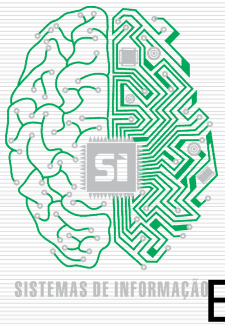

Estruturas de Controle

Fernando Del Moro

1

O comando if

- Os comandos condicionais permitem a execução seletiva de uma determinada parte de código de acordo com uma condição analisada numa expressão.
- A condição é uma expressão booleana ou um valor booleano, i.e., a expressão deve resultar em *true* ou *false.*

2

O comando if

- Sintaxe:

```
if (expressão booleana) {
    comando ou bloco;
}
else {
    comando ou bloco;
}
```

3

O comando if

- Exemplo:

```
int a, b;

a = 22;
b = 5;

if (a < b) {
        System.out.print("A é menor que B");
}
else {
        System.out.print("A é maior que B");
}
```

4

Múltiplas condições if

- O comando *if* pode ser utilizado para unir uma cadeia condições.

5

Múltiplas condições if

- Sintaxe:

```
if (expressão booleana) {
    comando ou bloco;
}
else{
    if (expressão booleana) {
            comando ou bloco;
    }
    else {
        comando ou bloco;
    }
}
```

6

Múltiplas condições if

- Exemplo:

```
int a = 13;
int b = 22;
int c = 3;
if (a < b) {
   System.out.println("a é menor que b");
}
else{
    if (a < c) {
      System.out.println("a é menor que c");
     }
     else {
     System.out.println("a não é menor que b nem c");
    }
}
```

7

if aninhado

- Sintaxe:

```
if (expressão booleana)
    if (expressão booleana)
        if (expressão booleana) {
                comando ou bloco;
        }
        else {
                comando ou bloco;
        }
```

8

if aninhado

- Exemplo:

```
int a = 4; int b = 5; int c = 7; int d = 3;
if(a < b)
    if(a < c)
      if(a < d) {
            System.out.println("a é menor que b, c, e d");
       }
        else {
            System.out.println("a é menor que b e c");
            System.out.println("a não é menor que d");
       }
```

9

if aninhado

- Alternativa (sintaxe):

```
if ((exp_bol_1) && (exp_bol_2) && (exp_bol_3)) {
        comando ou bloco;
}
else {
        comando ou bloco;
}
```

10

if aninhado

- Alternativa (exemplo):

```
if((a < b) && (a < c) && (a < d)) {
    System.out.println("a é menor que b, c, e d");
}
else {
    System.out.print("a não é menor que b ou c");
    System.out.println(" ou d");
}
```

11

O comando switch

- O comando *switch* é um caso especial de estrutura de controle de seleção que permite mais do que uma escolha quando uma condição é satisfeita.

12

O comando switch

- Sintaxe:

```
switch (expr1) {
    case constant1:
        seqüência de comandos;
        break;
    case constant2:
        seqüência de comandos;
        break;
    default:
        seqüência de comandos;
        break;
}
```

13

O comando switch

- No comando switch(expr1), expr1 deve ser compatível com um tipo int, byte, short, ou char.
- Não são permitidos os tipos ponto flutuante, expressões long, ou referências a classes.
- A partir da versão 7 do Java foi incluída nessa lista a classe **String**.
- O label opcional default especifica o segmento de código a ser executado quando o valor da variável ou expressão não for igual a nenhum dos valores dos cases.

14

O comando switch

- Exemplo1:

```
int opcao = 2;
switch (opcao) {
    case 1:
            System.out.print("Não Cliente");
            break;
    case 2:
            System.out.print("Cliente");
            break;
    default:
            System.out.print("Telefonista...");
            break;
}
```

15

O comando switch

- Exemplo1:

```
String s = "sim";
 switch (s){
    case "sim": System.out.println("sim");break;
    case "não": System.out.println("não");break;
    default:System.out.println("ERRO");
 }
```

16

O comando switch

- Há algumas palavras chave e símbolos que são usados na estrutura de seleção múltipla switch:
  - switch: começa o processo de seleção;
  - case: a condição é igual a constante; o símbolo dois pontos (:) é requerido;
  - break: termina a seqüência de ações e sai da estrutura de controle do switch;
  - default: se nenhum case for satisfeito, então a execução passa para o bloco default;

17

Laços

- As estruturas de repetição permitem realizar uma seqüência de comandos de forma repetida.
- A linguagem de programação Java suporta três tipos de construtores de laços:
  - for
  - while
  - do...while

18

Laços

- O laço for normalmente é utilizado quando o número de laços a ser executado for pré-determinado; já para as estruturas while e do...while, utiliza-se quando o número de vezes for indeterminado.

19

Laços

- Todas as estruturas possuem 4 elementos ou ações que ocorrem:
  - Inicialização.
  - Teste de uma condição ou expressão.
  - Execução de uma seqüência de comandos.
  - Alteração da condição ou expressão para realizar a saída do laço.

20

O comando do...while

- Sintaxe:

```
do {
    seqüência de comandos;
} while (teste booleano);
```

21

O comando do...while

- Exemplo:

```
int i = 0;//inicialização
do {
    System.out.println(i);
    i++;
} while (i < 10);      //teste
System.out.println("Fim");
```

22

O comando do...while

- Características
  - Pelo menos uma única vez a seqüência de comandos será executada.
  - O laço é controlado pelo teste da variável “i”.
  - Se a variável “i” for omitida, o laço será executado eternamente.
  - O controle de parada do laço consiste em alterar a condição de teste para *false*.

23

O comando while

- Sintaxe:

```
while (teste booleano) {
    seqüência de comandos;
}
```

24

O comando while

- Exemplo:

```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;   //atualiza a variável de controle
}
System.out.println("Fim");
```

25

O comando for

- Sintaxe:

```
for (ini_expr; teste booleano; alter_expr) {
    seqüência de comandos;
}
```

26

O comando for

- Exemplo:

```
for(int i = 0; i < 10; i ++) {
    System.out.println(i);
}
System.out.println("Fim");
```

27

O comando break

- O comando **break** normalmente é utilizado para efetuar a "saída" do bloco internamente mais aninhado dos comandos **switch**, **for**, **while** ou **do-while**.
  - Sintaxe:

  break;

28

O comando break

- Quando executado, um comando **break** faz com que o fluxo de controle seja desviado para a próxima linha depois do bloco de comandos que contém o **break**.

29

O comando break

- Exemplo:

```
public class ComandoBreak {

    public static void main(String[] args) {
        String saida = "";
        int contador;

        //repete 10 vezes
        for(contador=1; contador<=10; contador++) {
                //se contador for igual a 5, termina o laço
                if(contador == 5)
                        break;
                saida += contador + "-";
        }//fim do for
        System.out.println(saida);
        System.out.println("A variavel contador terminou com o valor:
    "+contador);
    }
}
```

30

O comando *continue*

- Outro comando que altera explicitamente o fluxo em um programa Java é o comando **continue**, que tem a seguinte sintaxe:

  continue;

- O comando **continue** só pode ser usado dentro de laços **for**, **while** e **do-while**.

31

O comando continue

- O comando continue faz com que a execução pule os passos restantes do laço na iteração atual.
- Exemplo:

```
public class ComandoContinue {
    public static void main(String[] args) {
        String saida = "";
        int contador;
        //repete 10 vezes
        for(contador=1; contador<=10; contador++) {
                //se contador for igual a 5, CONTINUA com a
                //próxima iteração do laço
                if(contador == 5)
                        continue;
                saida += contador + " ";
        }//fim do for
        System.out.println(saida);
        System.out.println("A variavel contador terminou com o valor:
    "+contador);
    }
}
```

32