LINGUAGEM DE PROGRAMAÇÃO 2
04– ESTRUTURAS DE CONTROLE

Francisco Barretto – francisco.barretto@udf.edu.br

Roteiro

1. Estruturas de Controle de Fluxo
    1. If
    2. Else
    3. If-else-if
    4. Switch
2. Estrutura de Controle de Repetição
    1. For
    2. While
    3. Do

Sintaxe Java

- O Método *main*
  - ```
    class HelloWorld {
        public static void main( String[] args ) {
            System.out.println( "Hello World!" );
        }
    }
    ```
  - Quando o interpretador Java executa uma aplicação, ele começa chamando o método main
  - O método main então chama todos os outros métodos necessários para executar a aplicação
    ```

1. Estruturas de Controle

- As estruturas de controle de fluxo são fundamentais para qualquer linguagem de programação.
- Sem elas só haveria uma maneira do programa ser executado: de cima para baixo comando por comando. Não haveria condições, repetições ou saltos.
- A linguagem Java possui diversos comandos de controle de fluxo.
- É possível resolver todos os problemas sem utilizar todas elas, mas devemos nos lembrar que a elegância e facilidade de entendimento de um programa dependem do uso correto das estruturas no local certo.

1.1 If

- O comando if representa uma tomada de decisão do tipo "SE isto ENTÃO aquilo";
- A sua forma geral é:

```
if (condicaoBooleana) {
  codigo;
}
```

- A condição do comando **if** é uma expressão que será avaliada:
  - Se o resultado for false a declaração não será executada.

1.1 If

- Uma condição booleana é qualquer expressão que retorne true ou false. Para isso, você pode usar os operadores <, >, <=, >= e outros.
- Um exemplo:

```
int idade = 15;
if (idade < 18) {
  System.out.println("Não pode entrar");
}
```

2.1 IF

- Para comparar se uma variável tem o mesmo valor que outra variável ou valor, utilizamos o operador ==. Repare que utilizar o operador = dentro de um if vai retornar um erro de compilação, já que o operador = é o de atribuição.

```
int mes = 1;
if (mes == 1) {
  System.out.println("Você deveria estar de férias");
}
```

Exemplo IF


```
public class UsandoIf {
    public static void main(String args[]) {
       int num = 15;
       if (num>10) System.out.println("\n\nO numero e
maior que 10");
       if (num==10)
       {
             System.out.println ("\n\nVoce acertou!\n");
             System.out.println ("O numero e igual a
10.");
        }
       if (num<10) System.out.println("\n\nO numero e
menor que 10");
       return (0);
}
```

1.2 Else

- Podemos pensar no comando **else** como sendo um complemento do comando **if;**

```
if (condição) declaração_1;
else declaração_2;
```

- A expressão da condição será avaliada:
  - Se ela for true será executada.
  - Se ela for false declaração 2 será executada.
- É importante nunca esquecer que, quando usamos a estrutura **if-else**, estamos garantindo que uma das duas declarações será executada. Nunca serão executadas as duas ou nenhuma delas.

1.2 Else

- É possível concatenar expressões booleanas através dos operadores lógicos "E" e "OU".
- O "E" é representado pelo && e o "OU" é representado pelo ||.
- Um exemplo seria verificar se ele tem menos de 18 anos e se ele não é amigo do dono:

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && amigoDoDono == false) {
  System.out.println("Não pode entrar");
}
else {
  System.out.println("Pode entrar");
}
```

1.2 Else

- Esse código poderia ficar ainda mais legível, utilizando-se o operador de negação, o !. Esse operador transforma o resultado de uma expressão booleana de false para true e vice versa.

```
int idade = 15;
boolean amigoDoDono = true;
if (idade < 18 && !amigoDoDono) {
System.out.println("Não pode entrar");
}
else {
System.out.println("Pode entrar");
}
```

1.3 If-Else-If

- A estrutura **if-else-if** é apenas uma extensão da estrutura **if-else;**
- ```
  if (condição_1) declaração_1;
  else if (condição_2)
  {declaração_2;}
  .
  .
  .
  else if (condição_n) declaração_n;
  else declaração_default;
  ```
  ```

1.4 Switch

- O comando **if-else** e o comando **switch** são dois comandos de decisão;
- O comando **switch** é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos.

1.4 Switch


```
switch (variável)
{
        case constante_1:
                declaração_1;
                break;
        case constante_2:
                declaração_2;
                break;
        (…)
        case constante_n:
                declaração_n;
                break;
        default
                declaração_default;
}
```

1.4 Switch

- Podemos fazer uma analogia entre o **switch** e a estrutura **if-else-if** apresentada anteriormente.
- A diferença fundamental é que a estrutura **switch** não aceita expressões. Aceita apenas constantes.
- O **switch** testa a variável e executa a declaração cujo case corresponda ao valor atual da variável.
- A declaração default é opcional e será executada apenas se a variável, que está sendo testada, não for igual a nenhuma das constantes.
- O comando break, faz com que o switch seja interrompido assim que uma das declarações seja executada. Se após a execução da declaração não houver um break, o programa continuará executando.

1.4 Switch


```
public class ExemploSwitch {
    public static void main(String args[]) {
        int diaDaSemana = 1;
        switch (diaDaSemana) {
            case 1:
                System.out.println("Domingo");
                break;
            case 2:
                System.out.println("Segunda-feira");
                break;
            case 3:
                System.out.println("Terça-feira");
                break;
            case 4:
                System.out.println("Quarta-feira");
                break;
            case 5:
                System.out.println("Quinta-feira");
                break;
            case 6:
                System.out.println("Sexta-feira");
                break;
             case 7:
                System.out.println("Sábado");
                break;
            default:
                 System.out.println("Este não é um dia válido!");
         }
    }
}
```

2.1 For

- **For** é a primeira de uma série de três estruturas para se trabalhar com loops de repetição. As outras são **while** e **do**.
- O loop **for** é usado para repetir um comando, ou bloco de comandos, diversas vezes, de maneira que se possa ter um bom controle sobre o loop.

```
for (inicializacao; condicao; incremento) {
  codigo;
}
```

Exemplo For


```
public class UsandoFor {
    public static void main(String[] args) {
        for(int count=1 ; count <= 10 ; count++){
            System.out.println(count);
        }
    }

}
```

2.2 While

- A estrutura **while** testa uma condição. Se esta for verdadeira a declaração é executada e faz-se o teste novamente, e assim por diante.

```
while (condição) declaração;
```

Exemplo: While


```
int idade = 15;
while (idade < 18) {
   System.out.println(idade);
   idade = idade + 1;
}
```

2.3 Do

- A estrutura **do-while** executa a declaração, testa a condição e, se esta for verdadeira, volta para a declaração.

```
do
{
declaração;
} while (condição);
```

- A diferença do comando do-while é que ele, ao contrário do for e do while garante que a declaração será executada pelo menos uma vez.

Bibliografia

Básica:

- DEITEL, H. M.; DEITEL, P. J. Java: Como Programar. 8. ed. São Paulo: Pearson Education, 2010
- MANZANO, José Augusto N. G.; COSTA JUNIOR, Roberto Afonso da. JAVA II: programação e computadores - guia básico de introdução, orientação e desenvolvimento. 1. ed. São Paulo: Érica, 2006
- SANTOS, R. Introdução à Programação Orientada a Objetos Usando Java. 1. Ed. Rio de Janeiro: Campus, 2003.

Bibliografia

Complementar:

- ARNOLD, Ken; GOSLING, James; HOLMES, David. A linguagem de programação Java. 4. ed. Porto Alegre: Bookman, 2007. xiii, 799 p. ISBN 978856003164l
- RODRIGUES FILHO, R. Desenvolva aplicativos com Java 2. São Paulo: Érica, 2005
- ROMAN, Ed; AMBLER, Scott W.; JEWELL, Tyler. Dominando Enterprise Javabeans. 2. ed. Porto Alegre: Grupo A, 2014
- RUTTER, Jake. Smashing jQuery: Interatividade Avançada com JavaScript Simples. Porto Alegre: Grupo A, 2012
- SIERRA, K.; BATES, B. Use a Cabeça! Java. 2. ed. Rio de Janeiro: Alta Books, 2007.

Contato

francisco.barretto@udf.edu.br