Curso de Pós-Graduação Lato Sensu a Distância em Engenharia de Software

Programação Orientada a Objetos com Java: 1ª parte

INTRODUÇÃO AOS APLICATIVOS JAVA

Data de Início:
08 de março de 2010 às 08h00min

Data de Encerramento:
09 de Abril de 2010 às 23h55min.

Data da Prova:
23 de Abril de 2010 de 17h30min às 20h30min.

Professor: Delano Brandes Marques

Tipos primitivos de dados

Os tipos de dados oferecidos pela linguagem Java, dividem-se basicamente em duas categorias: tipos primitivos e tipos de referências. Tipos primitivos de dados consistem em dados mais simples. As variáveis deste tipo armazenam em seu conteúdo um valor exatamente do tipo declarado. Os tipos por referência correspondem a arrays, classes e interfaces. São tipos definidos por classes que armazenam a localização de um objeto na memória. A grande diferença entre os dois tipos é que as variáveis do tipo primitivo armazenam o próprio valor e as de tipo por referência armazenam o endereço de um objeto.

Uma dica para diferenciar os dois tipos é através de seus nomes:

- Tipos primitivos começam com letra minúscula
- Tipos por referência começam com letras maiúsculas

Atenção! Toda classe tem seu nome iniciado com uma letra maiúscula. Mais a frente veremos mais detalhes a respeito das classes.

A seguir comentaremos um pouco a respeito de cada um dos tipos primitivos de dados.

boolean

É o tipo mais simples em Java. Uma variável deste tipo pode assumir apenas um entre dois valores ***true*** ou ***false***. As variáveis booleanas são utilizadas para sinalizar

condições ou ocorrência de eventos em um programa. Além disso, o resultado de algumas operações como **x<y** ou **x>=u** será um valor booleano, que pode ser armazenado para uso posterior em variáveis booleanas. Operações como estas, são chamadas operações lógicas.

Números Inteiros

Os tipos primitivos byte, int, short e long representam tipos de dados inteiros. As variáveis desses tipos podem conter valores numéricos inteiros dentro da faixa estabelecida para cada tipo.

- ✓ byte: de -128 a 127
- ✓ int: de -2147483648 a 21474836847
- ✓ short: de -32768 a 32767
- ✓ long: de -9223372036854775808 a 92233720036854775807

Números Reais

Para armazenamento de números decimais existem duas categorias de variáveis: **float** e **double**. Da mesma forma que os tipos de dados inteiros, há uma faixa estabelecida para os valores contidos para cada tipo:

- ✓ float: de -3.40292347E+38 a +3.40292347E+38
- ✓ double: de -1.79769313486231570E+308 a +1.79769313486231570E+308

A representação destes valores pode ser feita através da notação decimal (exemplo: -24.321) ou da notação científica (exemplo: 2.52E-31).

char

Variáveis deste tipo armazenam um caractere Unicode, ou seja, um caractere de 16 bits, sendo que de 0 a 225 correspondem aos caracteres do código da tabela ASCII.

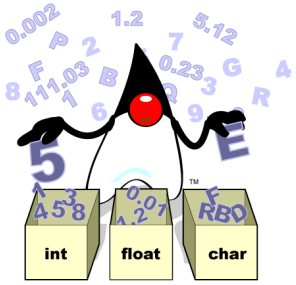
Caracter	Decimal	Hexadecimal
1	49	0031
2	50	0032
3	51	0033
4	52	0034
5	53	0035
6	54	0036
7	55	0037
8	56	0038
9	57	0039
A	65	0041
B	66	0042
C	67	0043
D	68	0044
...	...	...

O caracter armazenado em uma variável deste tipo pode ser representado na forma decimal, hexadecimal ou pelo próprio caracter. No caso da forma hexadecimal ou caracter, os valores armazenados devem estar entre aspas simples. A única diferença é que na forma hexadecimal o valor deve ser antecedido de **\u**. Na forma decimal, basta apenas utilizar o número, sem o uso de aspas simples. Por exemplo, uma variável do tipo **char** pode armazenar o caracter **D** de três formas:

- ✓ 'D'
- ✓ 68
- ✓ '\u0044'

Variáveis deste tipo são pouco utilizadas no dia a dia. Mais a frente veremos como utilizar as Strings, usadas constantemente, porém não são definidas por um tipo primitivo.

Declarando e Usando Variáveis



Para utilizar uma variável, a primeira coisa que devemos fazer é criá-la na memória. Isto é feito por meio da declaração de uma variável. Este procedimento é bem simples, e consiste em informar o tipo da variável e em seguida seu nome:

```
tipoDaVariavel nomeDaVariavel;
```

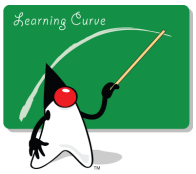
Por exemplo, vamos criar uma variável para armazenar a idade de uma pessoa:

```
int idade;
```

Na linha apresentada acima, acabamos de declarar uma variável chamada **idade** do tipo inteiro, que passa a existir após esta declaração. Abaixo apresentamos outros exemplos de declarações de variáveis:

- ✓ `byte a;`
- ✓ `short b;`
- ✓ `int c;`
- ✓ `long d;`
- ✓ `int e;`
- ✓ `byte contador;`
- ✓ `int angulo_em_graus;`

- ✓ *char indice;*
- ✓ *float notaDoBimestre;*
- ✓ *float peso_da_pessoa;*



O nome de variável, assim como o nome de um método ou classe é chamado de um **identificador**. Quando criado um identificador irá sempre representar o mesmo objeto a ele associado, independente do contexto em que seja empregado.

Cuidados na criação de um identificador:

- O primeiro caracter de um identificador deve ser uma letra, um underscore "_", ou um cifrão "$". Os demais caracteres podem ser quaisquer seqüências de numerais e letras;
- Os identificadores não podem ser palavras chave, como: class, for, while, public, etc

Operadores em Java

Símbolos utilizados para realização de diversas tarefas (cálculos, igualdades, concatenação de textos, operações lógicas, comparações, atribuições, entre outras) pode-se afirmar que os operadores estão presentes na grande maioria das classes. A seguir apresentamos os principais símbolos e operados utilizados em Java.

	Operador	Aplicação
Aritmético	+	Soma
	-	Subtração
	*	Multiplicação
	/	Divisão
	%	Resto da Divisão
	++	Incremento (pré ou pós)
	--	Incremento (pré ou pós)
Especial	+	Concatenação de Strings
	new	Instanciar objetos em memória
	.	Aplicação de referência
	instanceof	Verifica se um objeto pertence a uma classe
	this	Utilizado como uma referência para o próprio objeto que está sendo utilizado
de Atribuição	=	Atribuição
	+=	Operação composta de atribuição somando-se a si mesmo
	-+	Operação composta de atribuição subtraindo-se a si mesmo
	*=	Operação composta de atribuição multiplicando-se a si mesmo
	/=	Operação composta de atribuição dividindo-se a si mesmo
de Comparação	>	Maior que
	>=	Maior ou igual que
	<	Menor que
	<=	Menor ou igual que
	==	Igual
	!=	Diferente
Lógico	&	E lógico booleano: *devolve **true** se ambos operandos forem **true**, avaliando ambos*
	|	Ou inclusivo lógico booleano: *devolve **true** se um dos operandos for **true**, avaliando ambos*
	&&	E condicional: *devolve **true** se ambos operandos forem **true***
	||	Ou condicional: *devolve **true** se um dos operandos for **true***
	^	Ou exclusivo lógico booleano: *devolve **true** se e somente se um dos operandos for **true** e o outro **false***
	!	Negação

Quadro 01 – Operadores utilizados em Java

Bem, após sabermos como declarar uma variável e os principais operadores em Java, a partir de agora, poderemos utilizá-la. Primeiramente iremos declarar e atribuir um valor a variável.

```
int x=30;
```

Ao verificar a instrução acima, podemos perceber que é possível atribuir um valor a uma variável no momento de sua declaração. A partir deste momento a variável ***x*** passará a existir e irá armazenar o valor 30. Além de atribuir, podemos utilizar o valor armazenado pela variável. O código abaixo imprime seu valor na saída padrão por meio da chamada a ***System.out.println***.

```
System.out.println(x); // imprime a variável x
```

Agora, iremos fazer uso do valor da variável para outro propósito, alterar ou definir uma segunda variável. O código apresentado a seguir cria uma variável chamada ***y*** com o valor de ***x*** mais um.

```
int y=x+5;
```

Onde testar o código apresentado?
Crie a estrutura básica de uma classe, como vimos anteriormente na classe PrimeiroPrograma, e dentro do método **main** digite o código.
Não se esqueça que o nome da classe e do arquivo (.java) devem ser os mesmos.

Abaixo apresentamos a classe *DeclarandoVariaveis* para testarmos os trechos mostrados acima. O arquivo da classe deverá se chamar ***DeclarandoVariaveis.java***.

Também apresentamos a classe *OperadoresJava* que tem como objetivo exercitar o uso dos demais operadores e tipos de dados apresentados. Este arquivo deverá se chamar ***OperadoresJava.java***.

```
public class DeclarandoVariaveis{
    public static void main(String args[]) {
        int x=30; //declara e inicializa a variável x com o valor 30
        System.out.println("O valor da variável x é " + x); // imprime a variável x
        int y=x+5; //declara e inicializa a variável y com o valor de x + 5
        System.out.println("O valor da variável y é " + y); // imprime a variável y
    }
}
```

```
public class OperadoresJava {

    public static void main(String args[]) {

                double pi = 3.14;

                double x = 5 * 10;

                boolean verdade = true;

                int idade = 30;

                int a=1,b=0,c,d;

                boolean teste,menorDeIdade = idade < 18;

                char letra = 'a';

                a=b=c=d=0;

                ++a;

                --b;

                a+=b;

                c*=b;

                d+=4;

                teste=a==c;

                teste=((a>=c)&(d==a));

                teste=((b<10)|(d>0));

                teste=((d<=c)^(a!=b));

                teste=((b<10)&&(d>0));

                teste=((d<=c)||(a!=b));

    }

}
```

Após ter criado os dois arquivos com os códigos apresentados, comilado e executados as classes, altere o código da classe OperadoresJava de forma a exibir o valor de cada variável existente. Quais são os resultados armazenados em cada uma? Efetue alterações nas operações realizadas, teste outros operados, outros valores, e analise os novos resultados obtidos.