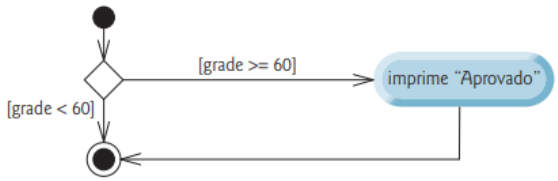
4.5 A instrução de seleção única `if`

Se a nota do aluno for maior que ou igual a 60
Imprime "Aprovado"

```
if (studentGrade >= 60)
   System.out.println("Passed");
```

Diagrama UML de atividades para uma instrução `if`

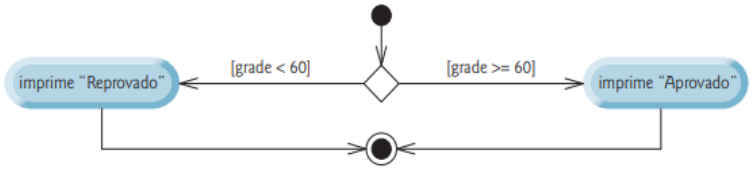


4.6 Instrução de seleção dupla `if...else`

```
Se a nota do aluno for maior que ou igual a 60
             Imprima "Aprovado"
Caso contrário
             Imprima "Reprovado"
```

```
if (grade >= 60)
             System.out.println("Passed");
else
   System.out.println("Failed");
```

Diagrama UML de atividades para uma instrução `if...else`



Instruções `if...else` aninhadas

```
Se a nota do aluno for maior que ou igual a 90
             Imprima "A"
caso contrário
             Se a nota do aluno for maior que ou igual a 80
                          Imprima "B"
             caso contrário
                          Se a nota do aluno for maior que ou igual a 70
                                       Imprima "C"
                          caso contrário
                                Se a nota do aluno for maior que ou igual a 60
                                             Imprima "D"
                                caso contrário
                                             Imprima "F"
```

Esse pseudocódigo pode ser escrito em Java como

```
if (studentGrade >= 90)
   System.out.println("A");
else
   if (studentGrade >= 80)
      System.out.println("B");
   else
      if (studentGrade >= 70)
         System.out.println("C");
      else
         if (studentGrade >= 60)
            System.out.println("D");
         else
            System.out.println("F");
```

```
if (studentGrade >= 90)
   System.out.println("A");
else if (studentGrade >= 80)
   System.out.println("B");
else if (studentGrade >= 70)
   System.out.println("C");
else if (studentGrade >= 60)
   System.out.println("D");
else
   System.out.println("F");
```

Blocos

```
if (x > 5)
   if (y > 5)
      System.out.println("x and y are > 5");
   else
      System.out.println("x is <= 5");
```

```
if (x > 5)
{
   if (y > 5)
      System.out.println("x and y are > 5");
}
else
   System.out.println("x is <= 5");
```

```
if (grade >= 60)
   System.out.println("Passed");
else
{
   System.out.println("Failed");
   System.out.println("You must take this course again.");
}
```

Operador condicional (?:)

```
System.out.println(studentGrade >= 60 ? "Passed" : "Failed");
```

```java
// Figura 4.4: Student.java
// Classe Student que armazena o nome e a média de um aluno.
public class Student
{
   private String name;
   private double average;

   // construtor inicializa variáveis de instância
   public Student(String name, double average)
   {
      this.name = name;

      // valida que a média é > 0.0 e <= 100.0; caso contrário,
      // armazena o valor padrão da média da variável de instância (0.0)
      if (average > 0.0)
         if (average <= 100.0)
            this.average = average; // atribui à variável de instância
   }

   // define o nome de Student
   public void setName(String name)
   {
      this.name = name;
   }

   // recupera o nome de Student
   public String getName()
   {
      return name;
   }

   // define a média de Student
   public void setAverage(double studentAverage)
   {
      // valida que a média é > 0.0 e <= 100.0; caso contrário,
      // armazena o valor atual da média da variável de instância
      if (average > 0.0)
         if (average <= 100.0)
            this.average = average; // atribui à variável de instância
   }

   // recupera a média de Student
   public double getAverage()
   {
      return average;
   }

   // determina e retorna a letra da nota de Student
   public String getLetterGrade()
   {
      String letterGrade = ""; // inicializado como uma String vazia

      if (average >= 90.0)
         letterGrade = "A";
      else if (average >= 80.0)
         letterGrade = "B";
      else if (average >= 70.0)
         letterGrade = "C";
      else if (average >= 60.0)
         letterGrade = "D";
      else
         letterGrade = "F";

      return letterGrade;
   }
} // finaliza a classe Student
```

```java
// Figura 4.5: StudentTest.java
// Cria e testa objetos Student.
public class StudentTest
{
   public static void main(String[] args)
   {
      Student account1 = new Student("Jane Green", 93.5);
      Student account2 = new Student("John Blue", 72.75);

      System.out.printf("%s's letter grade is: %s%n",
         account1.getName(),account1.getLetterGrade());
      System.out.printf("%s's letter grade is: %s%n",
         account2.getName(),account2.getLetterGrade());
   }
} // fim da classe StudentTest
```

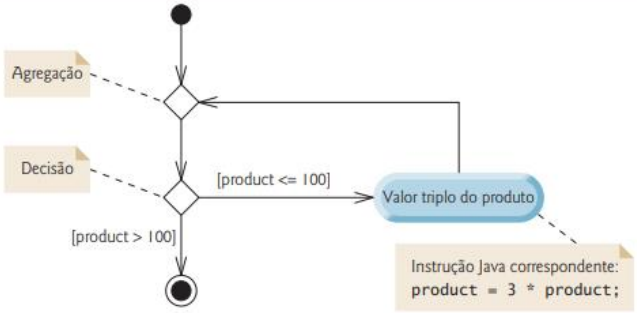
```
Jane Green's letter grade is: A
John Blue's letter grade is: C
```

4.8 Instrução de repetição `while`

Enquanto houver mais itens em minha lista de compras
Comprar o próximo item e riscá-lo da minha lista

```java
while (product <= 100)
   product = 3 * product;
```

Diagrama UML de atividades para a instrução `while`



4.9 Formulando algoritmos: repetição controlada por contador

Uma classe de dez alunos se submeteu a um questionário. As notas (inteiros no intervalo 0–100) para esse questionário estão disponíveis. Determine a média da classe no questionário.

O algoritmo em pseudocódigo com repetição controlada por contador

```
Configure o total como zero
Configure o contador de notas como um

Enquanto contador de notas for menor ou igual a dez
    Solicite para o usuário inserir a próxima nota
    Insira a próxima nota
    Adicione a nota ao total
    Adicione um ao contador de notas

Configure a média da classe como o total dividido por dez
Exibe a média da classe
```

Implementando repetição controlada por contador

```
1  // Figura 4.8: ClassAverage.java
2  // Resolvendo o problema da média da classe usando a repetição controlada por contador.
3  import java.util.Scanner; // programa utiliza a classe Scanner
4
5  public class ClassAverage
6  {
7     public static void main(String[] args)
8     {
9        // cria Scanner para obter entrada a partir da janela de comando
10       Scanner input = new Scanner(System.in);
11
12       // fase de inicialização
13       int total = 0;  // inicializa a soma das notas inseridas pelo usuário
14       int gradeCounter = 1; // inicializa nº da nota a ser inserido em seguida
15
16       // fase de processamento utiliza repetição controlada por contador
17       while (gradeCounter <= 10) // faz o loop 10 vezes
18       {
19          System.out.print("Enter grade: "); // prompt
20          int grade = input.nextInt(); // insere a próxima nota
21          total = total + grade; // adiciona grade a total
22          gradeCounter = gradeCounter + 1; // incrementa o contador por 1
23       }
24
25       // fase de término
26       int average = total / 10; // divisão de inteiros produz um resultado inteiro
27
28       // exibe o total e a média das notas
29       System.out.printf("%nTotal of all 10 grades is %d%n", total);
30       System.out.printf("Class average is %d%n", average);
31    }
32 } // fim da classe ClassAverage
```

```
Enter grade: 67
Enter grade: 78
Enter grade: 89
Enter grade: 67
Enter grade: 87
Enter grade: 98
Enter grade: 93
Enter grade: 85
Enter grade: 82
Enter grade: 100

Total of all 10 grades is 846
Class average is 84
```

4.10 Formulando algoritmos: repetição controlada por sentinela

Desenvolva um programa para tirar a média da classe que processe as notas de acordo com um número arbitrário de alunos toda vez que é executado.

```
Inicialize total como zero
Inicialize counter como zero

Solicite que o usuário insira a primeira nota
Insira a primeira nota (possivelmente o sentinela)

Enquanto o usuário não inserir o sentinela
    Adicione essa nota à soma total
    Adicione um ao contador de notas
    Solicite que o usuário insira a próxima nota
    Insira a próxima nota (possivelmente a sentinela)

Se o contador não for igual a zero
    Configure a média como o total dividido pelo contador
    Imprima a média
Caso contrário
    Imprima "Nenhuma nota foi inserida"
```

Implementando a repetição controlada por sentinela

```java
// Figura 4.10: ClassAverage.java
// Resolvendo o problema da média da classe usando a repetição controlada por sentinela.
import java.util.Scanner; // programa utiliza a classe Scanner

public class ClassAverage
{
   public static void main(String[] args)
   {
      // cria Scanner para obter entrada a partir da janela de comando
      Scanner input = new Scanner(System.in);

      // fase de inicialização
      int total = 0; // incializa a soma das notas
      int gradeCounter = 0; // inicializa o nº de notas inseridas até agora

      // fase de processamento
      // solicita entrada e lê a nota do usuário
      System.out.print("Enter grade or -1 to quit: ");
      int grade = input.nextInt();

      // faz um loop até ler o valor de sentinela inserido pelo usuário
      while (grade != -1)
      {
         total = total + grade; // adiciona grade a total
         gradeCounter = gradeCounter + 1; // incrementa counter

         // solicita entrada e lê a próxima nota fornecida pelo usuário
         System.out.print("Enter grade or -1 to quit: ");
         grade = input.nextInt();
      }

      // fase de término
      // se usuário inseriu pelo menos uma nota...
      if (gradeCounter != 0)
      {
         // usa número com ponto decimal para calcular média das notas
         double average = (double) total / gradeCounter;

         // exibe o total e a média (com dois dígitos de precisão)
         System.out.printf("%nTotal of the %d grades entered is %d%n",
            gradeCounter, total);
         System.out.printf("Class average is %.2f%n", average);
      }
      else // nenhuma nota foi inserida, assim gera a saída da mensagem apropriada
         System.out.println("No grades were entered");
   }
} // fim da classe ClassAverage
```

```
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 72
Enter grade or -1 to quit: -1

Total of the 3 grades entered is 257
Class average is 85.67
```

4.11 Formulando algoritmos: instruções de controle aninhadas

Uma faculdade oferece um curso que prepara os candidatos a obter licença estadual para corretores de imóveis. No ano passado, dez alunos que concluíram esse curso prestaram o exame. A universidade quer saber como foi o desempenho dos seus alunos nesse exame. Você foi contratado para escrever um programa que resuma os resultados. Para tanto, você recebeu uma lista desses 10 alunos. Ao lado de cada nome é escrito 1 se o aluno passou no exame ou 2 se o aluno foi reprovado.

Seu programa deve analisar os resultados do exame assim:

1. *Dê entrada a cada resultado do teste (isto é, um 1 ou um 2). Exiba a mensagem "Inserir resultado" na tela toda vez que o programa solicitar o resultado de outro teste.*
2. *Conte o número de cada tipo de resultado.*
3. *Exiba um resumo dos resultados do teste indicando o número de alunos aprovados e reprovados.*
4. *Se mais de oito estudantes forem aprovados no exame, imprima "Bonus to instructor!".*

Segundo refinamento completo do pseudocódigo e conversão para a classe `Analysis`

```
Inicialize as aprovações como zero
Inicialize as reprovações como zero
Inicialize o contador de alunos como um

Enquanto o contador de alunos for menor ou igual a 10
    Solicite que o usuário insira o próximo resultado de exame
    Insira o próximo resultado de exame

    Se o aluno foi aprovado
        Adicione um a aprovações
    Caso contrário
        Adicione um a reprovações

    Adicione um ao contador de aluno

Imprima o número de aprovações
Imprima o número de reprovações

Se mais de oito alunos forem aprovados
    Imprima "Bonus to instructor!"
```

```
// Figura 4.12: Analysis.java
// Análise dos resultados do exame utilizando instruções de controle aninhadas.
import java.util.Scanner; // classe utiliza a classe Scanner

public class Analysis
{
   public static void main(String[] args)
   {
      // cria Scanner para obter entrada a partir da janela de comando
      Scanner input = new Scanner(System.in);

      // inicializando variáveis nas declarações
      int passes = 0;
      int failures = 0;
      int studentCounter = 1;

      // processa 10 alunos utilizando o loop controlado por contador
      while (studentCounter <= 10)
      {
         // solicita ao usuário uma entrada e obtém valor fornecido pelo usuário
         System.out.print("Enter result (1 = pass, 2 = fail): ");
         int result = input.nextInt();

         // if...else está aninhado na instrução while
         if (result == 1)
            passes = passes + 1;
         else
            failures = failures + 1;

         // incrementa studentCounter até o loop terminar
         studentCounter = studentCounter + 1;
      }
```

```
33
34          // fase de término; prepara e exibe os resultados
35          System.out.printf("Passed: %d%nFailed: %d%n", passes, failures);
36
37          // determina se mais de 8 alunos foram aprovados
38          if (passes > 8)
39             System.out.println("Bonus to instructor!");
40       }
41    } // fim da classe Analysis
```

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed: 9
Failed: 1
Bonus to instructor!
```

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed: 6
Failed: 4
```

4.12 Operadores de atribuição compostos

Operador de atribuição	Expressão de exemplo	Explicação	Atribuições
Suponha: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	10 a `c`
`-=`	`d -= 4`	`d = d - 4`	1 a `d`
`*=`	`e *= 5`	`e = e * 5`	20 a `e`
`/=`	`f /= 3`	`f = f / 3`	2 a `f`
`%=`	`g %= 9`	`g = g % 9`	3 a `g`

4.13 Operadores de incremento e decremento

Operador	Nome do operador	Exemplo	Explicação
`++`	pré-incremento	`++a`	Incrementa `a` por 1, então utiliza o novo valor de `a` na expressão em que `a` reside.
`++`	pós-incremento	`a++`	Usa o valor atual de `a` na expressão em que `a` reside, então incrementa `a` por 1.
`--`	pré-decremento	`--b`	Decrementa `b` por 1, então utiliza o novo valor de `b` na expressão em que `b` reside.
`--`	pós-decremento	`b--`	Usa o valor atual de `b` na expressão em que `b` reside, então decrementa `b` por 1.

Diferença entre operadores de pré-incremento e operadores de pós-incremento

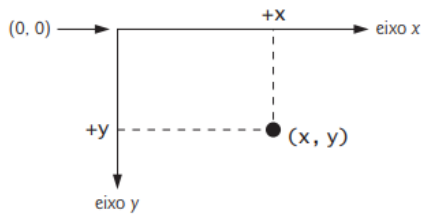
```
1  // Figura 4.15: Increment.java
2  // Operadores de pré-incremento e de pós-incremento.
3
4  public class Increment
5  {
6     public static void main(String[] args)
7     {
8        // demonstra o operador de pós-incremento
9        int c = 5;
10       System.out.printf("c before postincrement: %d%n", c); // imprime 5
11       System.out.printf("    postincrementing c: %d%n", c++); // imprime 5
12       System.out.printf(" c after postincrement: %d%n", c); // imprime 6
13
14       System.out.println(); // pula uma linha
15
16       // demonstra o operador de pré-incremento
17       c = 5;
18       System.out.printf(" c before preincrement: %d%n", c); // imprime 5
19       System.out.printf("     preincrementing c: %d%n", ++c); // imprime 6
20       System.out.printf("  c after preincrement: %d%n", c); // imprime 6
21    }
22 } // fim da classe Increment
```

```
c before postincrement: 5
   postincrementing c: 5
c after postincrement: 6

c before preincrement: 5
   preincrementing c: 6
c after preincrement: 6
```

4.15 (Opcional) Estudo de caso de GUIs e imagens gráficas: criando desenhos simples

Sistema de coordenadas do Java




```
1  // Figura 4.18: DrawPanel.java
2  // Utilizando DrawLine para conectar os cantos de um painel.
3  import java.awt.Graphics;
4  import javax.swing.JPanel;
5
6  public class DrawPanel extends JPanel
7  {
8     // desenha um X a partir dos cantos do painel
9     public void paintComponent(Graphics g)
10    {
11       // chama paintComponent para assegurar que o painel é exibido corretamente
12       super.paintComponent(g);
13
14       int width = getWidth(); // largura total
15       int height = getHeight(); // altura total
16
17       // desenha uma linha a partir do canto superior esquerdo até o inferior direito
18       g.drawLine(0, 0, width, height);
19
20       // desenha uma linha a partir do canto inferior esquerdo até o superior direito
21       g.drawLine(0, height, width, 0);
22    }
23 } // fim da classe DrawPanel
```

```java
// Figura 4.19: DrawPanelTest.java
// Criando JFrame para exibir um DrawPanel.
import javax.swing.JFrame;

public class DrawPanelTest
{
   public static void main(String[] args)
   {
      // cria um painel que contém nosso desenho
      DrawPanel panel = new DrawPanel();

      // cria um novo quadro para armazenar o painel
      JFrame application = new JFrame();

      // configura o frame para ser encerrado quando ele é fechado
      application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      application.add(panel); // adiciona o painel ao frame
      application.setSize(250, 250); // configura o tamanho do frame
      application.setVisible(true); // torna o frame visível
   }
} // fim da classe DrawPanelTest
```