que o programa Java funcione na máquina, que basicamente é um software que simula uma máquina física e consegue executar vários programas, gerenciar processos, memória e arquivos. Ela constitui uma plataforma onde a memória, o processador e seus outros recursos, são totalmente virtuais, não dependendo de hardwares.

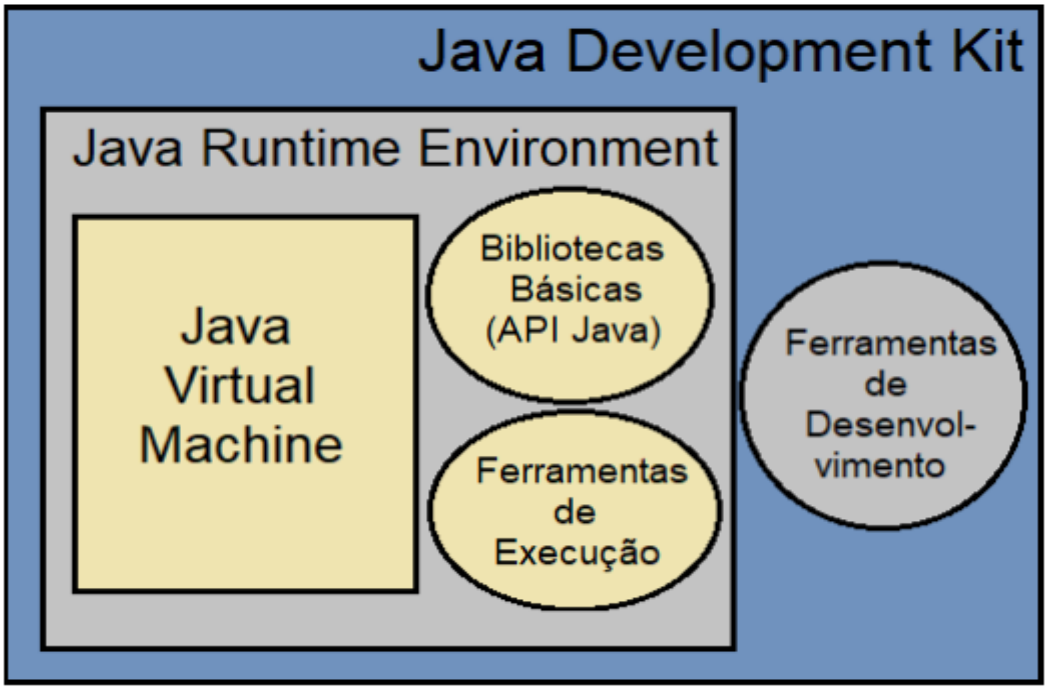
A API contém centenas de Interfaces e milhares de classes que acompanham o kit de desenvolvimento do Java e que podem ser empregadas para a realização de diversos tipos de tarefas durante a construção de um programa.

A execução de um código em Java não está diretamente relacionada com o Sistema Operacional, ele conversa diretamente com a JVM (Java Virtual Machine), possibilitando assim a portabilidade de seu código. O que for escrito em um sistema operacional Windows, irá rodar em um sistema operacional Linux (salvo algumas exceções de códigos nativos).

Para ter a Máquina Virtual Java em seu sistema é necessário instalar o JRE (Java Runtime Environment), um programa gratuito que basicamente permite ao usuário rodar aplicativos Java em seu computador. Ele é responsável por prover os requisitos mínimos para executar um programa Java, contém uma JVM e os pacotes básicos do Java (API core).

Para aqueles que desejam desenvolver aplicações, é necessário instalar o JDK (Java Development Kit), pacote que inclui tudo o que é necessário para escrever aplicações e também o JRE para poder rodá-los após finalizá-los. Muitos programadores fazem uso de IDEs (Integrated Development Environment) de programação, como Eclipse e Netbeans, para ajudá-los durante o desenvolvimento.

Figura 1: ilustração da relação entre os componentes.



Fonte: dicasdejava.com.br.

1.3 Começando com JAVA

1.3.1 Instalando JDK

1.3.1.1 Instalação JDK no Windows.

Vamos mostrar o passo a passo de como é feita a instalação do JDK que é o pacote que inclui tudo que precisaremos para desenvolver um código em Java no ambiente Windows.

1º Passo: acesse a página de download do JDK no site da Oracle e clique no link do JDK.

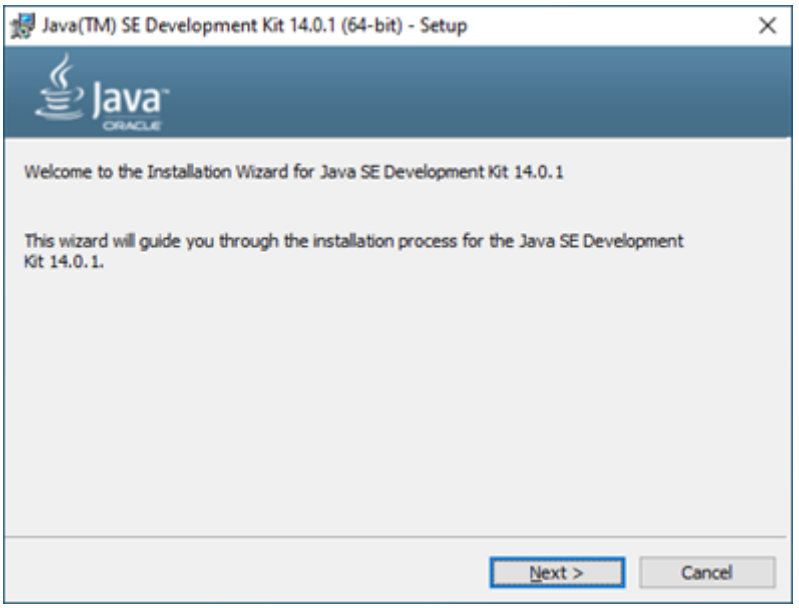
Na página seguinte temos que escolher a versão que seja adequada para a arquitetura do seu sistema operacional (32 bits ou 64 bits).

Java SE Development Kit 16.0.1

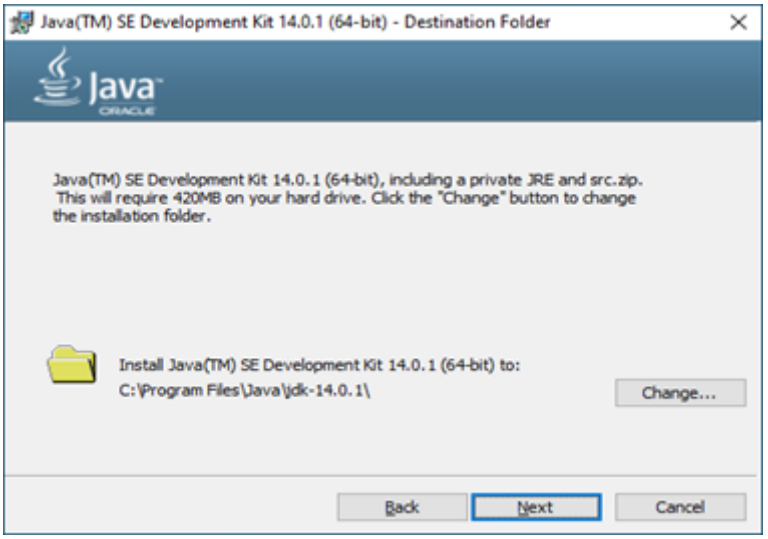
Este software é licenciado sob o Contrato de Licença de Rede de Tecnologia Oracle para Oracle Java SE

Descrição do produto / arquivo	Tamanho do arquivo	Download
Pacote Linux ARM 64 RPM	144,87 MB	jdk-16.0.1_linux-aarch64_bin.rpm
Arquivo compactado Linux ARM 64	160,72 MB	jdk-16.0.1_linux-aarch64_bin.tar.gz
Pacote Debian x64 para Linux	146,16 MB	jdk-16.0.1_linux-x64_bin.deb
Pacote Linux x64 RPM	152,99 MB	jdk-16.0.1_linux-x64_bin.rpm
Arquivo compactado Linux x64	170,02 MB	jdk-16.0.1_linux-x64_bin.tar.gz
MacOS Installer	166,58 MB	jdk-16.0.1_osx-x64_bin.dmg
Arquivo compactado macOS	167,2 MB	jdk-16.0.1_osx-x64_bin.tar.gz
Instalador do Windows x64	150,56 MB	jdk-16.0.1_windows-x64_bin.exe

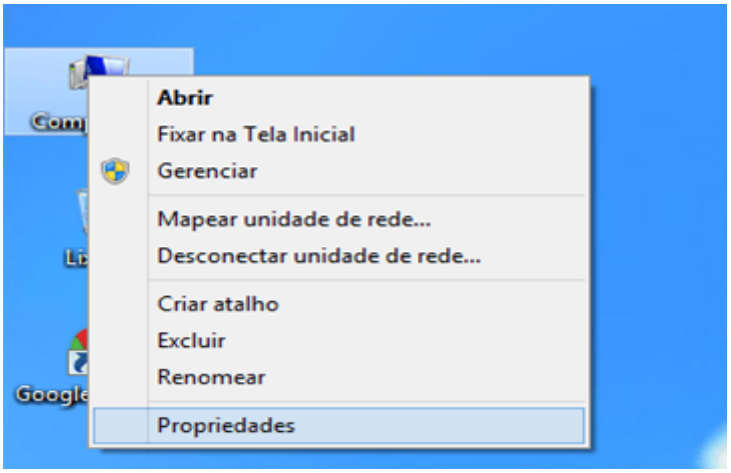
2º Passo: Após realizar o download do instalador do JDK, execute o mesmo e clique em “Next”.



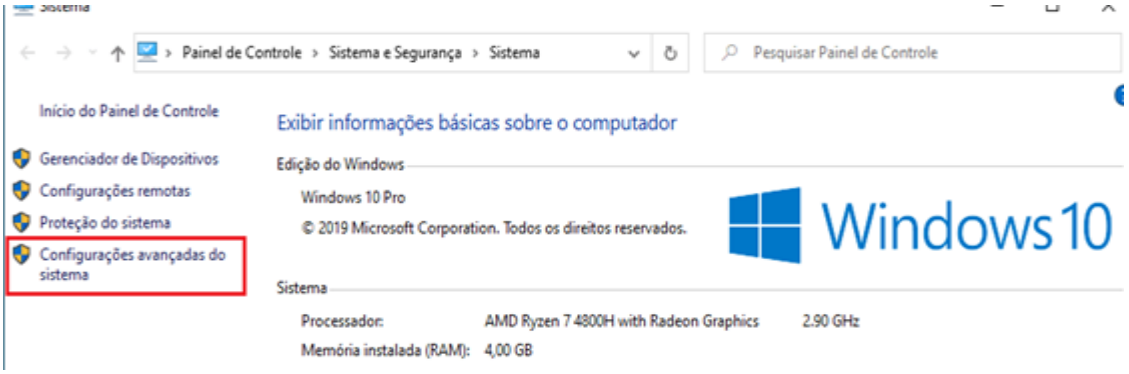
Na tela seguinte grave o local ao qual o Java está sendo instalado e clique em “Next”.



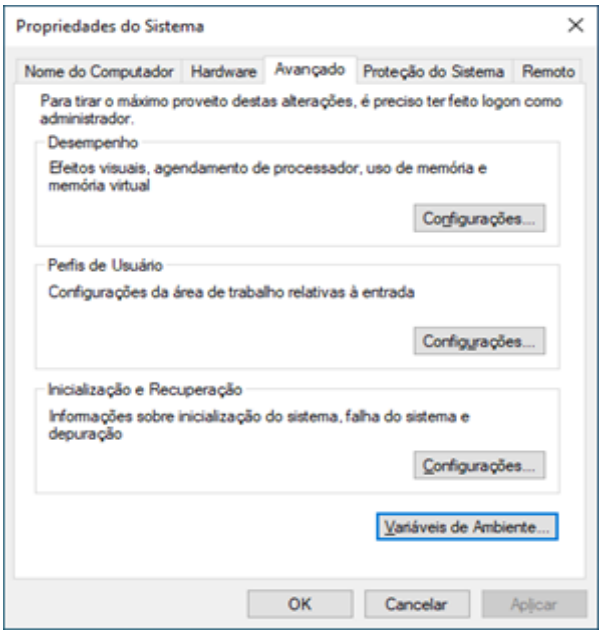
3º Passo: Configurando as variáveis de ambiente do Windows. Clique com o botão direito sobre o ícone de “Meu Computador” na sua área de trabalho, e no menu de contexto escolha a opção “Propriedades”.



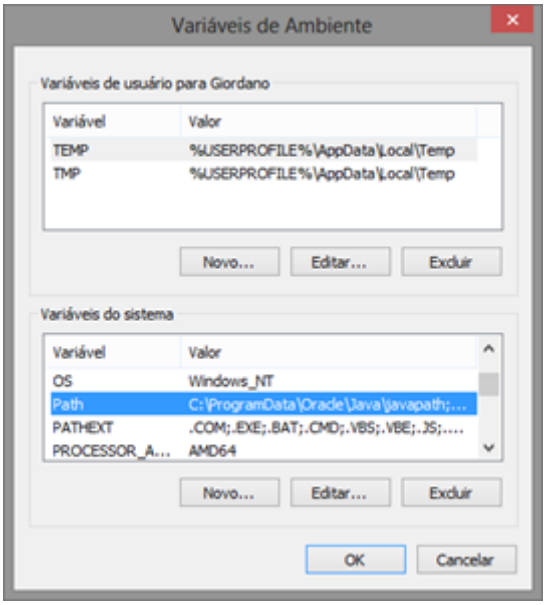
Na janela que se abrirá, acesse a opção Configurações avançadas do sistema.



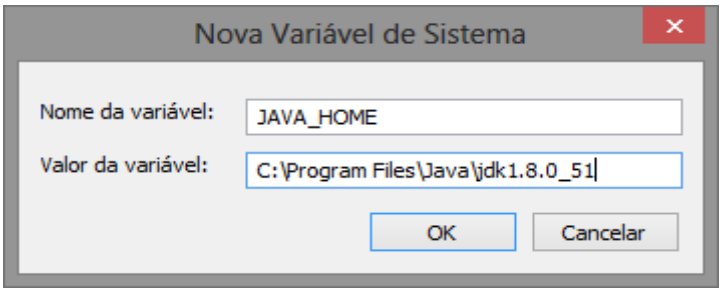
Na próxima tela clique no botão Variáveis de Ambiente que fica no canto inferior direito da janela.



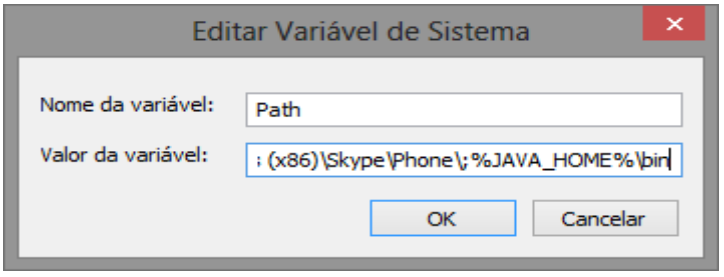
Teremos a janela de variáveis do ambiente exibida. Nela clique no botão Novo.



Na janela que se abre, definimos a variável JAVA_HOME, informando no valor da variável, aquele caminho de instalação do JDK que anotamos no início do processo de instalação.



Por fim teremos que selecionar a variável *Path* na lista de variáveis do sistema e clicamos no botão Editar**.** Tomando muito cuidado para não apagar nada do que já está escrito no valor da variável *Path*, senão diversas aplicações que dependem desta configurações podem parar de funcionar corretamente. O que deve ser feito no valor desta variável é ir até o final do valor da mesma e acrescentar a nossa variável JAVA_HOME ao Path.

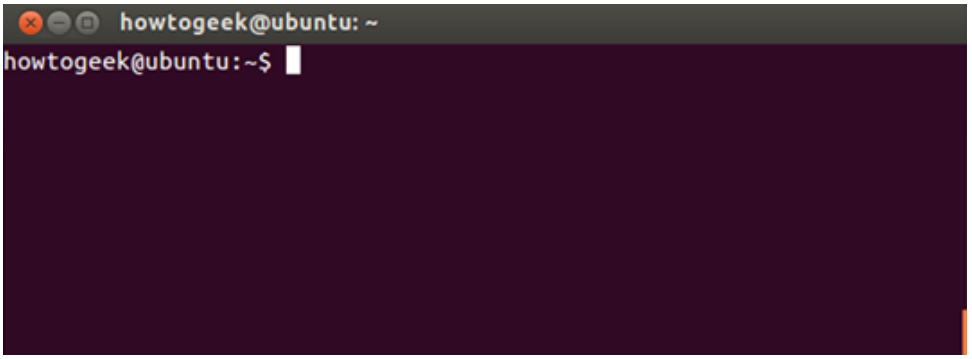


1.3.1.2 Instalação JDK no Linux.

Para a instalação no Linux temos dois tipos, o primeiro é o OpenJDK que é liberado sob a licença GPL v2, e o segundo é o Oracle JDK que é licenciado sob o Contrato de Licença de Código Binário da Oracle.

Para podermos fazer a instalação do OpenJDK seguiremos os passos abaixo.

Passo 1: Abrir o terminal.



Passo 2: Para instalar o Java 14, use o comando abaixo no terminal.

sudo apt install openjdk-14-jre

Podendo utilizar outras versões basta trocar o numero do Java para o que deseja utilizar. Exemplo de instalação Java 8.

sudo apt install openjdk-8-jre

Contudo agora veremos como podemos fazer a instalação do Oracle JDK e para isso seguiremos os passos abaixo.

Passo 1: Abrir o terminal do Linux.

Passo 2: digitar o comando abaixo para caso você tenha instalado o OpenJDK ele ser desinstalado.

sudo apt purge openjdk*

Passo 3: Utilizar o comando para abrir um app que possui 4 opções de versões do pacote Java.

sudo add-apt-repository ppa:linuxuprising/java && sudo apt update

- oracle-java11-installer-local
- oracle-java11-set-default-local
- oracle-java14-installer
- oracle-java14-set-default

Passo 4: comando para instalação. Utilizaremos o pacote oracle-java14-installer que é o recomendável a ser utilizado.

sudo apt install oracle-java14-installer

1.3.2 Preparando ambiente de programação (eclipse)

1.3.2.1 Instalação Eclipse IDE no Windows.

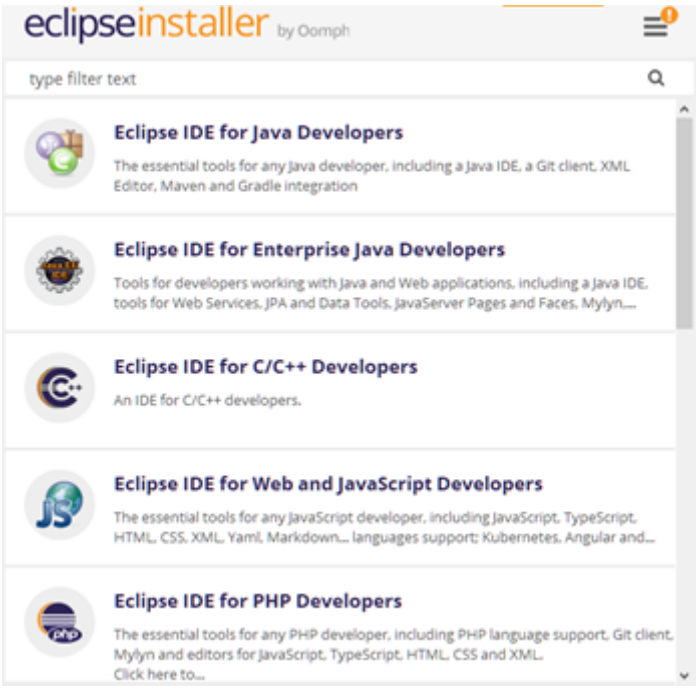
Vamos trazer o passo a passo da Instalação do Eclipse IDE no ambiente Windows.

Passo 1: acesse a página de download do Eclipse IDE no site do Eclipse. Clique sobre o botão “Download 64 bit”. Caso seu sistema operacional não seja 64 bits o site irá carregar o link para a versão 32 bits.

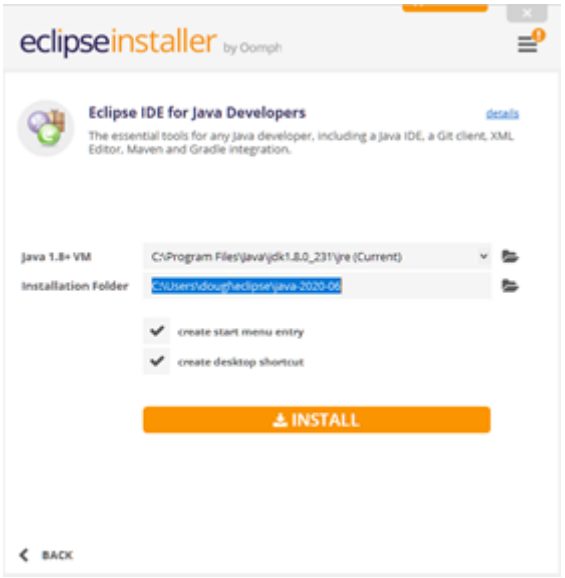
Passo 2: Aguarde o download do arquivo e clique sobre ele, após a finalização, para executar e começar a instalação.

Dentro de alguns instantes para o instalador abrir, surgirá a seguinte tela para escolha da plataforma a ser instalada.

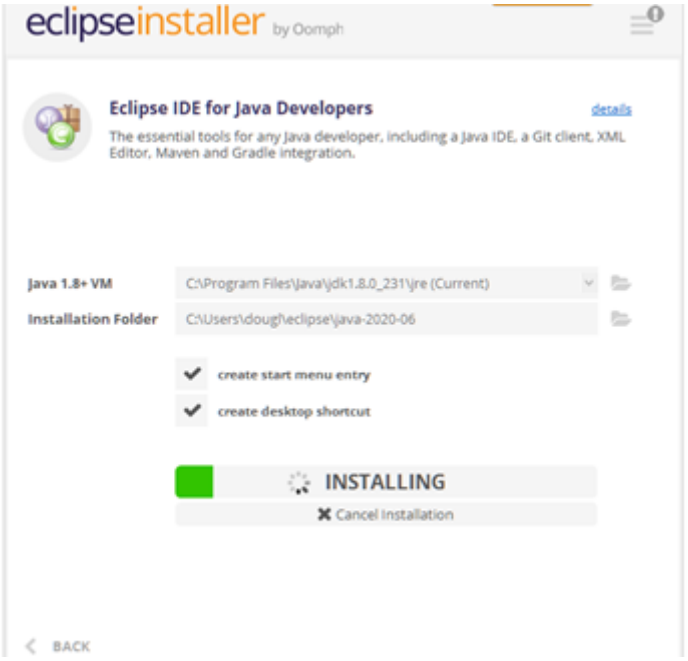
Utilizaremos a opção Eclipse IDE for Java Developers.



Passo 3: Como foi escolhida a opção para desenvolvimento Java, a instalação do Eclipse irá procurar automaticamente pela pasta do JDK (Java Development Kit) o qual contém as ferramentas necessárias para o desenvolvimento de programas na linguagem Java. Deixar as pastas e caixas de seleção todas como estão e clique em “Install”.



Passo 4: Aguarde a instalação acabar.



Em um dado momento será requisitado à confirmação da licença de uso marque a caixa "Remember accepted licenses" e clique em "Accept" para continuar. Caso apareça alguma outra licença, proceda da mesma forma.

Por fim, o Eclipse irá terminar com a mensagem de sucesso na instalação do programa.

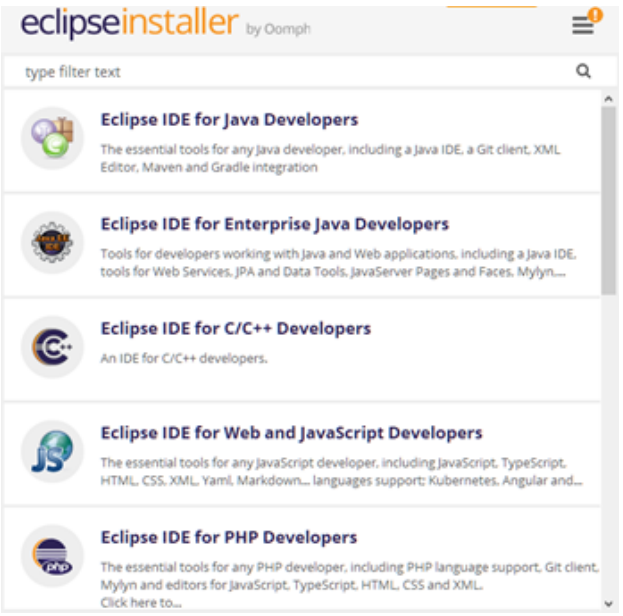
1.3.2.2 Instalação Eclipse IDE no Linux.

De maneira semelhante ao Windows iremos fazer a instalação do Eclipse IDE no ambiente Linux.

Passo 1: acesse a página de download do Eclipse IDE no site do Eclipse escolher o sistema operacional Linux. Muitas vezes ele já o reconheceu, basta realizar o download.

Passo 2: Depois de realizado o download, iremos ao diretório do download (pasta) e descompactaremos o arquivo eclipse-inst-linux64.tar.gz.

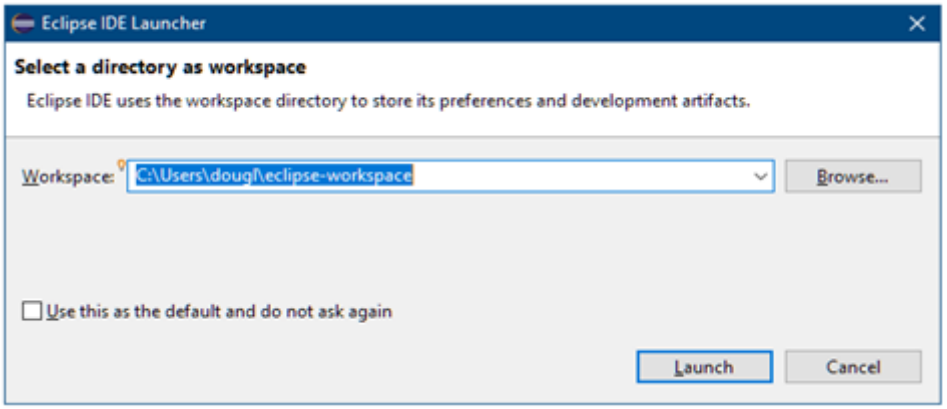
Depois desta tarefa você terá um diretório chamado eclipse-installer. Entre nele e dê um clique duplo no arquivo instalador eclipse-inst. A seguinte janela será aberta.



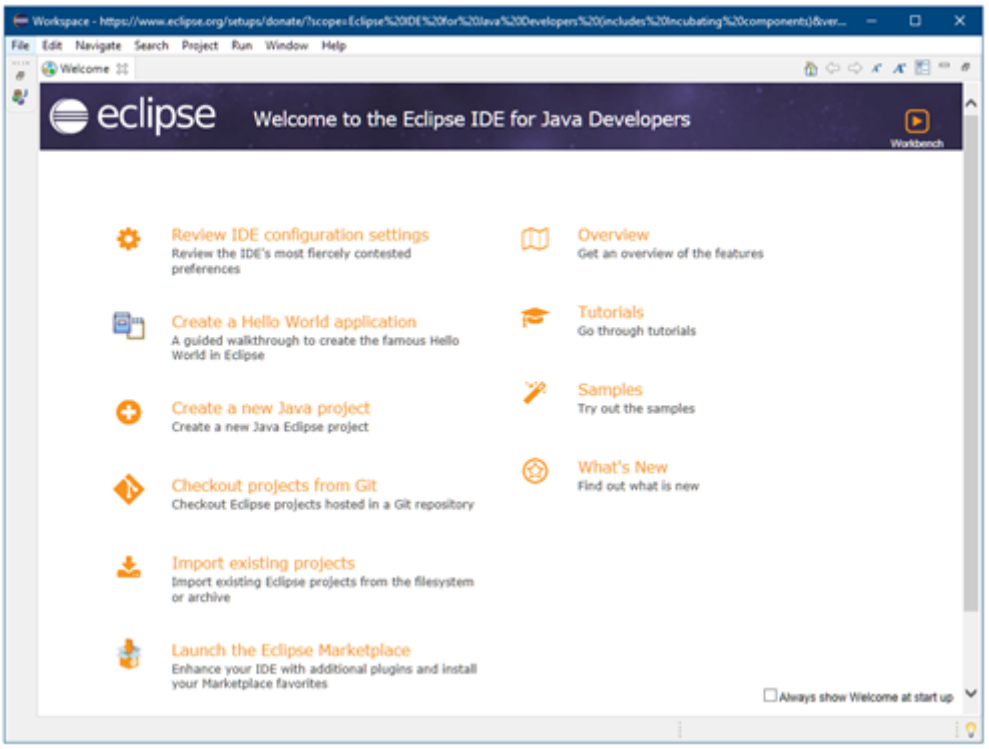
Passo 3: A partir daí é só seguir o mesmo passo a passo da instalação no ambiente Windows.

1.3.2.3 Abertura do Eclipse IDE.

Ao clicar e abrir o Eclipse pela primeira vez será preciso escolher uma pasta para o Workspace, ou seja, o ambiente/pasta de trabalho onde os projetos serão criados e armazenados. Escolha uma pasta que desejar ou deixe a pasta padrão. Após a escolha clique em Launch.

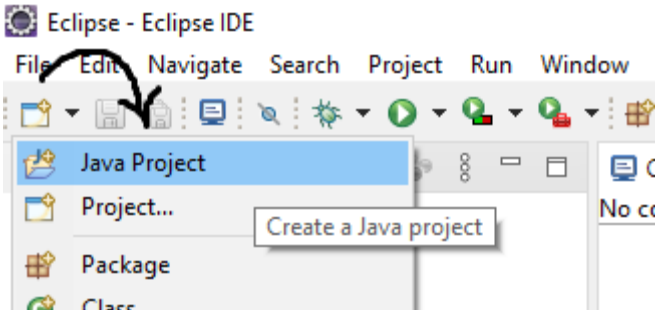


Após a inicialização do Eclipse, será exibida uma tela de boas vindas.

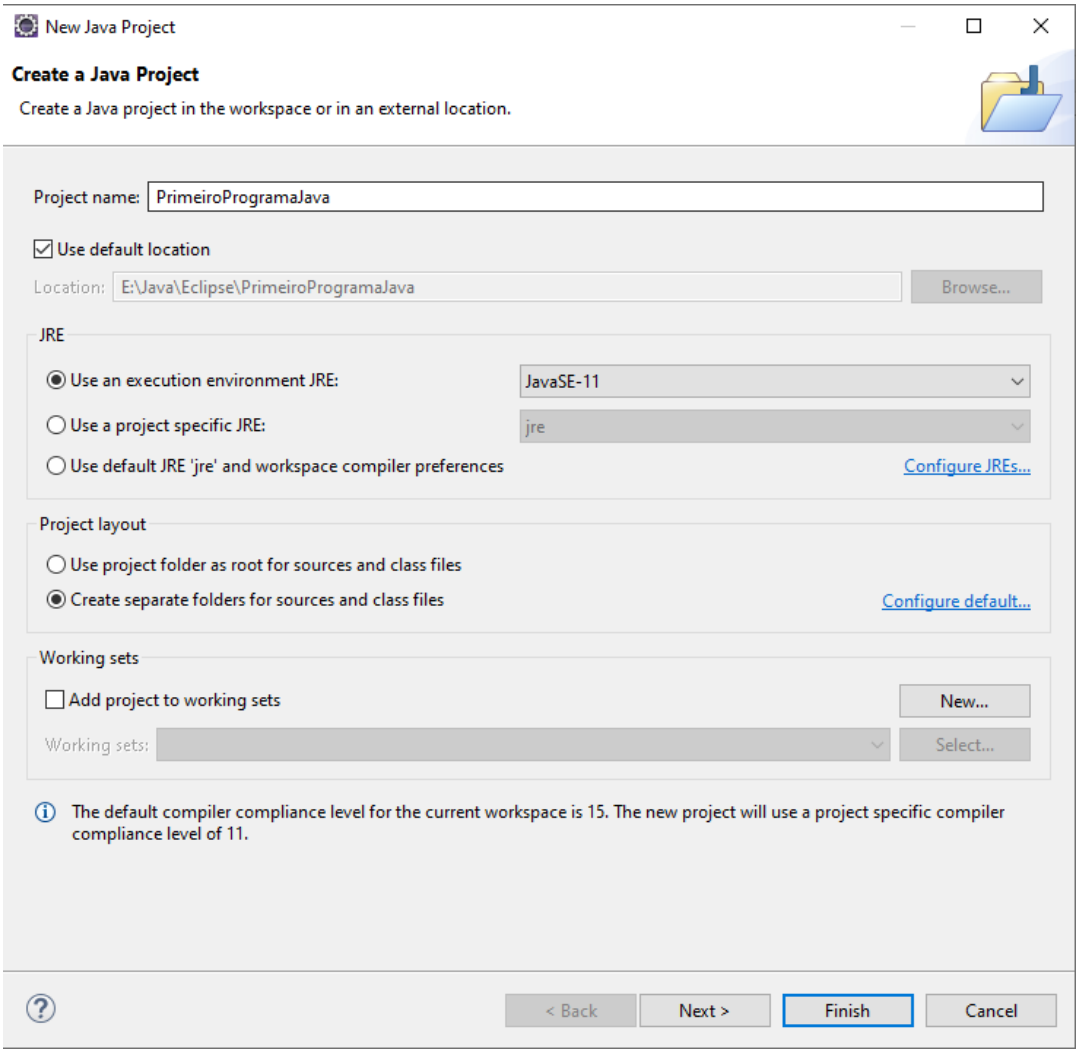


1.3.3 Hello World

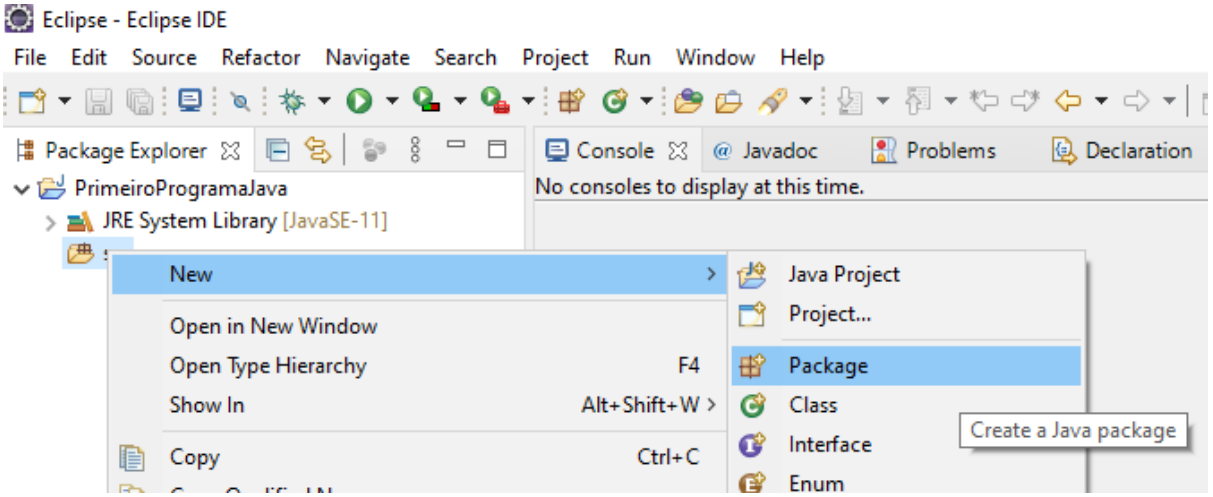
Agora que você instalou o Java e Eclipse e está com seu ambiente pronto, você deve abrir um novo projeto para começar a programar:



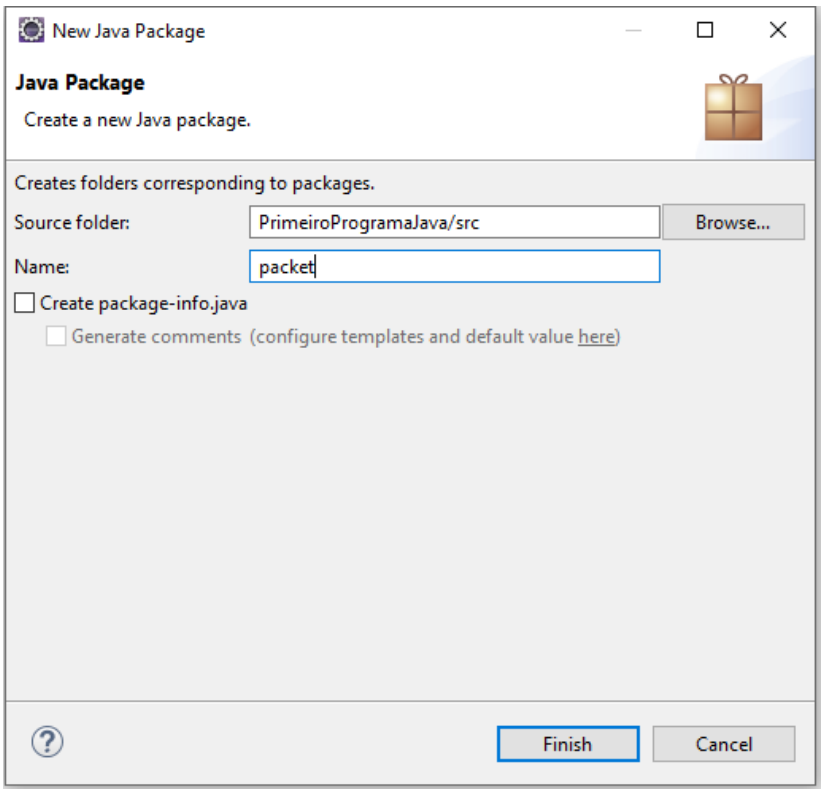
Em seguida deve abrir uma nova janela, coloque o nome de seu projeto, verifique se ele está usando corretamente a versão do Java que você instalou e finalize.



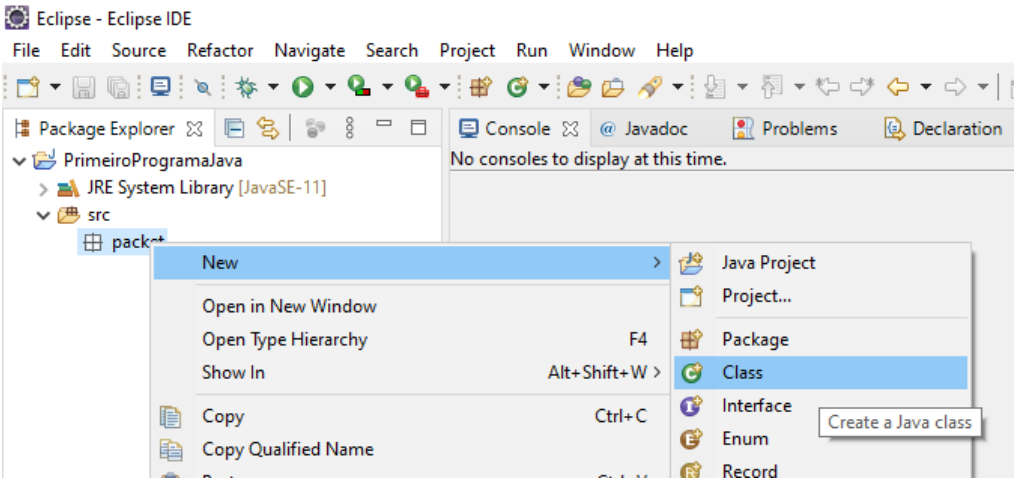
Agora que está com seu projeto criado, precisamos criar um novo pacote dentro dele:



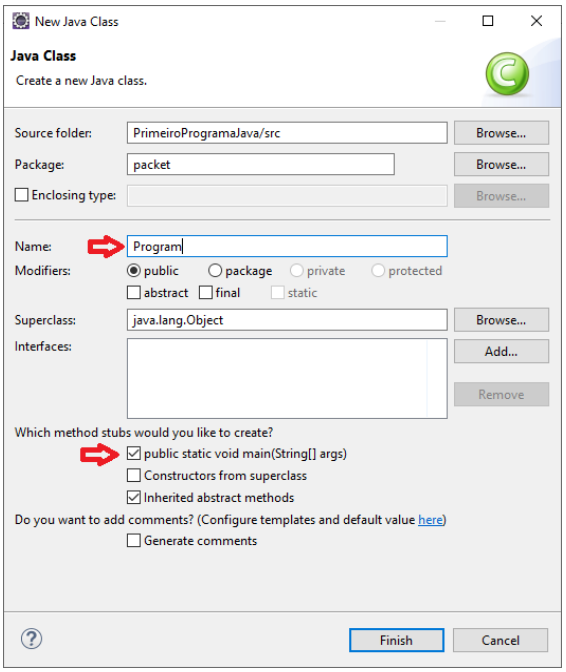
Dê um nome ao seu pacote, lembre-se, ele deve começar com letra minúscula:

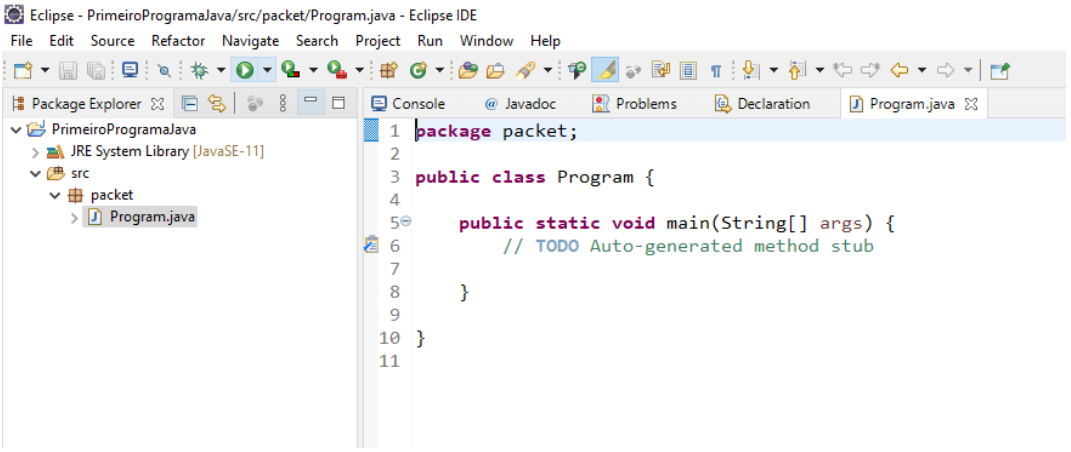


Após a criação do pacote, agora dentro dele você deve criar uma classe:



Dê um nome a sua classe, esta deve começar com letra maiúscula, após colocar o nome, você pode habilitar a caixa marcada na imagem, ela irá criar um método main, este que iremos usar para programar dentro.





Parabéns, quase terminando por aqui, você tem um novo projeto, um pacote e uma classe Program com o método main, vamos testar para ver se tudo está de acordo e funcionando, escreva isso dentro do método main:

System.out.println("Hello World");

Depois vá na hotbar e aperte nesse botão destacado para compilar seu código:

Se tudo ocorreu bem, no console deve ter impresso a frase "Hello World".

Capítulo 2

Variáveis:

Para começarmos a trabalhar com a linguagem java, precisamos criar variáveis, estas que assumem um valor qualquer e podem facilmente ser manipulados conforme a necessidade do programador, ou se estiver criando um sistema que funcione de forma mais autônoma, os dados passaram por esta variável para que possam ser computados de forma automática.

Uma variável pode funcionar da seguinte forma, você a declara com um nome qualquer, no entanto tome cuidado para não escolher nenhum nome reservado pela linguagem de programação, ou que comece com um número, exemplo “12doze”, deste modo a linguagem não conseguirá identificar como uma variável válida.

2.1 Int:

Em java, as variáveis têm tipos, estes que tem uma função específica, um dos tipos mais comuns utilizados é “int”, o inteiro, usado para armazenar números como 1 ou 2, ou outro qualquer, desde que seja inteiro.

Exemplo:

int numero_inteiro = 10;

Neste exemplo podemos observar que o int no começo do código serve para indicar a máquina que tipo de valor ela deve guardar [1], em seguida temos o nome da variável para que possamos usar posteriormente [2], o sinal de igual aqui, tem uma função um pouco diferente do igual que conhecemos na matemática, este aqui serve para atribuir um valor qualquer para a variável [3] que neste caso é o valor de número 10 [4], e para fechar a linha de código devemos colocar um ; (ponto e vírgula) [5]. Vale ressaltar que após atribuir o valor 10 para a variável “numero_inteiro” está agora é exatamente igual ao valor 10, no entanto diferente de 10 que é um valor absoluto, a variável pode ser mudada assim quando necessário:

numero_inteiro = 20;

Veja que desta vez não foi necessário colocar int antes do nome da variável, uma vez que estamos a reutilizando e não criando novamente, fato esse que se você tentar utilizar int, o compilador vai acusar um erro no terminal.

2.2 Long:

Essa variável ou tipo é bastante semelhante ao int, no entanto engloba uma gama de valores bem maior, pois diferente do int que reserva na memória um espaço de 4 bytes, o long reserva uma quantia de 8 bytes, o dobro do int.

Exemplo:

long numero_long = 9223372036854775807;

2.3 Outros tipos para valores inteiros

Aqui você pode observar nesta tabela, esta que dispõe todos os tipos de números inteiros existentes em Java:

Tipos	Memória utilizada	Valor Mínimo	Valor Máximo
byte	1 Byte	-128	127
short	2 Bytes	-32.768	32.767
int	4 Bytes	-2.147.483.648	2.147.483.647
long	8 Bytes	-9.223.372.036.854.775.808	9.223.372.036.854.775.807

2.4 Float:

Este tipo, diferente dos anteriormente citados, é diferente, além da capacidade de atribuir valores numéricos inteiros, podemos colocar valores fracionados em casas decimais, por exemplo, o valor 0,5 ou 6,73.

Exemplo:

float pi = 3,141592;

Conforme houver a necessidade de trabalhar com números fracionados, o tipo float é uma boa pedida.

2.5 Double:

Este tipo, assim como long é para o int, é uma versão que abrange uma gama maior de valores, onde o float possui 4 bytes e double possui 8 bytes.
Exemplo:

double pi = 3,141592653589793238 46;

Tabela de valores dos tipos fracionais:

Tipos	Memória Utilizada	Valor Mínimo	Valor Máximo	Precisão
float	4 Bytes	-3,4028 E + 38	3,4028 E + 38	6 − 7 dígitos
double	8 Bytes	-1,7976 E + 308	1,7976 E + 308	15 dígitos

2.6 Char:

Diferente das outras variáveis já explicadas, essa é bastante diferente, não armazena inteiros e nem números fracionados, e então, o que ela armazena? assim como números são muito importantes enquanto você programa, letras, isso mesmo, letras também tem seu grau de importância na hora de programar, como por exemplo na hora de comparar com outras letras para verificar se a senha está correta, no entanto, char só pode armazenar uma letra por vez, isso é bom quando você for criar estruturas e precisa de uma chave de valor único que não seja passível de fácil mudança como números.

Exemplo:

char chave = “a”;

Veja que diferente de atribuir um valor numérico, para atribuir uma letra ou um caractere como preferir, é necessário colocá-lo entre aspas para que a sintaxe da linguagem o possa reconhecer sem que dê algum tipo de erro. Uma curiosidade, char vem de caractere, por isso armazena apenas um caracter.

2.7 String:

Assim como char, este não armazena números inteiros ou fracionários, no entanto, diferente de char que pode apenas possuir apenas um caractere, o tipo String que vem de linha, literalmente, pode armazenar uma frase, ou até um parágrafo inteiro sem maiores problemas.

Exemplo:

String frase = "Esta é uma frase armazenada no tipo String";

String paragrafo = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."; [1].

Da mesma forma que char precisa de aspas duplas para inserir um valor, String funciona da mesma forma.

[1] ***Lorem Ipsum*** *é simplesmente uma simulação de texto da indústria tipográfica e de impressos, e vem sendo utilizado desde o século XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu não só a cinco séculos, como também ao salto para a editoração eletrônica, permanecendo essencialmente inalterado. Se popularizou na década de 60, quando a Letraset lançou decalques contendo passagens de Lorem Ipsum, e mais recentemente quando passou a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.*

2.8 Boolean:

Este é o tipo mais diferente de todos, este não armazena nem números ou números fracionários, e também não armazena caracteres ou frases inteiras, no entanto, pode se dizer que em termos de programação, é um dos tipos mais importantes, por que ele armazena um estado. Na computação conhecemos os números binários quase que por obrigação, já que se tratando de baixo nível, toda a programação é feita com eles, estes que são 0 ou 1, zero sendo nível lógico baixo e 1 sendo nível lógico alto. Neste caso, o tipo sendo boolean vem do nome do criador da álgebra booleana, George Boole, sendo ela que manipula números binários.

O tipo boolean armazena o estado de “false” ou “true”, sendo false o equivalente de 0 e true sendo o de 1 respectivamente.

Exemplo:

boolean george_boole = true;

boolean george_boole = false;

Veja que neste exemplo, não é necessário escrever true ou false entre aspas duplas, uma vez que essas duas palavras estão reservadas pelo sistema e muito provavelmente seu IDE vá destacá-las. Importante também ressaltar que este tipo será melhor abordado nos tópicos de operadores lógicos.

2.9 Outros tipos de Variáveis

Importante ressaltar que, a maioria destes tipos mostrados com exceção de String (único com letra maiúscula) são do tipo primitivo, a forma bruta e sem refinamento algum. Tendo isso em vista, podemos falar dos tipos “Wrapper”, que para cada tipo primitivo existe um “Wrap”.

Para os inteiros temos:

Primitivos	**Wrapper**
byte	Byte
short	Short
int	Integer
long	Long

É notória que na maioria, pelo menos visivelmente a diferença é apenas uma letra maiúscula, com exceção de int -> Integer. No entanto existe diferença prática, uma vez que tipos wrapper herdam do objeto “Object”, formato fundamental de objeto na linguagem Java, este que confere várias características e funções extras para as variáveis.

Exemplo:

int variavel_inteira = Integer.parseInt(“430”); //transforma string em inteiro

No entanto, por se tratar de um conteúdo para iniciantes não iremos abordar orientação a objetos neste documento.

Para fracionários temos:

Primitivos	**Wrapper**

float	Float
double	Double

Para caractere temos:

Primitivo	**Wrapper**
char	Char

Para booleano temos:

Primitivo	**Wrapper**
boolean	Boolean

Capítulo 3

Operações

3.1 - Manipulando variáveis com operadores

Os operadores aritméticos realizam as operações fundamentais da matemática entre duas variáveis e retornam o resultado. Assim nos possibilitando executar todo tipo de cálculo possível. Caso seja necessário escrever operações maiores ou mais complexas, podemos combinar esses operadores e criar expressões.

Tipo de operadores aritméticos:

+	Operador de adição
-	Operador de subtração
*	Operador de multiplicação
/	Operador de divisão
%	Operador de módulo (ou resto da divisão)

Exemplo de uso de operadores aritméticos:

int soma = 2 + 2; Somando um número com o outro.

int subtração = 2 - 2; subtraindo um número com o outro.

int mult = 2 * 2; multiplicando um número com o outro.

int divisão = 2 / 2; dividindo um número pelo o outro.

Temos outra forma para podermos demonstrar o cálculo de alguma operação utilizando operadores aritméticos em conjunto com o operador de atribuição. Vejamos abaixo:

int mult = 2;

mult *= 2; multiplicando um número 2 com o outro 2.

Temos também os operadores de incremento e decremento que são bastante utilizados. Eles basicamente incrementam ou decrescem em 1 o valor da variável.

++ Incrementa

-- Decrementa

Exemplo de uso de operadores de incremento e decremento:

int numero = 10;

número ++; Incrementando

número --; Decrementado

O valor continuará sendo o número 10, pois tivemos um incremento de 1 e depois decrementamos 1 novamente.

Capítulo 4

Estrutura Condicional

Até agora vimos que variáveis devem ter tipos, e como manipulá-las com operações (operadores) matemáticas:

```
Integer valor = 1;
valor++;
System.out.println(valor);

Terminal:
2
```

Ou até mesmo na hora de concatenar uma String:

```
String frase = “Concatenando”;
frase = frase + “ frase”;
System.out.println(frase);

Terminal:
Concatenando frase
```

No entanto, se quisermos fazer um programa que tenha outras funções além de armazenar valores dentro de variáveis e somá-las com outros valores ou variáveis precisamos avançar. Lembra-se do tipo de variável “boolean”? Esta que armazena valores do tipo “true” ou “false”, esse tipo de variável será bastante importante neste tópico.

Imagine a seguinte situação, você precisa entrar em uma sala, no entanto não possui as credenciais necessárias para entrar nela, logo será barrado e terá o acesso negado, as estruturas condicionais funcionam da mesma forma.

4.1 Bloco If/Else:

Como dito anteriormente, no exemplo da sala, as condicionais podem ser escritas desta forma no código:

```
String senha = “correta”;

if(senha == “correta”){
        System.out.println(“Acesso permitido”);
```

}else{System.out.println(“Acesso negado”);}

Veja que neste exemplo, nós criamos uma variável do tipo String para armazenar “correta”, neste exemplo, esta é a nossa senha, em seguida iniciamos o bloco if/else, primeiro vem o if (se) para indicar que vamos testar uma condição, em seguida abrimos um parênteses para colocarmos a condição, a condição é a seguinte, ele testa se o conteúdo dentro da variavel “senha” é igual (==, operador de igualdade) a String “correta”, se for igual realmente a condição nos levará a primeira opção dentro da primeira chave, esta irá “printar” a mensagem “Acesso permitido” indicando que a condição testada é “true”, nesse caso, verdadeira.

```
String senha = “incorreta”;

if(senha == “correta”){
        System.out.println(“Acesso permitido”);
}else{
        System.out.println(“Acesso negado”);
}
```

Nesse caso, se senha for diferente de correta, neste caso como a variável “senha” tem “incorreta armazenado", a condicional logo identifica que a sentença é “false”, ou seja, que é falsa e pula a primeira etapa e vai direto para a segunda. No bloco else (se não), já que a condição principal não foi atendida ele irá executar o outro bloco, assim o terminal irá “printar” “Acesso negado”.

Outra situação que pode acontecer, é você querer testar mais de uma condição no mesmo bloco if/else:

```
Integer valor = 10;

if(valor == 10){
        System.out.println(“Valor igual a 10”);
}else if(valor > 10){
        System.out.println(“Valor maior que 10”);
}else{
        System.out.println(“Valor menor que 10”);
}
```

Neste exemplo vemos que se o valor da variável for diferente de 10, caso da primeira condicional, ele em seguida irá testar a segunda condicional, se o valor for maior que (>, operador de maior que) 10, ele entrará nesta condição, se não for igual a 10, ou maior que 10, nesse caso só pode ser menor que 10, não precisamos explicitar isso no código, o programa entenderá e logo entrará na condição.

4.2 Bloco Switch Case:

Se você quiser testar várias condições em opções e tiver um padrão já pré-estabelecido, é bom utilizar o bloco switch case:

```
char key = “a”;
switch (key){
        case “a”:
                System.out.println(“Opção 1”);
                break;
        case “b”:
                System.out.println(“Opção 2”);
                break;
        default:
                System.out.println(“Opção inválida”);
}
```

Neste bloco vemos uma variável do tipo char chamada key, ela será nossa chave para acessar os casos dentro do bloco. Começamos o bloco escrevendo switch , em seguida abrimos parênteses para colocar a condição ou melhor, a chave para acessar as opções, abrimos então as chaves para irmos aos casos, cada “case” terá uma chave específica para acessá-lo, no primeiro caso a chave deve ser “a”, então iremos para seu bloco, você pode colocar o que precisar, no exemplo colocamos um print dizendo que você acessou a “opção 1”, se neste exemplo key fosse igual “b”, iríamos acessar a opção 2, caso a key tivesse um valor que não está em nenhum dos casos dentro do bloco, automaticamente cairá na condição default (é obrigatório no bloco).

4.3 Outro tipo de condicional:

Se você quiser utilizar algo menos verboso, talvez o if ternário seja uma boa opção, veja:

```
String senha = “correta”;
```

(senha == “correta”) ? Código 1 : Código 2;

Nesse if colocamos a condicional entre parênteses antes [1], após a condicional temos uma “?” (interrogação), imagine como se estivesse perguntando algo, esta condição é “true”? [2] se sim, o Código 1 é executado [3], após isso temos “:” (dois pontos) para separar da condição dois, caso esta seja “false”, então executa o Código 2 [4].

4.4 Operadores Lógicos

Agora veremos os operadores de igualdade que fazem a verificação se o valor ou o resultado da expressão lógica à esquerda é igual (“==”) ou diferente (“!=”) ao da direita, retornando um valor booleano.

==	Utilizamos quando desejamos verificar se uma variável é igual a outra.
!=	Utilizamos quando desejamos verificar se uma variável é diferente de outra.

Exemplo de uso de operadores de igualdade:

```
int valor1 = 1;
int valor2 = 2;
if(valor1 == valor2){
        System.out.println(“Valores iguais”);
} else {
        System.out.println(“Valores diferentes”);
}
```

Os operadores relacionais, parecidos com os de igualdade, avaliam dois operandos. Definem se o operando à esquerda é menor, menor ou igual, maior ou maior ou igual ao da direita, retornando um valor booleano.

>	verifica se uma variável é maior que outra.
>=	verifica se uma variável é maior ou igual a outra.
<	verifica se uma variável é menor que outra.
<=	verifica se uma variável é menor ou igual a outra.

Exemplo de uso de operadores relacionais:

```
if (valor1 > valor2) {
        System.out.println(“maior”);}
if (valor1 >= valor2) {
        System.out.println(“maior ou igual”);}
if (valor1 < valor2) {
        System.out.println(“menor”);}
if (valor1 <= valor2) {
        System.out.println(“menor ou igual”);}
```

Temos os operadores lógicos que representam um jeito que nos permite criar expressões lógicas maiores a partir da junção de duas ou mais expressões. Utilizamos as operações lógicas E (representado por “&&”) e OU (representado por “||”).

&&	Utilizado quando desejamos que as duas expressões sejam verdadeiras.
||	Utilizado quando precisamos que pelo menos uma das expressões seja verdadeira.

Exemplo de uso de operadores lógicos:

```
if((5 == (6 -1)) && (8 == (4 + 4))){
    System.out.println(“Ambas as expressões são verdadeiras”);
}
```

Capítulo 5

Laços de repetição

Como foi visto anteriormente podemos manipular variáveis através de operações matemáticas e operações condicionais. Uma outra função presente na programação é a repetição de determinado bloco de execução.

Na programação, é chamado de laço ou loop, um sistema que repete instruções por uma quantidade determinada ou não, de vezes. A linguagem Java oferece algumas dessas instruções, as que serão abordadas são, o for, o while e o foreach.

5.1 - For

O for tem uma declaração relativamente fácil, mas necessita da determinação da quantidade de vezes que deverá ser repetido. Como podemos ver a abaixo como funciona a declaração de um laço for:

for(Iniciação da variável; condição da repetição; incremento da variável;){Corpo do código; };

Por exemplo:

```
public static void main(String[] args) {

    for(int i=0;i<10;i++) {

        System.out.println("Numero: "+i);

    }

}
```

E o resultado da execução será:

```
Numero: 0
Numero: 1
Numero: 2
Numero: 3
Numero: 4
Numero: 5
Numero: 6
Numero: 7
Numero: 8
Numero: 9
```

A sequência de execução que ocorrerá no laço for pode ser dividido em algumas etapas:

1 - Inicia e atribui um valor a variável(int i = 0).

2 - Verifica se a condição de repetição foi satisfeita.

3.1 - se a variável for menor que o valor determinado(i<10).

4 - Executa o corpo do código(System.out.println("Numero: "+i)).

5 - executa a iteração da variável(i++).

6 - volta ao paço 2.

3.2 - se a variável não for menor que o valor determinado(i<10) - **FIM.**

5.2 - While

A forma de execução do while se assemelha a do for, mas a sua estrutura é o que os diferencia. O while também executa o código enquanto a condição expressa for true.

Iniciação da variável;

while(condição da repetição;){

Corpo do código;

interação da variável;

}

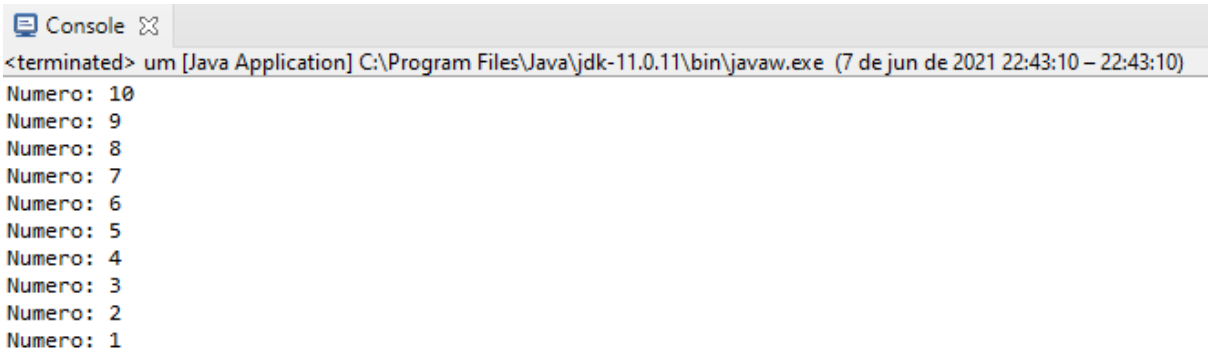
Por exemplo:

```
public static void main(String[] args) {
    int i=10;

    while(i>0) {
        System.out.println("Numero: "+i);
        i--;
    }

}
```

E o resultado da execução será:



O programa anterior se equipara em etapas ao for, ele irá verificar se o valor da variável é maior que zero. Caso seja, mostrará a mensagem na tela, e decrementa seu valor. E realiza isso repetidas vezes até que o valor seja menor ou igual a zero.

5.3 - Do-While

O laço Do-while funciona de maneira análoga ao while usado anteriormente, as diferenças entre eles estão na sua estrutura de declaração e na sua forma de execução, já que o do-while sempre executa o bloco uma vez antes de realizar a verificação.

A diferença para os outros, é que o bloco de instrução será executado no mínimo uma única vez. Como podemos ver no exemplo abaixo:

Iniciação da variável;
do{
    Corpo do código;
    interação da variável;
}while(condição da repetição);

por exemplo:

```
    public static void main(String[] args) {
        int i=10;
        do {

            System.out.println("Numero: "+i);
            i--;
        }
        while(i>0);

    }
```

O programa acima apresenta o mesmo resultado do anterior.

Capítulo 6

Arrays e listas:

6.1 - Array

Os **arrays ou matrizes (arrays bidimensionais), como são conhecidos pelo** Java, fazem parte do pacote java.util na coleção da API do Java. São objetos de recipientes que contém um número fixo de valores de um único tipo.

O comprimento de um **array é estabelecido quando criado** o tamanho de um array sempre deve ser um valor int, e nunca um long, short ou byte**, sendo que após a criação o seu comprimento fica fixo.** Os elementos em um array são indexados a partir de zero.

Aqui você pode observar um exemplo de array de 5 elementos.

5	6	7	8	9

.0 1 2 3 4

Declaramos um array de uma dimensão (“vetor”) da seguinte forma abaixo:

tipo nomeArray[ ];

ou tipo[ ] nomeArray;

Na parte do tipo fazemos a escolha de qual será o tipo de dados de todos os elementos que serão armazenados nas posições do array. Na parte do nomeArray colocaremos o nome que queremos que o array tenha.

double salarios[ ];

Depois temos que de fato efetivamente alocar espaço na memória para o array, onde tamanho se refere ao número de elementos do array, e nomeArray é o nome da variável que será ligada ao array físico. Quando usamos o operador **new** devemos especificar o número de elementos que serão alocados no array.

nomeArray = new tipoDado[tamanho];

Contudo, temos, que para criar um array em Java é um processo em duas etapas: Declaramos uma variável do tipo desejado para o array, e então alocamos memória para efetivamente armazenar o array, atribuindo-a à variável criada.

Na declaração de um array, cada elemento recebe um valor padrão, sendo 0 para números de tipo primitivo, falso (false) para elementos booleanos e vazio (null) para referências.

Vamos ver abaixo como declarar e inserir valores em um Array.

int[ ]nomeArray = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110};

Imprimir todos os valores presentes no Array utilizando laço de repetição.

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

        System.out.printf("%5d %4s %4d \n", i, "=>" , nomeArray [ i ]);

        }
```

Arrays multidimensionais.

Esse tipo de array é declarado como tendo duas dimensões e é usado para representar tabelas de valores que consistem em informações organizadas em linhas e colunas.

Os arrays bidimensionais precisam de dois índices para identificar um elemento particular. Vejamos abaixo:

int[ ][ ] nomeArray = { { 1,2,3 }, { 4, 5, 6 } };à Declarando um Array bidimensional de 3x3.

Consultar valores no Array bidimensional através de laço de repetição.

```
for(int linha = 0; linha < nomeArray.length; linha++)

    {       //FAZ LOOP PELAS COLUNAS DA LINHA ATUAL

    for( int coluna = 0; coluna < nomeArray [linha].length; coluna++)

        System.out.printf("%d ", nomeArray [linha][coluna]);
```

```
 System.out.println();

}

}
```

6.2 - Listas

Temos o List que é uma interface de Collection Framework da Java. Com ela podemos criar listas, que armazena dados, esses dados podendo ser elementos duplicados. Os índices de List são baseados em zero, isto é, o índice do primeiro elemento é zero. Declaramos da maneira abaixo:

List Nomelista;

6.3 - ArrayList

Agora veremos ArrayList que é a classe que implementa a interface List, ele tem características parecida com uma lista, principalmente de manter os dados inseridos ordenados e acessá-los através de índices. Conforme inserimos elementos na ArrayList, ela vai crescendo, não sendo necessário informar o seu tamanho na hora de criação, como ocorria com vetores e matrizes.

Podemos inserir, recuperar, remover, consultar e iterar sobre uma ArrayList através dos seus métodos. Vejamos abaixo alguns exemplos:

Declarando um ArrayList.

ArrayList< tipo_de_dados > NomeArrayList = new ArrayList();

O texto tipo_de_dados se refere ao tipo que deseja que seu ArrayList tenha, podendo ser Double, String, Int e etc.. Já o Nome ArrayList é qualquer nome que queira dar ao seu ArrayList.

Inserindo dados num ArrayList.

NomeArrayList.add("X "); -> X pode ser qualquer coisa que você desejar adicionar ao ArrayList.

Consultar dados do ArrayList utilizando laço de repetição.

```
int n = NomeArrayList.size();  à Mostra o tamanho do ArrayList.

for (i=0; i<n; i++) {   à Percorre o ArrayList todo.

System.out.printf("Posição %d- %s\n", i, NomeArrayList.get(i));

}
```

Removendo dados do ArrayList.

agenda.remove(Y); à Y representa o número do índice que deseja remover do ArrayList.

Capítulo 7

Funções

7.1 Criando Funções

Agora que você já viu como funcionam variáveis, como manipular elas com operações matemáticas, como criar condicionais e laços de repetição para manipular matrizes e listas, está na hora de dar um passo adiante.

Funções funcionam da seguinte forma, se você quiser economizar linhas de código e sempre precisar usar a mesma lógica, não há necessidade de repeti-las, você pode criar uma função da seguinte forma:

```
public static void main(String[] args) {
        int resultado = soma(5, 6);
        System.out.println(resultado);
        resultado = soma(10, 10);
        System.out.println(resultado);
}
public static int soma(int a, int b) {
        return a + b;
}
Console:
11
20
```

Veja na linha “int resultado = soma(5, 6);” chamamos a função soma descrita logo a baixo:

```
public static int soma(int a, int b) {
        return a + b;
}
```

Para declarar esta função devemos sair da função main, declarar logo acima ou abaixo, escrevemos “public static” e logo em seguida dizemos que tipo de variável desejamos que o programa retorne, no exemplo usamos o “int”, no entanto poderia ser qualquer tipo, String, double ou void (esse em específico é usado quando queremos que a função não retorna nenhum valor, logo veremos um exemplo), após isso nos damos o nome que queremos a função, observe que o nome assim como variáveis não devem usar letra maiúscula ou números no começo, depois de dar nome a sua função, de preferência que ela descreva de forma sucinta o que sua função vai realizar, abrimos parênteses, dentro dos parênteses vem o que chamamos de parâmetro da função, são os valores que ela vai receber, neste exemplo como estamos trabalhando com “int” estamos recebendo este tipo também no parâmetro.

(int a, int b)

No caso, estamos recebendo dois valores, um “a” e outro “b” do tipo inteiro, dentro do corpo desta função realizamos a soma de a e b (a+b) e retornamos seu resultado usando a expressão “return”.

return a + b;

Depois de descrever toda a lógica que sua função irá realizar podemos fechar as chaves. Na “main” onde chamamos anteriormente a função usamos alguma variável para capturar seu resultado.

int resultado = soma(5, 6);

Talvez você pense que utilizar função para algo tão trivial quanto somar dois valores seja perda de tempo, e realmente é, no entanto para problemas mais complexos as funções se mostram extremamente úteis.

Exemplo:

```
public static void main(String[] args) {
        String result = concatenar_varias_vezes("Java", true, 5);
        System.out.println(result);
```

```
    }
    public static String concatenar_varias_vezes(String a, boolean b, int
qtd) {
        String result = a;
        if (b) {
            for (int i = 0; i < qtd-1; i++) {
                result = result + a;
            }
        }
        return result;
    }
```

Console:

JavaJavaJavaJavaJava

Esse exemplo serve apenas para mostrar que você pode incorporar várias coisas dentro de uma mesma função, na passagem de parâmetro usamos String boolean e int para engrenar a função e no final retornamos uma String.

Capítulo 8

Referências

[1] *Kit para desenvolvimento de aplicativos Java montado pelos criadores da linguagem de programação. Acesso em:* https://www.oracle.com/br/java/technologies/ -

[2] *Tutorial da linguagem Java. Acesso em:* https://docs.oracle.com/javase/tutorial/index.html -

[3] *Links para documentações. Acesso em:* https://oracle.com/java/technologies/javase/javase-tech-database.html

[4] *Lista de drivers JDBC. Acesso em:* http://industry.java.sun.com/products/jdbc/drivers

[5] *Revista sobre a linguagem java. Acesso em:* https://www.infoworld.com/category/java/

[6] *Página de Download do eclipse. Acesso em:* https://www.eclipse.org/downloads/

[7] *Tutorial de instalação por terminal no linux. Acesso em:* https://www.howtogeek.com/140679/beginner-geek-how-to-start-using-the-linux-terminal/

[8] *Diferenças entre Java. Acesso em:* https://dicasdejava.com.br/qual-a-diferenca-entre-jdk-jre-e-jvm/

[9] *Link de download do Java. Acesso em:* https://www.oracle.com/java/technologies/javase-downloads.html

[10] *Tutorial de instalação do Java no Linux. Acesso em:* https://www.edivaldobrito.com.br/como-instalar-o-java-no-ubuntu-20-04-lts-e-derivados/