O Básico da Linguagem

Emerson C. Lima

Faculdade de Educação Tecnológica do Estado Rio de Janeiro

13 de agosto de 2020

Variáveis

Variáveis

A linguagem de programação Java possui os seguintes tipos de variáveis:

- Variáveis de instância (campos não estáticos)
- Variáveis de classe (campos estáticos)
- Variáveis locais
- Parâmetros

Regras e convenções para nomeclatura de variáveis em Java:

- Os nomes são sensíveis à caixa.
- Caracteres subsequentes podem ser letras, dígitos, sifrão ($) ou _.
- Se o nome que você escolheu é uma palavra, coloque-a em minúscula.
- Se é formado por mais de uma palavra, coloque a primeira letra de cada palavra subsequênte em maiúscula.
- Se é sua variável armazena uma constante, coloque em maiúscula separando as palavras com _.

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`
- `long`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`
- `long`
- `float`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`
- `boolean`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte`
- `short`
- `int`
- `long`
- `float`
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short`
- `int`
- `long`
- `float`
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int`
- `long`
- `float`
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long`
- `float`
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long` - complemento de dois 64-bit
- `float`
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long` - complemento de dois 64-bit
- `float` - ponto-flutuante IEEE 754 32-bit
- `double`
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long` - complemento de dois 64-bit
- `float` - ponto-flutuante IEEE 754 32-bit
- `double` - ponto-flutuante IEEE 754 64-bit
- `boolean`
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long` - complemento de dois 64-bit
- `float` - ponto-flutuante IEEE 754 32-bit
- `double` - ponto-flutuante IEEE 754 64-bit
- `boolean` - true ou false
- `char`

Variáveis

Tipos Primitivos

A Linguagem de Programação Java é "tipada-estaticamente", ou seja, precisamos declarar todas as variáveis antes que possam ser utilizadas.

```
int gear = 1;
```

Tipos primitivos suportados pela linguagem Java:

- `byte` - complemento de dois 8-bit
- `short` - complemento de dois 16-bit
- `int` - complemento de dois 32-bit
- `long` - complemento de dois 64-bit
- `float` - ponto-flutuante IEEE 754 32-bit
- `double` - ponto-flutuante IEEE 754 64-bit
- `boolean` - true ou false
- `char` - caracter Unicode 16-bit \u0000 (0) à (65.535)

Variáveis

Tipo `java.lang.String`

Java fornece suporte especial para cadeias de caracteres pela classe `java.lang.String`.

Variáveis

Tipo `java.lang.String`

Java fornece suporte especial para cadeias de caracteres pela classe `java.lang.String`.

Colocando uma cadeia de caracteres entre aspas automaticamente cria um objeto `String`.

Variáveis

Tipo `java.lang.String`

Java fornece suporte especial para cadeias de caracteres pela classe `java.lang.String`.

Colocando uma cadeia de caracteres entre aspas automaticamente cria um objeto `String`.

```
String s = "this is a string";
```

Variáveis

Tipo `java.lang.String`

Java fornece suporte especial para cadeias de caracteres pela classe `java.lang.String`.

Colocando uma cadeia de caracteres entre aspas automaticamente cria um objeto `String`.

```
String s = "this is a string";
```

Objetos `String` são imutáveis.

Variáveis

Tipo `java.lang.String`

Java fornece suporte especial para cadeias de caracteres pela classe `java.lang.String`.

Colocando uma cadeia de caracteres entre aspas automaticamente cria um objeto `String`.

```
String s = "this is a string";
```

Objetos `String` são imutáveis.

A classe `String` não é um tipo primitivo, mas considerando o tratamento especial dado pela linguagem, você poderia pensar que sim.

Operadores

Operadores

Operadores Aritméticos

```
+   Operador aditivo (usado também
       para concatenação de Strings)
-   Operador de subtração
*   Operador de multiplicação
/   Operador de divisão
%   Operador de resto
```

Podemos combinar os operadores aritméticos com o operador de atribuição (=):

```
x += 1; x = x + 1;
```

O operador + também pode ser usado para concatenar duas Strings:

```
class ConcatDemo {
    public static void main(String[] args){
        String firstString = "This is";
        String secondString = " a concatenated string.";
        String thirdString = firstString+secondString;
        System.out.println(thirdString);
    }
}
```

Operadores

Operadores Unários

```
-   Operador menos unário, negativa uma expressão
++  Operador de incremento
--  Operador de decremento
!   Operador de complemento lógico
```

Operadores

Operadores de Igualdade e Relacionais

```
==      igual à
!=      diferente
>       maior que
>=      maior ou igual à
<       menor que
<=      menor ou igual à
```

Operadores

Operadores Condicionais

`&&` E-Condicional

`||` Ou-Condicional

Operadores

Operador `instanceof`

O operador `instanceof` compara se um objeto é de um tipo especificado.

Operadores

Operador `instanceof`

O operador `instanceof` compara se um objeto é de um tipo especificado.

```
if (obj1 instanceof Parent) ...
```

Operadores

Operador `instanceof`

O operador `instanceof` compara se um objeto é de um tipo especificado.

```
if (obj1 instanceof Parent) ...
```

```
"oi "+ (obj1 instanceof Parent)
```

Expressões, Comandos e Blocos

Expressões, Comandos e Blocos

Uma expressão é uma construção feita de variáveis, operadores e chamadas de métodos, construída de acordo com a sintaxe da linguagem, que avalia para um simples valor.

```
int cadence = 0;
anArray[0] = 100;
System.out.println("Element 1 at index 0: " + anArray[0]);

int result = 1 + 2; // result is now 3
if (value1 == value2)
    System.out.println("value1 == value2");
```

Expressões, Comandos e Blocos

Comandos são equivalentes a sentenças em uma linguagem natural. Um comando forma uma unidade completa de execução.

```
// assignment statement
aValue = 8933.234;

// increment statement
aValue++;

// method invocation statement
System.out.println("Hello World!");

// declaration statement
double aValue = 8933.234;

// object creation statement
Bicycle myBike = new Bicycle();
```

Expressões, Comandos e Blocos

Um bloco (de código) é um grupo de zero ou mais comandos entre chaves.

```
class BlockDemo {
     public static void main(String[] args) {
          boolean condition = true;
          if (condition) { // begin block 1
               System.out.println("Condition is true.");
          } // end block one
          else { // begin block 2
               System.out.println("Condition is false.");
          } // end block 2
     }
}
```

Controle de Fluxo

Controle de Fluxo

- Tomada de decisão

Controle de Fluxo

- Tomada de decisão
  - `if-then`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`
  - `do-while`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`
  - `do-while`
- Ramificação

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`
  - `do-while`
- Ramificação
  - `break`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`
  - `do-while`
- Ramificação
  - `break`
  - `continue`

Controle de Fluxo

- Tomada de decisão
  - `if-then`
  - `if-then-else`
  - `switch`
- Repetição
  - `for`
  - `while`
  - `do-while`
- Ramificação
  - `break`
  - `continue`
  - `return`