Tipos de dados Java[1]

O Java conhece dois tipos de tipos: *tipos primitivos* e *tipos objetos*. Tipos primitivos são armazenados nas variáveis diretamente e têm semântica de valor (os valores são copiados quando atribuídos a outra variável). Os tipos objetos são armazenados nas referências ao objeto (não o próprio objeto). Quando atribuídos a outra variável, somente a referência é copiada, não o objeto.

1 Tipos primitivos

A tabela a seguir lista todos os tipos primitivos da linguagem Java:

Nome de tipo	Descrição	Exemplo
Números inteiros		
`byte`	inteiro de 1 byte (8 bits)	24, -2
`short`	inteiro curto (16bits)	137, -119
`int`	inteiro (32bits)	5409, -2003
`long`	inteiro longo (64 bits)	423266353L, 55L
Números reais		
`float`	ponto flutuante de precisão simples	43.889F
`double`	ponto flutuante de precisão dupla	45.63, 2.4e5
Outros tipos		
`char`	um único caractere (16 bits)	'm', '?', '\u00F6'
`boolean`	um valor booleano (v ou f)	`true`, `false`

Notas:

- Um número sem um ponto de fração decimal é geralmente interpretado como um `int`, mas automaticamente convertidos em tipos `byte`, `short` ou `long` quando atribuídos (se o valor se ajustar). Você pode declarar um literal como `long` colocando um 'L' depois do número.
- Um número com um ponto de fração decimal é do tipo `double`. Você pode especificar um literal `float` colocando um 'F' ou 'f' depois do número.
- Um caractere pode ser escrito como um único caractere Unicode entre aspas simples ou como um valor Unicode de quatro dígitos, precedido por '\u'.

A tabela a seguir detalha os valores mínimos e máximos disponíveis nos tipos numéricos.

[1]Fonte: Barnes, Kolling, Programação orientada a objetos com Java, Apendice B, 4ª ed. Editora Pearson.

Tipo	Mínimo	Máximo
`byte`	-128	127
`short`	-32768	32767
`int`	-2147483648	2147483647
`long`	-9223372036854775808	9223372036854775807
	Mínimo positivo	**Máximo positivo**
`float`	1.4e-45	3.4028235e38
`double`	4.9e-324	1.7976931348623157e308

2 Tipos objetos

Todos os tipos não listados na seção 1 são tipos objetos. Esses tipos incluem tipos de classes e tipos de interface da biblioteca Java padrão (como `String`) e tipos definidos pelo usuários.

Uma variável de um tipo objeto contém uma referência (ponteiro) a um objeto. Atribuições e passagem de parâmetro têm semântica de referência (isto é, a referência é copiada, não o objeto). Depois de atribuir uma variável a outra, as duas variáveis referenciam o mesmo objeto. As duas variáveis são conhecidas como *aliases* para o mesmo objeto.

As classes são as templates para objetos, definindo os campos e métodos que cada instância tem. Os arrays comportam-se como tipos objetos - eles também têm semântica de referência.

3 Classes empacotadoras

Todo tipo simples em Java tem uma classe empacotadora correspondente que representa o mesmo tipo, mas é um tipo de objeto real. Isso torna possível utilizar valores de tipos primitivos onde tipos objetos são requeridos por um processo conhecido como *autoboxing*. A tabela a seguir lista os tipos primitivos e seus tipos empacotadores correspondentes no pacote `java.lang`. Além de `Integer` e `Character`, os nomes das classes empacotadoras são os mesmos nomes dos tipos primitivos, mas com a primeira letras maiúscula.

Tipo primitivo	Tipo empacotador
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`boolean`	`Boolean`

Sempre que um valor de um tipo primitivo é utilizado em um contexto que exige um tipo de objeto, o compilador usa o *autoboxing* para empacotar automaticamente o valor

do tipo primitivo em um objeto empacotador apropriado. Isso significa que os valores do tipo primitivo podem ser adicionados diretamente a uma coleção, por exemplo. A operação inversa - conversão *unboxing* - também é realizada automaticamente quando um objeto do tipo empacotador é utilizado em um contexto que exige um valor do tipo primitivo correspondente.

4 Conversão de tipo dos tipos objeto

Como um objeto pode pertencer a uma hierarquia de herança dos tipos, às vezes é necessário converter uma referência de objeto de um tipo em uma referência de um subtipo em posição mais baixa na hierarquia de herança. Esse processo é chamado conversão de tipo (ou *downcasting*). O operador de conversão de tipo consiste no nome de uma classe ou tipo de interface escrito entre parênteses em frente de uma variável ou expressão. Por exemplo:

```
Carro c = (Carro) veiculo;
```

Se o tipo declarado da variável `veiculo` for `Veiculo` e `Carro` for uma subclasse de `Veiculo` então essa instrução será compilada. Uma verificação separada é feita em tempo de execução para assegurar que o objeto referido por `veiculo` é um `Carro` e não uma instância de um subtipo diferente.

É importante reconhecer que a conversão de tipo entre tipo objeto é completamente diferente da conversão de tipo entre tipos primitivos (*cast*). Especialmente, a conversão de tipo entre tipos objeto não envolve nenhuma modificação do objeto envolvido. Ela é puramente uma maneira de ganhar acesso às informações sobre o tipo que já é verdadeiro para o objeto - isto é, parte do seu tipo completamente dinâmico.