Tipos de Dados

Tipos de dados

- Os tipos de dados são particularmente importantes em Java porque essa é uma linguagem fortemente tipada.
- Ou seja, todas as operações têm a compatibilidade de seus tipos verificada pelo compilador.
- Logo, a verificação minuciosa dos tipos ajuda a impedir a ocorrência de erros e melhora a confiabilidade.
- Não há o conceito de uma variável “sem tipo” por exemplo.
- Uma operação aplicada a um tipo pode não ser permitida em outro

Tipos primitivos

- Java contém duas categorias gerais de tipos de dados internos: orientado a objetos e não orientado a objetos.
- Os tipos orientados a objetos são definidos por classes, mas a discussão das classes deixaremos para depois.
- Porém , na base de Java, temos oito, tipos de dados primitivos

Tipos primitivos

Tipo	Significado
boolean	Representa os valores verdadeiro/falso
byte	Inteiro de 8 bits
char	Caractere
double	Ponto flutuante de precisão dupla
float	Ponto flutuante de precisão simples
int	Inteiro
long	Inteiro longo

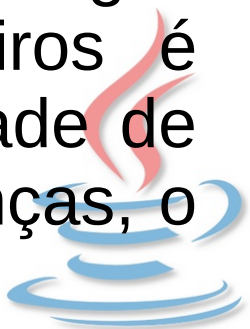
Inteiros

- Java define quatro tipos inteiros: byte, short, int e long, conforme tabela:

Tipo	Tamanho em bits	Intervalo
byte	8	-127 a 128
short	16	-32.768 a 32.767
int	32	-2.147.483.648 a 2.147.483.647
long	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

Inteiros

- O tipo inteiro mais usado é int. Variável de tipo int costumam ser empregadas no controle de laços, na indexação de arrays e na execução de cálculos de inteiros para fins gerais.
- Quando você precisar de um inteiro que tenha um intervalo maior do que o int, use long.
- A especificação formal de Java define uma categoria de tipo chamada tipos integrais, que inclui byte, short, long e char. No entanto, a finalidade dos quatro primeiros é representar quantidades inteiras numéricas. A finalidade de char é representada por caracteres. Devido às diferenças, o tipo de char é tratado separadamente.



Ponto Flutuante

- Os tipos de ponto flutuante podem representar números que têm componentes fracionários.
- Há duas espécies de tipos de ponto flutuante, float e double, que representam números de precisão simples e dupla, respectivamente. O tipo float, tem 32 bits e o tipo double tem 64 bits.
- Dos dois, o double é o mais usado, porque todas as funções matemáticas da biblioteca de classes Java usam valores double.

Caracteres

- Em Java, os caracteres não são valores de 8 bits como em muitas outras linguagens de computador, char é um tipo de 16 bits sem sinal com um intervalo que vai de 0 a 65.536. O conjunto de caracteres ASCII de 8 bits padrão é um subconjunto do Unicode e vai 0 a 127.Logo os caracteres ASCII ainda são caracteres Java válidos.
- Uma variável de caractere pode receber um valor pela inserção do caractere entre aspas simples. Por exemplo, este código atribui à variável ch a letra x:

```
char ch;

ch='x';
```

Caracteres

- Java foi projetado para uso mundial.
- Logo, tem de usar um conjunto de caracteres que possa representar os idiomas do mundo todo. O Unicode é o conjunto de caracteres padrão projetado especialmente para esse fim.
- É claro que o uso do Unicode é ineficiente para idiomas como o inglês, alemão, espanhol ou francês, cujo caracteres podem ser armazenados em 8 bits. Mas esse é o preço a ser pago pela portabilidade global.

Booleano

- O tipo boolean representa os valores verdadeiro/falso, usando as palavras reservadas true e false.
- Logo, uma variável ou expressão de tipo boolean terá um desses dois valores.
- Parece pouco mas veremos logo mais as utilidades deste tipo de dado.

Antes de vermos Literais vamos fazer mais um exercício!

Exercício

Neste exercício, você criará um programa que calcula a que distância, em pés, um ouvinte está da queda de um relâmpago.

O som viaja a aproximadamente 1.100 pés por segundo pelo ar.

Logo, conhecer o intervalo entre o momento em que você viu um relâmpago e o momento em que o som o alcançou lhe permitirá calcular a distância do relâmpago.

Exercício

1 – Crie um projeto chamado Relampago no Eclipse.

2 – Crie um novo arquivo chamado Som.java

3 – Para calcular a distância, você terá que usar valores de ponto flutuante. Por quê? Porque o intervalo de tempo, 7,2, tem um componente fracionário. Embora pudéssemos usar um valor do tipo **float**, usaremos **double** neste exercício.

4 – Para fazer o cálculo, você multiplicará 7,2 por 1.100. Em seguida atribuirá esse valor a uma variável.

Execute o programa. O resultado a seguir será exibido:

“O relâmpago está a 7920,0 pés de distância”.

Literais

- Em Java, os literais são valores fixos representados em sua forma legível por humanos. Por exemplo, o número 100 é um literal. Normalmente os literais também são chamados de constantes.
- Os literais Java podem ser de qualquer um dos tipos de dados primitivos. A maneira como cada literal é representado depende de seu tipo. As constantes de caracteres são delimitadas por aspas simples. Por exemplo, ‘a’ e ‘%’ são constantes de caracteres.

Literais

- Os literais inteiros são especificados como números sem componentes fracionários. Por exemplo, 10 e -100 são literais inteiros. Os literais de ponto flutuante requerem o uso do ponto decimal seguido pelo componente fracionário do número. Por exemplo, 11.123 é um ponto literal flutuante. Java também permite o uso de notação científica para números de ponto flutuante.

- Por padrão, os literais inteiros usam o tipo int. Se quiser especificar um literal long, acrescente um l ou L. Por exemplo, 12 é um int, mas 12L é um long.

Literais

- Também é padrão os literais de ponto flutuante serem de tipo double. Para especificar um literal float, acrescente um f ou F à constante. Por exemplo, 10.19F é de tipo float.
- Embora os literais inteiros criem um valor int por padrão, eles podem ser atribuídos a variáveis de tipo char, byte ou short contanto que o valor atribuído possa ser representado pelo tipo de destino. Um literal inteiro sempre pode ser atribuído a uma variável long.

Literais

- A partir do JDK 7, é permitido embutir um ou mais sublinhados em um literal inteiro ou de ponto flutuante. Isso pode facilitar a leitura de valores compostos por muitos dígitos. Quando o literal é compilado, os sublinhados são simplesmente descartados. Exemplo:

123_45_1234

- Essa linha especifica o valor 123.451.234. O uso dos sublinhados é particularmente útil na codificação de coisas como números de peças, identificações de clientes e códigos de status que normalmente são criados como uma combinação de subgrupos de dígitos.

Literais

- Java também permite o uso de literais de ponto flutuante hexadecimais, mas raramente eles são usados.
- A partir do JDK 7, é possível especificar um literal inteiro com o uso de binários. Para fazer isso, use um 0b ou 0B antes do número binário. Por exemplo, este número especifica o valor 12 em binário: 0b1100.

Sequência de escape de caracteres

- A inserção de constantes de caracteres entre aspas simples funciona para a maioria dos caracteres imprimíveis, mas alguns caracteres, como o retorno do carro, impõe um problema especial quando um editor de texto é usado.
- Além disso, outros caracteres específicos, como as aspas simples e duplas, têm um significado especial em Java, logo, você não pode usá-los diretamente. Essas sequências são usadas no lugar dos caracteres que representam.

Sequência de escape de caracteres

- Por exemplo, esta linha atribui a **ch** o caracter de tabulação:

```
Ch = '\t';
```

- O próximo exemplo atribui uma aspa simples a **ch**:

```
Ch = '\'';
```

Sequência de escape de caracteres

- Tabela de sequência de escape de caracteres.

Sequencia de escape	Descrição
\'	Aspas simples
\"	Aspas duplas
\\	Barra invertida
\r	Retorno de carro
\n	Nova linha
\f	Avanço de página
\t	Tabulação horizontal
\b	Retrocesso
\ddd	Constante octal (onde ddd é uma constante octal)
\uxxxx	Constante hexadecimal (onde xxxx é uma constante hexadecimal)

Literais de Strings

- Java dá suporte a outro tipo de literal: o string. Um string é um conjunto de caracteres inserido em aspas duplas. Por exemplo:

"isto é um teste" é um string.

- Além dos caracteres comuns, um literal string também pode conter uma ou mais das sequências de escape. Por exemplo, considere o programa a seguir. Ele usa a sequência de escape \n e \t.

Literais de Strings

```
// Demonstração de sequências de escape em strings.
class  StrDemo{
    public static void main(String args[]) {
     //Usa \n para gerar uma nova linha
       System.out.println(“First line\n Second line”);
     //Usa-se tabulações para alinhar a saída
     System.out.println(“A\tB\tC *);
      System.out.println(“D\tE\tF *);
    }
}
```

Literais de Strings

A saída será:

```
First line
Second line
A  B    C
D    E  F
```

- Você não deve confundir strings com caracteres. Um literal de caractere representa uma única letra de tipo char. Um string contendo apenas uma letra continua sendo um string. Embora os strings sejam compostos por caracteres, eles não são do mesmo tipo.