Literais

A organização das **variáveis**, espaços na memória volátil do computador destinado a um dado que é modificado durante a execução de um algoritmo, é feita com base nos **tipos de dados**. Cada tipo tem características diferentes e com isso ocupa espaços distintos na memória. Além disso, o uso de tipos auxilia na programação, pois evita que a pessoa programadora faça a manipulação incorreta dos dados, como por exemplo, carregar um número onde deveria ser um texto. E assim por diante.

Uma das características marcantes do Java é que ele é uma **linguagem fortemente tipada**. Na prática, isso significa que cada um dos dados manipulados dentro de um programa no Java precisar ter um tipo. Não há como fugir disso. As linguagens de programaçã

trabalham com alguns **tipos primários (ou primitivos)** de dados, e no Java isso não é diferente. Esses tipos são chamados de primários pois são a base de qualquer outro tipo de dado que pode ser estruturado usando os recursos da linguagem em questão.

No Java, é possível classificar os tipos em quatro grandes categorias e uma especial:

- Números inteiros *(byte, short, int, long);*
- Números flutuantes *(float/double);*
- Caracteres *(char);*
- Booleano *(bool);*
- Especial: Nulo *(null);*

Dentre todos os tipos que fazem parte da categoria de números inteiros, estão:

Tipo	Tamanho *(bits)*	Faixa de valores
byte	8	-128 a 127
short	16	-32.768 a 32.767
int	32	2^{31} a $2^{31} - 1$
long	64	-2^{63} a $2^{63} -1$

Tabela 1 - Tipos primitivos do Java para números inteiros.

O tipo *int* costuma ser a primeira escolha dos profissionais para armazenar os dados numéricos (inteiros) de um aplicação, no entanto, existem situações em que os outros tipos devem ser considerados, por exemplo, quando se quer otimizar a memória utilizada pelo programa ou então quando se quer atribuir e armazenar números inteiros muito longos. Neste caso é recomendado o uso da estrutura *long* (muito usada para mapear os *ids* de entidades em uma estrutura de banco de dados).

Além disso, existem os números que não são necessariamente inteiros. Para estes tipos de valores deve-se usar os tipos de números de ponto flutuante. **Ponto flutuante** diz respeito a maneira como os números reais são representados internamente na memória d

computador. Dentre os tipos primitivos de números de ponto flutuante, estão:

Tipo	Tamanho (*bits*)	Faixa de valores
float	32	IEEE 754 ±1,40129846432481707e-45 a 3,40282346638528860e+38
double	64	IEEE 754 ±4,94065645841246544e-324 a 1,79769313486231570e+308

Tabela 2 - Tipos primitivos do Java para números de ponto fluente.

Em termos de caracteres, o tipo *char* consegue representar apenas um caractere ***Unicode*** (padrão para representação de textos no computador). Para isso, ele utiliza um bloco de memória de 16 *bits*. Além disso, existe o tipo *booleano* que permite armazenar um valor lógico no estado verdadeiro (*true*) ou falso (*false*) usando apenas um único *bit*.

representar a inexistência de dados em uma determinada situação. Por exemplo, ao estruturar o cadastro de dados de um cliente n

seu sistema, pode ser que alguns dos dados sejam opcionais. Nestes casos, esses atributos podem ser atribuídos com o valor *null*. No entanto, os tipos primitivos por si só não aceitam esse valor. Só é possível atribuir o valor nulo às classes.

Para tornar a vida da pessoa desenvolvedora um pouco mais fácil, o Java em si possui classes que dão funções aos tipos primitivos para ajudar a manipular eles dentro dos algoritmos. Essas classes são chamadas de classes *wrapper*, pois elas “empacotam” os tipos primitivos e dão “funcionalidades” a mais para cada uma delas. A tabela abaixo mostra essa relação:

Tipo Primitivo	Classe *Wrapper*
boolean	Boolean
byte	Byte
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Tabela 3 - Relação entre tipos primitivos e classes wrappers do Java.

Cada classe deste tipo encapsula somente um único valor de um dado tipo primitivo, como é o caso do *int* para *Integer*, por exemplo.

Atividade Extra

- Artigo: A história do null
- Site: Exercícios e Desafios sobre Java
- Site: Exercícios de Desafios sobre Java (2)

Referência Bibliográfica

BARNES, D. J.; KOLLING, M. **Programação orientada a objetos com java: uma introdução prática usando o bluej.** 4.ed. Pearson: 2009.

FELIX, R. (Org.). **Programação orientada a objetos.** Pearson: 2017.

MEDEIROS, L. F. de. **Banco de dados: princípios e prática.** Intersaberes: 2013;

ORACLE. Java Documentation, 2021. **Documentação oficial da plataforma Java.** Disponível em: <https://docs.oracle.com/en/java/>. Acesso em 12 de Jul. de 2021.

Ir para questão