)

|  |  |
| --- | --- |
|  | FTCE  Faculdade de Tecnologia e Ciências Exatas |

USJT – 2021/1 –Programação de Soluções Computacionais

**Aula: 03 – Estruturas de seleção**

**1 Introdução**

É comum que programas de computador possuam trechos de código cuja execução seja condicional. Ou seja, de acordo com alguma condição especificada pelo programador, o trecho de código pode ou não ser executado. Isso é diferente do que vimos até agora. Os programas desenvolvidos até então consistem em uma única sequência de instruções, as quais executam uma após a outra, de maneira incondicional. A linguagem Java possui três estruturas de seleção: **if/else** (e suas variações), **switch/case** e o **operador ternário**. Nas seções a seguir iremos estudar as três estruturas.

**2 Possíveis representações**

Programas de computador podem ser representados com pseudocódigo, fluxogramas e outras alternativas. Vejamos alguns exemplos de representação para estruturas de seleção.

**2.1 (Estrutura Se: Pseudocódigo e Fluxograma)**

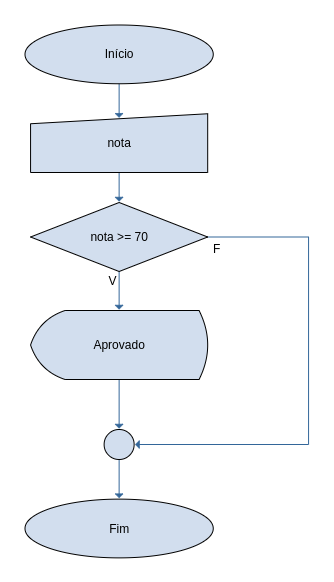
A Listagem 2.1.1 mostra um exemplo de representação de algoritmo que utiliza uma estrutura de seleção do tipo **Se**. O algoritmo opera sobre um valor de nota obtido por um aluno e decide se ele está aprovado. Note que esse tipo de notação admite muitas variações. O que importa é que o conteúdo apresentado seja claro e não ambíguo.

Listagem 2.1.1

|  |
| --- |
| **Var** nota: real;  **Início**  **Ler** nota;  **Se** nota maior ou igual a 70 **Então**  **Escrever** “Aprovado”  **Fim Se**  **Fim** |

A Figura 2.1.1 mostra o mesmo algoritmo utilizando a representação chamada fluxograma.

Figura 2.1.1



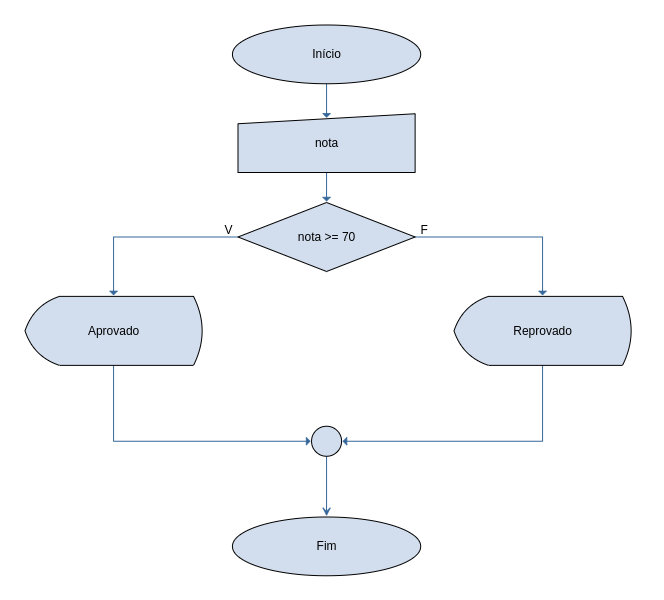
É possível associar um bloco **Senão** a uma estrutura de seleção do tipo **Se**. A execução deles é mutuamente exclusiva. Ou seja, se o bloco **Se** executar, o bloco **Senão** não executa. E vice-versa. Veja o exemplo da Listagem 2.1.2.

Listagem 2.1.2

|  |
| --- |
| **Var** nota: real;  **Início**  **Ler** nota;  **Se** nota maior ou igual a 70 **Então**  **Escrever** “Aprovado”  **Senão**  **Escrever** “Reprovado”  **Fim Se**  **Fim** |

A Figura 2.1.2 mostra essa variação como um fluxograma.

Figura 2.1.2



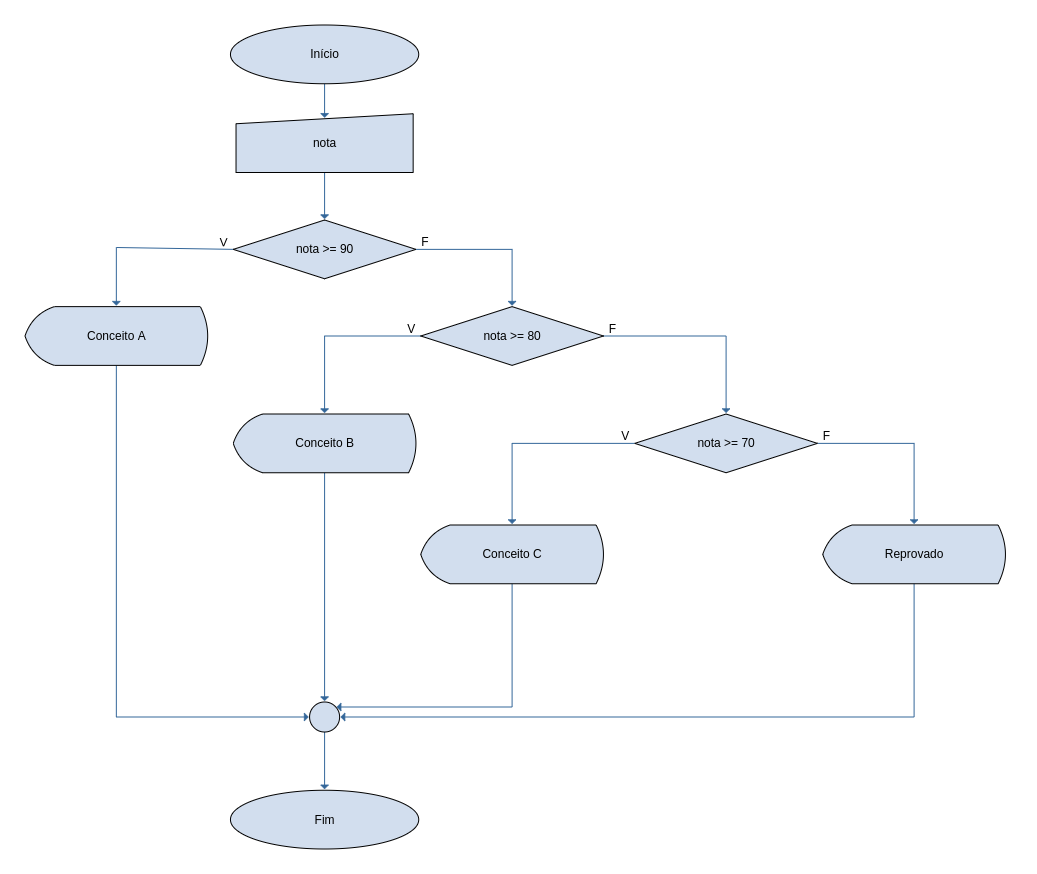
Há também uma variação chamada “**encadeada**”. Veja um exemplo na Listagem 2.1.3.

Listagem 2.1.3

|  |
| --- |
| **Var** nota: real;  **Início**  **Ler** nota;  **Se** nota maior ou igual a 90 **Então**  **Escrever** “Conceito A”  **Senão Se** nota maior ou igual a 80 **Então**  **Escrever** “Conceito B”  **Senão Se** nota maior ou igual a 70 **Então**  **Escrever** “Conceito C”  **Senão**  **Escrever** “Reprovado”  **Fim Se**  **Fim** |

A Figura 2.1.3 mostra um fluxograma que faz uso de uma estrutura de seleção encadeada.

Figura 2.1.3



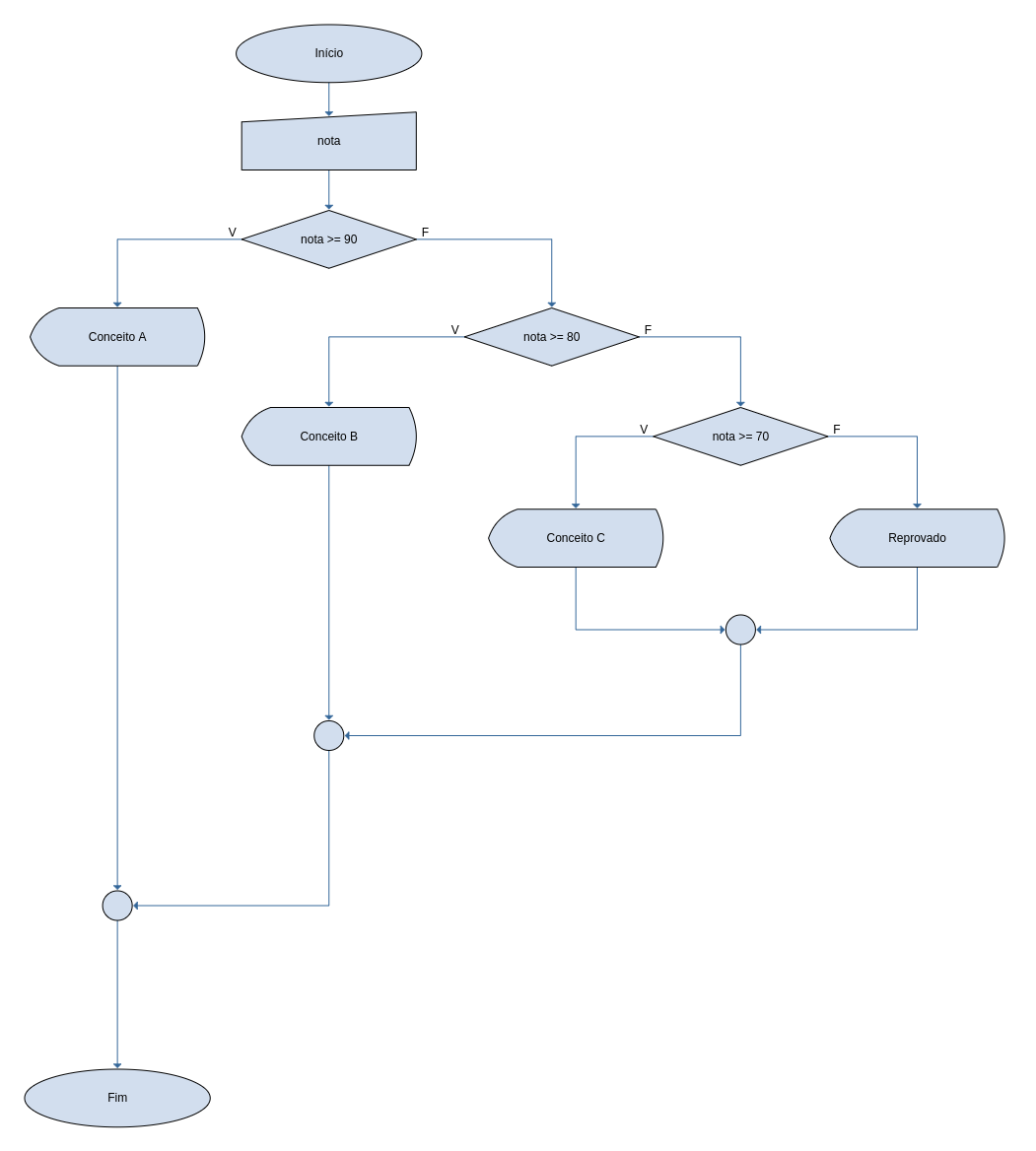
Por fim, vejamos a variação chamada “**aninhada**”. A Listagem 2.1.4 exibe um exemplo.

Listagem 2.1.4

|  |
| --- |
| **Var** nota: real;  **Início**  **Ler** nota;  **Se** nota maior ou igual a 90 **Então**  **Escrever** “Conceito A”  **Senão**  **Se** nota maior ou igual a 80 **Então**  **Escrever** “Conceito B”  **Senão**  **Se** nota maior ou igual a 70 **Então**  **Escrever** “Conceito C”  **Senão**  **Escrever** “Reprovado”  **Fim Se**  **Fim Se**  **Fim Se**  **Fim** |

A Figura 2.1.4 mostra um fluxograma que utiliza uma estrutura de seleção aninhada.

Figura 2.1.4



**2.2 (Estrutura de seleção Avalie/Caso: Pseudocódigo e fluxograma)**

Muitas linguagens de programação disponibilizam uma estrutura chamada switch/case, que pode ser útil quando, por exemplo, há uma lista de valores finita envolvida na seleção do bloco a ser executado. As Listagens 2.2.1 e 2.2.2 mostram exemplos de pseudocódigo para essa estrutura.

Listagem 2.2.1

|  |
| --- |
| **Var** nota: inteiro;  **Início**  **Ler** nota;  **Avalie** nota:  **Caso** 10:  **Escrever** Parabéns!  **Escrever** Conceito A!  **Pare**  **Caso** 9:  **Escrever** Conceito A.  **Pare**  **Caso** 8:  **Escrever** Conceito B.  **Pare**  **Caso** 7:  **Escrever** Conceito C.  **Pare**  **Caso Contrário**:  **Escrever** Reprovado.  **Fim Avalie**  **Fim** |

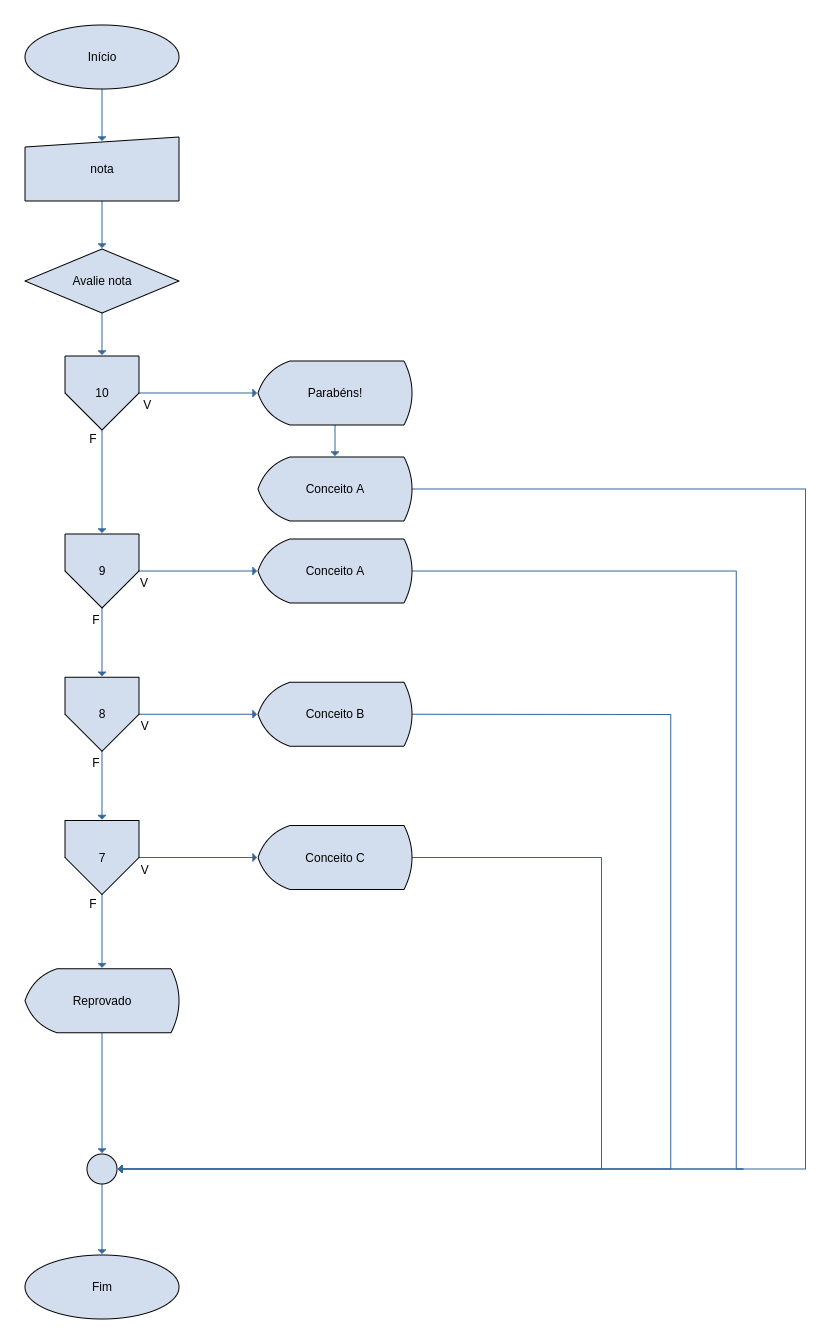
Na Listagem 2.2.2 a importância da instrução **Pare** fica evidente. A estrutura **Avalie/Caso** possui uma espécie de lógica em queda aplicada à sua execução: quando um caso é escolhido, a execução começa ali até uma de duas coisas acontecer: uma instrução **Pare** ser encontrada ou o fim da estrutura ser encontrado.

Listagem 2.2.2

|  |
| --- |
| **Var** nota: inteiro;  **Início**  **Ler** nota;  **Avalie** nota:  **Caso** 10:  **Escrever** Parabéns!  **Caso** 9:  **Escrever** Conceito A.  **Pare**  **Caso** 8:  **Escrever** Conceito B.  **Pare**  **Caso** 7:  **Escrever** Conceito C.  **Pare**  **Caso Contrário**:  **Escrever** Reprovado.  **Fim Avalie**  **Fim** |

A estrutura **Avalie/Caso** pode também ser representada por fluxogramas. Veja um exemplo na Figura 2.2.1.

Figura 2.2.1



A estrutura de seleção conhecida como **operador** **ternário** tem funcionamento análogo à estrutura **Se**. É comum utilizá-la para simplificar o código, muitas vezes fazendo seleções em uma linha só. Sua representação como pseudocódigo ou fluxograma é análoga àquelas da estrutura **Se**.

**3 Java: A estrutura de seleção if/else e suas variações**

Como vimos, uma linguagem de programação pode disponibilizar diferentes estruturas de seleção. No caso da linguagem Java, as três estruturas mencionadas até então estão disponíveis. Vejamos como elas podem ser utilizadas.

**3.1 (A estrutura if simples)** O programa da Listagem 3.1.1 mostra a estrutura de seleção If simples em uso.

Listagem 3.1.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** IfSimples {  **public** **static** **void** main(String[] args) {  **double** nota;  nota = Double.*parseDouble*(JOptionPane.*showInputDialog*("Digite a nota"));  **if** (nota >= 70) {  JOptionPane.*showMessageDialog*(**null**, "Aprovado");  }  }  } |

**3.2 (Associando um bloco else (senão) a um bloco if)** A Listagem 3.2.1 mostra como um bloco if pode ter a ele associado um bloco else. Lembre-se que a execução deles é mutuamente exclusiva. Se um executa, o outro não executa. Um deles sempre executa.

Listagem 3.2.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** IfElse {  **public** **static** **void** main(String[] args) {  **double** nota;  nota = Double.*parseDouble*(JOptionPane.*showInputDialog*("Digite a nota"));  **if** (nota >= 70) {  JOptionPane.*showMessageDialog*(**null**, "Aprovado");  }  **else** {  JOptionPane.*showMessageDialog*(**null**, "Reprovado");  }  }  } |

**3.3 (If/else encadeado)** Veja um exemplo de if/else encadeado na Listagem 3.3.1.

Listagem 3.3.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** IfElseEncadeado {  **public** **static** **void** main(String[] args) {  **double** nota;  nota = Double.*parseDouble*(JOptionPane.*showInputDialog*("Digite a nota"));  **if** (nota >= 90) {  JOptionPane.*showMessageDialog*(**null**, "Parabéns");  JOptionPane.*showMessageDialog*(**null**, "Conceito A");  }  **else** **if** (nota >= 80) {  JOptionPane.*showMessageDialog*(**null**, "Conceito B");  }  **else** **if** (nota >= 70) {  JOptionPane.*showMessageDialog*(**null**, "Conceito C");  }  **else** {  JOptionPane.*showMessageDialog*(**null**, "Reprovado");  }  }  } |

**3.4 (If/else aninhado)** Também é possível aninhar blocos. Na Listagem 3.4.1 ilustramos como isso pode ser feito com blocos if/else.

Listagem 3.4.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** IfElseAninhado {  **public** **static** **void** main(String[] args) {  **double** nota;  nota = Double.*parseDouble*(JOptionPane.*showInputDialog*("Digite a nota"));  **if** (nota >= 90) {  JOptionPane.*showMessageDialog*(**null**, "Parabéns");  JOptionPane.*showMessageDialog*(**null**, "Conceito A");  }  **else** {  **if** (nota >= 80) {  JOptionPane.*showMessageDialog*(**null**, "Conceito B");  }  **else** {  **if** (nota >= 70) {  JOptionPane.*showMessageDialog*(**null**, "Conceito C");  }  **else** {  JOptionPane.*showMessageDialog*(**null**, "Reprovado");  }  }  }  }  } |

**4 Java: A estrutura switch/case**

Nesta seção veremos como a estrutura switch/case pode ser utilizada na linguagem Java.

**4.1 (Estrutura switch/case: exemplo com cases independentes)** Na Listagem 4.1.1 o código exibido faz uso de uma estrutura switch/case em que, para cada case, um bloco específico é determinado. Assim, ela não faz uso da lógica em queda (fall-through), inerente à estrutura.

Listagem 4.1.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** SwitchCase {  **public** **static** **void** main(String[] args) {  **int** nota;  nota = Integer.*parseInt*(JOptionPane.*showInputDialog*("Digite a nota"));  **switch** (nota) {  **case** 10:  JOptionPane.*showMessageDialog*(**null**, "Parabéns");  JOptionPane.*showMessageDialog*(**null**, "Conceito A");  **break**;  **case** 9:  JOptionPane.*showMessageDialog*(**null**, "Conceito A");  **break**;  **case** 8:  JOptionPane.*showMessageDialog*(**null**, "Conceito B");  **break**;  **case** 7:  JOptionPane.*showMessageDialog*(**null**, "Conceito C");  **break**;  **default**:  JOptionPane.*showMessageDialog*(**null**, "Reprovado");  **break**;  }  }  } |

**4.2 (Estrutura switch/case: exemplo usando a lógica em queda)** Alguns cases podem ter funcionamento em comum, que é o que ocorre com os cases 10 e 9 da Listagem 4.1.1. Neste caso, podemos empregar a lógica em queda do switch e escrever menos código, como mostra a Listagem 4.2.1.

Listagem 4.2.1

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** SwitchCaseLogicaEmQueda {  **public** **static** **void** main(String[] args) {  **int** nota;  nota = Integer.*parseInt*(JOptionPane.*showInputDialog*("Digite a nota"));  **switch** (nota) {  **case** 10:  JOptionPane.*showMessageDialog*(**null**, "Parabéns");  **case** 9:  JOptionPane.*showMessageDialog*(**null**, "Conceito A");  **break**;  **case** 8:  JOptionPane.*showMessageDialog*(**null**, "Conceito B");  **break**;  **case** 7:  JOptionPane.*showMessageDialog*(**null**, "Conceito C");  **break**;  **default**:  JOptionPane.*showMessageDialog*(**null**, "Reprovado");  **break**;  }  }  } |

**5 Java: O operador ternário**

O operador ternário (ele opera sobre três operandos, daí o nome) é uma construção da linguagem que tende a simplificar determinados blocos que envolvem decisões. Ele permite que decisões sejam realizadas em uma única linha.

**5.1 (Atribuição simples com operador ternário**) Considere o seguinte exemplo. Desejamos guardar em uma variável uma informação que indica se o usuário pode dirigir em função de sua idade. Obviamente isso pode ser feito com um if/else. O operador ternário permite que isso seja feito de maneira mais simples. Na Listagem 5.1.1 fazemos uso da estrutura if/else já vista. A Listagem 5.1.2 mostra como fazer a mesma coisa usando o operador ternário.

Listagem 5.1.1

|  |
| --- |
| **public** **class** PodeDirigirIfElse {  **public** **static** **void** main(String[] args) {  **int** idade = Integer.*parseInt*(JOptionPane.*showInputDialog*("Quantos anos você tem?"));  String podeDirigir;  **if** (idade >= 18)  podeDirigir = "Sim, você pode dirigir";  **else**  podeDirigir = "Não, você não pode dirigir por enquanto";  JOptionPane.*showMessageDialog*(**null**, podeDirigir);  }  } |

Listagem 5.1.2

|  |
| --- |
| **import** javax.swing.JOptionPane;  **public** **class** PodeDirigirTernario {  **public** **static** **void** main(String[] args) {  **int** idade = Integer.*parseInt*(JOptionPane.*showInputDialog*("Quantos anos você tem?"));  String podeDirigir;  podeDirigir = idade >= 18 ? "Sim, você pode dirigir" : "Não, você não pode dirigir por enquanto";  JOptionPane.*showMessageDialog*(**null**, podeDirigir);  }  } |

***Exercícios***

1. Escreva um programa que obtém o salário de uma pessoa e diz se ela está ganhando pelo menos o salário-mínimo.

2. Escreva um programa que obtém a idade de uma pessoa e diz se ela é maior, segundo a legislação brasileira. Escreva uma versão com if/else e outra com o operador ternário.

***Referências***

DEITEL, P. e DEITEL, H. **Java Como Programar**. 8ª Edição. São Paulo, SP: Pearson, 2010.

LOPES, A. e GARCIA, G. **Introdução à Programação – 500 Algoritmos Resolvidos**. 1ª Edição. São Paulo, SP: Elsevier, 2002.