POO

Expressão lógica e comandos de desvio

Prof. Alcides Calsavara
PUCPR

Tipo para Valores Lógicos

Tipo	Bits	Valor mínimo	Valor máximo
`boolean`	`1`	`N.A.`	`N.A.`

Únicos valores possíveis: **`true`** e **`false`**

Variáveis Lógicas

```
public static void main( String[ ] args)
{
   boolean cheio = false;

   boolean aprovado = true;

   cheio = true;

   System.out.println( cheio );

   aprovado = false;

   System.out.printn( aprovado );
}
```

Operadores Lógicos

Os operadores lógicos são os seguintes:

!	: *não*	(NOT)
&&	: *e*	(AND)
||	: *ou*	(OR)

Podem-se usar parênteses (recursivamente) para alterar a precedência padrão entre operadores (*não*, *e*, *ou*).

Exemplos

```
boolean b = true;
boolean e = true;
boolean j = false;

boolean m;

m = b && e;

m = b && j;

m = b || e;

m = b || j;
```

```
m = b || j && e;

m = (b || j) && e;

m = !e;

m = ! ( e && j );

m = e || j;
```

Operadores de Comparação

Os operadores de comparação são os seguintes:

```
==  igual a
!=  diferente de
>   maior que
<   menor que
>=  maior ou igual a
<=  menor ou igual a
```

Exemplos

```
int k = 3;

m = ( k < 5 ); // k menor que 5 ?

m = ( k >= 7 ); // k maior ou igual a 7 ?

m = ( k == 10 ); // k igual a 10 ?

m = ( k != 3 ); // k diferente de 3 ?
```

Expressão Lógica

As formas mais simples de uma expressão lógica são os próprios valores **true** e **false**, ou uma variável do tipo **boolean**. Porém, o uso mais comum é com a aplicação de operadores de comparação e operadores lógicos. Por exemplo, se existir no programa Java uma variável **x** do tipo **int**, pode ser escrita a seguinte expressão lógica:

```
x < 10
```

A avaliação dessa expressão resultará verdadeiro se o valor de **x** for *menor que* **10**, ou falso, caso contrário.

Outra expressão lógica válida é a seguinte:

```
x <= 10
```

Nesse caso, a avaliação da expressão resultará verdadeiro se o valor de **x** for *menor ou igual a* **10**, ou falso, caso contrário.

Expressão Lógica

Também pode ser feita uma combinação lógica de termos em uma expressão lógica. Por exemplo, assumindo que, além da variável **x** do tipo **int**, exista também no programa Java uma variável **y** do tipo **double**, a seguinte expressão lógica é válida:

```
x == 10 && y > 5.0
```

A avaliação dessa expressão resultará em verdeiro se, e somente se, o valor de **x** for *igual a* **10** *e* o valor de **y** for *maior que* **5.0**, ou seja, é preciso que os dois termos da expressão (o primeiro termo da expressão compara **x** com **10**, enquanto o segundo termo compara **y** com **5.0**) sejam verdadeiros para que toda a expressão seja verdadeira. Observe que o operador de comparação de igualdade é representado por **==**, enquanto o operador lógico *e* é representado por **&&**.

Expressão Lógica

Outro operador lógico muito usado é o operador *ou*, representado por **||**. Como exemplo, considere a seguinte expressão lógica:

```
x == 10 || y > 5.0
```

A avaliação dessa expressão resultará verdadeiro se o valor de **x** for *igual a* **10** *ou* se o valor de **y** for *maior que* **5.0**, ou seja, basta que um dos dois termos seja verdadeiro para que toda a expressão seja verdadeira.

Expressão Lógica

Uma expressão lógica pode ser muito complexa, com diversos operadores de comparação e lógicos e, ainda, com uso de parênteses para alterar a precedência entre operadores lógicos (o operador *e* tem precedência sobre o operador *ou*). Por exemplo, considere a seguinte expressão:

```
x >= 10 && (y > x || y < 5.0)
```

A avaliação dessa expressão resultará verdadeiro se o valor de **x** for *maior ou igual* a **10** e se, pelo menos, uma das seguintes cláusulas for verdadeira:

i. o valor **y** é *maior que* o valor **x**
ii. o valor de **y** é *menor que* **5.0**

Observe que os parênteses garantem que, primeiramente, o operador *ou* é avaliado para, então, o operador *e* ser avaliado, invertendo a precedência normal. Equivalementemente, a expressão também poderia ser escrita da seguinte forma:

```
(x >= 10 && y > x) || (x >= 10 && y < 5.0)
```

Comando de desvio: **if**

```
if ( expressão-lógica )
     comando-1
else
     comando-2
```

Se *expressão-lógica* resulta verdadeiro (**true**), é executado o *comando-1*; **senão**, é executado o *comando-2*.

Observe o uso de parênteses para delimitar a expressão lógica.

Observe o uso da palavra-chave **else** (que significa *senão*) para indicar o *comando-2*.

Exemplo

Execute o programa três vezes, fornecendo como entrada os valores **200.00**, **50.00** e **100.00**. Note que a mensagem **"Obrigado"** é sempre impressa.

```
import java.util.Scanner;

public class Main {
   public static void main(String args[]) {
      Scanner teclado = new Scanner( System.in );

      System.out.print( "Digite o preço do produto: " );
      double preco = teclado.nextDouble();

      if (preco <= 100.00)
         System.out.println( "Preço bom!" );
      else
         System.out.println( "Muito caro!" );

      System.out.println( "Obrigado" );
   }
}
```

Comando Composto (Bloco de Comandos)

Execute o programa três vezes, fornecendo como entrada os valores **200.00**, **50.00** e **100.00**.

```
if (preco <= 100.00)
{
   System.out.println( "Preço bom!" );
   System.out.println( "Compre uma unidade");
}
else
   System.out.println( "Muito caro!" );
```

Encadeamento de desvios

Execute o programa para as seguintes entradas: **80.00, 50.00, 30.00, 100.00 e 150.00.**

```
if (preco <= 100.00)
{
   System.out.println( "Preço bom!" );

   if ( preco > 50.00)
      System.out.println( "Compre uma unidade.");
   else
      System.out.println( "Compre duas unidades");
}
else
   System.out.println( "Muito caro!" );
```

Comando-2 opcional

Teste o exemplo, fornecendo como entrada o valor **200.00**. Observe que somente a mensagem "Obrigado" é impressa.

```
if (preco <= 100.00)
{
   System.out.println( "Preço bom!" );

   if ( preco > 50.00)
      System.out.println( "Compre uma unidade.");
   else
      System.out.println( "Compre duas unidades");
}
```

Comando de desvio: switch

Verifica o valor fornecido para **k** e, dependendo do valor (0, 1, 2 ou outro qualquer, isto é, **default**), executa um bloco específico de comandos (cada bloco é encerrado com o comando **break**). Execute o programa com os seguintes dados de entrada: 0, 1, 2 e 4.

```
System.out.print("Digite o valor de k: ");
int k = teclado.nextInt();

switch ( k )
{
   case 0: k = (k + 5) * 3; break;
   case 1: k = k + 5; k = k * k; break;
   case 2: k = k * 3; k = k / 2; break;
   default: k = 0;
}

System.out.println( k );
```

```
switch ( k )
{
    case 0:
        k = (k + 5) * 3;
        break;
    case 1:
        k = k + 5;
        k = k * k;
        break;
    case 2:
        k = k * 3;
        k = k / 2;
        break;
    default: k = 0;
}
```