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 estudaremos as três estruturas.

2 Estruturas de seleção: fluxograma e pseudocódigo

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

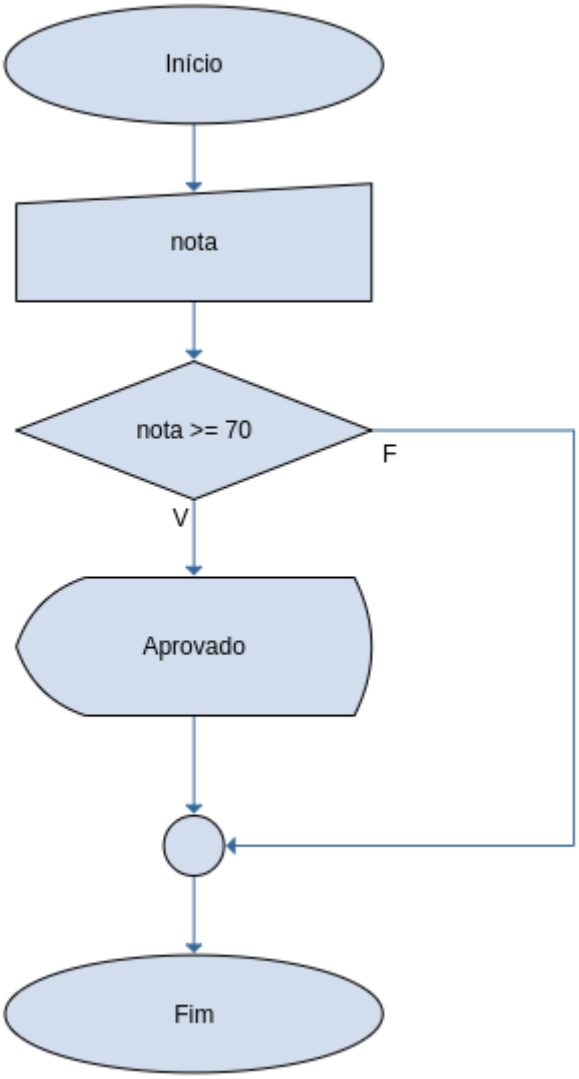
2.1 (Estrutura Se: Pseudocódigo e Fluxograma) O Bloco de Código 2.1.1 mostra um exemplo de representação de algoritmo que utiliza uma estrutura de seleção do tipo Se. O algoritmo opera sobre um valor de nota obtido por um aluno e decide se ele está aprovado. Note que esse tipo de notação admite muitas variações. O que importa é que o conteúdo apresentado seja claro e não ambíguo.

Bloco de Código 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



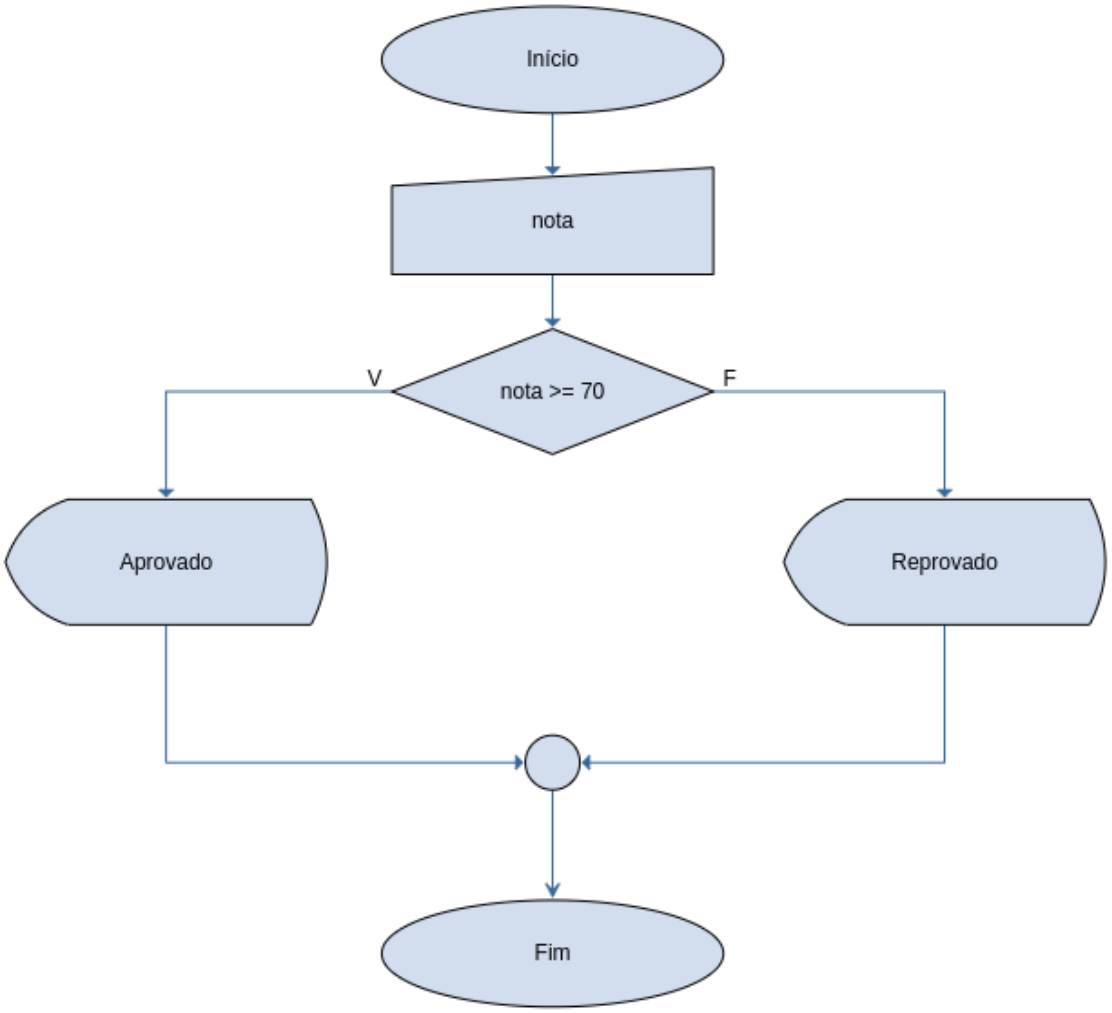
É possível associar um bloco “Senão” a uma estrutura de seleção do tipo Se. A execução deles é mutuamente exclusiva. Ou seja, se o bloco Se executar, o bloco Senão não executa. E vice-versa. Veja o Bloco de Código 2.1.2.

Bloco de Código 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



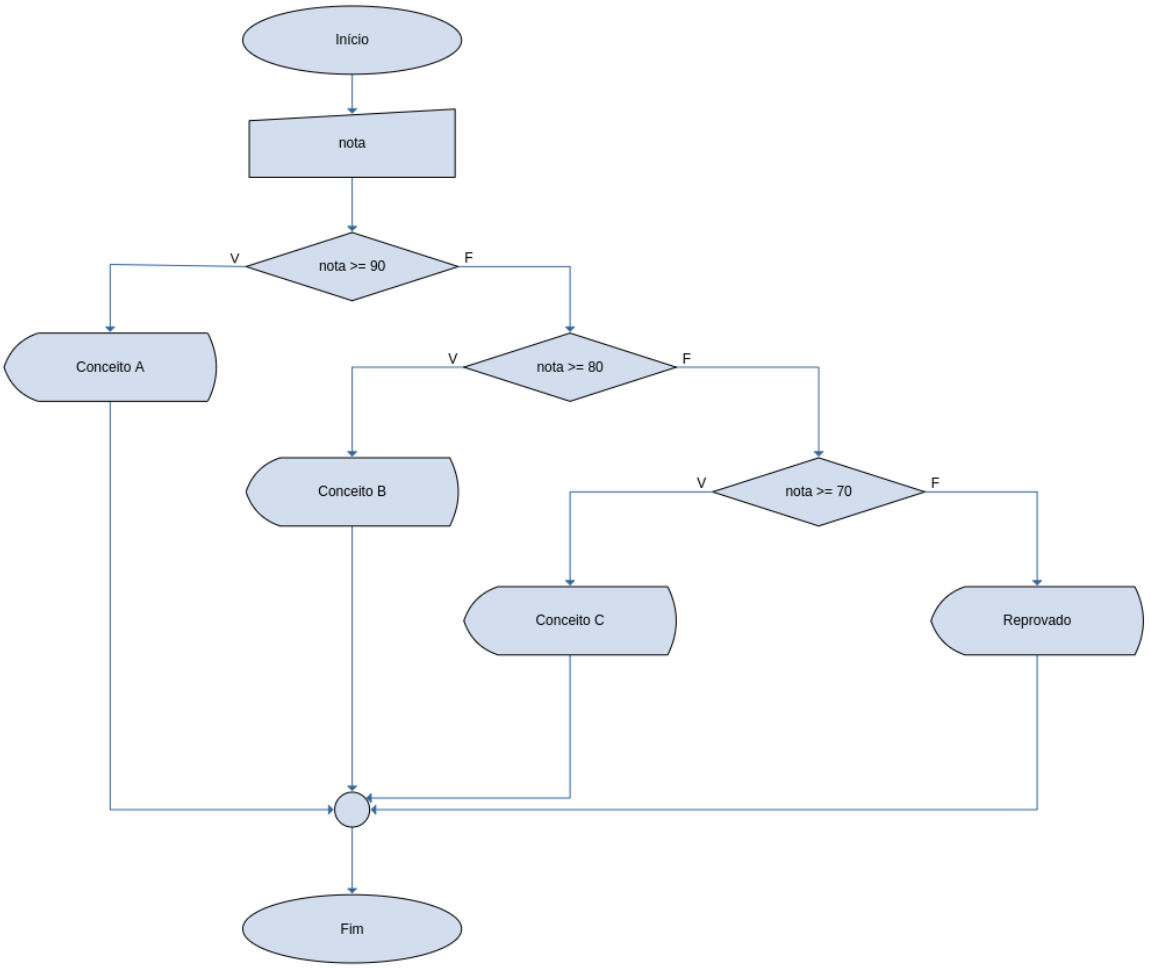
Há também uma variação chamada “**encadeada**”. Veja um exemplo no Bloco de Código 2.1.3.

Bloco de Código 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



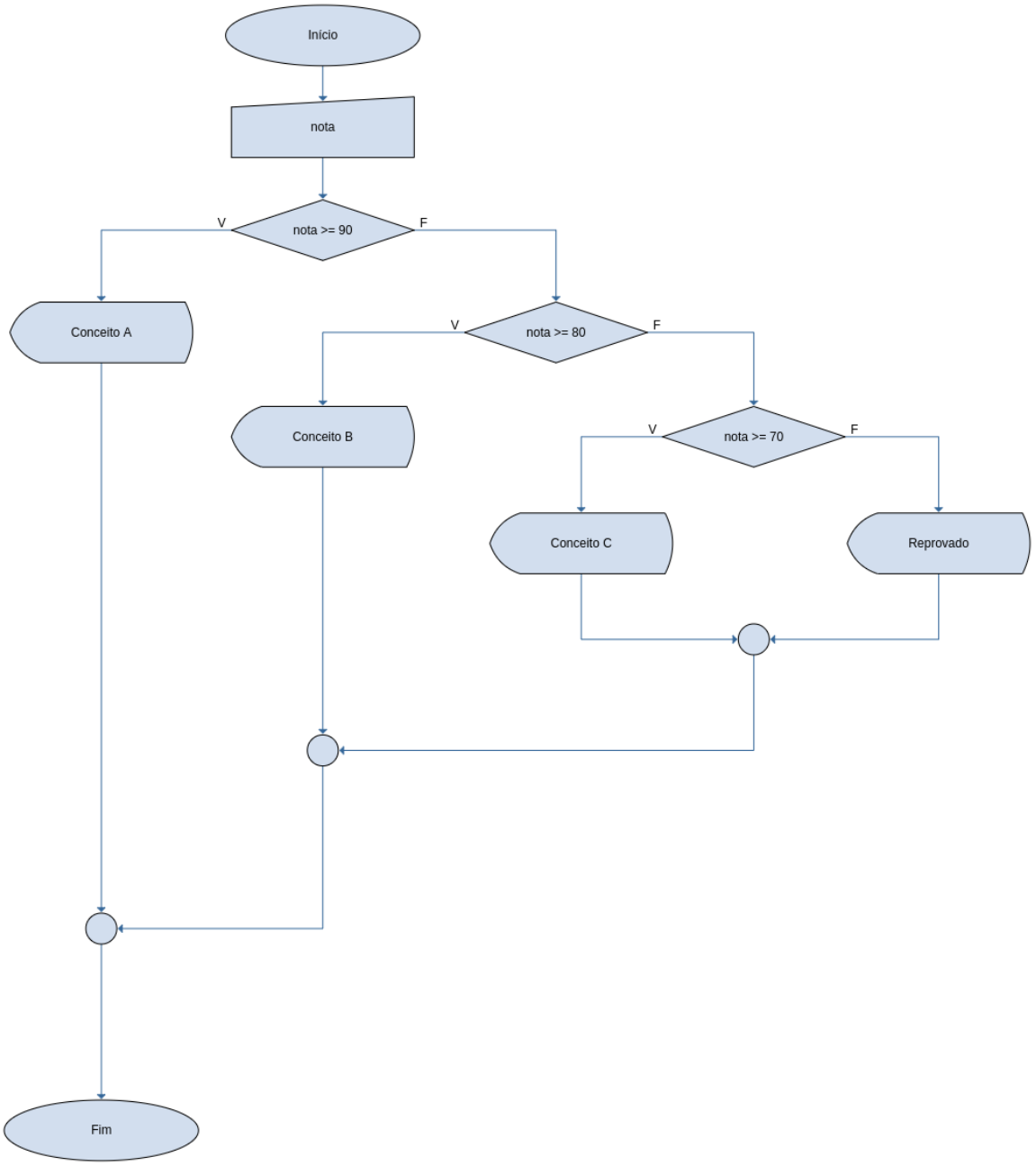
Por fim, vejamos a variação aninhada, ilustrada no Bloco de Código 2.1.4.

Bloco de Código 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. Os blocos de código 2.2.1 e 2.2.1 mostram exemplos de pseudocódigo para essa estrutura.

Bloco de Código 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
```

No Bloco de Código 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 "Avalie" ser encontrado.

Bloco de Código 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 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 única linha. 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 Bloco de Código 3.3.1 mostra a estrutura de seleção If simples em uso.

Bloco de Código 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) O Bloco de Código 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.

Bloco de Código 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 no Bloco de Código 3.3.1.

Bloco de Código 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. No Bloco de Código 3.4.1 ilustramos como isso pode ser feito com blocos if/else.

Bloco de Código 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) No Bloco de Código 4.1.1 o código exibido faz uso de uma estrutura switch/case em que, para cada case, um bloco específico é determinado.

Bloco de Código 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 (fall-through)) Alguns cases podem ter trechos de código em comum. Neste caso, podemos empregar a lógica em queda do switch e escrever menos código, como mostra o Bloco de Código 4.2.1.

Bloco de Código 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. No Bloco de Código 5.1.1 fazemos uso da estrutura if/else já vista. O Bloco de Código 5.1.2 mostra como fazer a mesma coisa usando o operador ternário.

Bloco de Código 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);
   }
}
```

Bloco de Código 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. Ler um número inteiro e exibir se ele é positivo, negativo ou neutro (0).
2. Ler coeficientes reais a, b e c de uma equação de segundo grau e exibir a(s) raiz(es), caso exista(m). Lembretes: Calcule o valor de delta. Se ele for negativo, não há raízes.. Se for igual a zero, há uma única raiz. Se delta for maior do que zero, então há duas raízes.
3. Ler três valores reais e exibir o maior valor entre eles. Suponha que eles sejam diferentes.
4. Ler um inteiro no intervalo [1, 7] e exibir o dia da semana associado a ele, como a seguir: 1: Domingo, 2: Segunda, 3: Terça. E assim por diante.
5. Ler um número inteiro no intervalo [1, 12]. Considerando que cada número representa um mês da seguinte forma: 1: Janeiro, 2: Fevereiro e assim por diante, exiba o número de dias que o mês cujo respectivo número digitado possui.
6. Ler um número inteiro e responder se ele é bissexto ou não. Um ano bissexto tem as seguintes características:

    - é múltiplo de quatro **e** não é múltiplo de 100 **ou**
    - é múltiplo de 400