

Desvio Condicional if / if – else / if – else if – else

Operadores Relacionais

Operadores Lógicos

Programação Orientada a Objetos – Aula 03
Professores: Hamilton Machiti da Costa

Desvio Condicional: if

- Server para mudar o fluxo de execução de um algoritmo baseado em uma ou mais condições. Considere o seguinte exemplo:
- Se um número for maior que 0 imprimir que ele é positivo. Em Java, escreve-se:

```
if (x > 0) {  
    System.out.println("O número é positivo");  
}
```

Desvio Condicional: if - else

- Caso ele não seja positivo, o número pode ser negativo ou nulo. Caso queiramos também escrever esta informação, fazemos:

```
if (x > 0) {  
    System.out.println("O número é positivo");  
} else {  
    System.out.println("O número é negativo ou nulo");  
}
```

Desvio Condicional: if – else aninhado

- Porém, se quisermos ser precisos, dizendo se ele é negativo, ou seja, x menor que 0, ou nulo, isto é, x igual a 0, temos que fazer mais um desvio. Usando desvios aninhados, fica assim:

```
if (x > 0) {  
    System.out.println("O número é positivo");  
} else {  
    if (x < 0) {  
        System.out.println("O número é negativo");  
    } else {  
        System.out.println("O número é nulo");  
    }  
}
```

Desvio Condicional: if – else if - else

- Podemos também fazer sem aninhar:

```
if (x > 0) {  
    System.out.println("O número é positivo");  
} else if (x < 0) {  
    System.out.println("O número é negativo");  
} else {  
    System.out.println("O número é nulo");  
}
```

Expressões Lógicas e Operadores Relacionais

- O desvio é sempre baseado em uma expressão lógica, isto é, que retorne verdadeiro ou falso. Para isso temos que usar os operadores relacionais, que são:

`> maior`

`>= maior ou igual`

`< menor`

`<= menor ou igual`

`== igual`

`!= diferente`

Combinação de Expressões Lógicas e Operadores Lógicos

- Além disso, podemos combinar expressões lógicas usando os operadores lógicos:

& & E

| | OU

! NÃO

- Quando usamos o operador &&, as duas expressões lógicas devem ser verdadeiras para o resultado ser verdadeiro. Quando usamos o ||, basta uma delas ser verdadeira. O ! inverte o resultado da expressão lógica. O verdadeiro vira falso e vice-versa.

Exemplo

- Dados 3 números inteiros diferentes, a, b e c, queremos saber quem é o maior. Usando operadores lógicos, o algoritmo fica assim:

```
if( a > b && a > c) {  
    System.out.println("O maior é o a");  
} else if( b > a && b > c) {  
    System.out.println("O maior é o b");  
} else {  
    System.out.println("O maior é o c");  
}
```


Outro Exemplo

- Dados 3 números inteiros, a, b e c, queremos saber se há pelo menos dois números iguais no grupo. Usando operadores lógicos, o algoritmo fica assim:

```
if( a == b || a == c || b == c) {  
    System.out.println("Há pelo menos 2 números  
iguais");  
} else {  
    System.out.println("Os números são diferentes");  
}
```

Mais um Exemplo

- Considere o mesmo exemplo anterior. Usando o operador de negação, podemos inverter a ordem das condições lógicas:

```
if( !(a == b || a == c || b == c) ) {  
    System.out.println("Os números são diferentes");  
} else {  
    System.out.println("Há pelo menos 2 números  
iguais");  
}
```

Comparação de Strings

- Não use operadores relacionais (`==`, `!=`, `>`, `<`, `>=`, `<=`) para comparar Strings - e nenhum outro objeto.
- Use o método `equals()`. Ex:
 - `"12345".equals("12345")` retorna `true`
 - `"12345".equals("123")` retorna `false`
 - `!"12345".equals("123")` retorna `true`
- Para saber se uma string é maior que outra:
 - `"12345".length() > "678".length()` retorna `true`

Exemplo

- Dadas 2 strings, queremos saber se são iguais:

```
String s1 = "aluno";  
String s2 = "aluno";  
if( s1.equals("aluno")) {  
    System.out.println("São iguais");  
} else {  
    System.out.println("São diferentes");  
}
```