

## Material de Apoio Aula 6

### Controle de fluxo de execução: Seleção múltipla

O comando `if` (ou `if/else`) permite realizar uma forma de seleção, considerando uma condição única, a qual pode conter apenas um de dois valores: `true` ou `false`, conforme a condição for verdadeira ou falsa. Quando se fizer necessário testar diversos valores possíveis, e a cada um destes valores associar um determinado conjunto de operações, um comando de seleção múltipla se faz necessário. Em Java este comando é o `switch`. O comando `switch` permite avaliar uma variável do tipo inteiro, ou uma expressão retornando um valor inteiro, e associar comportamentos (trechos de código) específicos para cada um dos valores possíveis.

Sintaxe do comando `switch`:

```
switch ( expressão ) {           // Avalia a expressão inteira (ou a variável inteira)
  case valor_1 : comando;        // Início do conjunto de comandos associado ao valor_1
    comando;
    ...
    break;                       // Necessário não desejar executar comandos da seqüência
  case valor_2 : comando;        // Início do conjunto de comandos associado ao valor_2
    comando;
    ...
    break;
  ...
  default : comando;            // Caso não tenha encontrado nenhum valor que case com a
    comando;                    // expressão, executa a opção default.
    ...
}
```

Na utilização do comando `switch`, observe os seguintes pontos:

- A expressão utilizada no `switch` deve retornar um valor inteiro.
- Os valores informados em cada `case` devem ser valores inteiros constantes.
- O comando `break` não faz parte do comando `switch`. Seu uso é opcional, no entanto, é necessário quando deseja-se interromper a execução de uma seqüência de instruções. O caso típico é, tendo encontrado um `case` cujo valor seja o esperado pela expressão, executar o conjunto de instruções a ele associado, não avançando sobre as instruções do `case` associado a um outro valor. Devido a isto, usa-se o `break`.
- O valor `default` é opcional. O conjunto de comandos a ele associado somente é executado caso não seja encontrado nenhum `case` que tenha o valor especificado na expressão.
- O valor `default` necessariamente o último no `switch`.
- É desnecessário `break` no último valor, pois a execução não irá avançar sobre outras opções de valor.

### Exercícios

1. Construa uma classe `Calculadora`. Implemente, nesta classe, métodos correspondendo às quatro operações básicas. Os métodos realizam as operações correspondentes sobre os parâmetros recebidos (assuma todos `int`) e retornam o resultado (igualmente `int`). Implemente outro método, `opera`, que recebe 3 parâmetros (todos `int`): os dois primeiros correspondem a dois operandos e o terceiro a uma operação. O método `opera` invoca a realização da operação correspondente, assumindo que 1:+, 2:-, 3:\* e 4:/. O resultado default é 0.
2. Modifique a classe `Venda` do Exercício 13 da lista de Apoio 5 de forma a utilizar o comando `switch` para verificar a prestação de associada a uma venda.

## Controle de fluxo de execução: Iteração

Em muitos algoritmos, uma estratégia muito comum é repetir um número finito de vezes um trecho de código em busca de um resultado. Esta repetição, usualmente chamada *loop* ou *laço*, requer a introdução no programa de um comando de controle de quantas vezes o corpo do loop deve ser executado. Cada execução é chamada de *iteração*. Para que a programação de uma iteração seja bem sucedida, o programador deve incluir no seu programa os seguintes elementos:

- Um comando de controle de fluxo por repetição. Existem três opções:
  - **while**
  - **do ... while**
  - **for**
- Uma variável de iteração
  - Para controlar o avanço da iteração.
- Um valor inicial para a variável de iteração.
  - Definindo qual o valor inicial da variável de iteração na primeira iteração.
- Modificação da variável de condição
  - Definir qual novo valor a variável de iteração deve assumir quando cada iteração for completada.
- Teste de continuação
  - A variável de iteração deve ser testada, de forma a avaliar se o loop deve ou não ser encerrado.

### Comando while

O comando `while` permite controlar a execução de um loop. O teste de continuação da repetição é realizado na entrada de cada iteração através de uma expressão que retorna um valor booleano (`true` ou `false`). Neste caso, o valor `true` permite a entrada para a próxima iteração e o valor `false` interrompe o processo iterativo. Note que, como o teste é realizado antes de entrar na iteração, pode ocorrer que, em determinadas situações, nenhuma iteração seja executada.

A sintaxe do comando `while` é a seguinte:

```
// Sintaxe 1: o loop é representado pela execução de uma única instrução.
while ( expressão )
    comando;

// Sintaxe 2: o loop é representado pela execução de um bloco de instruções.
while ( expressão ) {
    comando;
    comando;
    ...
}
```

Observe que na sintaxe apresentada os seguintes elementos necessários a um loop encontram-se implícitos:

- Variável de iteração: ela pode ser de qualquer tipo, mas deve, obrigatoriamente, ser utilizada na expressão.
- Valor inicial para a variável de iteração: deve ser informado antes do comando `while`.
- Modificação da variável de iteração: deve ser obrigatoriamente realizado em cada iteração, por um dos comandos executados pelo loop. Caso não seja realizado o incremento, existe grande probabilidade do programa entrar em loop infinito (*pendurar*).

Considerando a sintaxe do comando `while` e os elementos necessários a uma instrução de loop, a estrutura do comando `while` pode ser representada por:

```
// Exemplo 1: o loop é representado pela execução de uma única instrução.
variável_de_iteração = valor_inicial;
while ( expressão_que_testa_variável_de_iteração )
    comando_que_modifica_variavel_de_iteração;

// Exemplo 2: o loop é representado pela execução de um bloco de instruções.
variável_de_iteração = valor_inicial;
while ( expressão ) {
    comando;
    comando;
    ...
    comando_que_modifica_variavel_de_iteração;
}
```

## Exemplos:

<ul style="list-style-type: none"><li>• Implemente um método que imprima todos os números inteiros entre 0 e 10.</li></ul>	<ul style="list-style-type: none"><li>• Implemente um método que imprima todos os números inteiros pares maiores que 0 e menores que 10. Assuma que 0 é um número par.</li></ul>	<ul style="list-style-type: none"><li>• Implemente um método que imprima todos os números inteiros maiores que 0 e menores ou iguais a 10 em ordem invertida.</li></ul>
<pre>void imprimeInteiros() {     int i;     Saida s;      s = new Saida();      i = 0;     while ( i &lt;= 10 ) {         s.print(i);         i++;     } }</pre>	<pre>void imprimeInteirosPares() {     int i;     Saida s;      s = new Saida();      i = 0;     while ( i &lt; 10 ) {         s.print(i);         i = i + 2;     } }</pre>	<pre>void imprimeInteirosPares() {     int i;     Saida s;      s = new Saida();      i = 10;     while ( i &gt;= 0 ) {         s.print(i);         i--;     } }</pre>
<ul style="list-style-type: none"><li>• Implemente um método na classe <code>Desenha</code> que desenhe um quadrado de lado 10 com "*" (asteriscos) na tela (este programa tem um erro de lógica, ver Exercício 1).</li></ul>		
<pre>class Desenha {     private Saida s;      public Desenha() {         s = new Saida();     }      public void quadradoLadoDez() {         int i, j;          i = 1;          // Inicializa as variáveis de controle das ...         j = 1;          // ... iterações         while( i &lt; 10 ) { // Controla número de linhas             while( j &lt; 10 ) { // Controla número de colunas                 s.print("*");                 j++;        // Imprimiu a coluna j na linha i             } // while j             s.print("\n");  // A linha i foi impressa             i++;           // Ir para a linha i+1         } // while i     } // método } // classe</pre>		

## Exercícios

1. O método não realiza corretamente o problema proposto. Corrija a implementação.
2. (Com `while`) Implemente uma classe `Calculadora` que tenha os seguintes métodos:
  - a) Um método `Acumula`, que recebe como parâmetro um valor inteiro e retorna a soma de todos os valores entre 0 e o número lido. Não assuma que o valor seja positivo: caso o valor seja menor ou igual a zero, o valor retornado deve ser 0.
  - b) Um método `Fatorial`, que recebe como parâmetro um valor inteiro e retorna o fatorial deste número.  $Fat(n) = 1$ , se  $n = 1$  ou  $n = 0$ , e  $Fat(n) = n * Fat(n-1)$ . Não existe fatorial de número negativo.
  - c) Um método `maior`, que mantém um estado interno (`int`) no objeto que contém o maior valor informado, como parâmetro, para o método. No método `main`, números inteiros devem ser lidos do teclado até que seja informado -999 (que indica saída do programa).
3. (Com `switch`) Dentro do método `main`, implemente um trecho de código que leia um número inteiro. Se este número for 0, imprimir Domingo. Se for 1, imprimir Segunda, 2, Terça e assim por diante até 7, Sábado. Qualquer outro valor imprimir a mensagem "Dia inválido".
4. (Com `switch` e `while`) Dentro do método `main`, implemente um trecho de código que leia um número inteiro até que o usuário digite o número 99. Nos demais casos, avaliar se este número for 0, imprimir Domingo. Se for 1, imprimir Segunda, 2, Terça e assim por diante até 7, Sábado. Qualquer outro valor imprimir a mensagem "Dia inválido".

5. (Com `switch` e `while`) Dentro do método `main`, implemente um trecho de código que crie um objeto da classe `Carro`. A partir deste momento, execute em um loop, a leitura de um número do teclado e, conforme o valor deste número, faça as seguintes operações sobre o objeto criado:
- 1: avança o carro. Deve perguntar a distância a ser percorrida.
  - 2: abastece o carro. Deve perguntar a quantidade de quilômetros a ser percorrida.
  - 3: pede status do carro. Deve ser informado o status do carro (combustível, quilômetros já percorridos e quilômetros que podem ser percorridos com a quantidade de combustível disponível).

6. (Com `while`) Implemente uma classe `Desenha`, que possua três métodos que recebam um valor inteiro como parâmetro e que sejam capazes de desenhar, cada um, as seguintes forma geométricas usando o caracter `*`:

<code>void quadrado(int n)</code> n corresponde ao lado	<code>void triangulo(int n)</code> n corresponde a base e a altura	<code>void diagonal(int n)</code> n corresponde ao lado
<pre> ***** ***** ***** ***** ***** ***** ***** ***** </pre>	<pre> * ** *** **** ***** ***** ***** </pre>	<pre> \***** *\***** **\**** ***\*** ****\** *****\* *****\ </pre>

7. (Com `while`, `if` e uso de `String`) Utilize a classe `Aluno` já desenvolvida anteriormente. Construa, no método `main`, um algoritmo que leia o nome e as notas (GA e GB) dos alunos e imprima o GC, indicando se o aluno foi ou não aprovado.

- d) Considere que a turma tenha 5 alunos.  
e) Considere que o número de alunos não é conhecido, devem ser lidos nomes até que seja digitado como nome do aluno o nome “fim”.

8. Modifique o Exercício 7b, de forma que:

- a) Seja informado a média da turma.  
b) Sejam contados quantos alunos foram aprovados e quantos reprovados.  
c) Seja informado o nome dos alunos com a maior e a menor média (considere que não haverá empate).

9. Adicione, na classe `Calculadora`, um método `maximoValor`, que receba um número inteiro como parâmetro. Este número corresponde ao número de bits utilizado para armazenar um número. O método deve retornar o maior número que pode ser armazenada por uma variável que tenha o número de bits informados. Lembre que em uma representação de bits, para um comprimento de  $n$  bits, é possível obter  $2^n - 1$  valores. Por exemplo, para um comprimento de 8 bits, o maior número que pode ser obtido é:

$$2^8 - 1 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 - 1 = 255$$

10. Descubra o número. Jogo onde o jogador tenta descobrir qual foi o número gerado aleatoriamente pelo computador, para tanto, o jogador tem um número limitado de rodadas. Construa uma classe `Aposta`. O construtor deve inicializar o valor a ser “encontrado”. Um método `joga`, recebendo um inteiro como parâmetro, permite que o jogador jogue uma rodada. Este método retorna um valor inteiro – este valor pode ser negativo, positivo ou zero. Se for zero, o número apostado é igual ao número sorteado. Se for positivo, o número apostado é maior que o número sorteado. Se for negativo, o número apostado é menor que o número sorteado. Dica: obtenha, a partir de `Math.random()` um número inteiro multiplicando o valor por 100 – note que assim, o maior valor possível é 99.

- a) Fazer uma versão do método `main` que permita um número limitado de rodadas.  
b) Fazer uma versão do método `main` que force tantas rodadas quantas se fizerem necessárias para descobrir o número sorteado.