INE5603 Introdução à POO

Prof. A. G. Silva

13 de setembro de 2017

Estruturas de repeteção

(INTRODUÇÃO)

Comandos de repetição

- **Instrução de repetição** – repete uma ação enquanto uma condição permanecer verdadeira.

- **Pseudocódigo**
  Enquanto houver itens em minha lista de compras:
  Comprar o próximo item e riscar da minha lista.

- O corpo da declaração de repetição pode ser uma única instrução ou um bloco.

- Por fim, a condição se tornará falsa. Nesse ponto, a repetição termina e a primeira instrução depois da instrução de repetição é executada.

Comandos de repetição – do-while

- **do-while** – repita enquanto a expressão lógica for *true*
  - condição estabelecida no fim
  - deve executar ao menos uma vez

```
do
    comando ou bloco
while (expressao logica);
```

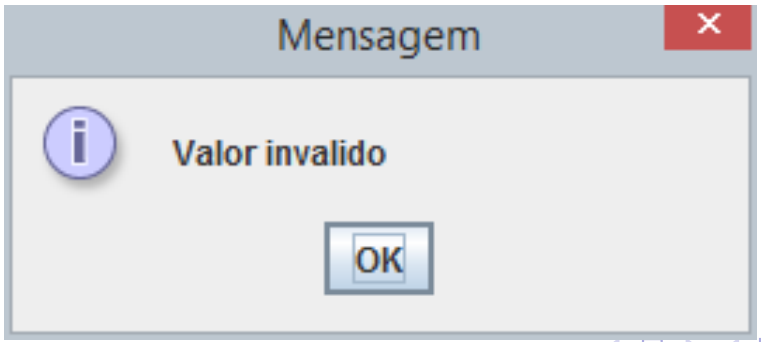
- Exemplo de trecho de código:

```
int n = umaInterface.leiaNaonegativo();
int fat = 1;
int cont = 0;
do {
    cont++;
    fat = fat * cont;
} while (cont < n);
umaInterface.mostreResultado(fat);
```

Comandos de repetição – do-while

- **do-while** – valor inteiro não negativo

```
public int leiaNaoNegativo() {
    String val;
    int n;
    do {
        val = JOptionPane.showInputDialog("Entre nao
            negativo: ");
        n = Integer.parseInt(val);
        if (n < 0)
            JOptionPane.showMessageDialog(null, "Valor
                invalido");
    } while (n < 0);
    return n;
}
```



Comandos de repetição – `do-while` – exemplo

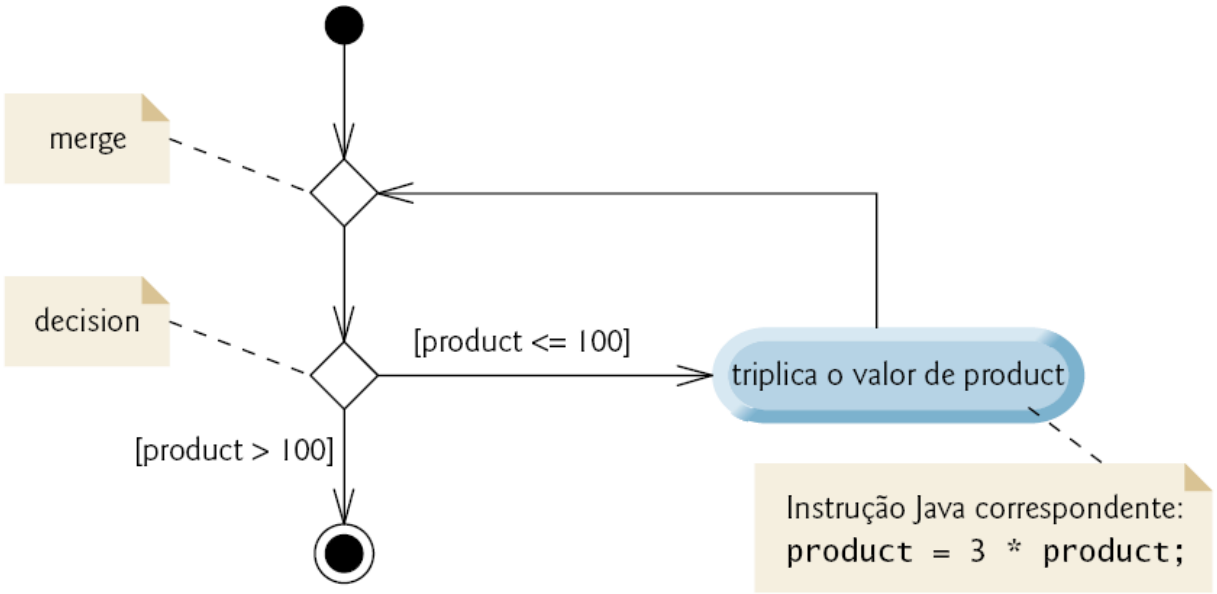


Diagrama de atividades UML da instrução de repetição `do-while`.

Comandos de repetição – do-while – exemplo (cont...)

- Exemplo da instrução de repetição **do-while** do Java: encontrar a primeira potência de 3 maior que 100. Assuma que a variável `int product` é inicializado como 3.

```
int product = 3;
do
    product = 3 * product;
while ( product <= 100 );
```

- Cada iteração multiplica product por 3, portanto `product` assume os valores 9, 27, 81 e 243 successivamente.
- Quando a variável product torna-se 243, a condição da instrução `do-while` – `product <= 100` – torna-se falsa.
- A repetição termina. O valor final de product é 243. A execução de programa continua com a próxima instrução depois da instrução `while`.

Comandos de repetição – `while`

- **while** – enquanto a expressão lógica for *true*, repita comando ou bloco de comandos
  - condição estabelecida no início
  - pode não executar nenhuma vez

```
while (expressao logica)
    comando ou bloco
```

- Exemplo de trecho de codigo:

```
double soma = 0;     Pessoa p;     int cont = 0;
int quant = umaInterface.leiaNaoNegativo();
while (cont < quant) {
    p = umaInterface.leiaPessoa();
    soma += p.informeIdade();
    cont++;
}
if (quant > 0) double media = soma / cont;
```

Comandos de repetição – `while`

- Exemplo:

```
public int mdc(int n, int m) {
    while (n != m)
        if (n > m)    n = n - m;
        else          m = m - n;

}
```

Comandos de repetição – `for`

- **for** – incialize variáveis na *expressao1*; enquanto a *expressao2* for *true*, repita e atualize as variáveis usando a *expressao2*

```
for (expressao1 ; expressao2 ; expressao3)
    comando ou bloco
```

- Exemplo 1:

```
int soma = 0;
for (int k=0; k < 5; k++)
    soma = soma + k;
```

- Exemplo 2:

```
long somaPar = 0;
for (int par=10; par > 2; par=par-2)
    somaPar += par;
```

Comandos de repetição – for

- Exemplo 3:

```
public long fornecaFatorial(int n) {
    long fat = 1;
    for (int i=1; i <= n; i++)
        fat = fat * i;
    return fat;
}
```

Observações

- Não fornecer, no corpo de uma instrução `while`, uma ação que consequentemente faz com que a condição torne-se falsa, normalmente resulta em um erro de lógica chamado **loop infinito** (a repetição pressegue indefinademente)

- Escolher um valor de sentinela que também seja um valor legítimo de dados é um erro de lógica. Por exemplo: digitar -1 para sair, quando -1 poderia ser um número válido em um somatório

- Muitos programas podem ser logicamente divididos em três fases: uma de inicialização das variáveis; uma de processamento com inserção dos valores dos dados e ajuste das variáveis; e uma de término que calcula e insere os resultados finais

Observações (cont...)

- Ao realizar a divisão por uma expressão cujo valor possa ser zero (p. ex., no cálculo de média, é necessário somar e dividir pela quantidade), é importante testar e tratar o denominador (p. ex., imprimindo uma mensagem de erro) em vez de permitir a ocorrência do erro

- Omitir as chaves que delimitam um bloco pode levar a erros de lógica, como *loops infinitos*. Para evitar esse problema, alguns programadores incluem o corpo de cada instrução de controle entre chaves mesmo que seja uma única instrução

- A experiência tem mostrado que a parte mais difícil de resolver um problema em um computador é desenvolver o algoritmo para a solução. Uma vez que um algoritmo foi especificado, produzir um programa em Java (ou qualquer outra linguagem) que execute tal algoritmo é relativamente simples

Comando break

- O comando **break**, ao ser executado, causa uma saída imediata de um processo de repetição definido por um **while**, **do-while** ou **for**, desviando o fluxo de execução para o primeiro comando após o laço repetitivo.

- Exemplo:

```
int soma = 0;
int n = umaInterface.leiaValor();
for (int cont=2; cont < n; cont++) {
    if (n % cont == 0)
        break;
    soma = soma + cont;
}
```

Comando continue

- O comando **continue**, ao ser executado, causa um saldo para o início da próxima iteração, ignorando o restante dos comandos dentro do laço. No exemplo a seguir, quando a variável `i` assume valor 7, hífens são exibidos (7 não é exibido, pois "`System.out.println(i);`" não é executado neste caso) e o laço inicia com o próximo valor de `i` (ou seja, `i` $\leftarrow$ 8).

- Exemplo:

```
for (int i=0; i<10; i++) {
    if (i==7) {
        System.out.println("--");
        continue;
    }
    System.out.println(i);
}
```