e tomar decisões com base em um conjunto de resultados de um exame. Apresentamos os operadores de atribuição compostos do Java e seus operadores de incremento e decremento. Por fim, discutimos os tipos primitivos do Java. No Capítulo 5, continuaremos nossa discussão sobre as instruções de controle, introduzindo as instruções `for`, `do...while` e `switch`.

Resumo

Seção 4.1 Introdução

- Antes de escrever um programa para resolver um problema, você deve ter um entendimento completo do problema e uma abordagem cuidadosamente planejada para resolvê-lo. Você também deve entender os blocos de construção que estão disponíveis e empregar técnicas de construção do programa comprovadas.

Seção 4.2 Algoritmos

- Qualquer problema de computação pode ser resolvido executando uma série de ações em uma ordem específica.
- O procedimento para resolver um problema em termos das ações a serem executadas e da ordem em que são executadas é chamado algoritmo.
- Especificar a ordem em que as instruções são executadas em um programa chama-se controle de programa.

Seção 4.3 Pseudocódigo

- Pseudocódigo é uma linguagem informal que ajuda a desenvolver algoritmos sem se preocupar com os estritos detalhes da sintaxe da linguagem Java.
- O pseudocódigo que usamos neste livro é simples — ele é conveniente e fácil de usar, mas não é uma linguagem de programação de computador real. Naturalmente, você pode usar seu próprio idioma nativo para desenvolver seu pseudocódigo.
- O pseudocódigo ajuda você a "estudar" um programa antes de tentar escrevê-lo em uma linguagem de programação como Java.
- O pseudocódigo cuidadosamente preparado pode ser facilmente convertido em um programa Java correspondente.

Seção 4.4 Estruturas de controle

- Normalmente, instruções em um programa são executadas uma após a outra na ordem em que são escritas. Esse processo é chamado execução sequencial.
- Várias instruções Java permitem a você especificar que a próxima instrução a executar não seja necessariamente a próxima na sequência. Isso é chamado transferência de controle.
- Bohm e Jacopini demonstraram que todos os programas poderiam ser escritos em termos de somente três tipos de estruturas de controle — a estrutura de sequência, a estrutura de seleção e a estrutura de repetição.
- O termo "estruturas de controle" vem do campo das ciências da computação. A *Java Language Specification* refere-se a "estruturas de controle" como "instruções de controle".
- A estrutura de sequência está incorporada ao Java. A não ser que seja instruído de outra forma, o computador executa as instruções Java uma depois da outra na ordem em que elas são escritas — isto é, em sequência.
- Em qualquer lugar que uma ação única pode ser colocada, várias ações podem ser colocadas em sequência.
- Diagramas de atividades fazem parte da UML. Um diagrama de atividades modela o fluxo de trabalho (também chamado atividade) de uma parte de um sistema de software.
- Os diagramas de atividades são compostos de símbolos — como símbolos de estado de ação, losangos e pequenos círculos — que são conectados por setas de transição, as quais representam o fluxo da atividade.
- Os estados de ação contêm expressões de ação que especificam determinadas ações a ser realizadas.
- As setas em um diagrama de atividades representam transições, que indicam a ordem em que ocorrem as ações representadas pelos estados da ação.
- O círculo sólido localizado na parte superior de um diagrama de atividade representa o estado inicial da atividade — o começo do fluxo de trabalho antes de o programa realizar as ações modeladas.
- O círculo sólido cercado por um círculo vazio que aparece na parte inferior do diagrama representa o estado final — o fim do fluxo de trabalho depois que o programa realiza suas ações.
- Os retângulos com o canto superior direito dobrados são notas da UML — observações explicativas que descrevem o objetivo dos símbolos no diagrama.
- O Java tem três tipos de instruções de seleção.
- A instrução de seleção única `if` escolhe ou ignora uma ou mais ações.
- A instrução de seleção dupla `if...else` seleciona entre duas ações ou grupos de ações.
- A instrução `switch` é chamada de instrução de seleção múltipla, uma vez que seleciona entre muitas ações diferentes (ou grupos de ações).

- O Java fornece as instruções de repetição `while`, `do...while` e `for` (também chamada iteração ou loop), que permitem que programas executem instruções repetidamente enquanto uma condição de continuação de loop permanece verdadeira.
- As instruções `while` e `for` realizam a(s) ação(ões) no seu zero de corpos ou mais horas — se a condição de continuação do loop for inicialmente falsa, a(s) ação(ões) não serão executadas. A instrução `do...while` realiza a(s) ação(ões) no seu corpo uma ou várias horas.
- As palavras `if`, `else`, `switch`, `while`, `do` e `for` são palavras-chave Java. As palavras-chave não podem ser utilizadas como identificadores, por exemplo, nos nomes de variáveis.
- Cada programa é formado combinando o número de instruções de sequência, seleção e de repetição, conforme apropriado no algoritmo que o programa implementa.
- As instruções de controle de entrada única/saída única são anexadas uma a outra conectando o ponto de saída de uma instrução ao ponto de entrada da instrução seguinte. Isso é conhecido como empilhamento de instruções de controle.
- Uma instrução de controle também pode ser aninhada dentro de outra instrução de controle.

Seção 4.5 A instrução de seleção única `if`

- Os programas utilizam instruções de seleção para escolher entre cursos alternativos de ações.
- O diagrama de atividade da instrução `if` de seleção única contém o símbolo de losango, que indica que uma decisão deve ser tomada. O fluxo de trabalho segue um caminho determinado pelas condições de guarda associadas do símbolo. Se uma condição de guarda for verdadeira, o fluxo de trabalho entra no estado de ação para o qual a seta de transição correspondente aponta.
- A instrução `if` é uma instrução de controle de uma única entrada e uma única saída.

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

- A instrução `if` de seleção única só realiza uma ação indicada quando a condição for `true`.
- A instrução de seleção dupla `if...else` executa uma ação quando a condição é verdadeira e outra ação quando a condição é falsa.
- Um programa pode testar múltiplos casos com instruções `if...else` aninhadas.
- O operador condicional (`?:`) é único operador ternário do Java — ele leva três operandos. Juntos, os operandos e o símbolo `?:` formam uma expressão condicional.
- O compilador Java sempre associa um `else` com o `if` imediatamente precedente, a menos que seja instruído a fazer de outro modo pela colocação de chaves (`{}`).
- O enunciado `if` espera uma instrução no seu corpo. Para incluir várias instruções no corpo de uma `if` (ou no corpo de um `else` de uma instrução `if...else`), inclua as instruções dentro de chaves (`{` e `}`).
- Um bloco de instruções pode ser colocado em qualquer lugar em que uma instrução única pode ser inserida.
- O efeito de um erro de lógica ocorre em tempo de execução. Um erro fatal de lógica faz com que um programa falhe e finalize prematuramente. Um erro de lógica não fatal permite a um programa continuar a executar, mas faz com que produza resultados incorretos.
- Assim como um bloco pode ser colocado em qualquer lugar em que uma instrução única pode ser colocada, você também pode utilizar uma instrução vazia, representada colocando-se um ponto e vírgula (`;`) onde normalmente entraria uma instrução.

Seção 4.8 Instrução de repetição `while`

- A instrução de repetição `while` permite especificar que um programa deve repetir uma ação enquanto alguma condição permanecer verdadeira.
- O símbolo de agregação da UML une dois fluxos de atividade em um único.
- Os símbolos de decisão e agregação podem ser distinguidos pelo número de setas de transição que entram e saem. Um símbolo de decisão contém uma seta de transição apontando para o losango e duas ou mais setas de transição apontando a partir do losango para indicar possíveis transições a partir desse ponto. Cada seta de transição que sai de um símbolo de decisão tem uma condição de guarda. Um símbolo de agregação contém duas ou mais setas de transição apontando para o losango e somente uma seta de transição apontando a partir do losango, para indicar a conexão de múltiplos fluxos de atividades a fim de continuar a atividade. *Nenhuma* das setas de transição associadas com um símbolo de agregação contém uma condição de guarda.

Seção 4.9 Formulando algoritmos: repetição controlada por contador

- A repetição controlada por contador utiliza uma variável chamada contador (ou variável de controle) para controlar o número de vezes que um conjunto de instruções é executado.
- A repetição controlada por contador costuma ser chamada de repetição definida, porque o número de repetições é conhecido antes de o loop começar a executar.
- Um total é uma variável utilizada para acumular a soma de vários valores. Variáveis utilizadas para armazenar totais normalmente são inicializadas como zero antes de serem utilizadas em um programa.
- A declaração de uma variável local deve aparecer antes de a variável ser utilizada nesse método. Uma variável local não pode ser acessada fora do método em que é declarada.
- Quando a divisão de um inteiro por outro resulta em um número fracionário, a parte fracionária do cálculo é truncada.

Seção 4.10 Formulando algoritmos: repetição controlada por sentinela

- Na repetição controlada por sentinela, um valor especial chamado de valor de sentinela (também chamado de valor de sinal, valor fictício ou valor de flag) é utilizado para indicar o "fim da entrada de dados".
- Deve-se escolher um valor de sentinela que não possa ser confundido com um valor aceitável de entrada.
- O refinamento passo a passo de cima para baixo é essencial para o desenvolvimento de programas bem estruturados.
- A divisão por zero é um erro de lógica.
- Para realizar um cálculo de ponto flutuante com valores inteiros, faça a coerção (conversão) de um dos números inteiros para o tipo `double`.
- O Java sabe como avaliar somente expressões aritméticas nas quais os tipos dos operandos são idênticos. Para assegurar isso, o Java realiza uma operação chamada de promoção em operandos selecionados.
- O operador de coerção unário é formado colocando-se parênteses em torno do nome de um tipo.

Seção 4.12 Operadores de atribuição compostos

- Os operadores de atribuição compostos abreviam expressões de atribuição. Instruções da forma

  variável = *variável* *operador* *expressão*;

em que *operador* é um dos operadores binários +, - *, / ou %, podem ser escritas na forma

  variável *operador*= *expressão*;

- O operador += adiciona o valor da expressão à direita do operador ao valor da variável à esquerda do operador e armazena o resultado na variável à esquerda do operador.

Seção 4.13 Operadores de incremento e decremento

- O operador de incremento unário, ++, e operador de decremento unário, --, adicionam 1 ao, ou subtraem 1 do, valor de uma variável numérica.
- O operador de decremento ou incremento que é prefixado a uma variável é o operador de incremento de prefixo ou decremento de prefixo, respectivamente. O operador de incremento ou decremento que é pós-fixado a uma variável é o operador de incremento pós-fixo ou decremento pós-fixo, respectivamente.
- Utilizar o operador de pré-incremento ou pré-decremento para adicionar ou subtrair 1 é conhecido como pré-incrementar ou pré-decrementar, respectivamente.
- Pré-incrementar (ou pré-decrementar) uma variável faz com que a variável seja incrementada ou decrementada por 1; então o novo valor da variável é utilizado na expressão em que ela aparece.
- Utilizar o operador de incremento ou decremento pós-fixo para adicionar ou subtrair 1 é conhecido como pós-incrementar ou pós-decrementar, respectivamente.
- Pós-incrementar ou pós-decrementar a variável faz com que seu valor seja utilizado na expressão em que ele aparece; então o valor da variável é incrementado ou decrementado por 1.
- Ao incrementar ou decrementar uma variável em uma instrução isolada, o pré-incremento ou o pós-incremento têm o mesmo efeito, assim como o pré-decremento ou o pós-decremento.

Seção 4.14 Tipos primitivos

- O Java requer que todas as variáveis tenham um tipo. Assim, o Java é conhecido como uma linguagem fortemente tipada.
- O Java utiliza caracteres de Unicode e números de ponto flutuante IEEE 754.

Exercícios de revisão

4.1 Preencha as lacunas em cada uma das seguintes afirmações:

a) Todos os programas podem ser escritos em termos de três tipos de estruturas de controle: _____, _____ e _____.

b) A instrução _____ é utilizada para executar uma ação quando uma condição for verdadeira e outra quando essa condição for falsa.

c) Repetir um conjunto de instruções por um número específico de vezes é chamado de repetição _____.

d) Quando não se sabe antecipadamente quantas vezes um conjunto de instruções será repetido, um valor de _____ pode ser utilizado para terminar a repetição.

e) A estrutura de _____ é construída em Java; por padrão, instruções são executadas na ordem que elas aparecem.

f) As variáveis de instância dos tipos `char`, `byte`, `short`, `int`, `long`, `float` e `double` recebem o valor _____ por padrão.

g) O Java é uma linguagem _____; ele requer que todas as variáveis tenham um tipo.

h) Se o operador de incremento for _____ para uma variável, primeiro a variável é incrementada por 1 e, então, seu novo valor é utilizado na expressão.

4.2 Determine se cada um dos seguintes é *verdadeiro* ou *falso*. Se *falso*, explique por quê.

a) Um algoritmo é um procedimento para resolver um problema em termos das ações a serem executadas e da ordem em que essas ações são executadas.

b) Um conjunto de instruções dentro de um par de parênteses é chamado bloco.

c) Uma instrução de seleção especifica que uma ação deve ser repetida enquanto algumas condições permanecem verdadeiras.

d) Uma instrução de controle aninhada aparece no corpo de uma outra instrução de controle.

e) O Java fornece os operadores aritméticos de atribuição compostos `+=`, `-=`, `*=`, `/=` e `%=` para abreviar expressões de atribuição.

f) Os tipos primitivos (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` e `double`) são portáveis somente em plataformas Windows.

g) Especificar a ordem em que as instruções são executadas em um programa é chamado controle de programa.

h) O operador de coerção unário `(double)` cria uma cópia temporária do tipo inteiro do seu operando.

i) Atribui-se às variáveis de instância do tipo `boolean` o valor `true` por padrão.

j) O pseudocódigo ajuda você a pensar sobre um programa antes de tentar escrevê-lo em uma linguagem de programação.

4.3 Escreva quatro instruções Java diferentes que adicionam 1 à variável de inteiro `x`.

4.4 Escreva instruções Java para realizar cada uma das seguintes tarefas:

a) Utilize uma instrução para atribuir a soma de `x` e `y` a `z`, em seguida, incremente `x` por 1.

b) Teste se a variável `contador` é maior do que 10. Se for, imprima `"Contador é maior que 10"`.

c) Utilize uma instrução para decrementar a variável `x` por 1, então subtraia-o da variável `total` e armazene o resultado na variável `total`.

d) Calcule o resto após `q` ser dividido por `divisor` e atribua o resultado a `q`. Escreva essa instrução de duas maneiras diferentes.

4.5 Escreva uma instrução Java para realizar cada uma das seguintes tarefas:

a) Declarar variáveis `sum` do tipo `int` e inicialize-as como `0`.

b) Declarar variáveis `x` do tipo `int` e inicialize-as como `0`.

c) Adicione a variável `x` à variável `sum`, e atribua o resultado à variável `sum`.

d) Imprima `"A soma é: "` seguido pelo valor da variável `sum`.

4.6 Combine as instruções escritas no Exercício 4.5 em um aplicativo Java que calcula e imprime a soma dos inteiros de 1 a 10. Utilize a instrução `while` para fazer loop pelas instruções de cálculo e incremento. O loop deve terminar quando o valor de `x` tornar-se 11.

4.7 Determine o valor das variáveis na instrução `product *= x++;` depois que o cálculo é realizado. Suponha que todas as variáveis sejam do tipo `int` e inicialmente tenham o valor 5.

4.8 Identifique e corrija os erros em cada um dos seguintes conjuntos de código:

a)
```
while (c <= 5)
{
   product *= c;
   ++c;
```

b)
```
if (gender == 1)
   System.out.println("Woman");
else;
   System.out.println("Man");
```

4.9 O que há de errado com a instrução `while` a seguir?

```
while (z >= 0)
   sum += z;
```

Respostas dos exercícios de revisão

4.1 a) sequência, seleção, repetição. b) `if...else`. c) controlada por contador (ou definida). d) de sentinela, de sinal, de flag ou fictício. e) sequência. f) `0` (zero). g) fortemente tipada. h) prefixado.

4.2 a) Verdadeiro. b) Falso. Um conjunto de instruções dentro de um par de chaves (`{` e `}`) é chamado bloco. c) Falso. Uma instrução de repetição especifica que uma ação deve ser repetida enquanto alguma condição permaneça verdadeira. d) Correto. e) Verdadeiro. f) Falso. Os tipos primitivos (`boolean`, `char`, `byte`, `short`, `int`, `long`, `float` e `double`) são portáveis em todas as plataformas de computador que suportam o Java. g) Verdadeiro. h) Falso. O operador de coerção unário `(double)` cria uma cópia temporária de ponto flutuante do seu operando. i) Falso. Variáveis de instância do tipo `boolean` recebem o valor `false` por padrão. j) Verdadeiro.

4.3
```
x = x + 1;
x += 1;
++x;
x++;
```

4.4 a)
```
z = x++ + y;
```

b)
```
if (count > 10)
   System.out.println("Count is greater than 10");
```

c)
```
total -= --x;
```

d)
```
q %= divisor;
q = q % divisor;
```

4.5 a)
```
int sum = 0;
```

b)
```
int x = 1;
```

c)
```
sum += x;
```
ou
```
sum = sum + x;
```

d)
```
System.out.printf("The sum is: %d%n", sum);
```

4.6 O programa é o seguinte:

```
1  // Exercício 4.6: Calculate.java
2  // Calcula a soma dos inteiros de 1 a 10
3  public class Calculate
4  {
5     public static void main(String[] args)
6     {
7        int sum = 0;
8        int x = 1;
9
10       while (x <= 10) // enquanto x é menor ou igual a 10
11       {
12          sum += x; // adiciona x a soma
13          ++x; // incrementa x
14       }
15
16       System.out.printf("The sum is: %d%n", sum);
17    }
18 } // fim da classe Calculate
```

```
The sum is: 55
```

4.7 `product` = 25, `x` = 6

4.8 a) Erro: está faltando a chave direita de fechamento do corpo da instrução `while`.
Correção: adicionar uma chave direita de fechamento depois da instrução `++c;`.

b) Erro: o ponto e vírgula depois de `else` resulta em um erro de lógica. A segunda instrução de saída sempre será executada.
Correção: remover ponto e vírgula depois de `else`.

4.9 O valor da variável `z` nunca é alterado na instrução `while`. Portanto, se a condição de continuação do loop (`z >= 0`) for verdadeira, um loop infinito é criado. Para evitar que um loop infinito ocorra, `z` deve ser decrementado de modo que acabe se tornando menor que 0.

Questões

4.10 Compare e contraste a instrução de seleção única `if` e a instrução de repetição `while`. Qual é a semelhança dessas duas instruções? Qual é a diferença?

4.11 Explique o que acontece quando um programa Java tenta dividir um inteiro por outro. O que acontece para a parte fracionária do cálculo? Como você pode evitar esse resultado?

4.12 Descreva as duas maneiras como as instruções de controle podem ser combinadas.

4.13 Que tipo de repetição seria apropriado para calcular a soma dos primeiros 100 inteiros positivos? Que tipo seria apropriado para calcular a soma de um número arbitrário de inteiros positivos? Descreva brevemente como cada uma dessas tarefas poderia ser realizada.

4.14 Qual é a diferença entre pré-incrementar e pós-incrementar uma variável?

4.15 Identifique e corrija os erros em cada um dos seguintes fragmentos de código. [*Observação:* pode haver mais de um erro em cada trecho de código.]

a)
```
if (age >= 65);
   System.out.println("Age is greater than or equal to 65");
else
   System.out.println("Age is less than 65)";
```

b)
```
int x = 1, total;
while (x <= 10)
{
   total += x;
   ++x;
}
```
c)
```
while (x <= 100)
   total += x;
   ++x;
```
d)
```
while (y > 0)
{
   System.out.println(y);
   ++y;
```

4.16 O que o seguinte programa imprime?

```
1  // Exercício 4.16: Mystery.java
2  public class Mystery
3  {
4  public static void main(String[] args)
5  {
6     int x = 1;
7     int total = 0;
8
9     while (x <= 10)
10    {
11       int y = x * x;
12       System.out.println(y);
13       total += y;
14       ++x;
15    }
16
17    System.out.printf("Total is %d%n", total);
18 }
19 } // fim da classe Mystery
```

Para os exercícios de 4.17 a 4.20, execute cada um dos seguintes passos:

a) Leia a declaração do problema.

b) Formule o algoritmo utilizando pseudocódigo e refinamento passo a passo de cima para baixo (top-down stepwise).

c) Escreva um programa Java.

d) Teste, depure e execute o programa Java.

e) Processe três conjuntos completos de dados.

4.17 (***Quilometragem de combustível***) Os motoristas se preocupam com a quilometragem obtida por seus automóveis. Um motorista monitorou várias viagens registrando a quilometragem dirigida e a quantidade de combustível em litros utilizados para cada tanque cheio. Desenvolva um aplicativo Java que receba como entrada os quilômetros dirigidos e os litros de gasolina consumidos (ambos como inteiros) para cada viagem. O programa deve calcular e exibir o consumo em quilômetros/litro para cada viagem e imprimir a quilometragem total e a soma total de litros de combustível consumidos até esse ponto para todas as viagens. Todos os cálculos de média devem produzir resultados de ponto flutuante. Utilize classe `Scanner` e repetição controlada por sentinela para obter os dados do usuário.

4.18 (***Calculador de limite de crédito***) Desenvolva um aplicativo Java que determina se um cliente de uma loja de departamentos excedeu o limite de crédito em uma conta-corrente. Para cada cliente, os seguintes dados estão disponíveis:

a) Número de conta.

b) Saldo no início do mês.

c) Total de todos os itens cobrados desse cliente no mês.

d) Total de créditos aplicados ao cliente no mês.

e) Limite de crédito autorizado.

O programa deve inserir todos esses dados como inteiros, calcular o novo saldo (= *saldo inicial* + *despesas* – *créditos*), exibir o novo saldo e determinar se o novo saldo excede ao limite de crédito do cliente. Para aqueles clientes cujo limite de crédito foi excedido, o programa deve exibir a mensagem `"Limite de crédito excedido"`.

4.19 (***Calculador de comissão de vendas***) Uma grande empresa paga seu pessoal de vendas com base em comissões. O pessoal de vendas recebe R$ 200 por semana mais 9% de suas vendas brutas durante esse período. Por exemplo, um vendedor que realiza um total de vendas

de mercadorias de R$ 5.000 em uma semana recebe R$ 200 mais 9% de R$ 5.000, um total de R$ 650. Foi-lhe fornecida uma lista dos itens vendidos por cada vendedor. Os valores desses itens são como segue:

```
Item    Value
1       239.99
2       129.75
3        99.95
4       350.89
```

Desenvolva um aplicativo Java que recebe entrada de itens vendidos por um vendedor durante a última semana e calcula e exibe os rendimentos do vendedor. Não existe nenhum limite para o número de itens que pode ser vendido.

4.20 **(*Calculador de salários*)** Desenvolva um aplicativo Java que determina o salário bruto de cada um de três empregados. A empresa paga as horas normais pelas primeiras 40 horas trabalhadas por cada funcionário e 50% a mais por todas as horas trabalhadas além das 40 horas. Você recebe uma lista de empregados, o número de horas trabalhadas por eles na semana passada e o salário-hora de cada um. Seu programa deve aceitar a entrada dessas informações para cada empregado e, então, determinar e exibir o salário bruto do empregado. Utilize a classe `Scanner` para inserir os dados.

4.21 **(*Localize o maior número*)** O processo de localizar o maior valor é muito utilizado em aplicativos de computador. Por exemplo, um programa que determina o vencedor de uma competição de vendas inseriria o número de unidades vendidas por cada vendedor. O vendedor que vende mais unidades ganha a competição. Escreva um programa em pseudocódigo e, então, um aplicativo Java que aceita como entrada uma série de 10 inteiros e determina e imprime o maior dos inteiros. Seu programa deve utilizar pelo menos as três variáveis a seguir:

a) `counter`: um contador para contar até 10 (isto é, monitorar quantos números foram inseridos e determinar quando todos os 10 números foram processados).

b) `number`: o inteiro mais recentemente inserido pelo usuário.

c) `largest`: o maior número encontrado até agora.

4.22 **(*Saída no formato de tabela*)** Escreva um aplicativo Java que utiliza um loop para imprimir a seguinte tabela de valores:

```
N      10*N    100*N   1000*N

1      10      100     1000
2      20      200     2000
3      30      300     3000
4      40      400     4000
5      50      500     5000
```

4.23 **(*Encontre os dois números maiores*)** Utilizando uma abordagem semelhante àquela do Exercício 4.21, encontre os *dois* maiores valores entre os 10 valores inseridos. [*Observação:* você só pode inserir cada número uma vez.]

4.24 **(*Validando entrada de usuário*)** Modifique o programa na Figura 4.12 para validar suas entradas. Para qualquer entrada, se o valor entrado for diferente de 1 ou 2, continue o loop até o usuário inserir um valor correto.

4.25 O que o seguinte programa imprime?

```
1  // Exercício 4.25: Mystery2.java
2  public class Mystery2
3  {
4  public static void main(String[] args)
5  {
6    int count = 1;
7
8    while (count <= 10)
9    {
10     System.out.println(count % 2 == 1 ? "****" : "++++++++");
11     ++count;
12   }
13 }
14 } // fim da classe Mystery2
```

4.26 O que o seguinte programa imprime?

```
1  // Exercício 4.26: Mystery3.java
2  public class Mystery3
3  {
4  public static void main(String[] args)
5  {
```

continua

continuação

```
6     int row = 10;
7
8     while (row >= 1)
9     {
10       int column = 1;
11
12       while (column <= 10)
13       {
14          System.out.print(row % 2 == 1 ? "<" : ">");
15          ++column;
16       }
17
18       --row;
19       System.out.println();
20    }
21 }
22 } // fim da classe Mystery3
```

4.27 (***Problema do `else` oscilante***) Determine a saída para cada um dos conjuntos dados de código quando `x` é `9` e `y` é `11` e quando `x` é `11` e `y` é `9`. O compilador ignora o recuo em um programa Java. Da mesma forma, o compilador Java sempre associa um `else` com o `if` imediatamente precedente a menos que instruído a fazer de outro modo pela colocação de chaves (`{}`). À primeira vista, o programador pode não ter certeza de qual `if` um `else` particular corresponde — essa situação é conhecida como "problema do `else` oscilante". Eliminamos o recuo do seguinte código para tornar o problema mais desafiador. [*Dica:* aplique as convenções de recuo que você aprendeu.]

a)
```
if (x < 10)
if (y > 10)
System.out.println("*****");
else
System.out.println("#####");
System.out.println("$$$$$");
```

b)
```
if (x < 10)
{
if (y > 10)
System.out.println("*****");
}
else
{
System.out.println("#####");
System.out.println("$$$$$");
}
```

4.28 (***Outro problema do `else` oscilante***) Modifique o código dado para produzir a saída mostrada em cada parte do problema. Utilize técnicas de recuo adequadas. Não faça nenhuma alteração além de inserir chaves e alterar o recuo do código. O compilador ignora recuo em um programa Java. Eliminamos o recuo do código fornecido para tornar o problema mais desafiador. [*Observação:* é possível que não seja necessária nenhuma modificação para algumas das partes.]

```
if (y == 8)
if (x == 5)
System.out.println("@@@@@");
else
System.out.println("#####");
System.out.println("$$$$$");
System.out.println("&&&&&");
```

a) Supondo que `x = 5` e `y = 8`, a seguinte saída é produzida:

```
@@@@@
$$$$$
&&&&&
```

b) Supondo que `x = 5` e `y = 8`, a seguinte saída é produzida:

```
@@@@@
```

c) Supondo que `x = 5` e `y = 8`, a seguinte saída é produzida:

```
@@@@@
```

d) Supondo que `x = 5` e `y = 7`, a seguinte saída é produzida. [*Observação:* todas as três últimas instruções de saída depois do `else` são partes de um bloco].

```
#####
$$$$$
&&&&&
```

4.29 (***Quadrado de asteriscos***) Escreva um aplicativo que solicita ao usuário que insira o tamanho do lado de um quadrado e, então, exibe um quadrado vazio desse tamanho com asteriscos. Seu programa deve trabalhar com quadrados de todos os comprimentos de lado possíveis entre 1 e 20.

4.30 (***Palíndromos***) Um palíndromo é uma sequência de caracteres que é lida da esquerda para a direita ou da direita para a esquerda. Por exemplo, cada um dos seguintes inteiros de cinco dígitos é um palíndromo: 12321, 55555, 45554 e 11611. Escreva um aplicativo que lê em um inteiro de cinco dígitos e determina se ele é ou não um palíndromo. Se o número não for de cinco dígitos, exiba uma mensagem de erro e permita que o usuário insira um novo valor.

4.31 (***Imprimindo o equivalente decimal de um número binário***) Escreva um aplicativo que insere um número inteiro que contém somente 0s e 1s (isto é, um número inteiro binário) e imprime seu equivalente decimal. [*Dica:* utilize os operadores de resto e divisão para pegar os dígitos do número binário um de cada vez da direita para a esquerda. No sistema de números decimais, o dígito mais à direita tem um valor posicional de 1 e o próximo dígito à esquerda um valor posicional de 10, depois 100, depois 1.000 e assim por diante. O número decimal 234 pode ser interpretado como 4 * 1 + 3 * 10 + 2 * 100. No sistema de número binário, o dígito mais à direita tem um valor posicional de 1, o próximo dígito à esquerda um valor posicional de 2, depois 4, depois 8 e assim por diante. O equivalente decimal do binário 1.101 é 1 * 1 + 0 * 2 + 1 * 4 + 1 * 8 ou 1 + 0 + 4 + 8 ou 13.]

4.32 (***Padrão de tabuleiro de damas de asteriscos***) Escreva um aplicativo que utiliza apenas as instruções de saída

```
System.out.print("* ");
System.out.print(" ");
System.out.println();
```

para exibir o padrão de tabuleiro de damas a seguir. Uma chamada de método `System.out.println` sem argumentos faz com que o programa gere saída de um único caractere de nova linha. [*Dica:* as instruções de repetição são requeridas.]

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

4.33 (***Múltiplos de 2 com um loop infinito***) Escreva um aplicativo que continue exibindo na janela de comando os múltiplos do inteiro 2 — a saber, 2, 4, 8, 16, 32, 64 e assim por diante. Seu loop não deve terminar (isto é, deve criar um loop infinito). O que acontece quando você executa esse programa?

4.34 (***O que há de errado com esse código?***) O que há de errado com a seguinte instrução? Forneça a instrução correta para adicionar 1 à soma de `x` e `y`.

```
System.out.println(++(x + y));
```

4.35 (***Lados de um triângulo***) Escreva um aplicativo que lê três valores diferentes de zero inseridos pelo usuário, determina e imprime se eles poderiam representar os lados de um triângulo.

4.36 (***Lados de um triângulo direito***) Escreva um aplicativo que lê três inteiros diferentes de zero, determina e imprime se eles poderiam representar os lados de um triângulo direito.

4.37 (***Fatorial***) O fatorial de um inteiro não negativo n é escrito como $n!$ (pronuncia-se "n fatorial") e é definido como segue:

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1 \quad \text{(para valores de } n \text{ maiores ou iguais a 1)}$$

e

$$n! = 1 \quad \text{(para } n = 0\text{)}$$

Por exemplo, $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$, o que dá 120.

a) Escreva um aplicativo que lê um inteiro não negativo, calcula e imprime seu fatorial.

b) Escreva um aplicativo que estima o valor da constante matemática *e* utilizando a fórmula a seguir. Permita ao usuário inserir o número de termos a calcular.

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

c) Escreva um aplicativo que computa o valor de e^x utilizando a fórmula a seguir. Permita ao usuário inserir o número de termos a calcular.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Fazendo a diferença

4.38 **(*Impondo privacidade com criptografia*)** O crescimento explosivo de comunicação e armazenamento de dados na internet e em computadores conectados por ela aumentou muito a preocupação com a privacidade. O campo da criptografia envolve a codificação de dados para torná-los difíceis de acessar (e espera-se — com os esquemas mais avançados — impossíveis de acessar) para usuários sem autorização de leitura. Nesse exercício, você investigará um esquema simples para criptografar e descriptografar dados. Uma empresa que quer enviar dados pela internet pediu-lhe que escrevesse um programa que criptografe dados a fim de que eles possam ser transmitidos com maior segurança. Todos os dados são transmitidos como inteiros de quatro dígitos. Seu aplicativo deve ler um inteiro de quatro dígitos inserido pelo usuário e criptografá-lo da seguinte maneira: substitua cada dígito pelo resultado da adição de 7 ao dígito, obtendo o restante depois da divisão do novo valor por 10. Troque então o primeiro dígito pelo terceiro e o segundo dígito pelo quarto. Então, imprima o inteiro criptografado. Escreva um aplicativo separado que receba a entrada de um inteiro de quatro dígitos criptografado e o descriptografe (revertendo o esquema de criptografia) para formar o número original. [*Projeto de leitura opcional:* pesquise a "criptografia de chave pública" em geral e o esquema de chave pública específica PGP (Pretty Good Privacy). Você também pode querer investigar o esquema RSA, que é amplamente usado em aplicativos robustos industriais.]

4.39 **(*Crescimento demográfico mundial*)** A população mundial cresceu consideravelmente ao longo dos séculos. O crescimento contínuo pode, por fim, desafiar os limites de ar respirável, água potável, terra fértil para agricultura e outros recursos limitados. Há evidência de que o crescimento tem reduzido a velocidade nos últimos anos e que a população mundial pode chegar ao ponto máximo em algum momento nesse século e, então, começar a diminuir.

Para esse exercício, pesquise questões de crescimento demográfico mundial on-line. *Não deixe de investigar vários pontos de vista.* Obtenha estimativas da população mundial atual e sua taxa de crescimento (a porcentagem pela qual provavelmente aumentará neste ano). Escreva um programa que calcule o crescimento demográfico mundial anual dos próximos 75 anos, *utilizando a premissa simplificadora de que a taxa de crescimento atual ficará constante*. Imprima os resultados em uma tabela. A primeira coluna deve exibir os anos do ano 1 ao ano 75. A segunda coluna deve exibir a população mundial esperada no fim desse ano. A terceira deve exibir o aumento numérico na população mundial que ocorreria nesse ano. Utilizando os seus resultados, determine o ano em que a população dobraria com relação ao número de hoje se a taxa de crescimento do ano atual persistisse.