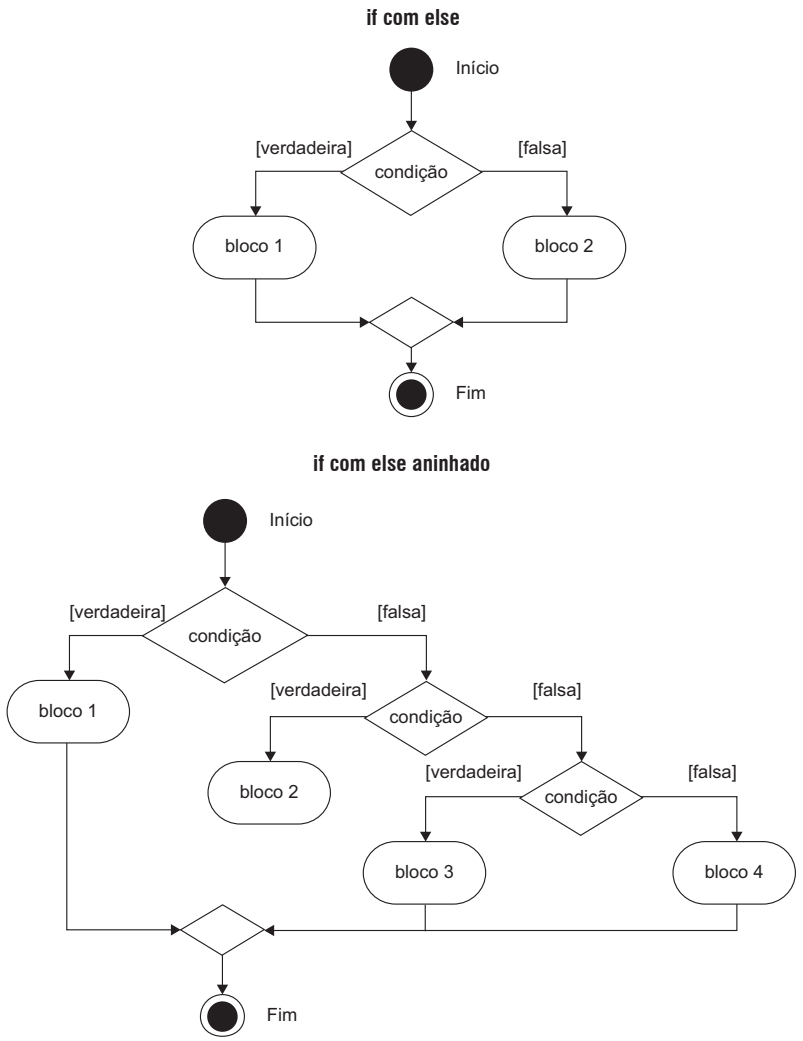

Figura 3.2 – Variações da estrutura condicional.

No primeiro caso é executado um bloco de instruções somente se a condição for verdadeira; no segundo caso serão executados um bloco de instruções para a condição verdadeira e outro para a falsa; já no terceiro caso, quando encontrada a primeira condição verdadeira todas as outras serão desconsideradas. Seja o caminho que for, apenas um bloco de instruções será executado. O terceiro caso demonstra que para cada condição falsa é executada uma outra condição, mas poderia ser representado ao contrário, ou seja, a condição verdadeira poderia levar à execução de uma outra condição.

Os exemplos seguintes demonstram esses três tipos de estrutura em Java.

Estrutura 1: if sem else

O Exemplo 3.1 mostra um uso prático do if sem a presença do else. Trata-se de uma classe em que o usuário seleciona uma opção (Masculino ou Feminino) e a partir disso é usada a instrução if para executar instruções diferentes. Vamos analisar o exemplo.

Exemplo 3.1 – Listagem da classe If

```
1  package cap03;
2  import javax.swing.JOptionPane;
3  public class If {
4      public static void main(String[] args) {
5          Object[] op = {"Masculino", "Feminino"};
6          String resp = (String) JOptionPane.showInputDialog(null,
7                  "Selecione o sexo:\n", "Pesquisa",
8                  JOptionPane.PLAIN_MESSAGE,
9                  null, op, "Masculino");
10         if (resp == null) {
11             JOptionPane.showMessageDialog(null, "Você pressionou Cancel");
12         }
13         if (resp == "Masculino") {
14             JOptionPane.showMessageDialog(null, "Você é homem.");
15         }
16         if (resp == "Feminino") {
17             JOptionPane.showMessageDialog(null, "Você é mulher.");
18         }
19         System.exit(0);
20     }
21 }
```

Funcionalidades comentadas do Exemplo 3.1:

- **Linha 5:** cria um array de objetos chamado **op** contendo os valores "Masculino" e "Feminino". Esses são os itens que aparecem para o usuário escolher. Se precisar de outras opções, basta adicionar seguindo o mesmo padrão. O estudo detalhado de arrays é apresentado no Capítulo 5. Por enquanto, considere que um array permite criar uma relação de valores.
- **Linhas 6 a 9:** conforme abordado no capítulo anterior, o método showInputDialog da caixa de diálogo JOptionPane permite que um valor seja fornecido pelo usuário. Nesse caso em específico, estamos apresentando uma maneira de customizar essa caixa de diálogo, pois, em vez de apresentar uma caixa de texto para digitação de um valor (a forma padrão), é apresentada uma lista de opções. O estudo mais aprofundado da classe JOptionPane é dado no Capítulo 8. Nesse momento vamos apresentar apenas alguns aspectos relacionados ao método showInputDialog. As linhas 6 e 9 formam uma única linha, mas, por questões didáticas, achamos melhor dividi-la em partes. Vamos entender isso seguindo as instruções da esquerda para a direita.

  String resp: declara uma variável do tipo String chamada resp. Essa variável receberá o conteúdo selecionado pelo usuário na caixa de diálogo.

  (String): um mecanismo de conversão, isto é, o resultado da instrução à direita será convertido para o tipo String.

  JOptionPane.showInputDialog: método showInputDialog da classe JOptionPane que abrirá a caixa de opções para o usuário. Neste exemplo, o método recebe 7 valores (tecnicamente chamados de parâmetros): o 1.º se refere ao local em que a caixa de mensagens será exibida: como está definido null a caixa de diálogo será exibida no centro da tela; o 2.º se refere à mensagem que será exibida ao usuário, no caso "selecione o sexo"; o 3.º se refere ao título que será exibido na caixa de mensagem, no caso "Pesquisa"; o 4.º se refere a um valor inteiro que define qual ícone será exibido na caixa de mensagem, no caso uma interrogação por meio da constante (JOptionPane. QUESTION_MESSAGE); o 5.º se refere a um ícone externo que pode ser usado, isto é, pode ser definida uma imagem que será exibida na caixa de diálogo; o 6.º se refere à lista de opções que serão exibidas para o usuário selecionar, no caso definido pela variável op (um array de objetos); o 7.º se refere à opção que aparecerá selecionada por default. Esse texto deve ser igual a uma das opções possíveis de ser selecionada.
- **Linha 10:** a instrução if verifica se foi pressionado o botão *Cancel* da caixa de diálogo, ou seja, caso o usuário pressione o botão *Cancel* no momento em que a caixa de diálogo aparece, então a variável resp recebe o valor null (isso ocorre na linha 6).

Dica

Para realizar várias comparações em um mesmo if, podem ser utilizados os operadores lógicos nas seguintes formas: if (x>y && x<7), if (x>y || x>z || x>k). Todas as comparações devem estar envolvidas por um único parêntese.

- **Linhas 13 a 15:** a instrução if verifica se o conteúdo da variável resp é igual a "Masculino" e mostra a mensagem de "Você é homem".
- **Linhas 16 a 18:** a instrução if verifica se o conteúdo da variável resp é igual a "Feminino" e mostra a mensagem de "Você é mulher".
- **Linha 19:** encerra a aplicação.

A Figura 3.3 ilustra a execução do Exemplo 3.1.

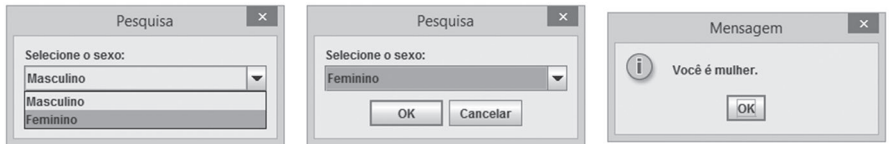

Figura 3.3 – Tela de execução do Exemplo 3.1 com seleção de Feminino.

O Exemplo 3.2 mostra um uso prático do if-else para validar a entrada do usuário. São realizadas três validações: em primeiro lugar, verifica se o usuário realmente entrou com um valor na caixa de diálogo, depois verifica se o valor digitado é numérico, logo a seguir verifica se esse valor está entre 1 e 12 (a faixa de valores possíveis, uma vez que um mês deve assumir apenas valores entre 1 e 12).

Exemplo 3.2 – Listagem da classe IfComElse

```
1   package cap03;
2   import javax.swing.*;
3   public class IfComElse {
4       public static void main(String args[]) {
5           String aux = JOptionPane.showInputDialog("Forneça o número do mês");
6           if (aux != null) {
7               try {
8                   int mes = Integer.parseInt(aux);
9                   if (mes >= 1 && mes <= 12) {
10                      JOptionPane.showMessageDialog(null, "Número do mês válido!\n " + mes);
11                  } else {
12                      JOptionPane.showMessageDialog(null, "Número do mês inválido!\n " + mes);
13                  }
14              } catch (NumberFormatException erro) {
15                  JOptionPane.showMessageDialog(null, "Digite apenas valores inteiros "+erro);
16              }
17          } else {
18              JOptionPane.showMessageDialog(null, "Operação Cancelada.");
19          }
20          System.exit(0);
21      }
22  }
```

Funcionalidades comentadas do Exemplo 3.2:

- Quando este exemplo for executado, o usuário deve entrar com um valor numérico entre 1 e 12 na caixa de diálogo gerada pela classe JOptionPane.
- **Linha 5:** conforme abordado anteriormente, o método showInputDialog da *classe JOptionPane* permite que um valor seja fornecido pelo usuário. Esse valor é armazenado na variável aux. Como se notou, a caixa de diálogo possui os botões *OK* e *Cancel*. Caso o usuário pressione o botão *Cancel*, o valor da variável aux será nulo (null).
- **Linha 6:** verifica se o usuário pressionou o botão *Cancel* da caixa de diálogo presente na linha 5, ou seja, se o valor de aux é diferente de nulo (!=null). Se for diferente de nulo, executa o trecho entre chaves composto pelas linhas 7 a 17.

  O bloco **try catch** (entre as linhas 7 e 16) é o responsável por verificar se o usuário digitou um valor numérico, uma vez que valores indevidos provocam erros de conversão. Maiores detalhes sobre o bloco try catch são apresentados na Seção 3.2 deste capítulo.
- **Linha 9:** contém a instrução **if** responsável por verificar se o número referente ao mês, digitado pelo usuário, está compreendido entre os valores 1 e 12. Se a comparação for verdadeira, envia uma mensagem positiva (linha 10); caso seja falsa, executa a instrução **else** (linha 11) e envia uma

mensagem negativa (linha 12). Observe que no interior dos parênteses da instrução if existem duas comparações unidas pelo operador lógico e (&&).

- Dessa forma, quando o usuário fornecer um valor inteiro entre 1 e 12 aparecerá a mensagem indicando que o mês é válido, caso contrário aparecerá a mensagem indicando que o mês é inválido.

A Figura 3.4 ilustra a execução do Exemplo 3.2.

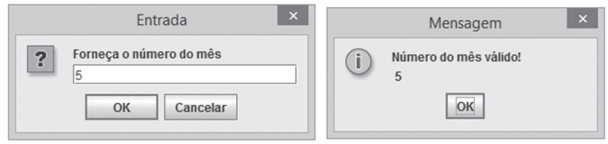

Figura 3.4 – Tela de execução do Exemplo 3.2.

O Exemplo 3.3 mostra como é possível criar uma estrutura em que cada instrução else realiza a abertura de um novo if. Ao analisar essa estrutura, podemos notar que existe um (ou mais) if dentro de um else.

Exemplo 3.3 – Listagem da classe IfComElseAninhado

```
1  package cap03;
2  import javax.swing.*;
3  public class IfComElseAninhado {
4      public static void main(String args[]) {
5          String aux = JOptionPane.showInputDialog("Forneça o número do mês");
6          if (aux != null) {
7              try {
8                  int mes = Integer.parseInt(aux);
9                  if (mes == 1) {
10                     aux = "Janeiro";
11                 } else if (mes == 2) {
12                     aux = "Fevereiro";
13                 } else if (mes == 3) {
14                     aux = "Março";
15                 } //inserir todos os outros meses
16                 else if (mes == 12) {
17                     aux = "Dezembro";
18                 } else {
19                     aux = "Mês Desconhecido!";
20                 }
21                 JOptionPane.showMessageDialog(null, aux);
22             } catch (NumberFormatException erro) {
23                 JOptionPane.showMessageDialog(null, "Digite apenas valores inteiros "+erro);
24             }
25         }
26         System.exit(0);
27     }
28 }
```

Conforme citamos anteriormente, nesse tipo de estrutura, ao se encontrar a primeira condição verdadeira, todas as outras são desprezadas e o controle da execução é levado ao final da primeira instrução if que iniciou o processo. Por exemplo: se na linha 9 a condição mes == 1 for verdadeira, todas as condições abaixo são desprezadas e a execução do programa salta para a linha 21. Essa característica da estrutura permite economizar tempo, pois se a condição verdadeira já foi encontrada não existe necessidade de se testar outras condições. Se fosse utilizada a estrutura do if sem o else (um if simples para cada condição como no Exemplo 3.1), todas as 12 comparações seriam executadas sempre, uma a uma, mesmo que a condição verdadeira já tivesse sido encontrada anteriormente.

3.1.2 Estrutura if resumido

Agora que o leitor já conhece o funcionamento da estrutura condicional if, vamos apresentar uma sintaxe alternativa por meio do operador interrogação (?).

Para facilitar a compreensão, vamos analisar o Exemplo 3.4, cuja listagem aparece em seguida.

Exemplo 3.4 – Listagem da classe IfResumido

```
1  package cap03;
2  import javax.swing.JOptionPane;
3  public class IfResumido {
4      public static void main(String[] args) {
5          int a = 10, b = 15, maior;
6          if (a > b) {
7              maior = a;
8          } else {
9              maior = b;
10         }
11         JOptionPane.showMessageDialog(null, "Usando um if comum: " + maior);
12
13         int c = 10, d = 8;
14         maior = (c > d) ? c : d;
15         JOptionPane.showMessageDialog(null, "Usando um if resumido: " + maior);
16     }
17 }
```

Funcionalidades comentadas do Exemplo 3.4:

- O exemplo realiza a comparação de dois valores e apresenta o maior deles em tela. Esse processo é realizado duas vezes, uma usando a estrutura if-else que apresentamos anteriormente e outra por meio do if reduzido.
- **Linhas 5 a 11:** são declaradas duas variáveis inteiras com os valores 10 e 15. Na estrutura if compara-se se o valor das variáveis (**a** e **b**). O maior valor dentre as duas variáveis é armazenado na variável **maior**. A linha 11 imprime na caixa de diálogo o valor da maior variável, no caso o valor 15.
- **Linhas 13 a 15:** fazem exatamente a mesma coisa do tópico anterior, porém de forma resumida. A linha 13 declara e inicializa o valor nas variáveis **c** e **d**. Na linha 14, o operador ? faz o papel do if do tópico anterior. Da esquerda para a direita podemos interpretar a sintaxe da forma seguinte:

  (c > d) ? → o conteúdo da variável **c** é maior do que o conteúdo da variável **d**?

  c : d → caso a pergunta anterior seja verdadeira, retorna o conteúdo da variável c, caso contrário, o conteúdo da variável d. Dessa forma, a variável **maior** receberá o maior valor dentre as duas variáveis.

3.1.3 A estrutura switch-case

A estrutura switch-case se refere a uma outra modalidade de desvio da execução do programa de acordo com certas condições, semelhante ao uso da instrução if. Ao trabalhar com uma grande quantidade de desvios condicionais contendo instruções if, pode-se comprometer a inteligibilidade do programa, dificultando sua interpretação. A estrutura switch-case possibilita uma forma mais adequada e eficiente de atender a esse tipo de situação, constituindo-se uma estrutura de controle com múltipla escolha.

A estrutura switch-case equivale a um conjunto de instruções if encadeadas, fornecendo maior inteligibilidade. Sua sintaxe é a seguinte:

```
switch (<expressão>){
  case 1: instruções; break;
  case 2: instruções; break;
  case 3: instruções; break;
  default: instruções;
}
```

Na primeira linha do **switch** é avaliado o resultado da expressão, que é comparado nas diretivas case, executando o bloco de instruções quando a expressão coincidir com o valor colocado ao lado direito do case. Em outras palavras, supondo que o valor da expressão seja igual a 2, serão executadas as instruções **localizadas entre case 2: e break**. A cada case o programa compara o valor da expressão com o valor colocado no case. Caso os valores sejam iguais, todas as instruções são executadas até que se encontre uma instrução break, que encerra o **switch** e faz a execução do programa desviar para o