



Estruturas condicionais e de repetição

Operadores Relacionais e Lógicos

- Operadores Relacionais
- Operadores Aritméticos

< (menor)

> (maior)

<= (menor ou igual)

>= (maior ou igual)

== (igual)

!= (diferente)

&& (E lógico)

|| (OU lógico)

! (NÃO lógico)

- Operadores relacionais para a classe String

equals

if (nome.equals("Maria"))

!equals

if (!nome.equals("Maria"))

Operadores Relacionais e Lógicos

■ Operadores Relacionais: == , != , > , < , >= , <=

- Quando aplicados a tipos nativos se comportam da forma esperada.
- Quando aplicados a referências ...

12/01/2004
↑ ↑
p q
p == q resulta true

12/01/2004 12/01/2004
↑ ↑
p q
p == q resulta false

só == e != podem ser aplicados a referências
!!!
só == e != podem ser aplicados a valores
booleanos !!!

Estruturas de decisão e controle

- if – else

Estrutura do
Comando

```
if (expressão_booleana )  
{ bloco de comandos do if;  
}  
[else]  
{ bloco de comandos do else;  
}
```

- Observações:

- O comando *else* é opcional.
- Na construção de *if*'s aninhados, o *else* refere-se sempre ao *if* mais próximo. Procure usar chaves para delimitar os blocos.
- O operador de comparação **igual a** é representado por `==` e não por `=`.
- Bloco de comandos pode ser um único comando (terminado por ponto-e-vírgula) ou vários comandos (delimitados por `{}`).
- É indispensável o uso de parênteses `()` na expressão_booleana.

Estruturas de Decisão e Controle

```
if ( .... )  
{ .....  
}
```

bloco de comandos

```
if ( ... )  
{ .....  
}  
else  
{ ....  
}
```

apenas um comando

```
if ( .... )  
.... ;
```

```
if ( .... )  
.... ;  
else  
.... ;
```

```
if ( ... )  
{ .....  
}  
else  
.... ;
```

mista

```
if ( ... )  
.... ;  
else  
{ .....  
}
```

Operador Condicional ?

- Usado quando o objetivo é verificar o valor que será atribuído a uma variável.
- Forma básica
 - `variável = (expressão ? Valor se verdadeiro: valor se falso)`
`int maior = (a > b? a : b)`

Estruturas de decisão e controle

- switch

Estrutura do
Comando

```
switch(variávelDeControle)
{ case constante1:
    bloco1;
    break;
  case constante2:
    bloco2;
    break;
  [default:
    bloco3;
    break;]
}
```

- Observações:

- Usual para selecionar alguma ação de um número de alternativas.
- A variável de controle só pode ser inteira, byte, short ou char.
- O case define o ponto de entrada da execução. Se você quiser que só um bloco de declarações seja executado use **break**.
- A opção **default** é opcional.

Contadores

Variáveis que recebem um valor inicial e são incrementadas a cada interação de uma repetição.

contador = contador + 1;

- **Exemplos:**

- **++**

incrementa o valor em 1

A = 5; A++; => A vale 6

Pode ser colocado antes ou depois da variável

A = 5;

B = A++; => B vale 5 e A vale 6

B = ++A; => B e A valem 6

- **+=**

soma o valor à variável

A = 5; A += 3 => A vale 8

- **--**

decrementa o valor de 1

A = 5; A--; => A vale 4

Contadores

- -= subtrai o valor à variável
A = 5; A -= 3 => A vale 2
- *= multiplica a variável pelo valor
A = 5; A *= 2; => A vale 10
- /= divide a variável pelo valor
A = 10; A /= 2 => A vale 5

Contadores

■ Exercícios

- ☐ `a = 3; a++; System.out.println(a);`
- ☐ `a = 3; ++a; System.out.println(a);`
- ☐ `a = 3; System.out.println(a++);`
- ☐ `a = 3; System.out.println(++a);`
- ☐ `a = 3; val = a++; System.out.println(val);`
- ☐ `a = 3; val = ++a; System.out.println(val);`
- ☐ `a = 3; a += 3; System.out.println(a);`

Contadores

■ Resposta

`a = 3; a++; System.out.println(a);`

imprime 4

`a = 3; ++a; System.out.println(a);`

imprime 4

`a = 3; System.out.println(a++);`

imprime 3!!!

`a = 3; System.out.println(++a);`

imprime 4

`a = 3; val = a++; System.out.println(val);`

imprime 3!!!

`a = 3; val = ++a; System.out.println(val);`

imprime 4

`a = 3; a += 3; System.out.println(a);`

imprime 6

Estruturas de Repetição

- while

Estrutura do
Comando

```
while (expressão_booleana)
{ Bloco de comandos;
}
```

Observações:

- É indispensável o uso de parênteses () na expressão_booleana.
- O laço permanece em execução enquanto a expressão_booleana for verdadeira.
- Um erro comum é não atualizar as variáveis de controle do laço, o que acarreta um *loop* infinito.
- Outro erro freqüente é colocar o ponto e vírgula após o while. A fase de atualização das variáveis de controle ficam fora do laço, gerando um *loop* infinito.
- Use sempre chaves para delimitar o Bloco de comandos.
- Break força a saída do laço

Estruturas de repetição

- do - while

do

{ Bloco de comandos;

} while (expressão_booleana);

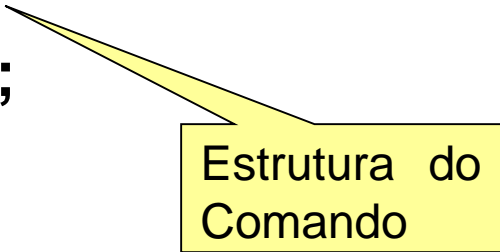
Estrutura do
Comando

- Observações:

- É semelhante ao comando while, sendo que a condição de parada do laço é testada após o bloco de comandos.
- Pelo menos uma vez o bloco de comandos será executado.
- Observe as mesmas considerações do comando while.

Estruturas de repetição

- **for** **for(inicialização; terminação; iteração)**
 { bloco de comandos;
 }



Estrutura do
Comando

- Observações:
 - Num *loop* **for** é explicita as quatro partes de uma iteração.
 - Um erro comum é colocar o ponto e vírgula após o **for**, ficando o bloco de comandos fora do laço. O resultado é um *loop* que nada realiza.

Exemplos (decisão e repetição)

```
public class exemploCondicaoRepeticao
{
    public static void main (String args[])
    {
        double valor = 1;
        char letra='A';
        int contador;
        while(valor <= 20)
        {
            System.out.println(valor);
            valor *=2; / valor = valor * 2
            if (valor >= 20)
            {
                System.out.println("Fim da exemplificação do while e if");
            }
        }
        for (contador=0; contador < 10; contador++)
        {
            System.out.println(contador);
        }
        System.out.println("Fim da exemplificação do for");
    }
}
```

Exemplos (decisão e repetição)

```
int i = 0;
do
{
    switch (letra)
    {
        case 'A' : System.out.println(letra + " é uma vogal");
                    break;
        case 'E' : System.out.println(letra + " é uma vogal");
                    break;
        case 'I' : System.out.println(letra + " é uma vogal");
                    break;
        case 'O' : System.out.println(letra + " é uma vogal");
                    break;
        case 'U' : System.out.println(letra + " é uma vogal");
                    break;
    }
    letra = args[i];
} while (letra != 'Z');
System.out.println("Fim da exemplificação do switch e do-while");
}
```


Laços “Infinitos”

```
public class ExemploLacoInfinito
{ private int a;
  private int b;
  public ExemploLacoInfinito(int vA, int vB)
  { setA(vA);
    setB(vB);
  }
  public void setA(int vA){
    a = vA;}
  public void setB(int vB) {
    b = vB;}
  public int getA(){
    return a;}
  public int getB(){
    return b;}
```

Laços “Infinitos”

```
public void testaLacoInfinito(){
    int valor = getA();
    while (true){
        valor++;
        if (valor>getB())
            break;
    }
    System.out.println(valor);
}

public void testaLacoInfinito2(){
    int valor = getA();
    for (;;){
        valor++;
        if (valor>getB())
            break;
    }
    System.out.println(valor);
}
}
```

Exercícios

construir uma classe que represente os clientes de um cinema. Todo cliente possui os atributos nome e idade. Construir um construtor para a classe e um método para calculo do valor do ingresso sabendo-se que:

- Crianças (menos 14 anos) e idosos (mais de 65 anos) pagam meia entrada R\$ 6,00
- Existe uma promoção onde todos pagam meia nas quartas e quintas feiras.
- Receber como argumento o dia da semana.

Considere uma locadora de vídeo com o seguinte cenário. A locadora trabalha com filmes. Todo filme possui um titulo um ator principal, o número de copias disponíveis e o numero de copias locadas. Os atores são identificados por seu nome, idade e sexo.

Divida as responsabilidades e construa:

Um membro que crie um filme que já tem copias locadas;

Um membro que permita criar um filme sem nenhuma copia locada;

Um membro que permita saber se o filme foi estrelado pelo mesmo ator.

Um membro que permita saber se o ator é mais velho que uma idade qualquer informada.

5) Construir uma classe comparação que terá um atributo um valor e métodos para:

Construtor

Receber um número e verificar se ele é igual ao da classe

Receber dois números e verificar se eles são iguais

Receber um número e verificar se ele é menor que o da classe

Receber dois números e verificar se o primeiro é maior que o segundo.

Quarta lista de Exercícios



Bibliografia

SANTOS, Rafael. *Introdução à Programação Orientada a Objetos Usando Java*. 1a Edição, Editora Campus. 2003.

Notas de aula do professor Frederico Barbosa