**Preparando o ambiente: instalação do Java e do IntelliJ**

O **JDK** (Java Development Kit) é um conjunto de ferramentas e bibliotecas de software que permite o desenvolvimento de aplicativos Java. Nesse preparando o ambiente vamos instalar o JDK para começarmos a desenvolver nossos primeiros programas em Java!

**Linux**

1) No terminal, vamos executar o seguinte comando para atualizar a lista de pacotes disponíveis para download nos repositórios do sistema:

sudo apt **update**

2) Feito isso, agora vamos instalar a versão 17 do JDK através do comando:

sudo apt install openjdk-17-jdk

3) Dessa forma, o sistema irá baixar e instalar o pacote OpenJDK 17 JDK, permitindo que o usuário comece a desenvolver aplicativos Java no Linux. Terminado esse processo de instalação, agora precisamos configurar a variável de ambiente **JAVA\_HOME**, utilizada para indicar o caminho de instalação do JDK. Em uma tradução literal seria o “lar do java” e ela será necessária para utilizarmos os recursos do JDK como o **javac**.

Para isso, iremos executar o comando ***sudo update-alternatives --config java*** que mostrará o caminho onde o JDK foi instalado.

Agora, precisamos copiar esse caminho até /bin, por exemplo: ***/usr/lib/jvm/java-17-openjdk-amd64/bin*** e digitar no seu terminal o comando: ***export JAVA\_HOME=***, sendo que após o sinal de igual, sem deixar espaços, cole o caminho que você copiou da instalação do JDK. Por exemplo:

export JAVA\_HOME=/usr/lib/jvm/java-17-openjdk-amd64/bin

Vamos pressionar a tecla “enter” e pronto, o JDK está instalado e configurado. Para testá-lo, digite os seguintes comandos no terminal:

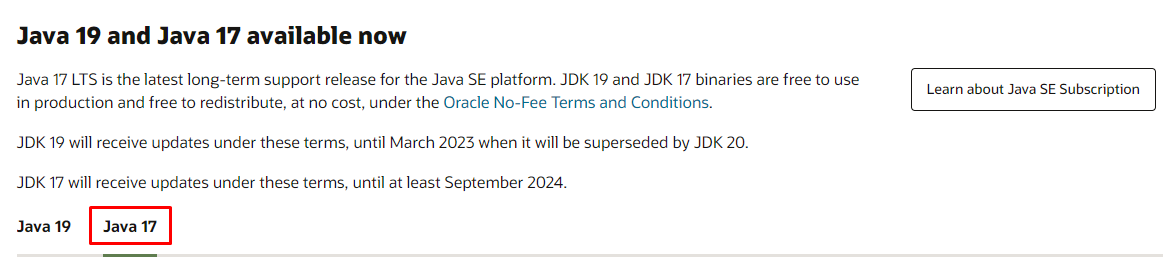
javac -version

java -version

**Mac**

1) Para instalar no Mac, podemos acessar o [site da Oracle](https://www.oracle.com/java/technologies/downloads/#jdk17-mac) ou buscar no navegador por “Java Download Oracle” e acessar o primeiro link.

2) Agora, precisamos escolher a versão do Java. Neste curso vamos utilizar o **Java 17 LTS**, por ser a versão de suporte de longo prazo mais recente para a plataforma Java SE.



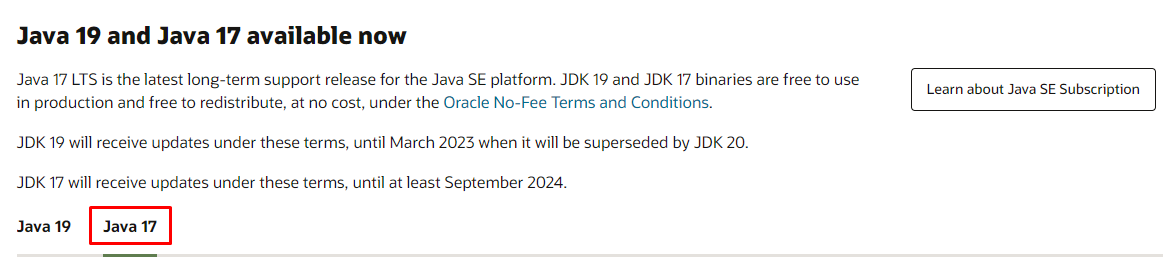
3) Selecione o Mac como sistema operacional e faça o download.



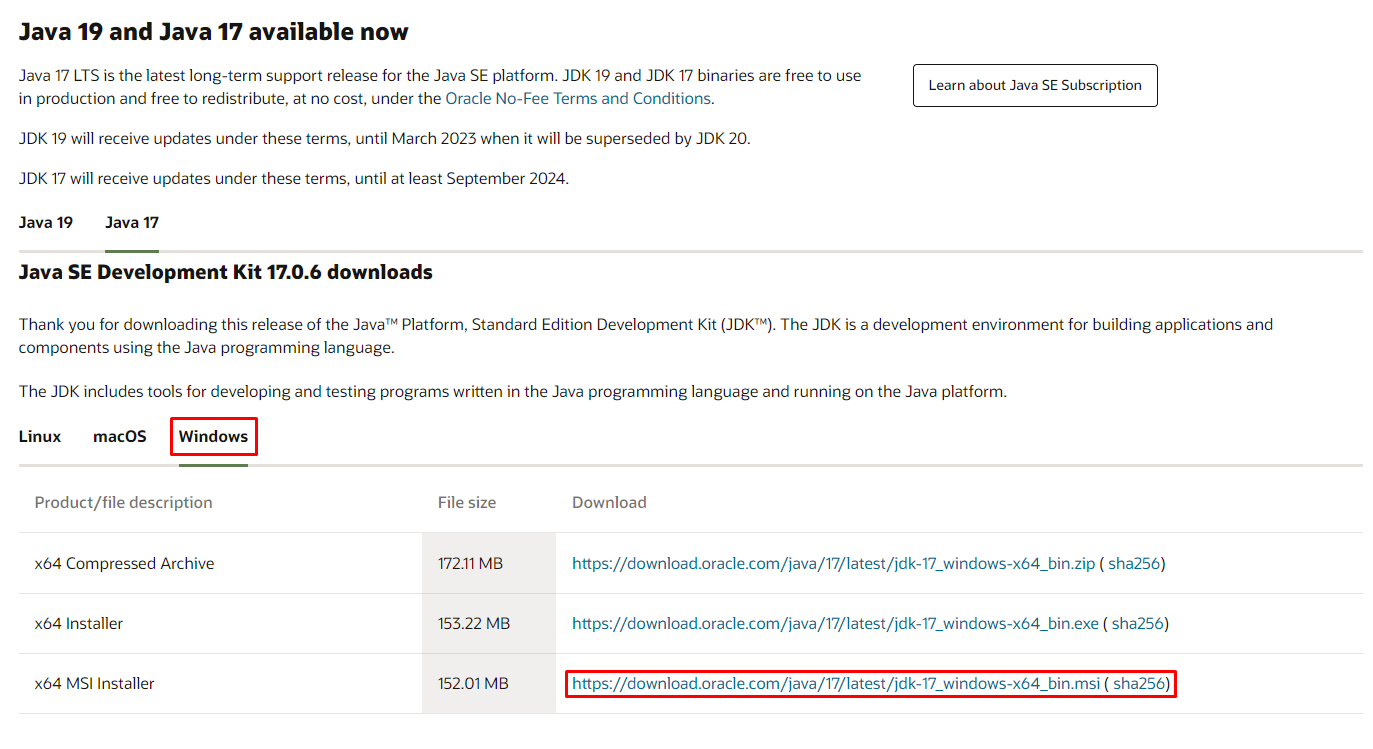
**Windows**

1) Para instalar no Windows, podemos acessar o [site da Oracle](https://www.oracle.com/java/technologies/downloads/#jdk17-windows) ou buscar no navegador por “Java Download Oracle” e acessar o primeiro link.

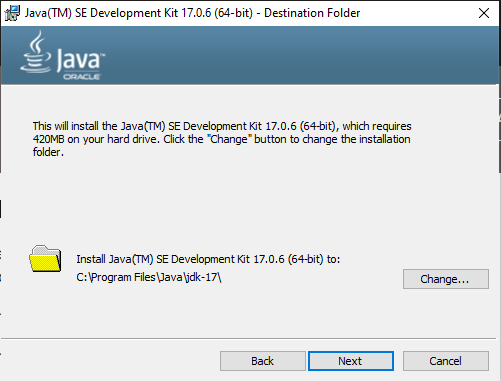
2) Agora, precisamos escolher a versão do Java. Neste curso vamos utilizar o **Java 17 LTS**, por ser a versão de suporte de longo prazo mais recente para a plataforma Java SE.



3) Selecione o Windows como sistema operacional e faça o download.



4) Após baixado, vamos executar o instalador e prosseguir com a instalação.





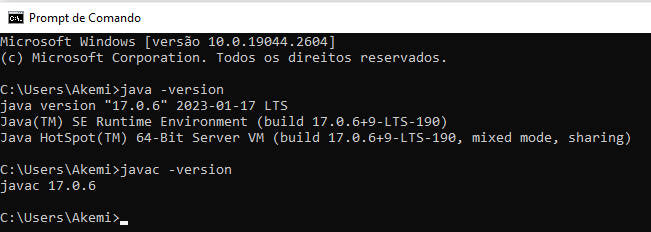


5) Após a instalação, vamos testar o nosso Java. Para isso, utilizaremos o prompt de comando. Teste os seguintes comandos e verifique se tudo correu como o esperado:

java -version

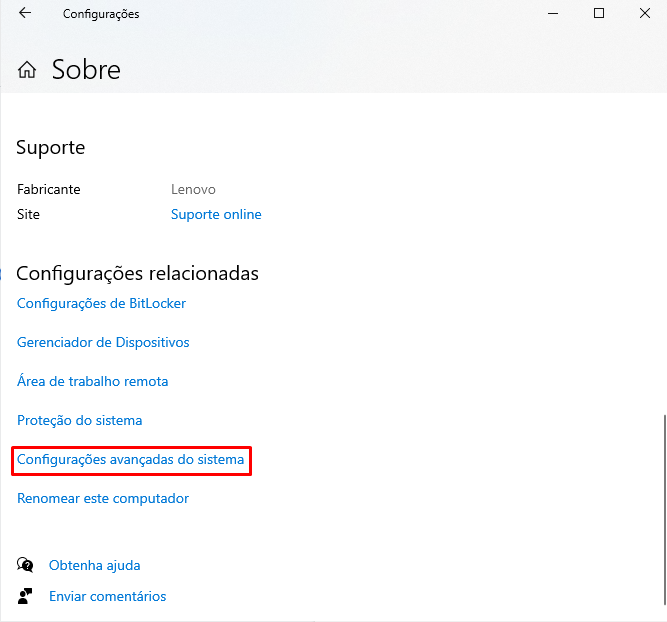
javac -version

Resultado:

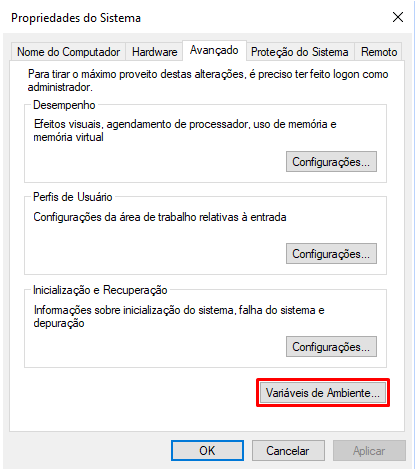


**Configurações do PATH no Windows**

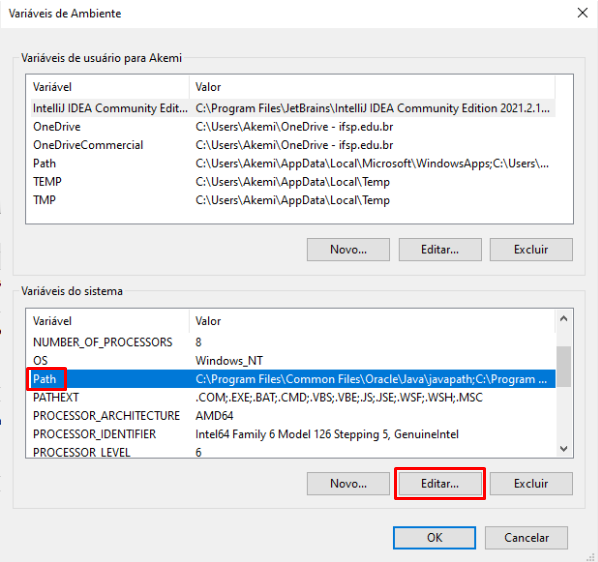
1) Vá em painel de controle e procure por “Sistema”, após isso, clique em “Configurações avançadas do sistema”.



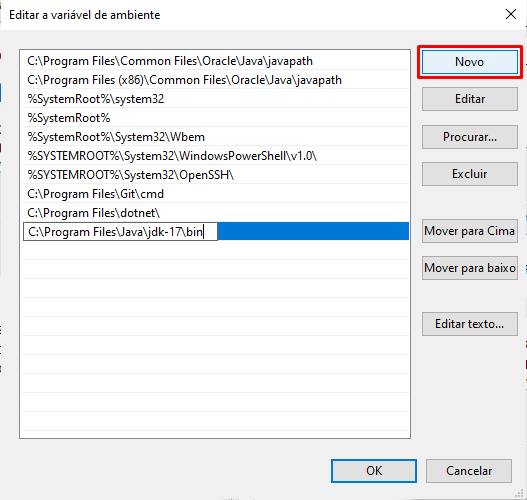
2) Na aba "Avançado", clique em “Variáveis de Ambiente”.



3) Agora foi aberta uma nova janela na parte inferior da tela, selecione a variável de ambiente chamada “Path” e clique em “Editar”.



4) Nessa nova janela, clique no botão “Novo” e na linha que foi selecionada, coloque o caminho para seu diretório bin dentro da pasta jdk, que está dentro da pasta Java.



5) Após feito, feche o prompt de comando e abra novamente. Teste os seguintes comandos:

java -version

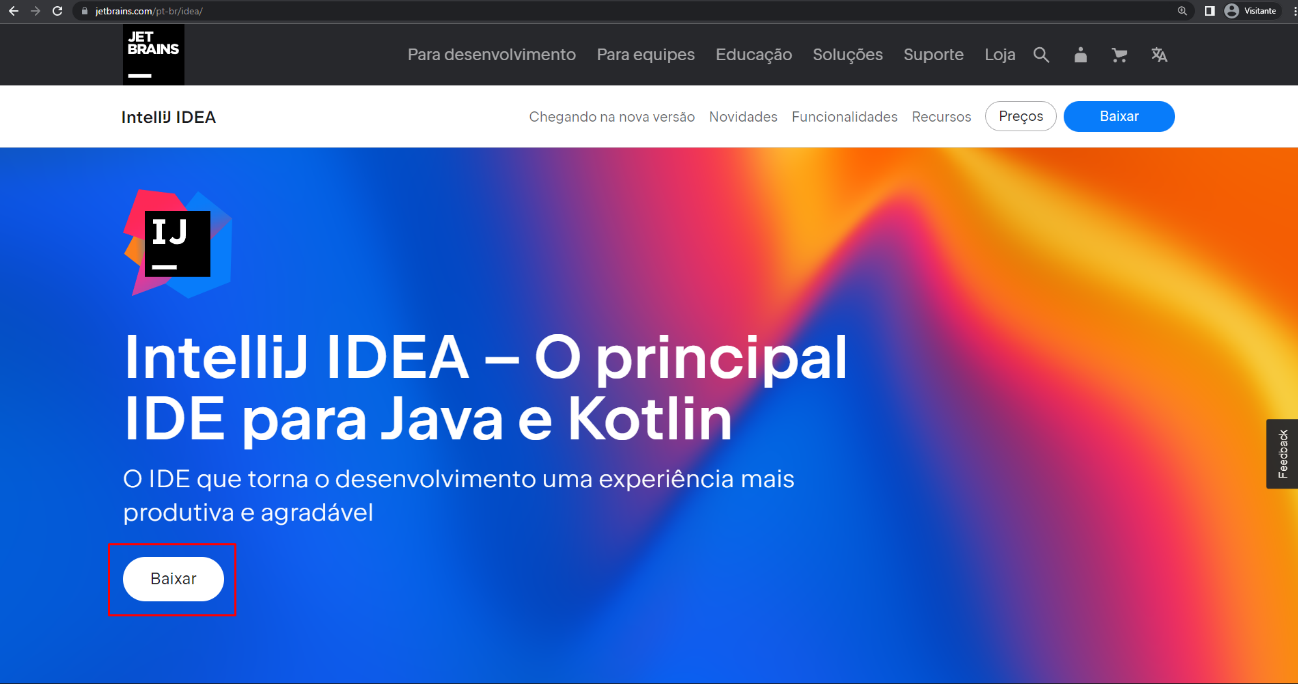
javac -version

**Instalação do IntelliJ**

Durante o curso os instrutores utilizam a IDE IntelliJ IDEA, da empresa JetBrains, muito conhecida por desenvolver IDE's e por ser a empresa que criou e mantém uma das linguagens filhas do Java, o [Kotlin](https://cursos.alura.com.br/formacao-kotlin).

Nesse preparando o ambiente vamos aprender a instalar o IntelliJ, para facilitar o acompanhamento do curso.

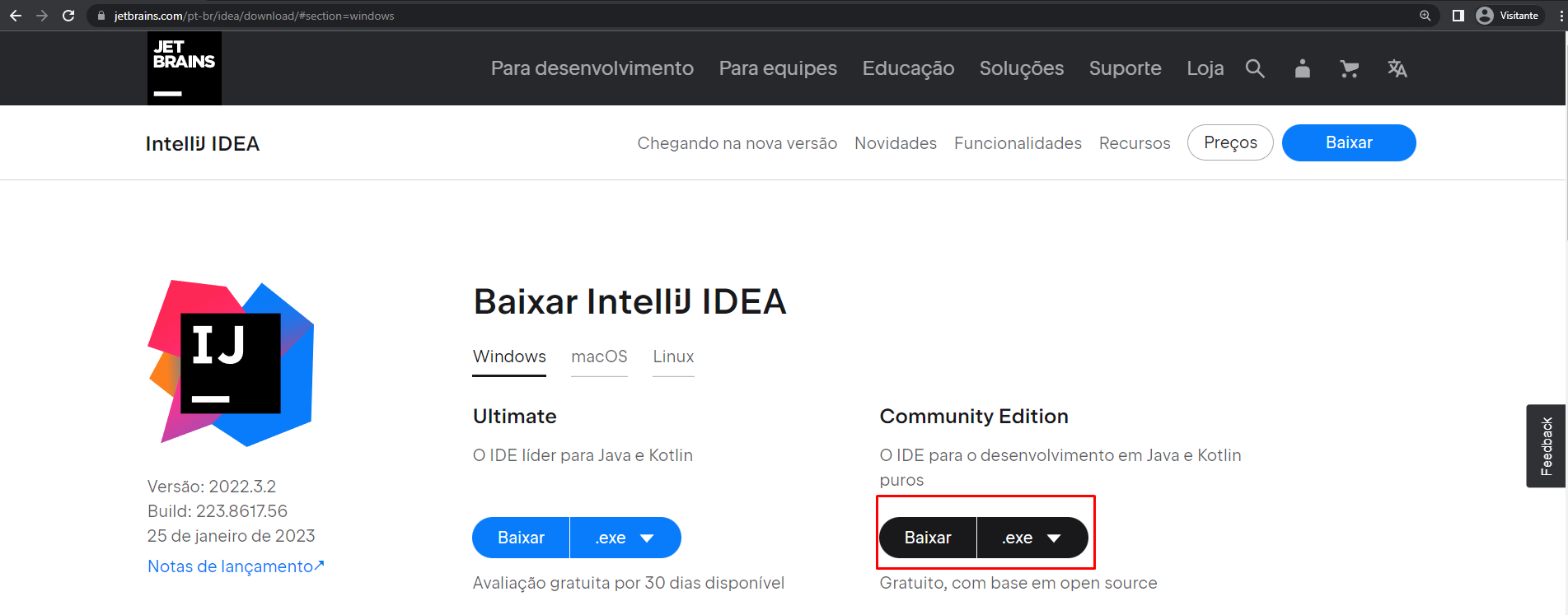
Para começar, precisamos entrar na [página principal do IntelliJ](https://www.jetbrains.com/pt-br/idea/), onde encontraremos algo parecido com a imagem abaixo, onde vamos clicar no botão "Baixar":



Após clicar, seremos levados à página de downloads da ferramenta, que apresenta opções para os sistemas Windows, Linux e MacOS em um menu que deve ser clicado baseado em seu sistema operacional. Logo abaixo do menu vemos duas opções de download, que se trata das duas versões da IDE: a edição **Ultimate** e **Community**.

A versão Ultimate apresenta mais recursos, como suporte a mais linguagens e gerenciamento de banco de dados embutido, no entanto é paga; A edição Community é a opção gratuita da ferramenta e apresenta todos os recursos que vamos precisar durante o curso, por este motivo vamos instalar esta versão.

Para fazer o download o IntelliJ Community Edition, vamos clicar no botão "Baixar" como apresentado na imagem abaixo:

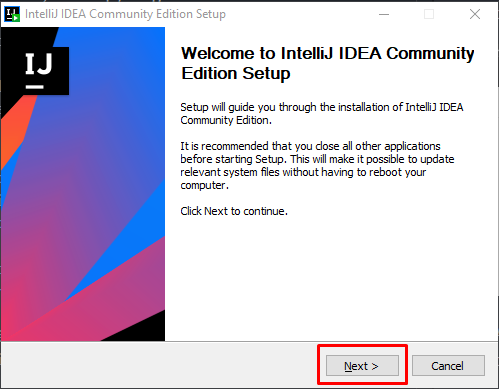


Após clicar em "Baixar", será feito o download da ferramenta e sua instalação varia com o sistema operacional.

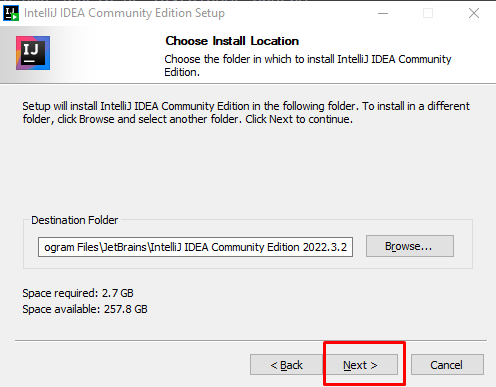
**Windows**

No Windows será baixado o instalador da ferramenta, com nome similar a "ideaIC-2022.3.2", dependendo da versão baixada. Para iniciar a instalação, vamos executar esse arquivo.

Após executado, vai ser aberta uma janela de instalador padrão do Windows, como na imagem abaixo, onde vamos clicar no botão "Next":

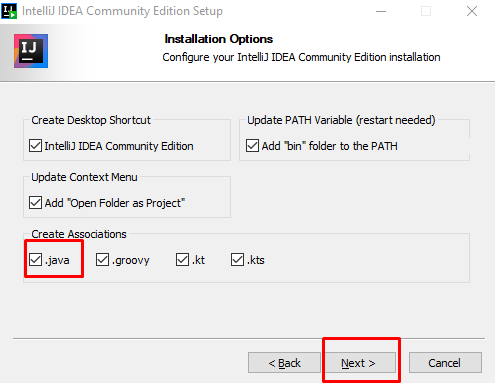


Que vai nos direcionar para uma página onde vamos selecionar em qual pasta será feita a instalação. Aqui nós podemos deixar no diretório padrão e novamente clicar no botão "Next", como mostrado abaixo:



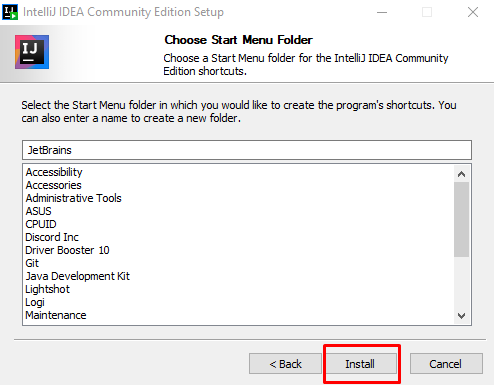
Após isso, seremos direcionados para uma tela com várias caixas de seleção, com opções de instalação da ferramenta. São opções referentes à se queremos criar um atalho no desktop, adicionar o IntelliJ ao PATH, para que possa ser acessado via linha de comando, adicionar uma nova ação ao menu de contexto, para que quando clicarmos em um arquivo com o botão direito do mouse aparece a opção de abri-lo no IntelliJ, e criar associações aos tipos de arquivo, para que eles sejam abertos diretamente na IDE.

Aqui podemos selecionar todas as caixas, como na imagem:



Caso você não queira marcar todas as caixas, a mais importante é na aba "Create Associations", onde vamos criar as associações dos arquivos **.java** ao IntelliJ, como destacado na imagem acima. Após marcar as opções desejadas, podemos seguir clicando no botão "Next".

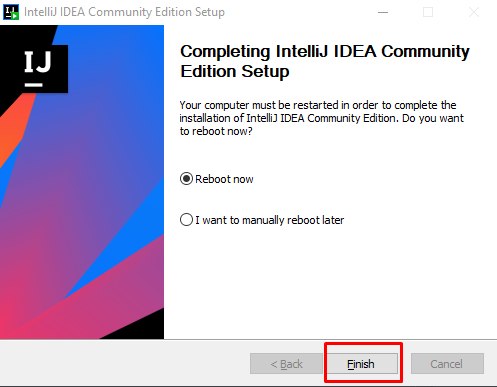
Após isso, vamos para a seleção da pasta do menu iniciar onde poderemos localizar a nossa IDE. Nessa tela podemos simplesmente deixar a opção padrão e clicar no botão "Install" para instalar a ferramenta no computador, como mostrado abaixo:



Será iniciada a instalação dos arquivos da ferramenta, o que pode demorar um tempinho. Hora de pegar um café e esperar!

Ao final da instalação, seremos direcionados a uma tela com a opção de reiniciar o computador agora ou depois. É recomendável reiniciar o computador, para que as alterações no PATH funcionem corretamente, no entanto a ferramenta já pode ser utilizada mesmo sem a reinicialização.

Por fim, podemos clicar no botão "Finish", para finalizar o instalador, como na imagem abaixo:



E prontinho, o IntelliJ já pode ser utilizado para desenvolver suas aplicações!

**Linux**

Em sistemas Linux, o arquivo baixado é um arquivo compactado já com a IDE pré-instalada. Para conseguir executá-la, precisamos navegar até a pasta onde o arquivo baixado está e descompactá-lo com o comando:

tar -xf <arquivoBaixado>.tar.gz

O nome do arquivo pode variar conforme a versão baixada, mas sempre segue um padrão como "**ideaIC-2022.3.2.tar**".

Após descompactar o arquivo, será criada uma pasta com nome similar a "idea-IC-xxx.xxx.xx" onde os "x" representam a versão da ferramenta. Para abrir a IDE, é necessário navegar até a pasta, através do comando:

cd idea-IC-xxx.xxx.xx

E acessar a subpasta "bin", a partir do comando:

cd bin

Onde está localizado o arquivo "idea.sh", que é o script de inicialização do IntelliJ. Agora, para abrir a IDE basta rodar o script com o comando:

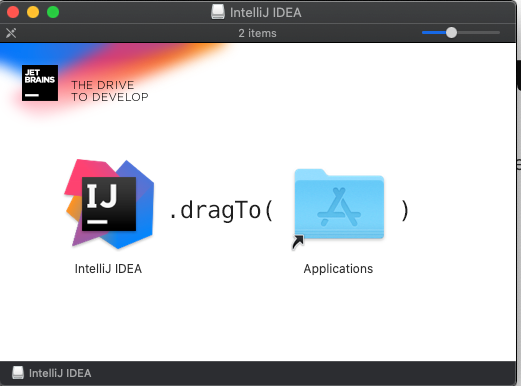
./idea.sh

Que vai inicializar a ferramenta que já está pronta para ser utilizada!

**MacOS**

No MacOS, assim como no Windows, é baixado um instalador com a instalação ".dmg" e a instalação é bastante simples.

Para isso, vamos navegar até a pasta de download do arquivo e executá-lo com duplo clique. Será aberta uma janela pedindo para que o ícone do IntelliJ seja arrastado até o ícone da pasta **Applications**, como mostrado na imagem:



Após arrastar, vai ser iniciado o processo de cópia de arquivos e a IDE será instalada. Ao término da instalação basta pesquisar no menu de pesquisa por "IntelliJ" e executar o programa "IntelliJ IDEA CE", que já teremos acesso aos recursos da ferramenta.

E prontinho, o IntelliJ já está instalado e estamos prontos para estudar Java!

**Um guia para iniciar em Java**

O Java é uma plataforma de programação poderosa e versátil, com quase 30 anos de história e evolução. Se você deseja saber mais detalhes sobre a trajetória do Java, leia o artigo [Java: Tudo o que você precisa saber sobre a linguagem](https://www.alura.com.br/artigos/java?_gl=1*71yht3*_ga*NjUzNTI4NjEzLjE3MTIwODAxNDE.*_ga_1EPWSW3PCS*MTcxMjIyOTg5My42LjEuMTcxMjIzMzMyMi4wLjAuMA..*_fplc*UDFQbTRBNkclMkJXOWVMcjE4R1JodndLYmVNM1M0c0JFTVZRQTRlcEpmWEVZJTJCNkt4JTJCYkFqSm03QUxSZElyd2dIUTR4YkdVZWVTUnJZak1SWHYxR21lY0pTZFZKTUlNQmlUWGtRYkNSRDNNQVRuVElkaGQ0Y3JNaVVaJTJCaGVBdHclM0QlM0Q.). Descubra a história e arquitetura da linguagem, conheça suas principais características e vantagens, e ainda encontre dicas valiosas que vão te ajudar em sua jornada de aprendizado.

**IntelliJ IDEA - Dicas e truques**

Existem diversas funcionalidades que podemos utilizar no IntelliJ IDEA no cotidiano que aumentam a produtividade no trabalho em projetos Java. Para complementar seus estudos, separei um artigo que mostra as facilidades que o IntelliJ nos fornece na hora de declarar variáveis, estrutura de controle de fluxo e até mesmo nas próprias estruturas de código.

Confira o artigo [IntelliJ IDEA: dicas e truques para usar no dia a dia](https://www.alura.com.br/artigos/intellij-idea-dicas-truques-usar-no-dia-a-dia?_gl=1*midm0h*_ga*NjUzNTI4NjEzLjE3MTIwODAxNDE.*_ga_1EPWSW3PCS*MTcxMjIyOTg5My42LjEuMTcxMjIzMzQwNS4wLjAuMA..*_fplc*UDFQbTRBNkclMkJXOWVMcjE4R1JodndLYmVNM1M0c0JFTVZRQTRlcEpmWEVZJTJCNkt4JTJCYkFqSm03QUxSZElyd2dIUTR4YkdVZWVTUnJZak1SWHYxR21lY0pTZFZKTUlNQmlUWGtRYkNSRDNNQVRuVElkaGQ0Y3JNaVVaJTJCaGVBdHclM0QlM0Q.).

**Bytecode e a JVM**

A JVM é uma ferramenta muito importante no mundo Java, pois ela permite que uma apĺicação possa ser executada em múltiplos sistemas operacionais distintos. Para complementar seus estudos, separei um artigo que mostra mais detalhes sobre o processo de compilação e execução de uma aplicação Java pela JVM. Vamos conferir?

[JVM: conhecendo o processo de execução de código](https://www.alura.com.br/artigos/jvm-conhecendo-processo-execucao-de-codigo?_gl=1*q8pen1*_ga*NjUzNTI4NjEzLjE3MTIwODAxNDE.*_ga_1EPWSW3PCS*MTcxMzYxNTI0MS4yNy4xLjE3MTM2MTUzNDUuMC4wLjA.*_fplc*TlJWdlElMkZIdjhxJTJCcXAyOElvVWdYalpldzNxb1hPRmQ3UHNHU1NjWlBSZWh2dHVWT3JoVjBUMWp0dm1HVnRCVHE5QmkxUWVPRlp0Z2xhY1RRR2J6VXBsa1h2MzdlenBtcHNCTHhMQjFweTE3Tkpka2hGcE5zb25VVnVSUnJGUSUzRCUzRA..)

**Imprimindo no console**

Considere o seguinte trecho de código em Java:

**public** **class** **Exemplo** {

**public** void **main**(String[] args) {

**System**.**println**("Oi, meu nome é Alura!");

}

}

* O método main foi declarado de maneira incorreta. Faltou a palavra reservada static na declaração do método main.
* O programa apresentará um erro de compilação. O comando correto para imprimir uma mensagem é System.out.println. O comando System.println, utilizado no exemplo de código, é inválido e vai causar um erro de compilação.

**Operadores**

No Java temos diversos tipos de operadores para lidar com os dados que estamos trabalhando em nossa aplicação. Vou detalhar melhor alguns deles aqui.

**Operadores de atribuição:** Os operadores de atribuição são usados para atribuir um valor a uma variável. O operador de atribuição básico é o "=" (sinal de igual). Por exemplo:

**int** valor = 5; //Atribui o valor 5 à variável valor

Existem também operadores de atribuição combinados, que são uma forma abreviada de atribuição. Por exemplo, o operador "+=" adiciona um valor à variável existente. Assim:

**int** valor = 10;

valor += 15; //Equivalente a valor = valor + 15, atribui o valor 25 à variável valor

**Operadores aritméticos:** Os operadores aritméticos são usados para realizar operações matemáticas básicas. São eles:

* "+" (adição)
* "-" (subtração)
* "\*" (multiplicação)
* "/" (divisão)
* "%" (resto da divisão)

Por exemplo:

**int** a = 10 + 5; // Atribui o valor 15 à variável a

**int** b = 10 - 5; // Atribui o valor 5 à variável b

**int** c = 10 \* 5; // Atribui o valor 50 à variável c

**int** d = 10 / 5; // Atribui o valor 2 à variável d

**int** e = 10 % 3; // Atribui o valor 1 à variável e (o resto da divisão de 10 por 3 é 1)

**Operadores relacionais:** Os operadores relacionais são usados para comparar valores. Eles retornam um valor booleano (verdadeiro ou falso). Trabalharemos melhor com eles quando estivermos na aula de condicionais, onde vamos modificar o fluxo da aplicação dada alguma condição. São eles:

* "==" (igual a)
* "!=" (diferente de)
* ">" (maior que)
* ">=" (maior ou igual a)
* "<" (menor que)
* "<=" (menor ou igual a)

Exemplo:

**int** a = 10; // Atribui o valor 10 à variável a

**int** b = 5; // Atribui o valor 5 à variável b

**int** c = 30; // Atribui o valor 30 à variável c

**boolean** igual = (b == a); //Nesse caso a variável igual ficará com o valor \*false\*, pois o valor de b não é igual o valor de a.

**boolean** diferente = (b != c); //A variável diferente ficará com o valor \*true\*, pois o valor de b é diferente do valor de c.

**boolean** maior = (b > a); //A variável maior ficará com o valor \*false\*, pois o valor de b é menor que o valor de a.

**boolean** menorIgual = (b <= c); //A variável menorIgual ficará com o valor \*true\*, pois o valor de b é menor que o valor de c.

**Operadores lógicos:** Esses operadores são usados quando queremos verificar duas ou mais condições e/ou expressões na aplicação. Eles fazem a comparação de valores booleanos e retornam também um resultado booleano.

São três operadores: AND (&&), OR (||) e NOT (!).

O operador AND (&&), que traduzindo para o português seria o E, é usado para verificar se duas condições são verdadeiras. Se ambas as condições forem verdadeiras, o resultado será verdadeiro. Caso contrário, o resultado será falso. Aqui está um exemplo:

**boolean** a = **true**;

**boolean** b = **false**;

**if** (a && b) {

// Este código não será executado, já que a é verdadeiro e b é falso.

}

O operador OR (||), que traduzindo para o português seria o OU, é usado para verificar se pelo menos uma das condições é verdadeira. Se pelo menos uma das condições for verdadeira, o resultado será verdadeiro. Caso contrário, o resultado será falso. Aqui está um exemplo:

**boolean** a = **true**;

**boolean** b = **false**;

**if** (a || b) {

// Este código será executado, já que a é verdadeiro, mesmo que b seja falso.

}

O operador NOT (!) é usado para negar uma condição. Se a condição for verdadeira, o resultado será falso. Se a condição for falsa, o resultado será verdadeiro. Aqui está um exemplo:

**boolean** a = **true**;

**if** (!a) {

// Este código não será executado, já que a é verdadeiro.

}

**Operadores de incremento:** Além dos operadores citados anteriormente, o operador de incremento é usado para aumentar o valor de uma variável em 1. Existem dois tipos de operadores de incremento: o operador de pré-incremento (++variavel) e o operador de pós-incremento (variavel++).

O operador de pré-incremento (++variavel) aumenta o valor da variável em 1 antes de usar a variável em uma expressão. Aqui está um exemplo:

int num = 5;

int resultado = ++num; //num é incrementado para 6 e depois atribuído a resultado

System.**out**.println(num); // imprime 6

System.**out**.println(resultado); // imprime 6

Já o operador de pós-incremento (variavel++) aumenta o valor da variável em 1 depois de usar a variável em uma expressão. Aqui está um exemplo:

int num = 5;

int resultado = num++; //num é atribuído primeiramente à variável resultado e depois incrementado para 6

System.**out**.println(num); // imprime 6

System.**out**.println(resultado); // imprime 5

**Convenção de código**

A convenção de código do Java é um conjunto de regras recomendadas para escrever código Java que é fácil de ler, entender e manter. Essas regras foram definidas pela Oracle, a empresa que mantém a linguagem Java, e são amplamente seguidas pela comunidade de desenvolvedores Java.

Aqui estão algumas das principais convenções de código do Java:

* Nomes de classes devem começar com letra maiúscula e usar a convenção PascalCase (também conhecida como Upper CamelCase).
  + Exemplo: **MinhaClasse.**
* Nomes de métodos devem começar com letra minúscula e usar a convenção camelCase.
  + Exemplo: meuMetodo().
* Nomes de constantes devem ser totalmente em letras maiúsculas, separadas por underline.
  + Exemplo: MINHA\_CONSTANTE.
* Nomes de variáveis devem começar com letra minúscula e usar a convenção camelCase.
  + Exemplo: minhaVariavel.
* Todas as linhas de código devem ter no máximo 80 caracteres de largura para facilitar a leitura.
* Recomenda-se usar espaços em branco para separar operadores, palavras-chave e elementos de controle de fluxo.
  + Exemplo: if (condicao) {.
* Use comentários para documentar seu código, explicando o que ele faz e por que ele faz isso. Comentários devem ser claros e concisos.

Esse ponto anterior, referente aos comentários, é polêmico, pois muitas pessoas desenvolvedoras consideram que um bom código deve ser autoexplicativo. Se você utiliza nomes intuitivos e descritivos para suas variáveis e métodos, fica mais claro de entender o que está acontecendo no código, dispensando a necessidade de uso de comentários.

Essas são apenas algumas das convenções de código mais comuns no Java. Seguir essas regras pode tornar seu código mais fácil de ler e entender, o que pode economizar tempo e esforço no longo prazo.

É importante lembrar que as convenções de código são apenas recomendações, e não regras obrigatórias. No entanto, seguir essas convenções pode ajudar a criar um padrão consistente em todo o seu código Java e também torná-lo mais fácil de compartilhar e colaborar com outros desenvolvedores.

**Tipos primitivos**

Em Java, assim como na maioria das linguagens de programação, existem os tipos primitivos, que são os tipos de dados mais básicos e fundamentais da linguagem. Eles são utilizados para representar valores simples e são definidos pela própria linguagem.

Java possui oito tipos primitivos diferentes: boolean, byte, char, short, int, long, float e double. Cada um desses tipos possui suas próprias características e faixa de valores permitidos, conforme será descrito a seguir.

**boolean:** O tipo boolean é utilizado para representar valores lógicos, podendo assumir apenas dois valores: true ou false. É utilizado em expressões condicionais, loops e outros casos em que se deseja avaliar se uma determinada condição é verdadeira ou falsa.

**byte:** O tipo byte é utilizado para representar valores numéricos inteiros de 8 bits. Ele possui uma faixa de valores de -128 a 127.

**char:** O tipo char é utilizado para representar caracteres individuais. Ele pode armazenar qualquer caractere Unicode e é representado por aspas simples ('').

**short:** O tipo short é utilizado para representar valores numéricos inteiros de 16 bits. Ele possui uma faixa de valores de -32.768 a 32.767.

**int:** O tipo int é utilizado para representar valores numéricos inteiros de 32 bits. É um dos tipos de dados mais utilizados para representar números inteiros em Java e possui uma faixa de valores de -2.147.483.648 a 2.147.483.647.

**long:** O tipo long é utilizado para representar valores numéricos inteiros de 64 bits. Ele é utilizado para representar valores inteiros muito grandes e possui uma faixa de valores de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Os valores devem ser declarados com “l” ou “L” no final. Exemplo: ***long variável = 123l***

**float:** O tipo float é utilizado para representar valores numéricos de ponto flutuante, ou seja, valores com casas decimais, sendo que ocupa 32 bits de memória. Ele pode representar números decimais com até sete dígitos e tem uma precisão limitada, o que significa que ele pode arredondar os números se eles forem muito grandes ou muito pequenos. Os valores devem ser declarados com “f” ou “F” no final. Exemplo: ***float variável = 123f***

**double:** O tipo double é similar o float, entretanto ele ocupa 64 bits de memória e pode representar números decimais com até 15 dígitos.

**Comentários no código fonte**

Ao escrever código em Java, é importante que o seu código seja legível e compreensível para outras pessoas, além de você mesmo no futuro. Uma das maneiras de fazer isso é por meio do uso de comentários de código, pois eles ajudam a explicar o que o seu código está fazendo e por que ele foi escrito daquela maneira.

**Como funcionam os comentários em Java?**

O compilador Java ignora todos os trechos de comentários de código, então eles não afetam o funcionamento do seu programa. Existem duas principais formas de se escrever comentários em Java:

**Comentários de linha única:** São comentários que aparecem em uma única linha e começam com duas barras "//". O compilador Java ignora todo o texto que aparece após as duas barras até o final da linha. Exemplo:

// Esta é uma linha de comentário que será ignorada pelo compilador

**Comentários de várias linhas:** Esses comentários podem abranger várias linhas e são delimitados por "/\*" e "\*/". O compilador Java ignora todo o texto que aparece entre esses dois símbolos.

Exemplo:

/\* Este é um exemplo de comentário

de várias linhas em Java

que será ignorado pelo compilador \*/

**Comentários são importantes?**

Os comentários são importantes porque ajudam a tornar o código mais legível e compreensível para outras pessoas que vão precisar ler e dar manutenção no código. Isso é especialmente importante quando várias pessoas trabalham no mesmo projeto. Comentários claros e concisos ajudam a explicar o que o código está fazendo, por que ele foi escrito daquela maneira e como ele funciona.

**Strings e text blocks**

Em Java, a classe String é uma das mais importantes e utilizadas. Ela é responsável por representar uma sequência de caracteres, ou seja, um texto.

**Criação de Strings**

Para criar uma String em Java, basta utilizar aspas duplas para delimitar o texto. Por exemplo:

**String** nome = "Alura";

Nesse exemplo, a variável nome recebe uma String com o texto "Alura". É possível concatenar duas ou mais Strings utilizando o operador +, por exemplo:

**String** saudacao = "Olá, ";

**String** nome = "Alura";

**String** mensagem = saudacao + nome + "!";

Nesse caso, a variável mensagem receberá a String "Olá, Alura!".

**Comparação de Strings**

Em Java, é possível comparar duas Strings utilizando o operador ==. Porém, esse operador verifica apenas se as duas variáveis apontam para o mesmo objeto na memória, e não se o conteúdo das Strings é igual. Para comparar o conteúdo de duas Strings, é necessário utilizar o método equals(). Por exemplo:

String senha = "12345";

**if** (senha.**equals**("12345")) {

System.**out**.println("Acesso autorizado!");

} **else** {

System.**out**.println("Senha incorreta.");

}

Nesse caso, o método equals() é utilizado para comparar o conteúdo da variável senha com a String "12345". Se as duas Strings forem iguais, a mensagem "Acesso autorizado!" será impressa, caso contrário, a mensagem "Senha incorreta." será impressa.

Mais adiante trabalharemos também com o método equalsIgnoreCase(), que é usado para que a comparação de Strings desconsidere as letras maiúsculas e minúsculas. Utilizando apenas o equals, as String “alura” e “Alura” seriam consideradas diferentes.

**Text Block**

Introduzido na versão 15 do Java, o Text Block é uma nova forma de representar Strings que facilitam a escrita de textos com múltiplas linhas. Em vez de utilizar aspas duplas para delimitar o texto e inserir quebras de linha manualmente, ou utilizar concatenações, é possível utilizar uma sintaxe mais simples que permite inserir o texto exatamente como ele é.

**Sintaxe do Text Block**

Para criar um Text Block em Java, basta utilizar três aspas duplas para delimitar o texto, seguidas de uma quebra de linha. Por exemplo:

String mensagem = """

Olá, mundo!

Este é um Text Block.

Ele permite escrever textos com múltiplas linhas

sem precisar usar caracteres de escape ou quebras de linha manualmente ou concatenações.

""";

Nesse exemplo, a variável mensagem recebe um Text Block com o texto "Olá, mundo! Este é um Text Block. Ele permite escrever textos com múltiplas linhas sem precisar usar caracteres de escape ou quebras de linha manualmente!".

**Formatação de textos**

Em Java, é possível formatar textos e números de diversas maneiras. Isso pode ser útil em diversas situações, como ao exibir valores para o usuário de uma maneira mais legível.

Uma das maneiras mais comuns de se formatar textos em Java é utilizando o método format(), da classe String. Esse método permite formatar um texto utilizando diversos placeholders, que são representados pelo caractere % seguido de uma letra que indica o tipo de dado que será inserido no placeholder. Por exemplo, %s indica que uma String será inserida no placeholder, %d indica um valor inteiro e %f indica um valor de ponto flutuante. Vamos ver um exemplo:

String nome = "Maria";

int idade = 30;

double valor = 55.9999;

System.**out**.println(String.format("Meu nome é %s, eu tenho %d anos e hoje gastei %.2f reais", nome, idade, valor));

Nesse exemplo, os valores das variáveis nome, idade e valor são passados como parâmetros para o método String.format, substituindo os placeholders %s, %d e %.2f, respectivamente. O resultado impresso será "Meu nome é Maria, eu tenho 30 anos e hoje gastei 55,99 reais". Perceba também que o placeholder %.2f indica que o valor deve ser formatado com duas casas decimais.

Esse exemplo do que foi feito para o String.format também pode ser usado com Text Block, onde usa o método que citei em aula, o formatted, para informar as variáveis que deverão ser utilizadas no lugar dos placeholders. Veja esse exemplo:

String nome = "João";

int aulas = 4;

String mensagem = """

Olá, %s!

Boas vindas ao curso de Java.

Teremos %d aulas para te mostrar o que é preciso para você dar o seu primeiro mergulho na linguagem!

""".formatted(nome, aulas);

System.out.println(mensagem);

O resultado impresso será:

Texto

Descrição gerada automaticamente

Olá, João!

**Casting**

Casting é um recurso utilizado em Java para converter um tipo de dado em outro. Essa conversão pode ser feita de forma automática pelo compilador (conversão implícita), quando o tipo de dado de destino é compatível com o tipo de dado de origem, ou de forma manual (conversão explícita), utilizando o operador de casting.

O casting é utilizado para permitir que tipos de dados incompatíveis possam ser utilizados em uma mesma operação ou expressão. Por exemplo, se um método espera um parâmetro do tipo int e o valor que se deseja passar é do tipo double, é necessário fazer um casting para converter o valor em int.

**Casting implícito**

O casting implícito é realizado automaticamente pelo compilador quando o tipo de dado de origem é compatível com o tipo de dado de destino. Por exemplo, é possível atribuir um valor de tipo int a uma variável do tipo double, pois o tipo double é maior e suporta todos os valores que o tipo int pode armazenar:

**int** x = 10;

**double** y = x; // casting implícito

**Casting explícito**

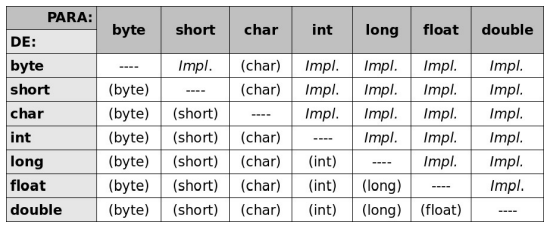
O casting explícito é realizado quando o tipo de dado de origem é incompatível com o tipo de dado de destino. Nesse caso, devemos utilizar o operador de casting para realizar a conversão:

**double** x = 10.5;

**int** y = (**int**) x; // casting explícito

No exemplo anterior, o valor da variável x é convertido em um valor inteiro utilizando o casting explícito. É importante notar que, neste caso, a parte decimal será descartada e o valor atribuído à variável y será 10.

Abaixo tem uma tabela, onde você pode visualizar mais facilmente as conversões que são implícitas e as que necessitam ser feitas de forma explícita.



**Concatenação de String e inteiros**

Alice resolveu praticar os conceitos de concatenação de Strings e elaborou o seguinte trecho de código:

**String** saudacao = "Olá, meu nome é ";

**String** nome = "Alice ";

**String** continuacao = "e minha idade é ";

**int** idade = 17;

System.out.println(saudacao + nome + continuacao + idade);

* Não há problemas, a concatenação pode ser feita sem problemas.

O resultado será:

**Ol**á, meu nome é Alice e minha idade é 17

**Switch Case**

Uma alternativa ao if/else é o switch case, que é uma estrutura de controle de fluxo que permite executar diferentes ações com base no valor de uma expressão. É uma forma mais simplificada e legível de escrever vários blocos if/else encadeados.

A sintaxe do switch case em Java é a seguinte:

**switch** (expressão) {

**case** valor1:

// código a ser executado se a expressão for igual a valor1

**break**;

**case** valor2:

// código a ser executado se a expressão for igual a valor2

**break**;

**case** valor3:

// código a ser executado se a expressão for igual a valor3

**break**;

...

**default**:

// código a ser executado se a expressão não for igual a nenhum valor

**break**;

}

A expressão é uma variável ou uma expressão de código que será avaliada. Cada case é uma possível condição que pode ser atendida pela expressão. Quando a expressão é igual ao valor especificado em um determinado case, o código correspondente a esse case será executado. A palavra-chave break é usada para sair do switch case após a execução do código correspondente.

O case default é opcional e é executado quando nenhum dos cases especificados é atendido.

Veja um exemplo simples de uso do switch case em Java para verificar o dia da semana com base em um número inteiro:

int dia = 3;

String nomeDia;

**switch** (dia) {

**case** 1:

nomeDia = "domingo";

**break**;

**case** 2:

nomeDia = "segunda-feira";

**break**;

**case** 3:

nomeDia = "terça-feira";

**break**;

**case** 4:

nomeDia = "quarta-feira";

**break**;

**case** 5:

nomeDia = "quinta-feira";

**break**;

**case** 6:

nomeDia = "sexta-feira";

**break**;

**case** 7:

nomeDia = "sábado";

**break**;

**default**:

nomeDia = "Dia inválido";

**break**;

}

System.**out**.println("O dia " + dia + " é " + nomeDia);

Nesse exemplo, a expressão é a variável dia, que contém o valor 3. O switch case verifica o valor da variável dia e executa o código correspondente ao caso em que dia é igual a 3. O resultado será a impressão no console: "O dia 3 é terça-feira".

**Vantagens do switch case:**

Em resumo, o switch case torna o código mais fácil de entender e mais legível, em comparação ao if/else, especialmente quando há várias condições possíveis.

**A classe Scanner**

A classe Scanner do Java é utilizada para ler dados de entrada em um programa Java. Esses dados podem ser lidos a partir de várias fontes de entrada, como arquivos, fluxos de entrada, Strings e até mesmo a entrada do usuário através do teclado, como vimos em aula.

Ela oferece uma série de métodos para ler dados de diferentes tipos, como inteiros, números de ponto flutuante, strings e caracteres.

Para utilizar a classe Scanner, primeiro é necessário importá-la no início do seu programa. Provavelmente ao incluir a mesma no código, a IDE já vai sugerir o import. Esse import ficará como descrito abaixo:

**import** java.util.Scanner;

Veja um exemplo básico de como utilizar a classe Scanner para ler dados distintos:

**public** **class** **ExemploScanner** {

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

Scanner scanner = **new** Scanner(System.**in**);

System.**out**.print("Digite seu nome: ");

String nome = scanner.nextLine();

System.**out**.print("Digite sua idade: ");

int idade = scanner.nextInt();

System.**out**.print("Digite o valor que pretende investir esse mês: ");

double valor = scanner.nextDouble();

System.**out**.println(nome + " que tem " + idade + " anos, irá investir R$ " + valor + " esse mês.");

scanner.close();

}

}

Nesse exemplo, primeiro importamos a classe Scanner e, em seguida, criamos uma instância dela passando o objeto System.in' como parâmetro para indicar que queremos ler a entrada do usuário pelo teclado.

Depois, usamos o método nextLine() para ler uma linha de texto. Além desse, utilizamos também o nextInt() para ler um número inteiro e o nextDouble() para ler um número decimal.

Você pode encontrar a lista completa de métodos na [documentação oficial do Java](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Scanner.html).

**Infinitos 1…**

Alice estava praticando seus conhecimentos sobre a estrutura de repetição while e teve a ideia de fazer um programa que apresentasse os números de 1 a 10. Assim, ela escreveu o seguinte trecho de código:

**public** **class** **Programa** {

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

int contador = 1;

**while**(contador <= 10) {

System.**out**.println(contador);

}

}

}

No entanto, seu programa só imprime no console o número 1 infinitamente. Qual problema no código de Alice?

Falta incrementar dentro do while +1 a cada lood na variável contador

**Transformando While em For**

Agora que Alice está ficando craque em laços, decidiu desafiar seu amigo Bernardo a transformar o código a seguir, que utiliza um laço while, em um código que utiliza o for.

***Código com while:***

**public** **class** **Programa** {

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

int contador = 1;

**while**(contador <= 10) {

System.**out**.println(contador);

contador++;

}

}

}

Você consegue ajudar Bernardo a chegar a resposta do desafio?

**public** **class** **Programa** {

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

**for**(int contador = 1; contador <= 10; contador++) {

System.**out**.println(contador);

}

}

}