

Antes de iniciar a aula, vou passar algumas coisas importantes:

- O que é uma variável em Java?

Uma **variável** é uma área de memória, associada a um **nome**, que pode armazenar **valores** de um determinado **tipo**. Um tipo de dado, define um conjunto de valores e um conjunto de operações. Java é uma linguagem fortemente tipada, diferente de linguagens como JavaScript, onde declarar o tipo da variável não é obrigatório.

- Declaração de Variáveis:

A declaração de uma variável é a maneira de informar ao compilador que você pretende reservar um espaço na memória para armazenar um determinado tipo de dado. A sintaxe básica para declarar uma variável em Java é:

```
<tipoDaVariavel> <nomeDaVariavel>;
```

Por exemplo, para declarar uma variável inteira (**int**) chamada **idade**, você faria:

```
int idade;
```

- Atribuição de Variáveis:

A atribuição de variáveis é o processo de armazenar um valor na variável previamente declarada. Utilizamos o operador de atribuição (=) para realizar essa tarefa. Por exemplo:

```
idade = 25;
```

 Aqui, atribuímos o valor 25 à variável **idade**.

Também pode **combinar a declaração e a atribuição** em uma única linha:

```
int idade = 25;
```

- Tipos de Variáveis em Java:

TIPO	VALORES	OBSERVAÇÃO	BITS EM MEMÓRIA	EXEMPLO
boolean	true ou false	O valor booleano assume valores true (verdadeiro) ou false (falso).	Podem variar entre diferentes JVMs e compiladores Java (1 bit ou 8 bits)	true
byte	Entre -128 e 127	Números inteiros de 8 bits de precisão	8	108
char	Entre 0 a 65535	Na teoria, char armazena números inteiros de 16 bits de precisão, mas na prática, se utilizam para qualquer caracter alfanumérico (char = character)	16	'a'
double	Entre 4.94065645841246544e-324 e 1.7976931348623157e+308	Pontos flutuantes de precisão dupla, usados para representar valores decimais	64	3.37
float	Entre 1.40239846e-46 e 3.40282347e+38	Pontos flutuantes de precisão simples, usados para representar valores decimais	32	3.37F
Int	Entre 2.147.483.648 e 2.147.483.647	Números inteiros de 32 bits de precisão	32	10
long	Entre -9.223.372.036.854.775.808 e +9.223.372.036.854.775.807	Números inteiros de 64 bits de precisão	64	7234829L
short	Entre -32768 e 32767	Números inteiros de 16 bits de precisão	16	10

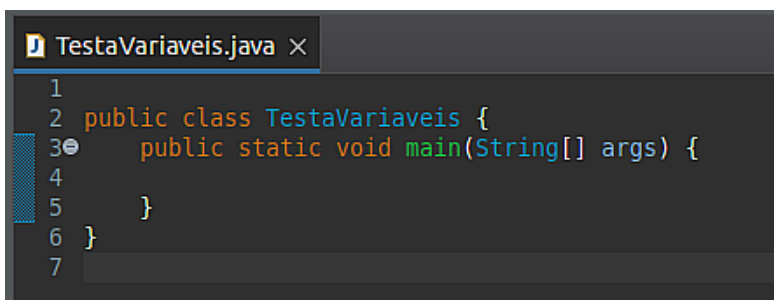
Agora vamos para a aula: nessa aula, abordaremos a criação de um novo projeto Java no Eclipse, abordando a criação de variáveis e operações básicas.

1- Criação do Projeto:

- Iniciamos criando um novo projeto chamado "sintaxe-variaveis-e-tipos" no Eclipse, acessando "New > Java Project".
- Observamos que o diretório "src" é visível na view de "Package Explorer", enquanto o diretório "bin" está oculto, pois o Eclipse prioriza mostrar o código fonte Java. O diretório 'bin' pode ser vista no Project Explorer (foi explicado na aula passada).

2- Criação da Classe "TestaVariaveis":

- Cria uma nova classe chamada "TestaVariaveis" dentro do diretório "src" para trabalhar com variáveis.
- A estrutura básica da classe deve incluir o método `main`:
(O método `main` é o ponto de entrada principal para a execução do programa Java. Ele é onde a execução do programa começa. E todo o código que deve ser executado quando o programa é iniciado está contido no corpo do método `main`.)
- Até então temos isso:

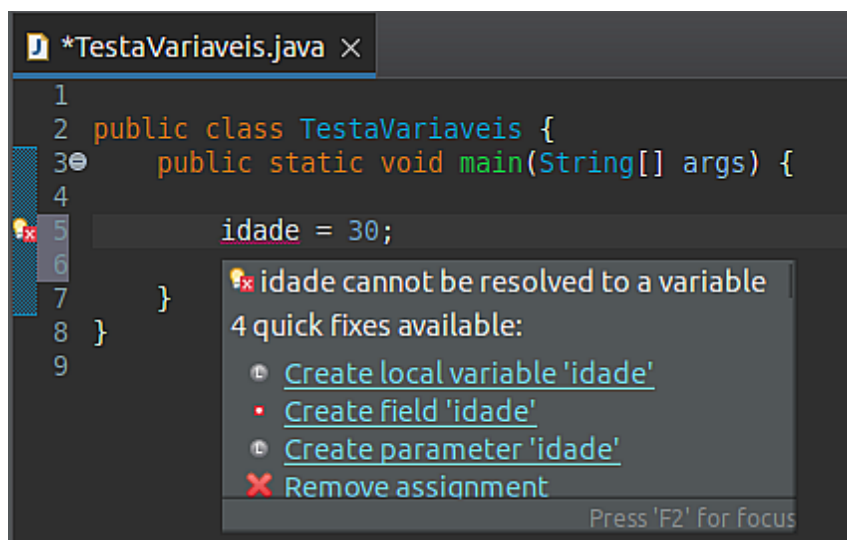


```
1
2 public class TestaVariaveis {
3     public static void main(String[] args) {
4
5     }
6 }
7
```

3- Declaração e Uso de Variáveis:

Objetivo: criar uma variável para guardar minha idade.

- Tentamos **atribuir** (guardar) um valor à variável `idade` dessa forma: `idade = 30;`



```
1
2 public class TestaVariaveis {
3     public static void main(String[] args) {
4
5         idade = 30;
6
7     }
8 }
9
```

idade cannot be resolved to a variable
4 quick fixes available:

- Create local variable 'idade'
- Create field 'idade'
- Create parameter 'idade'
- Remove assignment

Press 'F2' for Focus

No Java, como o Eclipse já está dando a entender sublinhando idade com vermelho, não compila isto, pois trata-se de uma linguagem **estaticamente ou fortemente tipada**, ou seja, que necessita da **declaração** de todas as variáveis e **tipos** a serem utilizados.

Passando o mouse sobre a palavra sublinhada (idade), lê-se a mensagem de erro que aparece na imagem acima "**idade cannot be resolved to a variable**".

Significa que "idade não pode ser entendida como uma variável", pois **não foi declarada**.

O Eclipse inclusive dará algumas opções de "rápido conserto", ou *quick fix*, como a criação local da variável, ou remoção da linha, por exemplo.

`idade = 37` é uma **atribuição**, em que 37 se encontra dentro de `idade`.

Precisaremos **declará-la** informando que ela é do **tipo** numérico e que guarda um valor inteiro, sem decimais ou pontos flutuantes.

Como foi visto no início desse pdf, em Java, o tipo de uma variável refere-se ao tipo de dado que a variável pode armazenar. Os tipos de dados em Java são divididos em dois grupos principais: tipos primitivos e tipos de referência, até então vamos ver os tipos primitivos.

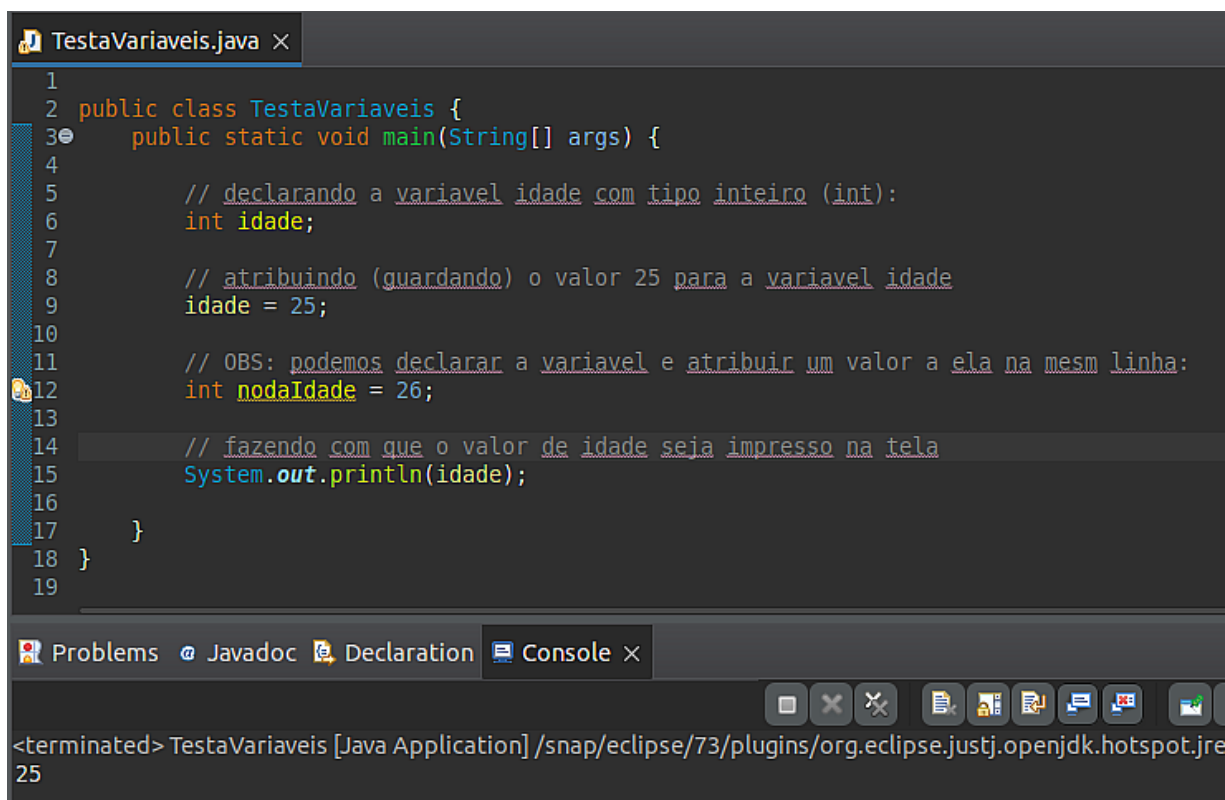
Então para atribuímos o valor da idade, primeiro devemos **declarar** a variável. Para declarar usamos o tipo e o nome da variável:

```
int idade;
```

Agora sim podemos **atribuir** um valor a ela:

```
int idade = 25;
```

Por fim será impresso através do "System.out.println()" o valor da variável idade. Então teremos o seguinte código:



```
1
2 public class TestaVariaveis {
3     public static void main(String[] args) {
4
5         // declarando a variavel idade com tipo inteiro (int):
6         int idade;
7
8         // atribuindo (guardando) o valor 25 para a variavel idade
9         idade = 25;
10
11        // OBS: podemos declarar a variavel e atribuir um valor a ela na mesm linha:
12        int nodaIdade = 26;
13
14        // fazendo com que o valor de idade seja impresso na tela
15        System.out.println(idade);
16
17    }
18 }
19
```

Problems Javadoc Declaration Console

<terminated> TestaVariaveis [Java Application] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre

25

Obs: as linhas que começam com " //" é vista apenas como comentários, não faz parte do programa

E após a execução (que foi aprendemos em aulas passadas) a saída do console será o valor da variável idade: 25

Trabalhando Operadores Aritméticos:

Temos nossa variável idade com valor = 25.

Agora iremos somar +1 a essa variável:

```
idade = idade + 1; → idade tinha valor 25, ao somar +1 ela deve ter 26
```

Podemos utilizar a variável em uma expressão numérica também.

Em uma expressão onde temos multiplicação (*), Divisão (/), Adição (+) e Subtração (-), A ordem de prioridade será:

* ou / depois vem + ou -

Caso queira priorizar a soma ou a subtração deve colocá-las entre **parênteses**.

```
int idade = 2 + 3 * 4; // Aqui, a multiplicação ocorre primeiro
```

```
int idade = (2 + 3) * 4; // Aqui, a adição ocorre primeiro
```

Operador (+) vai além da soma:

No Java o operador + também serve para concatenar(juntar) coisas.

Por exemplo digamos que queremos imprimir o valor da idade após a expressão numérica, podemos especificar com um texto entre aspas e em seguida concatenar com a variável, assim:

```
System.out.println("valor de idade após expressão numérica: " + idade);
```

Após fazer esses passos nosso código ficou assim:

!º Parte:

```
TestaVariaveis.java x
1
2 public class TestaVariaveis {
3     public static void main(String[] args) {
4
5         // declarando a variavel idade com tipo inteiro (int):
6         int idade;
7
8         // atribuindo (guardando) o valor 25 para a variavel idade
9         idade = 25;
10
11        // OBS: podemos declarar a variavel e atribuir um valor a ela na mesma linha:
12        int nodaIdade = 26;
13
14        // fazendo com que o valor de idade seja impresso na tela
15        System.out.println(idade);
16    }
```

2º Parte:

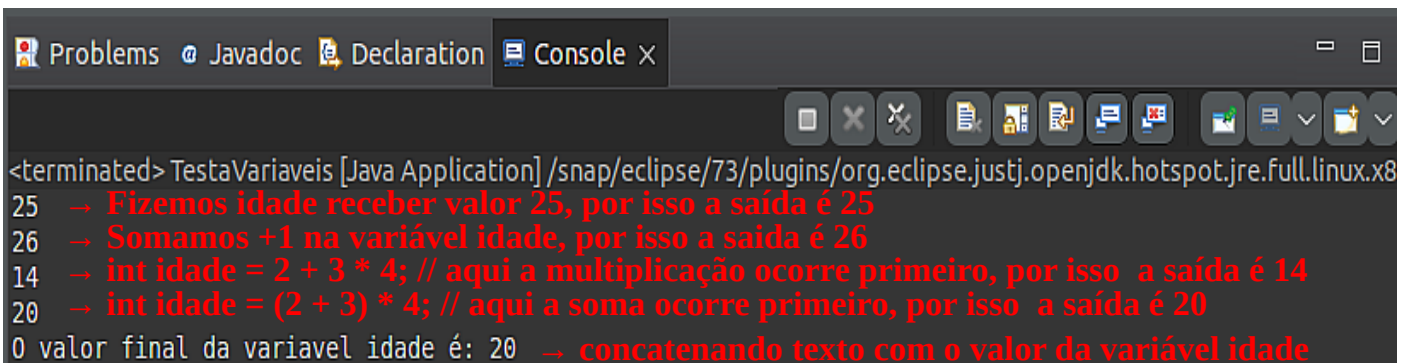
```
16
17 // ##### TRABALHANDO COM OPERADORES ARITMETICOS #####
18
19 // somar +1 a variável idade:
20 idade = idade +1;
21
22 // imprimindo o valor da idade apos ser somado +1
23 System.out.println(idade);
24
25 /* Podemos utilizar a variável em uma expressão numérica também.
26  * Onde temos multiplicação (*), Divisão (/), Adição (+) e Subtração (-),
27  * A ordem de prioridade será:
28  * multiplicação (*) ou Divisão (/) depois vem Soma (+) ou Subtração (-)
29  *
30  * Caso queira priorizar a soma ou a subtração deve colocá-las entre parênteses.
31  */
32 idade = 2 + 3 * 4; // Aqui, a multiplicação ocorre primeiro
33
34 // imprimindo valor de idade apos a expressao onde a multiplicacao ocorre primeiro
35 System.out.println();
36
37 idade = (2 + 3) * 4; // Aqui, a adição ocorre primeiro
38
39 // imprimindo valor de idade apos a expressao onde a soma ocorre primeiro
40 System.out.println(idade);
41
```

3º Parte:

```
42
43 // ##### No Java o operador + também serve para concatenar(juntar) coisas #####
44
45 // Juntando o texto entre aspas + o valor da variavel idade:
46 System.out.println("O valor final da variavel idade é: " + idade);
47 }
48 }
```

Resultado no console após executar o programa:

(A parte de vermelho é apenas uma explicação, ela não faz parte do resultado)



```
<terminated> TestaVariaveis [Java Application] /snap/eclipse/73/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64
25 → Fizemos idade receber valor 25, por isso a saída é 25
26 → Somamos +1 na variável idade, por isso a saída é 26
14 → int idade = 2 + 3 * 4; // aqui a multiplicação ocorre primeiro, por isso a saída é 14
20 → int idade = (2 + 3) * 4; // aqui a soma ocorre primeiro, por isso a saída é 20
0 valor final da variavel idade é: 20 → concatenando texto com o valor da variável idade
```

Mais do Java:

CamelCase:

É uma convenção de nomenclatura em que as palavras são unidas sem espaços e cada palavra subsequente é capitalizada.

Exemplo de CamelCase: `nomeDaVariavel`, `calcularTotal`, `minhaClasse`.

Palavras longas e não abreviadas:

Em Java, é comum usar nomes descritivos e completos para variáveis, métodos e classes. Isso torna o código mais legível e compreensível.

Por exemplo, em vez de usar `idadeUsr` como nome de variável, prefira algo mais descritivo como `idadeDoUsuario` ou ao invés de `totItens` use `totalDeItens`.

Nome de Classe:

O nome de uma classe em Java geralmente começa com uma letra maiúscula e segue a convenção CamelCase.

Exemplo: `MinhaClasse`, `UsuarioController`, `Calculadora`.

Nome de Variável:

Nomes de variáveis em Java geralmente começam com letra minúscula e seguem a convenção CamelCase.

Exemplo: `idadeDoUsuario`, `nomeCompleto`, `quantidadeItens`.

System.out.print:

`System.out.print` é usado quando você deseja imprimir uma mensagem **sem** adicionar uma nova linha ao final, diferente do `print` com o `ln` no final: `println`