

transforme ■ se



Java

Aula 1 - Introdução



A LINGUAGEM JAVA



A Linguagem Java

É bem provável que você já tenha ouvido falar da linguagem Java diversas vezes. Basta ler uma revista especializada na área de informática que lá está ela. A linguagem tem tido muito sucesso no mercado, e diversas ferramentas têm surgido para manipular ou gerar código Java.

Um dos motivos da grande aceitação da linguagem Java é que a tornou tão atraente é o fato de que programas podem ser executados virtualmente em muitas plataformas, aceitos em muitos tipos de equipamento.



A Linguagem Java

O aspecto da utilização de Java em multiplataforma é muito importante, porque os programadores não necessitam ficar preocupados em saber em qual máquina o programa será executado, uma vez que um mesmo programa pode ser usado num PC, num Mac ou em um computador de grande porte.

É muito melhor para uma empresa desenvolver um software que possa ser executado em “qualquer lugar”, independentemente da máquina do cliente. Seguindo essa mesma linha, a maioria das linguagens de programação desenvolvidas recentemente também é multiplataforma.

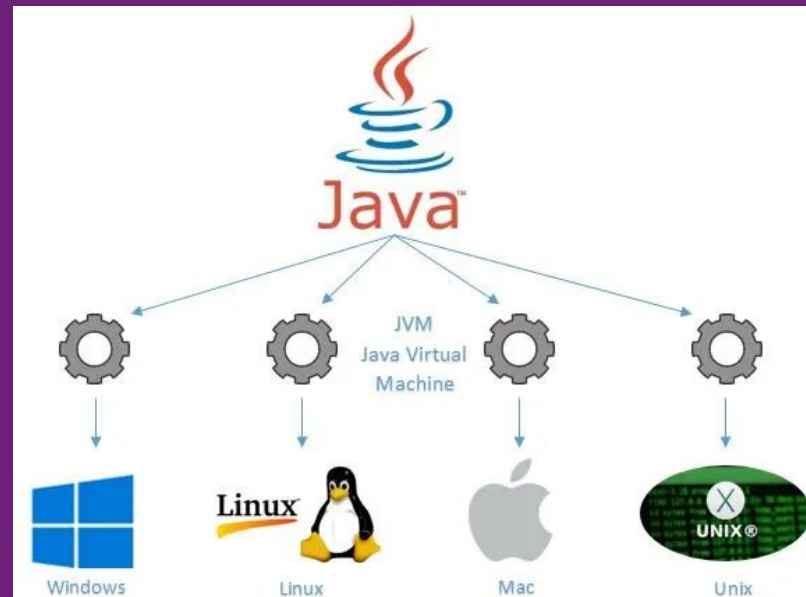
AS CARACTERÍSTICAS DA LINGUAGEM JAVA



A Linguagem Java

O aspecto da utilização de Java em multiplataforma é muito importante, porque os programadores não necessitam ficar preocupados em saber em qual máquina o programa será executado, uma vez que um mesmo programa pode ser usado num PC, num Mac ou em um computador de grande porte.

É muito melhor para uma empresa desenvolver um software que possa ser executado em “qualquer lugar”, independentemente da máquina do cliente. Seguindo essa mesma linha, a maioria das linguagens de programação desenvolvidas recentemente também é multiplataforma.



As características da linguagem Java

Orientação a objetos, Portabilidade, Multithreading, Suporte à comunicação

As características da linguagem Java

Orientação a objetos: é um paradigma de programação já sólido no mercado, e a maioria das linguagens de programação atuais permite trabalhar dessa forma.

Como conceito inicial, imagine a orientação a objetos como uma prática de programação que torna possível elaborar um software a partir da geração de objetos que se comunicam entre si. Esses objetos podem simular, apesar de não ser apenas isso, um objeto do mundo real, como um automóvel, uma casa, uma pessoa etc.

Mais à frente será apresentado o conceito de classe, um trecho de código a partir do qual os objetos são criados. Imagine a classe como uma forma de pão que possibilita fazer pães com as mesmas características da forma.

As características da linguagem Java

Portabilidade: Java é uma linguagem multiplataforma, ou seja, uma mesma aplicação pode ser executada em diferentes tipos de plataforma sem a necessidade de adaptação de código. Essa portabilidade permite que um programa escrito na linguagem Java seja executado em qualquer sistema operacional que possua uma máquina virtual Java.

As características da linguagem Java

Multithreading: threads (linhas de execução) são o meio pelo qual se consegue fazer com que mais de um evento aconteça simultaneamente em um programa. Assim, é possível criar servidores de rede multiusuário, em que cada thread, por exemplo, cuida de uma conexão de um usuário ao servidor, isto é, um mesmo programa pode ser executado várias vezes ao mesmo tempo e cada execução pode processar uma instrução em um ponto diferente do mesmo programa.

As características da linguagem Java

Suporte à comunicação: uma das vantagens do Java é fornecer um grande conjunto de classes com funcionalidades específicas, ou seja, muitos detalhes de programação são encapsulados em classes já prontas. Nesse contexto, a linguagem oferece um conjunto de classes para programação em rede, o que agiliza a implementação de sistemas multiusuários. Tais classes são desenvolvidas para suportar tecnologias avançadas de comunicação, como protocolos TCP/IP (Transport Control Protocol/Internet Protocol), HTTP, FTP (File Transfer Protocol), entre outros.

Diversas outras características poderiam ser citadas, tais como robustez, segurança, alto desempenho etc.

Java tem muitos frameworks

Em função do grande sucesso e aceitação que Java obteve no mercado, foram desenvolvidos muitos frameworks que visam à facilitação e automação do processo de desenvolvimento de software.

Um framework pode ser entendido como um ambiente de trabalho que torna transparente o processo de desenvolvimento, isto é, o desenvolvedor normalmente não precisa se preocupar em como “as coisas” funcionam e como elas se integram umas com as outras, o próprio ambiente se encarrega disso, reduzindo o trabalho e o tempo de desenvolvimento do software.

Existem muitos frameworks (talvez centenas) desenvolvidos para Java, dentre os quais podemos citar Struts, JSF, JUnit, Jasper Reports, Hibernate, Prevaier, Spring, GWT, MyFaces. Cada framework traz sua contribuição para o aprimoramento no desenvolvimento de aplicações em Java e exige muita dedicação por parte dos profissionais da área.

O mais utilizado e mais robusto atualmente é o Spring, que é o FW que vamos utilizar mais para frente.

Java e suas edições

Outra questão importante se refere às edições da linguagem Java, isto é, a divisão existente entre os diferentes ambientes em que o Java pode ser empregado. As principais divisões da linguagem Java são:

- JSE (Java Standard Edition): pode ser considerada o core (a base principal) da linguagem, projetada para execução em máquinas simples e estações de trabalho.
- JEE (Java Enterprise Edition): voltada para o desenvolvimento de aplicações baseadas em servidor, tais como páginas JSP (JavaServer Pages), Servlets, XML (Extensible Markup Language) etc.
- JME (Java Micro Edition): projetada para dispositivos com menor poder de processamento e memória, tais como dispositivos móveis, celulares etc.

Java e suas edições

- JavaFX: uma plataforma que suporta o desenvolvimento de aplicações ricas que podem ser executadas em vários dispositivos diferentes, tornando as interfaces gráficas mais interativas e dinâmicas. Fornece suporte a qualquer biblioteca Java, o que torna possível a criação de interfaces gráficas para diversos ambientes, incluindo desktop, browser, celulares, TVs, videogames, entre outros.
- Java Card: uma plataforma voltada a dispositivos embarcados com limitações de processamento e armazenamento, como smart cards e o Java Ring.

Saiba mais sobre o Java Ring: <https://acervolima.com/o-que-e-java-ring/>

CRIAÇÃO DE PROGRAMAS EM JAVA E COMO SÃO EXECUTADOS



Criação de programas em Java

Programas em Java na verdade são classes. Aliás, como praticamente tudo no java. Podemos utilizar um editor de textos simples como o bloco de notas para escrever nossos programas e após terminar, temos que salvar esse programa como um arquivo *.java e o nome deve ser o mesmo que demos à classe.

Esse código que escrevemos é o nosso código fonte e precisará ser passar por um processo de análise para verificar se existem erros. Esse processo é chamado de compilação e é realizado por meio de um compilador Java, normalmente o compilador presente no kit de desenvolvimento do Java. Todo programa Java deve ser compilado, assim como ocorre com outras linguagens de programação, como Pascal, C, entre outras.

Criação de programas em Java

Com o compilador é realizada a tradução do programa escrito em Java para uma linguagem intermediária chamada Java bytecodes – um código independente de plataforma que é decifrado por um interpretador Java, isto é, para que um programa em Java seja executado, é necessário possuir uma outra ferramenta chamada interpretador, responsável por executar o programa escrito em Java em que cada instrução do bytecode é interpretada e executada no computador.

A Figura a seguir ilustra a sequência de desenvolvimento de um programa em Java. Como um programa em Java deve ser criado na forma de uma classe, conceito estudado mais à frente, deste ponto em diante os programas serão chamados de classes.

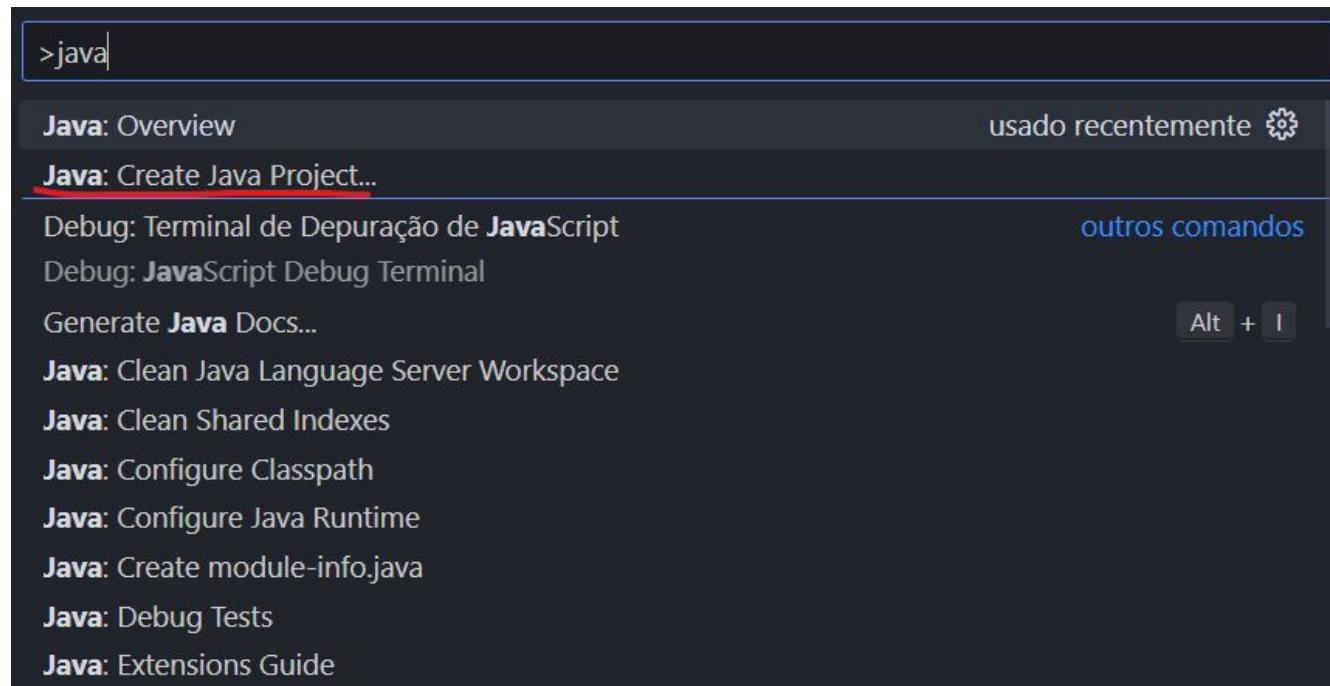
Criação de programas em Java



Na maioria das principais ferramentas de desenvolvimento, o processo de compilação é automático, isto é, ocorre durante a digitação do código-fonte. A compilação vai sendo executada automaticamente durante a digitação da mesma forma que o corretor ortográfico dos editores de texto.

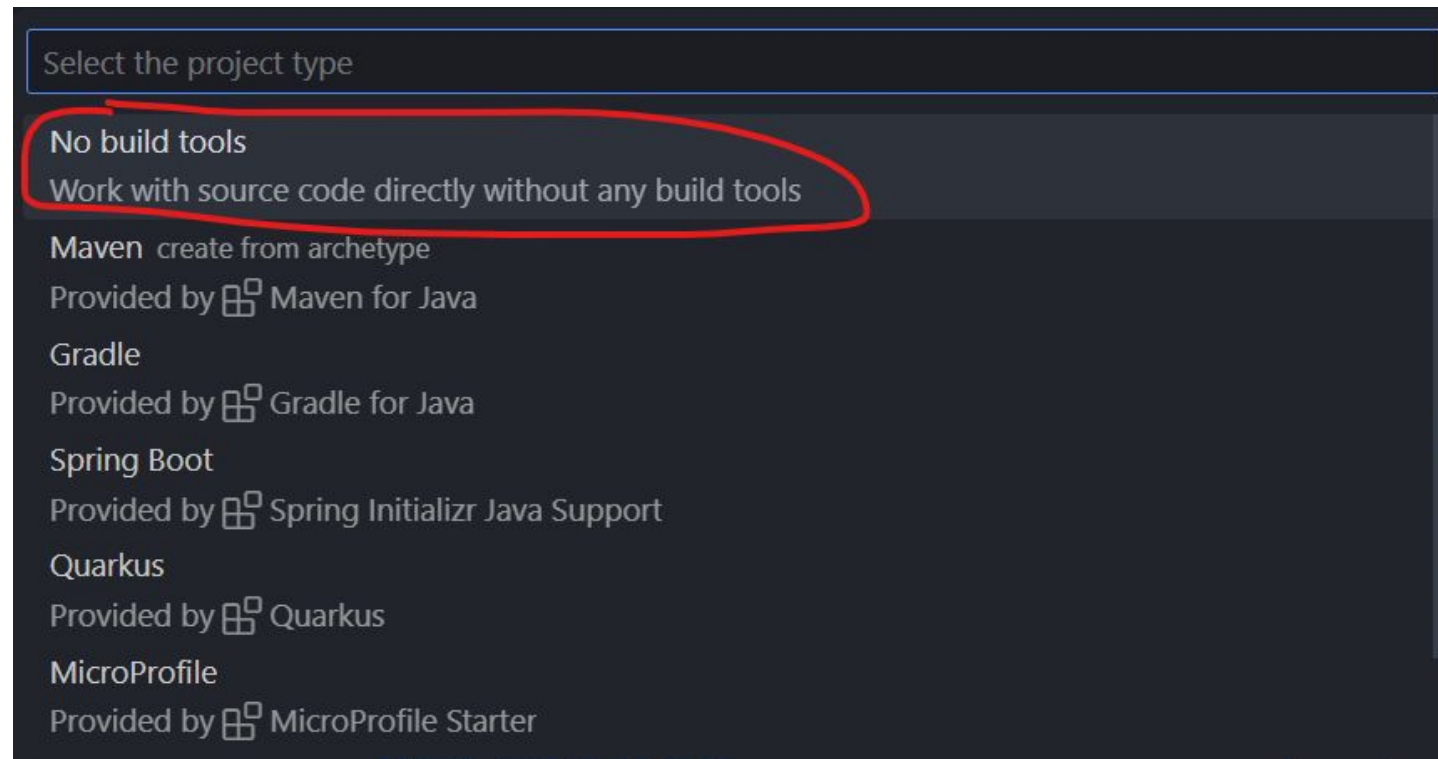
Criação de programas em Java

Para começarmos a programar em java, vamos começar criando um projeto onde vamos guardar nossas classes. No Vs Code, podemos usar a opção “Create Java Project” para criar o nosso projeto. Para isso, pressione a tecla F1 e na caixa que vai surgir digite: Java. Algumas opções aparecerão como na imagem abaixo. Clique na opção Java: Create Java Project.



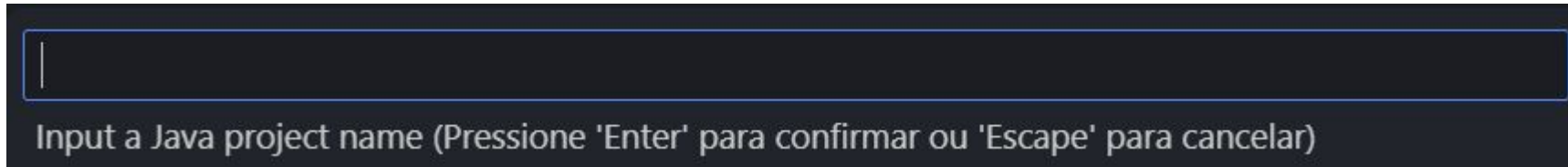
Criação de programas em Java

Na sequência, clique na opção “No Build Tools”. Uma janela se abrirá pedindo para você escolher uma pasta para salvar o seu projeto.



Criação de programas em Java

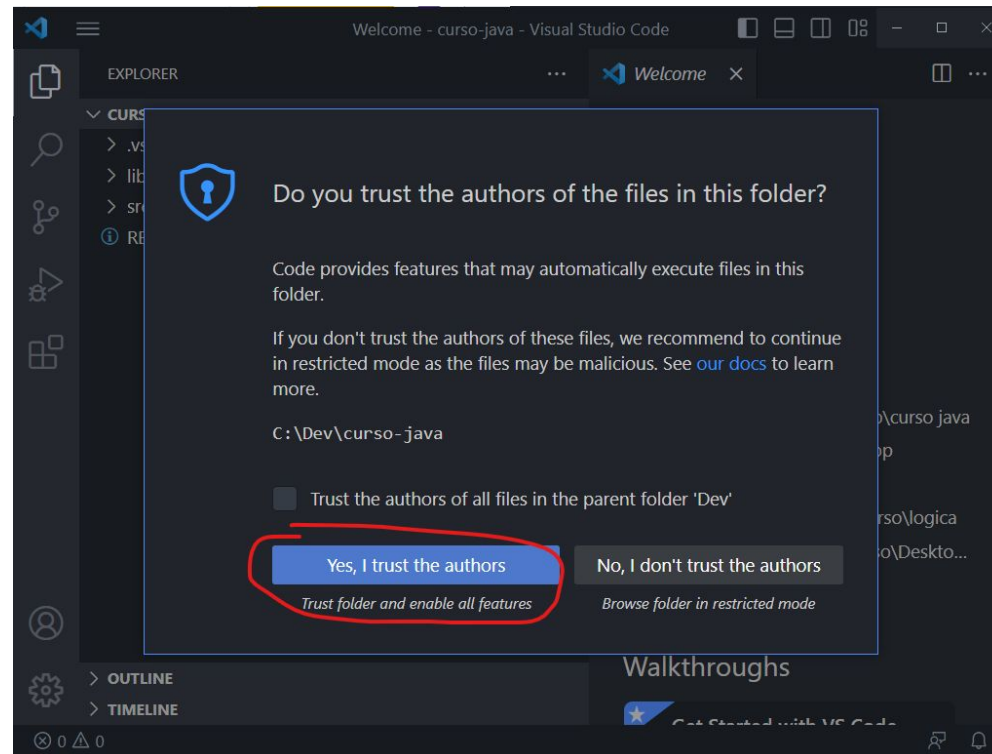
O próximo passo pede para que você escolha o nome do seu projeto. Digite o nome e pressione "Enter".

A screenshot of a Java IDE's project creation dialog box. It features a dark background with a light blue rectangular input field at the top. Below the input field, the text "Input a Java project name (Pressione 'Enter' para confirmar ou 'Escape' para cancelar)" is displayed in a light gray font.

Input a Java project name (Pressione 'Enter' para confirmar ou 'Escape' para cancelar)

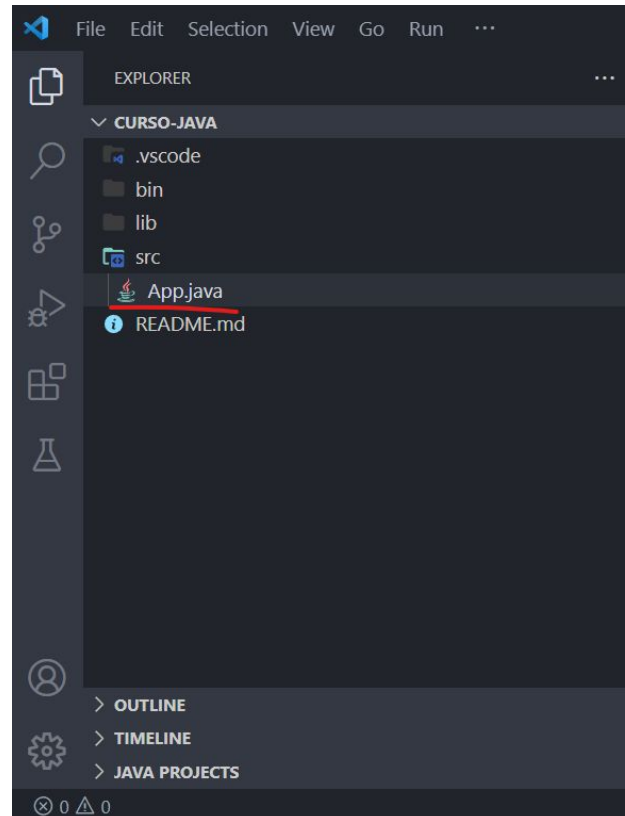
Criação de programas em Java

O VsCode vai abrir uma nova janela com o seu projeto já carregado e vai pedir pra você confirmar que confia nos autores dos arquivos na pasta do projeto. Pode clicar no botão azul.



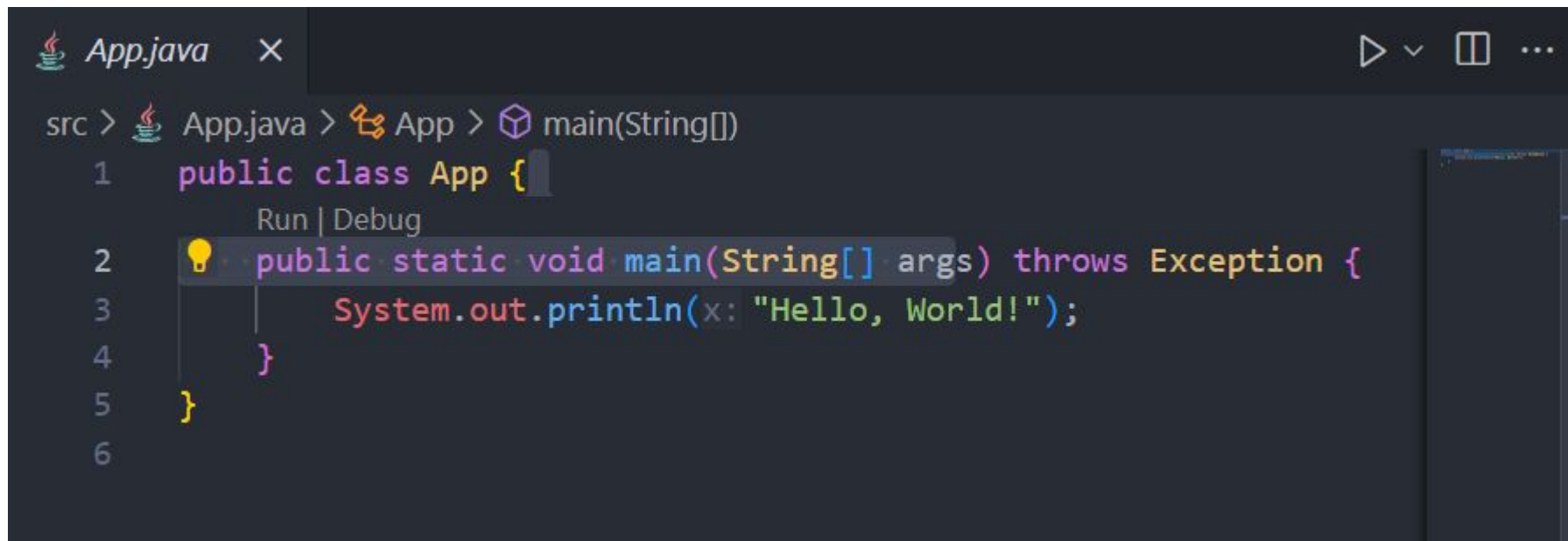
Criação de programas em Java

Pronto. Projeto criado e já aberto no Vs Code. Note que o Vs Code já cria uma classe inicial para nós. Com o nome App.java.



Criação de programas em Java

Ao clicar nesse arquivo, é aberto seu conteúdo no lado direito e podemos ver a estrutura de nosso pequeno programa inicial.



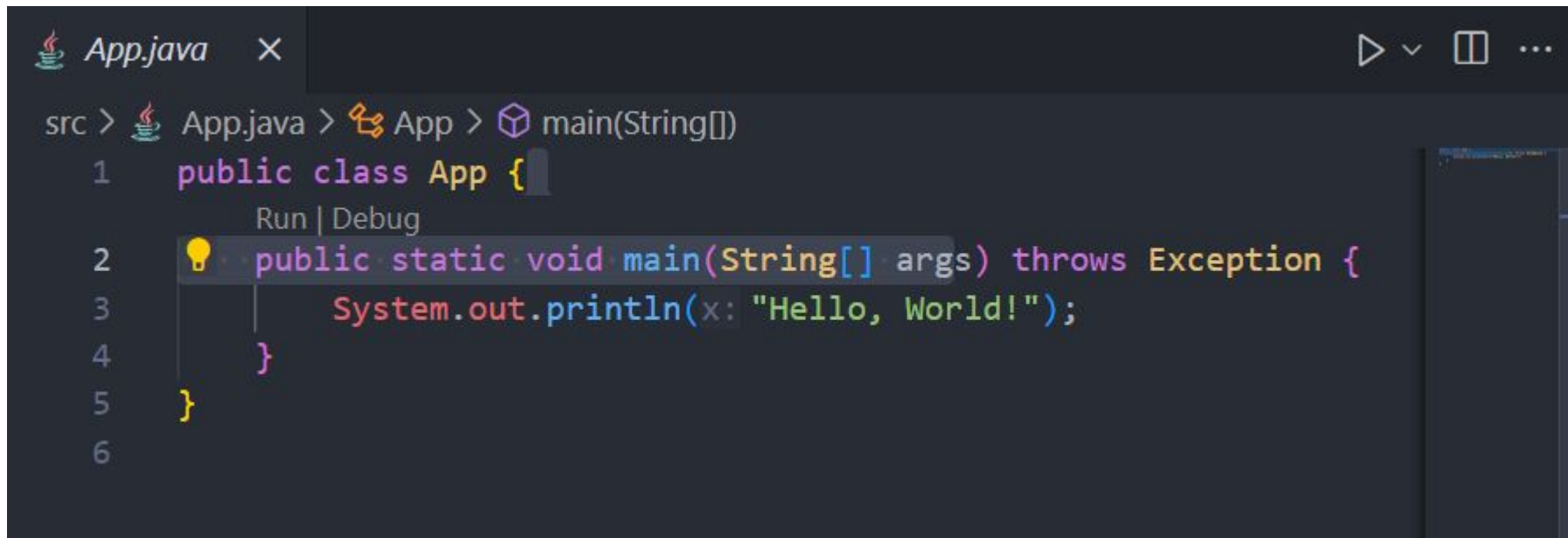
The screenshot shows an IDE window with a tab labeled 'App.java'. The breadcrumb navigation at the top reads 'src > App.java > App > main(String[])'. The code editor displays the following Java code:

```
1 public class App {  
2     public static void main(String[] args) throws Exception {  
3         System.out.println("Hello, World!");  
4     }  
5 }  
6
```

Line 2 is highlighted with a light blue background. A tooltip 'Run | Debug' is visible above line 2. The right side of the IDE shows a file explorer with a tree structure containing 'App.java'.

Criação de programas em Java

Vamos analisar a estrutura dessa classe. Toda classe em Java exige a presença da palavra reservada `class` seguida do nome da classe. Por convenção, todo nome de classe em Java inicia-se com letra maiúscula. Um par de chaves envolve todo o código da classe – sempre uma classe em Java possui uma chave que envolve o código.

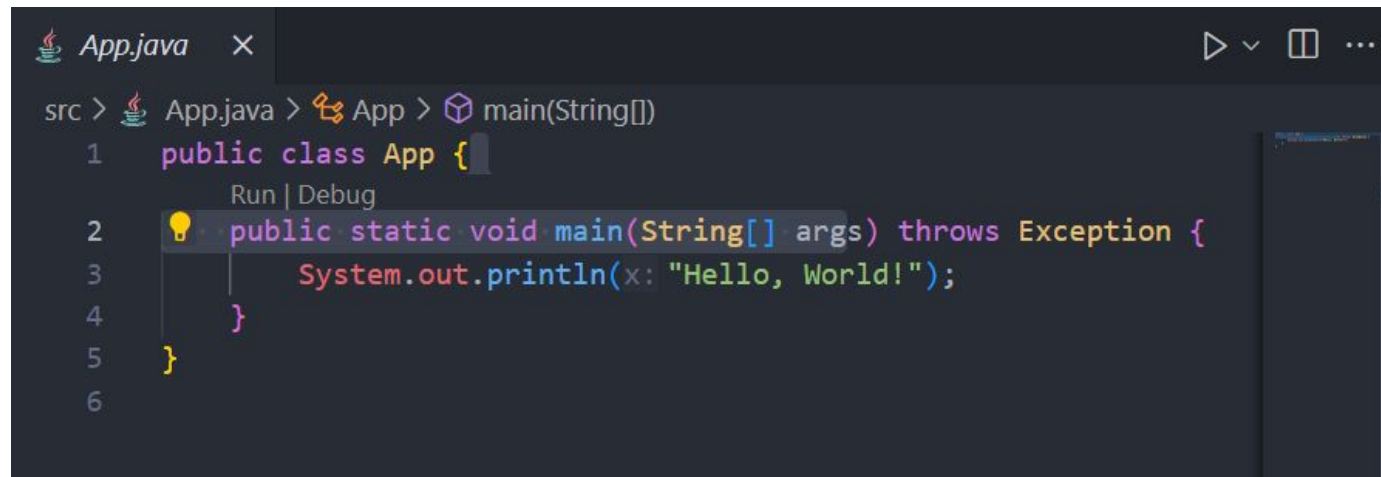


```
App.java ×
src > App.java > App > main(String[])
1 public class App {
    Run | Debug
2     public static void main(String[] args) throws Exception {
3         System.out.println(x: "Hello, World!");
4     }
5 }
6
```

Criação de programas em Java

Toda classe executável, isto é, toda classe que será interpretada e executada deve, obrigatoriamente, possuir o método main (principal), invocado quando a classe é executada.

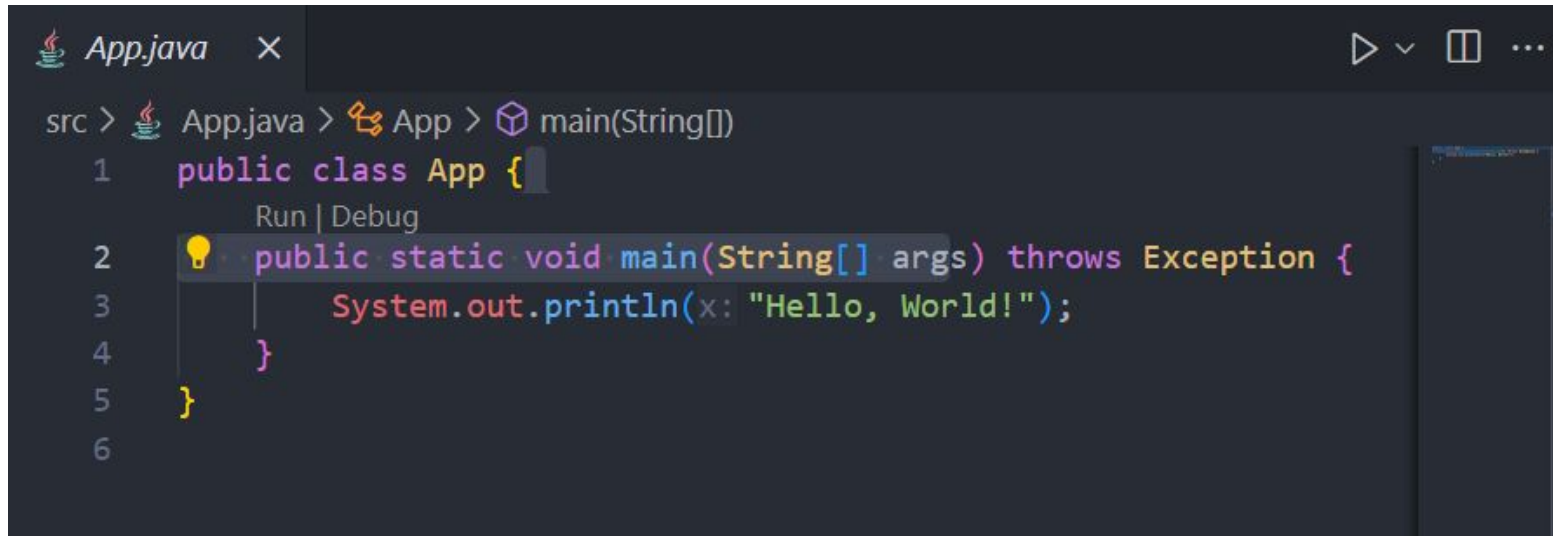
No exemplo, quando a classe for executada, será invocado o método main que possui uma instrução para envio de mensagem na tela (System.out.println). Não é exatamente uma instrução simples e sim uma classe da linguagem que possui um método especializado em saída de dados, mas vamos por partes.



```
App.java × [Run] [Debug] [Tools] [Help]
src > App.java > App > main(String[])
1 public class App {
2     public static void main(String[] args) throws Exception {
3         System.out.println("Hello, World!");
4     }
5 }
6
```

Criação de programas em Java

Procure salvar os arquivos em Java com o mesmo nome definido na classe. Por exemplo, se você criar uma classe com o nome Veiculo (class Veiculo), salve-a com o nome Veiculo.java. Esse procedimento é obrigatório quando uma classe é declarada como pública (public class Veiculo).



```
App.java x
src > App.java > App > main(String[])
1 public class App {
    Run | Debug
2 public static void main(String[] args) throws Exception {
3     System.out.println(x: "Hello, World!");
4 }
5 }
6
```

TIPOS DE DADOS



Tipos de dados

Antes de começarmos, vamos fazer uma pequena analogia. Um caminhão pode transportar diferentes objetos, e cada um deles pode exigir uma embalagem diferente para ser armazenado. Em função disso, cada objeto pode exigir um espaço físico diferente no compartimento de carga. Ao sair de viagem com o caminhão, seu Francisco carrega diversos produtos alimentícios, todos acondicionados em embalagens especiais. Isso é importante para que os produtos permaneçam bem acomodados no compartimento de carga.

Dessa forma, seu Francisco tem certeza de que os produtos chegarão ao destino da mesma maneira como iniciaram a viagem, isto é, suas características permanecerão as mesmas, os produtos chegarão ao destino de maneira intacta, sem sofrerem nenhum dano.

Tipos de dados

Para que o exposto seja verdade, é essencial utilizar a embalagem correta para cada tipo de produto. Por exemplo, a embalagem usada para transporte de bananas será diferente da usada para peras. O que aconteceria se seu Francisco tentasse colocar uma melancia numa caixa destinada a guardar caquis? A embalagem ideal para acondicionar caquis não pode ser a mesma usada para melancias, que seriam muito grandes para a embalagem; não é possível guardar uma melancia numa caixa de caquis.

De forma semelhante, em computação existem diferentes tipos de dados (tipos numéricos, caracteres, textuais etc.), e cada um precisa de um espaço diferente na memória do computador, cada um exige uma “embalagem” adequada para seu armazenamento. Cada embalagem pode ser considerada um tipo de dado diferente.

Tipos de dados

Considere que seja necessário armazenar na memória do computador os seguintes dados de uma pessoa: nome, sexo e idade. O nome contém uma sequência de caracteres, o sexo contém uma única letra (M ou F) e a idade, um valor numérico. Note que cada informação a ser armazenada exige uma “embalagem” diferente: o sexo, apenas uma letra (uma embalagem pequena), o nome, uma sequência de letras (uma embalagem maior), enquanto a idade deve manter um valor numérico inteiro (uma embalagem de tipo diferente).

Para armazenar esses dados, um programa deve definir variáveis, isto é, identificadores de um determinado tipo de dado.

Tipos de dados

Resumindo, antes de utilizar variáveis (a embalagem) é necessário definir que tipo de dado elas vão armazenar. Caso seja necessário armazenar a quantidade em estoque de um determinado produto, pode-se declarar uma variável do tipo inteira, ou para armazenar valores monetários é possível declarar uma variável do tipo double e assim por diante.

Na maioria das linguagens, quando um tipo de dado é utilizado, por exemplo, um tipo inteiro, pode ser que para uma determinada plataforma esse número seja armazenado com 16 bits e em outra com 32 bits. Em Java isso não ocorre, uma vez que um tipo de dado terá sempre a mesma dimensão, independentemente da plataforma utilizada. Esse e muitos outros recursos de portabilidade de Java permitem que os programadores escrevam programas sem conhecer a plataforma de computador em que eles serão executados.

Tipos de dados

Os tipos de dados no java podem ser primitivos ou não primitivos.

Tipo de dado primitivo: Incluem boolean, char, byte, short, int, long, float, e double.

Tipo de dado não primitivo: Incluem classes, wrappers, interfaces e arrays.

Tipos de dados

Tipo	Quantidade de bits	Valores
char	16	'\u0000' a '\uFFFF'
byte	8	-128 a + 127
int	32	-2.147.483.648 a +2.147.483.647
short	16	-32.768 a + 32.767
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
float	32	-3.40292347E+38 a +3.40292347E+38
double	64	-1.79769313486231570E+308 a +1.79769313486231570E+308
boolean	8	true ou false

Definição de variáveis e constantes

Uma variável ou constante é um tipo de identificador cujo nome, escolhido pelo programador, é associado a um valor pertencente a um certo tipo de dado. Em outras palavras, um identificador é a localização da memória (um endereço ou vários deles) capaz de armazenar o valor de um certo tipo, para o qual se dá um nome (o nome da variável, constante, objeto etc.) que usualmente descreve seu significado ou propósito. Dessa forma, todo identificador possui um nome, um tipo e um conteúdo. Os identificadores não podem utilizar as palavras reservadas da linguagem Java, assim como ocorre com outras linguagens de programação. A relação de palavras reservadas do Java é apresentada na próxima tela.

Palavras reservadas

As palavras reservadas em Java referem-se a nomes de instruções utilizadas na linguagem que não podem ser usadas como nomes de identificadores (variáveis, objetos etc.). O número de palavras reservadas aumentou em decorrência das diferentes versões do Java.

No link a seguir você encontrará todas as palavras reservadas do java.

[Palavras reservadas do java - Como Programar Java](#)

Definição de variáveis e constantes

A linguagem Java exige que todos os identificadores tenham um tipo de dado definido antes de serem utilizados no programa, ou seja, eles devem ser obrigatoriamente declarados, independentemente do ponto do programa, seja no início, no meio ou no final, desde que antes de sua utilização no programa.

Declaração de variáveis

Quando as variáveis são declaradas, a linguagem Java atribui a elas valores padrão, a menos que especificado de maneira contrária pelo programador. Atribui-se a todas as variáveis dos tipos char, byte, short, int, long, float e double o valor 0 por default. Já às variáveis do tipo boolean, por default, atribui-se false. Entretanto, dependendo do ponto do programa em que a variável é utilizada, torna-se obrigatória sua inicialização com algum valor, mesmo que a linguagem possua esses valores padrão.

As variáveis também possuem sensibilidade, isto é, ao declarar uma variável com o nome dolar, ela deve ser utilizada sempre da mesma forma. Não pode ser usada como Dolar, DOLAR, dOlar ou qualquer outra variação, apenas com todas as letras minúsculas, como realizado em sua declaração.

Declaração de variáveis

Quando as variáveis são declaradas, a linguagem Java atribui a elas valores padrão, a menos que especificado de maneira contrária pelo programador. Atribui-se a todas as variáveis dos tipos char, byte, short, int, long, float e double o valor 0 por default. Já às variáveis do tipo boolean, por default, atribui-se false. Entretanto, dependendo do ponto do programa em que a variável é utilizada, torna-se obrigatória sua inicialização com algum valor, mesmo que a linguagem possua esses valores padrão.

As variáveis também possuem sensibilidade, isto é, ao declarar uma variável com o nome dolar, ela deve ser utilizada sempre da mesma forma. Não pode ser usada como Dolar, DOLAR, dOlar ou qualquer outra variação, apenas com todas as letras minúsculas, como realizado em sua declaração.

Declaração de variáveis

Os nomes de variáveis devem começar com letra, caractere de sublinhado (_) ou cifrão (\$). Não é permitido iniciar o nome de uma variável com número. Por convenção, a linguagem Java utiliza o seguinte padrão para nomear as variáveis (e outros identificadores):

- Quando o nome da variável for composto apenas por um caractere ou palavra, os caracteres devem ser minúsculos.
- Quando o nome da variável tiver mais de uma palavra, a primeira letra, da segunda palavra em diante, deve ser maiúscula. Todos os outros caracteres devem ser minúsculos.

Exemplos de nomes de variáveis: a, a1, \$preco, nome, valorVenda, codigoFornecedor.

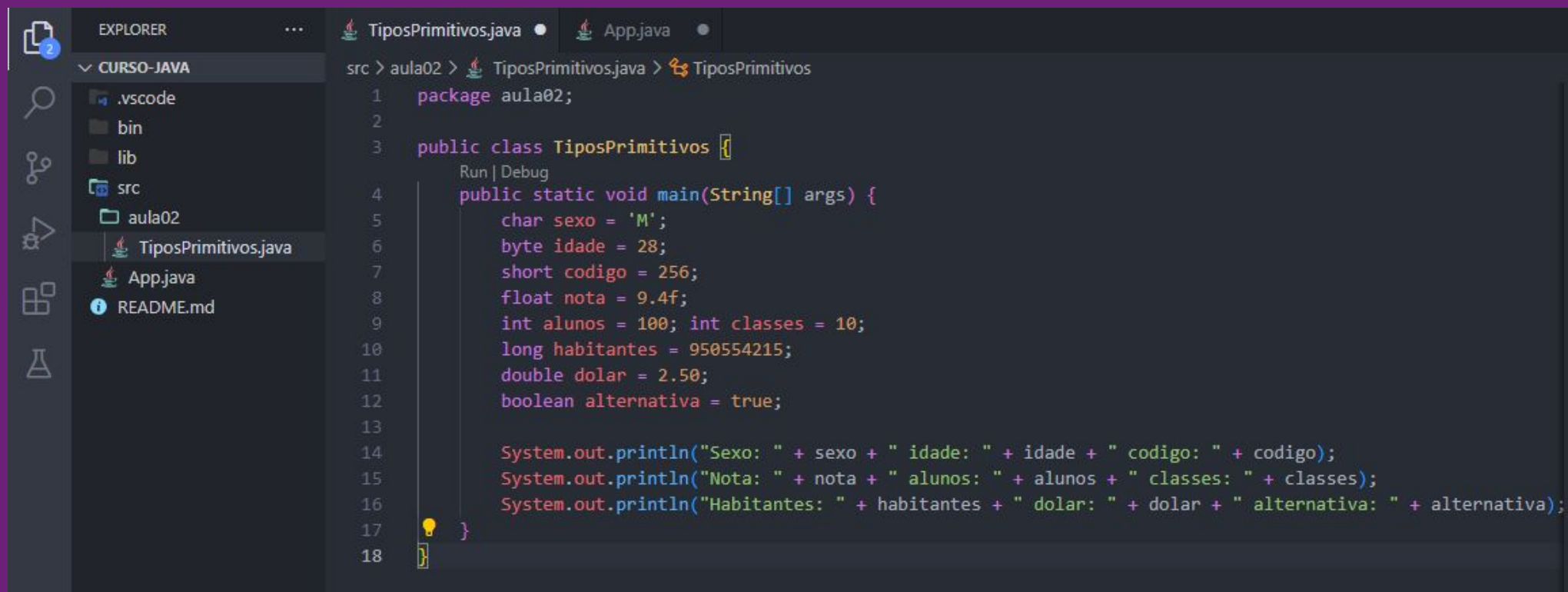
Declaração de variáveis



```
1  public class App {  
2      public static void main(String[] args) throws Exception {  
3          char sexo = 'M';  
4          byte idade = 28;  
5          short codigo = 256;  
6          float nota = 9.4f;  
7          int alunos = 100; int classes = 10;  
8          long habitantes = 950554215;  
9          double dolar = 2.50;  
10         boolean alternativa = true;  
11  
12         System.out.println("Sexo: " + sexo + " idade: " + idade + " codigo: " + codigo);  
13         System.out.println("Nota: " + nota + " alunos: " + alunos + " classes: " + classes);  
14         System.out.println("Habitantes: " + habitantes + " dolar: " + dolar + " alternativa: " + alternativa);  
15     }  
16 }
```

Declaração de variáveis

Para testar, crie um novo pacote com o nome aula02 e dentro dele, crie uma classe com o nome TiposPrimitivos, em seguida, crie variáveis e atribua valores a elas e use o comando `System.out.println()` para mostrar o resultado. A estrutura dos arquivos ficará como essa abaixo:



The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar displays the project structure for 'CURSO-JAVA'. It includes folders for '.vscode', 'bin', 'lib', and 'src'. Inside 'src', there is a folder named 'aula02' which contains the file 'TiposPrimitivos.java'. Other files in the project are 'App.java' and 'README.md'. The main editor area shows the code for 'TiposPrimitivos.java'. The code defines a package 'aula02', a public class 'TiposPrimitivos', and a main method. Inside the main method, several primitive variables are declared and assigned values: 'sexo' (char), 'idade' (byte), 'codigo' (short), 'nota' (float), 'alunos' (int), 'classes' (int), 'habitantes' (long), 'dolar' (double), and 'alternativa' (boolean). These values are then printed to the console using `System.out.println()`.

```
src > aula02 > TiposPrimitivos.java > TiposPrimitivos
1  package aula02;
2
3  public class TiposPrimitivos {
4      public static void main(String[] args) {
5          char sexo = 'M';
6          byte idade = 28;
7          short codigo = 256;
8          float nota = 9.4f;
9          int alunos = 100; int classes = 10;
10         long habitantes = 950554215;
11         double dolar = 2.50;
12         boolean alternativa = true;
13
14         System.out.println("Sexo: " + sexo + " idade: " + idade + " codigo: " + codigo);
15         System.out.println("Nota: " + nota + " alunos: " + alunos + " classes: " + classes);
16         System.out.println("Habitantes: " + habitantes + " dolar: " + dolar + " alternativa: " + alternativa);
17     }
18 }
```

Declaração de constantes

Uma constante é um tipo de variável que não pode alterar seu conteúdo depois de ter sido inicializado, ou seja, o conteúdo permanece o mesmo durante toda a execução do programa. Na realidade não existem constantes em Java, o que existe é um tipo de variável com comportamento semelhante a uma constante de outras linguagens. Em Java, essa variável é definida como final.

Essas constantes são usadas para armazenar valores fixos, geralmente definidos no início de uma classe. Em Java, padronizou-se identificar as variáveis do tipo final com todas as letras maiúsculas, e quando existe mais de uma palavra elas são separadas pelo caractere de underline (_). Exemplos de valores constantes são: o valor de PI na matemática (3.14), o valor da aceleração na física (9.81 m/s²), a quantidade de milissegundos existentes em um segundo (1.000) etc.

Declaração de constantes

veja alguns exemplos seguintes de declaração de variáveis do tipo final:



```
1  package aula02;
2
3  public class Constantes {
4      public static void main(String[] args) {
5          final double PI = 3.14;
6          final int MILISEGUNDOS_POR_SEGUNDO = 1000;
7          final long MILISEGUNDOS_POR_DIA = 24 * 60 * 60 * 1000;
8
9          System.out.println("PI : " + PI);
10         System.out.println("Milissegundos por segundo : " + MILISEGUNDOS_POR_SEGUNDO);
11         System.out.println("Milissegundos por dia : " + MILISEGUNDOS_POR_DIA);
12     }
13 }
```

Declaração de constantes

Caso um segundo valor seja atribuído a uma variável final no decorrer da classe, o compilador gera uma mensagem de erro. Não é obrigatório inicializar o conteúdo de uma variável final no momento de sua declaração. É possível realizar esses dois processos em locais distintos, conforme apresentado em seguida.

```
1  package aula02;  
2  
3  public class Constantes {  
4      Run | Debug  
5      public static void main(String[] args) {  
6  
7          final int MILIMETROS_POR_CENTIMETRO = 10;  
8  
9          MILIMETROS_POR_CENTIMETRO = 20;  
10     }  
11 }
```

Operadores

A linguagem Java oferece um amplo conjunto de operadores destinados à realização de operações aritméticas, lógicas e relacionais, com a possibilidade de formar expressões de qualquer tipo. Além dos operadores matemáticos (aritméticos), existem também operadores lógicos e relacionais, conforme abordado em seguida.

Função	Sinal	Exemplo
Adição	+	$x + y$
Subtração	-	$x - y$
Multiplicação	*	$x * y$
Divisão	/	x / y
Resto da divisão inteira	%	$x \% y$
Sinal negativo	-	$-x$
Sinal positivo	+	$+x$
Incremento unitário	++	$++x$ ou $x++$
Decremento unitário	--	$--x$ ou $x--$

Operadores relacionais

Os operadores relacionais possibilitam comparar valores ou expressões, retornando um resultado lógico verdadeiro ou falso.

Função	Caractere(s) utilizado(s)	Exemplo
Igual	<code>==</code>	<code>x == y</code>
Diferente	<code>!=</code>	<code>x != y</code>
Maior que	<code>></code>	<code>x > y</code>
Maior ou igual a	<code>>=</code>	<code>x >= y</code>
Menor que	<code><</code>	<code>x < y</code>
Menor ou igual a	<code><=</code>	<code>x <= y</code>

Operadores lógicos

São operadores que permitem avaliar o resultado lógico de diferentes operações aritméticas em uma expressão.

Função	Caractere(s) utilizado(s)	Exemplo
E lógico ou AND	&&	x && y
Ou lógico ou OR		x y
Negação ou NOT	!	!x

Sequências de escape

A linguagem de programação Java também possui algumas sequências de escape, que são atalhos utilizados para representar um caractere especial como, por exemplo, uma quebra de linha (`\n`).

Sequência	Caractere especial
<code>\b</code>	Espaço
<code>\f</code>	Form feed
<code>\n</code>	Nova linha
<code>\r</code>	Retorno
<code>\t</code>	Tabulação
<code>\"</code>	Aspas duplas
<code>\'</code>	Aspas simples
<code>\\</code>	Barra invertida


Conversão de tipos

Em diversas aplicações em Java é preciso realizar a conversão dos diversos tipos primitivos existentes. A linguagem Java possui uma série de classes que realizam essa tarefa. A Tabela apresenta algumas conversões de tipos. As palavras principais usadas na conversão aparecem em **negrito**.

Supondo a variável x	Converter em	y recebe o valor convertido
int x = 10	float	float y = (float) x
int x = 10	double	double y = (double) x
float x = 10.5	int	int y = (int) x
String x = "10"	int	int y = Integer.parseInt (x)
String x = "20.54"	float	float y = Float.parseFloat (x)
String x = "20.54"	double	double y = Double.parseDouble (x)
String x = "Java"	Vetor de bytes	byte b[] = x.getBytes ()
int x = 10	String	String y = String.valueOf (x)
float x = 10.35	String	String y = String.valueOf (x)
double x = 254.34	String	String y = String.valueOf (x)
byte x[] – (x é um vetor de bytes)	String	String y = new String (x)

Entrada de dados pelo teclado

Para recebermos a entrada de dados via teclado o java nos disponibiliza algumas classes. Vamos utilizar a classe Scanner.



```
1 package aula02;
2
3 import java.util.Scanner;
4
5 public class LeituraDeDados {
6
7     public static void main(String[] args) {
8         int idade = 0;
9         Scanner scan = new Scanner(System.in);
10
11         System.out.println("Insira sua idade: ");
12         idade = scan.nextInt();
13
14         System.out.println("A idade digitada foi: " + idade);
15     }
16
17 }
```

A classe Scanner possui métodos específicos para leitura de diferentes tipos de dados. Exemplos de alguns métodos similares para leitura de valores numéricos são: `nextByte`, `nextShort`, `nextInt`, `nextLong`, `nextDouble`, entre outros.

Caixa de diálogo para a entrada de dados

A linguagem Java dispõe de uma forma gráfica para receber dados do usuário. Trata-se da utilização de caixas de diálogo, no caso a caixa gerada a partir da classe JOptionPane.



```
1  package aula02;
2
3  import javax.swing.JOptionPane;
4
5  public class LeituraDeDados {
6
7      public static void main(String[] args) {
8          int idade = 0;
9
10         idade = Integer.parseInt(JOptionPane.showInputDialog("Insira sua idade: "));
11         System.out.println("A idade digitada foi: " + idade);
12     }
13
14 }
```

Caixa de diálogo para a entrada de dados

O método `showInputDialog("msg")` pode ser usado para coletar dados e, para salvar numa variável, temos sempre que verificar o tipo e fazer a conversão antes pois este método retorna sempre uma `String`.



```
1 package aula02;
2
3 import javax.swing.JOptionPane;
4
5 public class LeituraDeDados {
6
7     public static void main(String[] args) {
8         int idade = 0;
9
10        idade = Integer.parseInt(JOptionPane.showInputDialog("Insira sua idade: "));
11        System.out.println("A idade digitada foi: " + idade);
12    }
13
14 }
```

Caixa de diálogo para a entrada de dados

Para saber o que um método está retornando, pare o cursor do mouse sobre ele um instante e logo uma pequena janela com a definição do método se abrirá, podemos ver no início da definição o tipo de retorno. Veremos melhor isso mais à frente.

```
String javax.swing.JOptionPane.showInputDialog(Object message) throws HeadlessException
```

Shows a question-message dialog requesting input from the user. The dialog uses the default frame, which usually means it is centered on the screen.

- **Parameters:**
 - **message** the `Object` to display
- **Returns:**
 - user's input
- **Throws:**
 - `HeadlessException` - if `GraphicsEnvironment.isHeadless` returns `true`
- **See Also:**

Exercícios

1 - Crie uma classe que receba o valor de um produto e a porcentagem de desconto, calcule e mostre o valor do desconto e o valor do produto com o desconto. Observação: o valor do desconto é calculado por meio da fórmula: $\text{valor do desconto} = \text{valor do produto} * \text{percentual de desconto} / 100$.

2 - Usando a classe Scanner para entrada de dados, faça uma classe que receba dois valores inteiros. O primeiro valor corresponde à quantidade de pontos do líder do campeonato brasileiro de futebol. O segundo valor corresponde à quantidade de pontos do time lanterna. Considerando que cada vitória vale 3 pontos, elabore uma classe que calcule o número de vitórias necessárias para que o time lanterna alcance (ou ultrapasse) o líder. Por exemplo, supondo que as quantidades de ponto fornecidas sejam 40 e 22, então o número de vitórias apresentada na saída deverá ser 6, pois $(40-22) / 3 = 6$.

Exercícios

1 - Crie uma classe que receba o valor de um produto e a porcentagem de desconto, calcule e mostre o valor do desconto e o valor do produto com o desconto. Observação: o valor do desconto é calculado por meio da fórmula: $\text{valor do desconto} = \text{valor do produto} * \text{percentual de desconto} / 100$.

2 - Usando a classe Scanner para entrada de dados, faça uma classe que receba dois valores inteiros. O primeiro valor corresponde à quantidade de pontos do líder do campeonato brasileiro de futebol. O segundo valor corresponde à quantidade de pontos do time lanterna. Considerando que cada vitória vale 3 pontos, elabore uma classe que calcule o número de vitórias necessárias para que o time lanterna alcance (ou ultrapasse) o líder. Por exemplo, supondo que as quantidades de ponto fornecidas sejam 40 e 22, então o número de vitórias apresentada na saída deverá ser 6, pois $(40-22) / 3 = 6$.

transforme ■ se

O conhecimento é o poder
de transformar o seu futuro.