

Sumário

[Variáveis 3](#_Toc130752605)

[Tipos de Dados Primitivos 3](#_Toc130752606)

[Tipos de Dados não Primitivos 4](#_Toc130752607)

[Diferença Entre as Variáveis de Classe, Globais e Locais 4](#_Toc130752608)

[Declaração das Variáveis 5](#_Toc130752609)

[Uma pequena introdução ao camelCase 5](#_Toc130752610)

[O Uso da Palavra-Chave static 6](#_Toc130752611)

[Constantes 6](#_Toc130752612)

[A Palavra-Chave final 6](#_Toc130752613)

[Métodos 6](#_Toc130752614)

[Os Modificadores de Acesso 7](#_Toc130752615)

[A Palavra-Chave Return 7](#_Toc130752616)

[Os Tipos de retorno 7](#_Toc130752617)

[Básicos 7](#_Toc130752618)

[Arrays 7](#_Toc130752619)

[A Declaração dos Métodos 7](#_Toc130752620)

[TypeCasting 8](#_Toc130752621)

[Widening Casting (automaticamente) 8](#_Toc130752622)

[Narrowing Casting (manualmente) 8](#_Toc130752623)

# Variáveis

Bem, as variáveis dentro da programação são espaços reservados dentro da memória, com o intuito de armazenar valores que serão utilizados posteriormente para os mais diversos fins.

## Tipos de Dados Primitivos

Como na língua portuguesa temos os substantivos primitivos, que são a essência dos substantivos dentre estes se tornarão outros posteriormente. Um bom exemplo seguindo essa lógica seria o substantivo roupa, pois dele podemos retirar o substantivo camisa, calça, casaco... Bom desta mesma forma são os tipos de dados primitivos, que podem também ser chama-los de tipo de dados básicos. São eles:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **boolean** | **byte** | **char** | **double** | **float** | **int** | **long** | **short** |

Boolean

O tipo de dado boolean é usado para armazenar somente dois possíveis valores: verdadeiro e falso. Este tipo de dado é usado por sinalizadores simples que localizam condições verdadeira/falsa.

Especifica um bit de informação, mas o "tamanho" não pode ser especificado precisamente.

**Exemplo:** boolean check = false;

Byte

É um inteiro de complemento de dois com assinatura de 8 bits. Sua faixa de valor está entre -128 até 127. Valor mínimo é -128 e o valor máximo é 127. Seu valor padrão é 0.

O tipo de dado byte é usado para salvar em memória grandes matrizes onde a memória salva é muito requisitada. Isso economiza espaço porque um byte é quatro vezes menor que um número inteiro. Também pode ser usado no lugar de tipos de dados "int".

**Exemplo:** byte a = 10;

Char

O tipo de dado char é um único caractere unicode de 16 bits. Sua faixa de valor está entre '\u0000' (0) até '\uffff' (65.535). O tipo de data char é usado para para armazenar caracteres.

**Exemplo:** char tamanho = ‘P’;

Double

O tipo de dado double é um ponto flutuante IEEE 754 de 64 bits de precisão dupla. Seu valor é ilimitado. O tipo de dado double é geralmente usado para valores decimais assim como float. O tipo de dado double também não deve ser usado para precisar valores, como moeda. Seu valor padrão é 0.0d.

**Exemplo:** double d1 = 1.92d;

Float

O tipo de dado float é um ponto flutuante IEEE 754 de 32 bits de precisão única. Sua faixa de valor é ilimitada. É recomendado usar um ponto flutuante (ao invés de double) se você precisar economizar memória em grandes arrays de número de ponto flutuante. O tipo de dado float nunca deve ser usado para valores precisos, como moeda. Seu valor padrão é 0.0f.

**Exemplo:** float f1 = 1.92f;

Int

O tipo de dado int é um inteiro de complemento de dois com assinatura de 32 bits. Seu valor mínimo é -2.147.483.648 e valor máximo é 2.147.483.647. Seu valor padrão é 0.

O tipo de dado int geralmente é usado como um tipo de dado padrão para valores inteiros sem nenhum problema de memória.

**Exemplo:** int numero = 140000;

Long

O tipo de dado long é um inteiro de complemento de dois com assinatura de 64 bits. Seu valor mínimo é -9.223.372.036.854.775.808 e o valor máximo é de 9.223.372.036.854.775.807. O valor padrão é 0. O tipo de dado long é usado quando você precisa de uma faixa de valores maior do que a fornecida por int.

**Exemplo:** long lnum = 14320050L;

Short

O tipo de dado short é um inteiro de complemento de dois com assinatura de 16 bits. Seu valor mínimo é -32.768 e valor máximo é 32.767. Seu valor padrão é 0. O tipo de data short também pode ser usado para economizar memória assim como o tipo de dado byte. Um tipo de dado short é duas vezes menor que um inteiro.

**Exemplo:** short r = 14000;

Muito bem, mas por que não seria o tipo de dado String primitivo? Isto é até controverso, porém a resposta é até simples, "primitivo" é definido como qualquer tipo de dado não derivado da classe Object. String é derivada, então não pode ser um tipo de dado primitivo.

## Tipos de Dados não Primitivos

Os tipos de dados não primitivos são aqueles que derivam dos tipos de dados primitivos, como por exemplo, a String, que é uma cadeia ou array de caracteres (char), enumeradores, classes, arrays...

## Diferença Entre as Variáveis de Classe, Globais e Locais

As variáveis de classe são definidas dentro da classe e são compartilhadas por todos os objetos dessa classe. Elas são acessíveis a partir de qualquer método da classe ou objeto. Essas variáveis podem ser usadas para armazenar informações que precisam ser compartilhadas entre diferentes objetos da mesma classe. As variáveis de classe são precedidas pelo modificador de acesso "static".

As variáveis globais são definidas fora de qualquer função ou classe e podem ser acessadas a partir de qualquer lugar no programa. Elas são úteis quando você precisa armazenar informações que devem ser compartilhadas por várias funções ou classes no programa. No entanto, o uso excessivo de variáveis globais pode tornar o programa difícil de entender e depurar.

As variáveis locais são definidas dentro de uma função ou bloco de código e só podem ser acessadas dentro desse bloco de código. Elas são usadas para armazenar informações temporárias que só são necessárias dentro do escopo em que foram definidas. Quando o bloco de código é encerrado, as variáveis locais são automaticamente destruídas.

### Declaração das Variáveis

### Uma pequena introdução ao camelCase

CamelCase é um estilo de digitação muito utilizado na computação, usufruído quando há duas palavras que não podem conter um espaço entre elas. Seu principal ponto positivo é conseguir ler grandes “Strings” ou facilitar o entendimento do nome de uma variável que foi escrita.

A sua aplicação é: primeira palavra em minúsculo + segunda palavra com a primeira letra em maiúsculo. Sendo que se houver mais de duas palavras, estas tem a mesma regra da segunda palavra.

**Exemplo:** nomeDoCliente, numeroCliente...

Tendo em mente os conteúdos anteriores agora sim podemos ver como se declara uma variável:

Se pensarmos no caso mais simples, que seria uma variável local sua declaração seria:

tipo de dado + nome da variável (camelCase opcional)

Exemplo: int v10 = 1000;

**As variaveis de classe/globais a partir de agora serão chamadas apenas de variáveis globais**

Se o caso for uma variável global sua declaração seria:

Modificador de acesso + tipo de dado + nome da variável (camelCase opcional)

Exemplo: public static int a = 123;

Obs: uma variável pode conter um número em seu nome, desde que ele não seja o primeiro caractere.

## O Uso da Palavra-Chave static

A tão temida keyword static, ela permite que não haja a intervenção de um objeto ou da execução de uma instância de uma classe podendo chamar diretamente qualquer método da classe e também manipular alguns campos da classe. Sendo assim sua declaração seria:

Modificador de acesso + static + tipo de dado + nome da variável (camelCase opcional) = valor;

# Constantes

As constantes são valores fixos que não podem ser modificados após serem definidos. Elas são usadas para representar valores que nunca mudam durante a execução do programa, como números ou strings que representam configurações, valores de configuração ou outras informações que não devem ser alteradas.

As constantes em Java são usadas em muitos lugares, incluindo cálculos matemáticos, comparações e definição de valores padrão. Elas são úteis para tornar o código mais legível e fácil de entender, evitando que valores mágicos sejam usados diretamente no código e permitindo que mudanças em valores fixos sejam feitas de forma mais centralizada.

## A Palavra-Chave final

As constantes em Java são declaradas usando a palavra-chave "final". Isso significa que, depois de definir uma constante, ela não pode ser alterada. A sintaxe para declarar uma constante em Java é a seguinte na:

Variável local

final + tipo de dado + nome da variável (camelCase opcional)

Exemplo: final int numero = 20;

Variável global

Modificador de acesso + static (opcional) + final + tipo de dado + nome da variável (camelCase opcional)

Exemplo: public static final int segundoNumero = 1250;

# Métodos

Um método é um bloco de código ou coleção de instruções ou um conjunto de códigos agrupados para executar uma determinada tarefa ou operação. É usado com o intuito de reutilizar o código, se escreve um método uma vez e é utilizado muitas vezes. Ele também fornece a fácil modificação e legibilidade do código, apenas adicionando ou removendo um pedaço de código. O método é executado somente quando o chamamos ou invocamos.

O método mais importante em Java é o método main();

## Os Modificadores de Acesso

**Default**: todas as classes veem não precisando escrever nada na inicialização

**Public**: todas as classes veem precisando escrever na inicialização

**Private**: nenhuma classe além de si mesma pode ver

**Protected**: nenhuma classe além de si mesma pode ver, podendo a partir da instanciação ser acessada.

## A Palavra-Chave Return

A palavra-chave return retorna um valor de um método. O método retornará o valor imediatamente quando a palavra-chave for encontrada. Isso significa que o método não executará mais nenhuma instrução além da palavra-chave return e quaisquer variáveis ​​locais criadas no método serão descartadas.

## Os Tipos de retorno

### Básicos

int: o retorno tem que ser um numero inteiro.

String: o retorno tem que ser uma String.

char: o retorno tem que ser um caracter.

double: o retorno tem que ser um numero com 1 ou mais casas decimais.

float: o retorno tem que ser um numero com 1 ou mais casas decimais contendo "f" em sua terminação.

void: não há o retorno de nada.

### Arrays

int[]: o retorno tem que ser uma array de numeros inteiros.

String[]: o retorno tem que ser uma array de Strings.

char[]: o retorno tem que ser uma array de caracteres.

double[]: o retorno tem que ser uma array de numeros com 1 ou mais casas decimais.

float[]: o retorno tem que ser uma array de numeros com 1 ou mais casas decimais contendo "f" em sua terminação.

**Além deste muitos outros existem e serão apresentados ao longo do tempo**

### A Declaração dos Métodos

Modificador de Acesso | "static ou não" | tipo de retorno | nome da função(

tipo do argumento1 argumento1...) {

return se o tipo de retorno não for void, retorna algum valor com o tipo da função;

}

Exemplos:

public static void ola() {

System.out.println(“Ola!”);

}

public static int soma(int a, int b) {

return a+b;

}

public static boolean divisivelPor2(int num) {

boolean check = false;

if(num%2 == 0) {

check = true;

} else {

check = false;

}

return check;

}

public static String maiusculo(String texto) {

texto = texto.toUpperCase();

return texto;

}

public static String minusculo(String texto) {

texto = texto.toLowerCase();

return texto;

}

# TypeCasting

Typecasting (ou conversão de tipos) é o processo de converter um valor de um tipo de dado para outro tipo de dado. Isso é necessário porque, às vezes, os tipos de dados são incompatíveis, o que significa que não é permitido que você atribua diretamente um valor de um tipo de dado a uma variável de outro tipo de dado.

Existem dois tipos de TypeCasting:

## Widening Casting (automaticamente)

A conversão implícita ocorre automaticamente, quando é seguro fazê-lo, sem a necessidade de qualquer instrução explícita por parte do programador. Ela é permitida quando a perda de informação não ocorre, ou quando há um aumento de precisão ou uma ampliação de tamanho. Por exemplo, se você atribuir um valor inteiro a uma variável do tipo double, o Java fará a conversão de forma automática.

byte -> short -> char -> int -> long -> float -> double

## Narrowing Casting (manualmente)

A conversão explícita é quando o programador precisa especificar explicitamente que deseja converter um valor de um tipo de dados para outro tipo de dados. Isso pode ser necessário quando você precisa converter um tipo de dados em outro que seja incompatível ou quando a conversão implícita não ocorrerá. A conversão explícita é feita através da utilização de uma expressão especial que é chamada de cast. O cast é representado colocando o tipo de dados entre parênteses antes do valor que deseja ser convertido.

double -> float -> long -> int -> char -> short -> byte

**Exemplo:**

public static void main(String[] args) {

double ddouble = 9.78d;

int inteiro = (int) ddouble;

System.out.println(ddouble);

System.out.println(inteiro);

}

**Output:**

9.78

9