**OS 8 TIPOS DE DADOS PRIMITIVOS EM JAVA**

A linguagem de programação Java é fortemente tipada, o que significa que todas as variáveis ​​devem primeiro ser declaradas antes de poderem ser utilizadas. Isto implica que em Java devemos primeiro informar o tipo da variável e o nome, por exemplo:

int x = 1; Neste caso a variável x é de tipo inteira e possui o valor inicial de 1. Um tipo de dado determina quais valores podem ser armazenados em uma determinada variável, além das operações que podem ser executadas com a mesma. Além de int, a linguagem de programação Java suporta sete outros tipos de dados primitivos. Um tipo de dado primitivo é pré-definido pela linguagem e é chamado por uma palavra reservada.

Os oito tipos primitivos de dados suportados pela linguagem de programação Java são:

byte: O tipo de dado byte pode assumir valores inteiros positivos ou negativos e requer 8 bits para ser implementado. O valor mínimo de uma variável byte é de -128 e o valor máximo é de 127. O tipo de dado byte pode ser útil para economizar espaços de memória em matrizes de grandes dimensões, em casos que sejam viáveis. Variáveis tipo byte também podem ser usadas ​​no lugar de variáveis int de tal modo que seus limites ajudem a clarear o seu código, o fato de que uma variável byte é limitada pode servir como uma forma de documentação.

short: O tipo de dado short pode assumir valores inteiros positivos ou negativos e requer 16 bits para ser implementado. O valor mínimo de uma variável short é de -32.768 e o valor máximo de 32.767. Tal como acontece com byte, as mesmas diretrizes se aplicam: você pode usar um short para economizar memória em arrays grandes, em situações em que as economias seja realmente importante.

int: O tipo de dado int pode assumir valores inteiros positivos ou negativos e requer 32 bits para ser implementado. O valor mínimo de uma variável int é de -2.147.483.648 e o valor máximo de 2.147.483.647. Para valores inteiros, esse tipo de dado é geralmente a escolha padrão, a menos que haja uma razão (como citado acima) para escolher outra tipo. Este tipo de dado provavelmente será grande o suficiente para os números que seu programa irá utilizar.

long: O tipo de dado long pode assumir valores inteiros positivos ou negativos e requer 64 bits para ser implementado. O valor mínimo de uma variável long é de -9.223.372.036.854.775.808 e o valor máximo de 9.223.372.036.854.775.807. Use este tipo de dado é utilizado quando você precisa de um intervalo de valores maior do que os previstos por int.

float: O tipo de dado float é de precisão simples de 32 bits IEEE 754 ponto flutuante(traduzindo, é preciso). Tal como acontece com as recomendações para byte e short, use float (em vez de double), se você precisa economizar memória em grandes conjuntos de números decimais. Este tipo de dado nunca deve ser usado para valores muito preciso, como moeda. Para isso, você vai precisar usar a classe java.math.BigDecimal(se isso não faz muito sentido não se preocupe, iremos falar disso em vídeo aulas futuras).

double: O tipo de dado double é de dupla precisão de 64 bits IEEE 754 ponto flutuante(traduzindo, é preciso pra car\*\*ho). Para valores decimais, este tipo de dado é geralmente a opção padrão. Como mencionado acima, este tipo de dado não deve ser utilizado para valores como moeda.

boolean: O tipo de dado boolean tem apenas dois valores possíveis: true e false(verdadeiro/falso). Este tipo de dado representa um bit de informação, mas o seu “tamanho” não é algo que é definido com precisão.

char: O tipo de dado char é um caractere de 16 bits único Unicode. Ela tem um valor mínimo de ‘\ u0000’ (ou 0), e um valor máximo de ‘\ uffff’ (ou 65535.

**PALAVRAS RESERVADAS**

Palavras reservadas são as palavras que estão definidas na sintaxe da linguagem, isto é, essas palavras fazem parte da linguagem e assim, se nós utilizarmos, estaremos criando um conflito de nomes.

Quando estudamos as regras de nomenclatura de variáveis, aprendemos que as palavras reservadas da linguagem, não podem ser utilizadas.

Porém, se você realmente quiser utilizar um nome que é uma palavras reservada, a solução é inicializar com um underline. Como já estudado, o nome de variáveis pode inicializar com um caractere ou então, com o underline.

Mesmo que não tenhamos estudado, desde o primeiro programa que fizemos em Java, fizemos uso das palavras reservadas, por exemplo: int é uma palavra reservada utilizada para definir que uma variável é do tipo inteiro. if é uma outra palavra reservada que utilizamos nas tomadas de decisões. Nós também temos as palavras public, static, void que utilizamos para declarar o método que inicializa o programa Java.

Num primeiro momento, a lista contendo todas as palavras pode assustar um pouco, porém, não há necessidade de decorar ou então memorizar essa lista, até porque, cada palavra é uma instrução e para programar em Java, somos obrigados a estudar cada palavra reservada. Assim, não cometa o erro de tentar memorizar uma única instrução contida nessa lista.

Toda vez que digitamos uma palavras reservada no Eclipse, ou em qualquer editor Java, o mesmo atribui uma coloração diferente para o grupo de palavras reservadas. Assim é fácil perceber se o nome que estamos querendo utilizar é ou não uma palavra reservada. Se você está programando utilizando o Eclipse, não há motivos para preocupação! Se a palavra tiver a sua cor alterada, isso significa que não podemos fazer uso da mesma.

Somente se estivéssemos trabalhando com o Bloco de Notas ou então, algum editor genérico, precisaríamos memorizar ou ter um pouco de conhecimento acerca dessas palavras.

TIPOS PRIMITIVOS

boolean: tipo de dado lógico.

byte: inteiro de 8 bits que pode armazenar valores positivos e negativos (signed).

char: inteiro de 16-bit que armazena o valor unicode de um único caracteres. Podemos utilizar o tipo char para trabalharmos com números inteiros, porém, somentes os valores positivos (16-bit unsigned).

double: número de ponto flutuante de 64 bits (signed).

float: número de ponto flutuante de 32 bits (signed).

int: inteiro de 32 bits.

long: inteiro de 64 bits.

short: inteiro de 32 bits.

**MODIFICADORES DE ACESSO**

private: restringe o acesso ao membro (variável ou função) , fazendo que seja visível somente dentro da classe onde o mesmo foi definido.

protected: restringe o acesso para as classes filhas ou então, que estejam no mesmo pacote.

public: não há restrições quanto ao acesso

**MODIFICADORES DE CLASSE, VARIÁVEIS E/OU MÉTODOS**

abstract: membro que não possui implementação, somente a sua definição.

class: palavra reservada utilizada para definir uma estrutura tipo classe.

extends: utilizado para herdar de outra classe.

final: utilizado para definir constantes e impossibilitar, por exemplo, que um método seja redefinido numa subclasse, ou então, que uma classe seja estendida. A classe String do Java possui em seu construtor a instrução final. Logo, não é possível estender a mesma.

implements: define a interface que deve ser implementada ou então, as interfaces.

interface: palavra reservada para definir uma estrutura do tipo interface, isto é, uma estrutura que irá conter somente as definições e não as implementações propriamente ditas.

native: palavra utilizada para indicar que uma função está definida num outra linguagem, como por exemplo, C ou C++.

new: cria uma nova instância de um objeto e retorna o identificador a quem invocou o construtor da classe.

static: define que um membro será instanciado no momento em que a estrutura da classe for carregada. Assim, para fazer acesso a um método ou então, a uma variável, não é necessário criar uma instância antes.

strictfp:

synchronized: palavra reservada utilizada no manuseio de threads. Assim, conseguimos definir que um método só será invocado numa determinada instância, quando este não estiver sendo por uma outra instância.

transient: utilizado nos processos de serialização para indicar que um determinado campo não pode ou não será serializado.

volatile: palavra reservada que funciona de maneira contrária a palavra synchronized.

**CONTROLE DE FLUXO**

break: finaliza a iteração do bloco de instrução contido, só pode ser usado em, while, do while, for e switch.

case: estrutura utilizada para definir as condições dentro de uma estrutura switch.

continue: finaliza a execução de um ciclo e já pulando para o início do próximo.

default: bloco que será executado quando nenhuma condição for satisfeita na estrutura switch.

do:estrutura de repetição que executado um determinado bloco de instrução até que a condição definida através da instrução while seja atendida.

else: define o bloco a ser executado caso a condição verificada não seja satisfeita.

for: estrutura de iteração utilizada para definir onde um laço começa e onde o mesmo irá terminar.

if: estrutura utilizada para verificar condições. O valor de retorno sempre será do tipo primitivo Boolean, ou seja, um valor lógico.

instanceof: retorna True quando uma instância é instância oriúnda de uma classe, superclasse ou interface

return: valor a ser retornado por uma função. Também é utilizado para encerrar a execução de funções.

switch: estrutura que permite a verificação de diversas condições e também, capaz de definir uma condição padrão, ou seja, a condição default.

while: iterador que executa um determinado bloco até que uma condição seja verdadeira.

**TRATAMENTO DE ERROS**

try: define que um bloco de instrução deve ser executado de forma que se houver um erro, este deverá ser tratado.

finally: instrução utilizada em conjunto com a instrução try e sempre será executada, indiferente se houve ou não exceções durante a execução.

throws: levanta uma exceção

assert - função assertiva para definir determinadas condições e levantamento de exceções em tempo de desenvolvimento.

**PACOTES**

import: importa um determinado pacote e dispobiliza seus membros para utilização numa determinada classe.

**VARIÁVEIS**

super: faz menção a superclasse, ou seja, a classe pai.

this: do inglês, this significa este. Logo, utilizamos a instrução this quando estamos trabalhado dentro de uma determinada classe e queremos fazer menção a mesma.

void: representa um tipo sem valor. Comumente utilizamos para trabalharmos com procedimentos ao invés de funções. Assim, como a instrução void indica que não iremos retornar qualquer valor, entende-se que a estrutura em questão, é um procedimento.

**PALAVRAS RESERVADAS EM DESUSO**

const: palavra reservada que está em desuso. Sua função, seria para a declaração de constantes. Atualmente, recomenda-se a utilizaçao da palavra reservada final para a declaração de constantes.

goto: a linguagem Java tem a palavra goto como sendo reservada, porém, não implementou a mesma. Provavlmente, a principal razão pela não implementação é que com a estrutura goto é possível fazer um tipo de programação não estruturada, o que não é mais recomendado.

**LITERAIS RESERVADOS**

null: esta palavra define que uma variável não está apontando para nada.

true: valor lógico que indica verdadeiro

false: valor lógico que indica falso

**OPERADORES**

* ++ Increment
* -- Decrement
* + Plus
* \* Multiplicação
* / Divisão
* % Resto divisão
* () Cast
* ! NOT
* < Menor Que
* > Maior que
* <= Menor ou igual
* >= Maior ou igual
* != Diferente
* == Igual

**OPERADORES LÓGICOS**

* && And
* || OR

**OPERADORES DE ATRIBUIÇÃO**

* = Igual
* -= Menos Igual
* \*= Vezes Igual
* /= Dividido Igual
* % Resto Igual

**CONTROLE DE FLUXO**

* if
* else
* else if

**­ESTRUTURAS DE REPETIÇÃO**

* for
* while (Para ser executado precisa ser validado, valor dever ser verdadeiro)
* do while (Executa pelo menos uma vez para ser validado, sendo verdadeiro ou falso)

**ARRAYS**