**Capítulo 01**

**Exercícios de revisão**

**1.1** Preencha as lacunas em cada uma das seguintes afirmações:

a) Os computadores processam dados sob o controle de conjuntos de instruções chamados programas.

b) As principais unidades lógicas do computador são unidade de entrada, unidade de saída, unidade de memória, unidade de central de processamento, unidade aritmética e lógica e unidade de armazenamento secundário.

c) Os três tipos de linguagens discutidas no capítulo são linguagens de máquina, linguagens de assembly e linguagens de alto nível.

d) Os programas que traduzem programas de linguagem de alto nível em linguagem de máquina são chamados compiladores.

e) Android é um sistema operacional para dispositivos móveis baseados no kernel do Linux e Java.

f) O software candidato a lançamento em geral tem todos os recursos, sendo (supostamente) livre de erros e pronto para uso pela comunidade.

g) O Wii Remote, bem como muitos smartphones, usa um(a) acelerômetro que permite ao dispositivo responder ao movimento.

**1.2** Preencha as lacunas em cada uma das seguintes frases sobre o ambiente Java:

a) O comando java do JDK executa um aplicativo Java.

b) O comando javac do JDK compila um programa Java.

c) Um arquivo de código-fonte aberto Java deve terminar com a extensão .java.

d) Quando um programa Java é compilado, o arquivo produzido pelo compilador termina com a extensão .class.

e) O arquivo produzido pelo compilador Java contém bytecodes, que são executados pela Java Virtual Machine.

**1.3** Preencha as lacunas de cada uma das sentenças a seguir (com base na Seção 1.5):

a) Os objetos permitem a prática de ocultamento de informações — embora eles possam se comunicar entre si por meio de interfaces bem definidas, normalmente não têm autorização para descobrir como outros objetos são implementados.

b) Os programadores Java concentram-se na criação de classes, que contêm campos e o conjunto de métodos que manipulam esses campos, além de fornecer serviços para clientes.

c) O processo de analisar e projetar um sistema de um ponto de vista orientado a objetos é chamado análise e projeto orientados a objetos (OOAD).

d) Uma nova classe de objetos pode ser convenientemente criada por herança — a nova classe (chamada subclasse) começa com as características de uma classe existente (chamada superclasse), personalizando-as e talvez adicionando características próprias.

e) Unified Modeling Language (UML) é uma linguagem gráfica que permite às pessoas que projetam sistemas de software utilizar uma notação padrão da indústria para representá-las.

f) O tamanho, forma, cor e peso de um objeto são considerados atributos da classe dele.

**Questões**

**1.4** Preencha as lacunas em cada uma das seguintes afirmações:

a) A unidade lógica que recebe informações de fora do computador para uso por ele é a \_\_\_\_\_\_\_\_.

b) O processo de instrução do computador para resolver um problema específico é chamado \_\_\_\_\_\_\_\_.

c) \_\_\_\_\_\_\_\_ é um tipo de linguagem de computador que utiliza abreviações em inglês para instruções de linguagem de máquina.

d) \_\_\_\_\_\_\_\_ é uma unidade lógica que envia informações que já foram processadas pelo computador para vários dispositivos, de modo que possam ser utilizadas fora da máquina.

e) \_\_\_\_\_\_\_\_ e \_\_\_\_\_\_\_\_ são unidades lógicas do computador que retêm informações.

f) \_\_\_\_\_\_\_\_ é uma unidade lógica do computador que realiza cálculos.

g) \_\_\_\_\_\_\_\_ é uma unidade lógica do computador que toma decisões lógicas.

h) As linguagens mais convenientes para que o programador escreva programas rápida e facilmente são as \_\_\_\_\_\_\_\_.

i) A única linguagem que um computador pode entender diretamente é a \_\_\_\_\_\_\_\_ dele.

j) \_\_\_\_\_\_\_\_ é uma unidade lógica do computador que coordena as atividades de todas as outras unidades lógicas.

**1.5** Preencha as lacunas em cada uma das seguintes afirmações:

a) A linguagem de programação \_\_\_\_\_\_\_\_ é agora utilizada para desenvolver aplicativos corporativos de grande porte, aprimorar a

funcionalidade de servidores da web, fornecer aplicativos a dispositivos de consumo popular e para muitos outros propósitos.

b) Inicialmente, o \_\_\_\_\_\_\_\_ tornou-se muito conhecido como a linguagem de desenvolvimento do sistema operacional UNIX.

c) O \_\_\_\_\_\_\_\_ garante que as mensagens, que consistem em partes sequencialmente numeradas chamadas bytes, sejam adequadamente

encaminhadas do emissor para o receptor, cheguem intactas e sejam montadas na ordem correta.

d) A linguagem de programação \_\_\_\_\_\_\_\_ foi desenvolvida por Bjarne Stroustrup no início dos anos 1980 na Bell Laboratories.

**1.6** Preencha as lacunas em cada uma das seguintes afirmações:

a) Os programas Java normalmente passam por cinco fases: \_\_\_\_\_\_\_\_*,* \_\_\_\_\_\_\_\_*,* \_\_\_\_\_\_\_\_*,* \_\_\_\_\_\_\_\_ e \_\_\_\_\_\_\_\_*.*

b) Um(a) \_\_\_\_\_\_\_\_ fornece muitas ferramentas que suportam o processo de desenvolvimento de software, como editores para escrever

e editar programas, depuradores a fim de localizar erros de lógica em programas e muitos outros recursos.

c) O comando java invoca \_\_\_\_\_\_\_\_, que executa programas Java.

d) Um(a) \_\_\_\_\_\_\_\_ é um aplicativo de software que simula um computador, mas oculta o sistema operacional e o hardware subjacentes

dos programas que interagem com ela(e).

e) O \_\_\_\_\_\_\_\_ transfere os arquivos .class contendo os bytecodes do programa para a memória principal.

f) O \_\_\_\_\_\_\_\_ examina bytecodes para assegurar que eles são válidos.

**1.7** Explique as duas fases de compilação de programas Java.

**1.8** Um dos objetos mais comuns do mundo é um relógio de pulso. Discuta como cada um dos seguintes termos e conceitos se aplicam à noção

de um relógio: objeto, atributos, comportamentos, classe, herança (considere, por exemplo, o alarme dele), modelagem, mensagens,

encapsulamento, interface e ocultamento de informações.

**Fazendo a diferença**

Ao longo deste livro incluiremos exercícios “Fazendo a diferença”, pelos quais você será convidado a trabalhar em problemas que realmente importam a indivíduos, comunidades, países e ao mundo. Para informações adicionais sobre organizações em todo o planeta que trabalham a fim de fazer a diferença, além de ideias de projetos de programação relacionadas, visite nosso Making a Difference Resource Center em [www.deitel.com/makingadifference](http://www.deitel.com/makingadifference).

**1.9** (*Test-drive: calculadora de emissão de carbono*) Alguns cientistas acreditam que as emissões de carbono, em especial da queima de combustíveis fósseis, contribuem significativamente para o aquecimento global, e que isso pode ser combatido se as pessoas tomarem medidas a fim de limitar o uso de combustíveis com base de carbono. As organizações e pessoas estão cada vez mais preocupadas com suas “emissões de carbono”. Sites (em inglês) como o TerraPass <http://www.terrapass.com/carbon-footprint-calculator/> e o Carbon Footprint <http://www.carbonfootprint.com/calculator.aspx> fornecem calculadoras para estimar a “pegada de carbono”. Teste essas calculadoras com o intuito de determinar as suas emissões de carbono.

Os exercícios nos próximos capítulos solicitarão que você programe sua própria calculadora de emissões de carbono. A fim de se preparar, use a web como recurso de pesquisa de fórmulas para esse cálculo.

**1.10** (*Test-drive: calculadora de índice de massa corporal*) A obesidade causa agravamentos significativos de problemas de saúde como diabetes e doenças cardíacas. Para determinar se uma pessoa está acima do peso ou obesa, você pode utilizar uma medida chamada índice de massa corporal (IMC). Os departamentos de assistência social e de saúde norte-americanos fornecem uma calculadora do IMC em <http://www.nhlbi.nih.gov/guidelines/obesity/BMI/bmicalc.htm>. Utilize-a para calcular seu próprio IMC. Um exercício adiante solicitará que você programe sua própria calculadora. A fim de se preparar, use a web como recurso de pesquisa de fórmulas para esse cálculo.

**1.11** (*Atributos dos veículos híbridos*) Neste capítulo você aprendeu alguns dos conceitos básicos das classes. Agora você começará a estabelecer os aspectos de uma classe chamada “veículos híbridos”. Eles estão se tornando cada vez mais populares, porque muitas vezes têm um desempenho por litro de combustível muito melhor do que veículos que só utilizam gasolina. Navegue pela web e estude os recursos de quatro ou cinco carros híbridos atuais mais populares, e então liste o maior número possível de atributos relacionados a eles. Alguns mais comuns incluem desempenho urbano por litro de combustível e desempenho em estradas. Liste também as particularidades das baterias (tipo, peso etc.).

**1.12** (*Neutralidade de gêneros*) Muitas pessoas querem eliminar o machismo em todas as formas de comunicação. Solicitaram a você que criasse um programa que pudesse processar um parágrafo de um texto e substituir palavras definidoras de gênero por outras neutras. Supondo que você recebeu uma lista de termos específicos quanto a gênero e seus equivalentes neutros (por exemplo, substituir “wife” e “husband” por “spouse”, “man” e “woman” por “person”, “daughter” e “son” por “child”), explique o procedimento que usaria para examinar um parágrafo e fazer manualmente as substituições. Como essa tarefa pode gerar um termo estranho como “woperchild?” Você aprenderá que uma palavra mais formal para “procedimento” é “algoritmo”, e que um algoritmo estabelece os passos a serem executados e a ordem na qual eles acontecem. Mostraremos como desenvolver algoritmos e então convertê-los em programas Java que podem ser executados em computadores.

**Capítulo 02**

**Exercícios de revisão**

**2.1** Preencha as lacunas em cada uma das seguintes afirmações:

a) Um(a) { começa o corpo de cada método e um(a) } termina o corpo de cada método.

b) Você pode usar a declaração if para tomar decisões.

c) // começa em um comentário de fim de linha.

d) Caracteres de espaço, novas linhas e tabulações são chamados espaço em branco.

e) Palavras-chave são reservadas para uso pelo Java.

f) Aplicativos Java iniciam a execução no método main.

g) Os métodos System.out.print, System.out.println e System.out.printf exibem informações em uma janela de comando.

**2.2** Determine se cada uma das seguintes afirmações é *verdadeira* ou *falsa*. Se *falsa*, explique por quê.

a) Os comentários fazem com que o computador imprima o texto depois das // na tela quando o programa executa. Falso. Os comentários não causam nenhuma ação quando o programa executa. Eles são utilizados para documentar programas e melhoram sua legibilidade.

b) Todas as variáveis devem ser atribuídas a um tipo quando são declaradas. Verdadeiro.

c) O Java considera que as variáveis number e NuMbEr são idênticas. Falso. Java diferencia letras maiúsculas de minúsculas, então essas variáveis são distintas.

d) O operador de resto (%) pode ser utilizado apenas com operandos inteiros. Falso. O operador de resto também pode ser utilizado com operandos não inteiros em Java.

e) Os operadores aritméticos \*, /, %, + e - têm, todos, o mesmo nível de precedência. Falso. Os operadores \*, / e % têm uma precedência mais alta que os operadores + e -.

**2.3** Escreva instruções para realizar cada uma das tarefas a seguir: Resolução: Ex02\_03.java

a) Declare que as variáveis c, thisIsAVariable, q76354 e number serão do tipo int.

int c, thisIsAVariable, q76354, number;

ou

int c;

int thisIsAVariable;

int q76354;

int number;

b) Solicite que o usuário insira um inteiro. System.out.print(“Enter an integer: “);

c) Insira um inteiro e atribua o resultado à variável int value. Suponha que a variável Scanner input possa ser utilizada para ler um valor digitado pelo usuário. value = input.nextInt();

d) Imprima “This is a Java program" em uma linha na janela de comando. Use o método System.out.println. System.out.println(“This is a Java program”);

e) Imprima “This is a Java program" em duas linhas na janela de comando. A primeira deve terminar com Java. Utilize o método System.out.printf e dois especificadores de formato %s.

System.out.printf(“%s%n%s%n”, “This is a Java”, “program”);

f) Se a variável number não for igual a 7, exiba “The variable number is not equal to 7".

if (number != 7)

System.out.println(“The variable number is not equal to 7”);

**2.4** Identifique e corrija os erros em cada uma das seguintes instruções: Resolução: Ex02\_04.java

a) if (c < 7);

System.out.println(“c is less than 7”);

Erro: o ponto e vírgula depois do parêntese direito da condição (c < 7) no if. Correção: remova o ponto e vírgula depois do parêntese direito. [*Observação:* como resultado, a instrução de saída executará independentemente de a condição em if ser verdadeira.]

b) if (c => 7)

System.out.println(“c is equal to or greater than 7”);

Erro: o operador relacional => é incorreto. Correção: altere => para >=.

**2.5** Escreva declarações, instruções ou comentários que realizem cada uma das tarefas a seguir: Resolução: Ex02\_05-06

a) Declare que um programa calculará o produto de três inteiros. // Calcula o produto de três inteiros

b) Crie um Scanner chamado input que leia valores a partir da entrada padrão. Scanner input = new Scanner(System.in);

c) Declare as variáveis x, y, z e result como tipo int.

int x, y, z, result;

ou

int x;

int y;

int z;

int result;

d) Solicite que o usuário insira o primeiro inteiro. System.out.print(“Enter first integer: “);

e) Leia o primeiro inteiro digitado pelo usuário e armazene-o na variável x. x = input.nextInt();

f) Solicite que o usuário insira o segundo inteiro. System.out.print(“Enter second integer: “);

g) Leia o segundo inteiro digitado pelo usuário e armazene-o na variável y. y = input.nextInt();

h) Solicite que o usuário insira o terceiro inteiro. System.out.print(“Enter third integer: “);

i) Leia o terceiro inteiro digitado pelo usuário e armazene-o na variável z. z = input.nextInt();

j) Compute o produto dos três inteiros contidos nas variáveis x, y e z e atribua o resultado à variável result. result = x \* y \* z;

k) Use System.out.printf para exibir a mensagem “Product is” seguida pelo valor da variável result.

System.out.printf(“Product is %d%n”, result);

**2.6** Usando as instruções que você escreveu no Exercício 2.5, elabore um programa completo que calcule e imprima o produto de três inteiros. Resolução: Ex02\_05-06

A solução para o exercício de revisão 2.6 é a seguinte:

// Exercício 2.6: Product.Java

// Calcula o produto de três inteiros.

import java.util.Scanner; // programa utiliza Scanner

public class Product

{

public static void main(String[ ] args)

{

// cria Scanner para obter entrada a partir da janela de comando

Scanner input = new Scanner(System.in);

int x; // primeiro número inserido pelo usuário

int y; // segundo número inserido pelo usuário *continua*

int z; // terceiro número inserido pelo usuário

int result; // produto dos números

System.out.print("Enter first integer: "); // solicita entrada

x = input.nextInt(); // lê o primeiro inteiro

System.out.print("Enter second integer: "); // solicita entrada

y = input.nextInt(); // lê o segundo inteiro

System.out.print("Enter third integer: "); // solicita entrada

z = input.nextInt(); // lê o terceiro inteiro

result = x \* y \* z; // calcula o produto dos números

System.out.printf("Product is %d%n", result);

} // fim do método main

} // fim da classe Product

Enter first integer: **10**

Enter second integer: **20**

Enter third integer: **30**

Product is 6000

**Questões**

**2.7** Preencha as lacunas em cada uma das seguintes afirmações:

a) /\* \*\*/ são utilizados para documentar um programa e aprimorar sua legibilidade.

b) Uma decisão pode ser tomada em um programa Java com um(a) if.

c) Os cálculos normalmente são realizados pelas instruções +, -, \* e /.

d) Os operadores aritméticos com a mesma precedência da multiplicação são / e %.

e) Quando parênteses em uma expressão aritmética estão aninhados, o conjunto de parênteses interior é avaliado primeiro.

f) Uma posição na memória do computador que pode conter valores diferentes várias vezes ao longo da execução de um programa é chamada variável.

**2.8** Escreva instruções Java que realizem cada uma das seguintes tarefas: Resolução: Ex02\_08.java

a) Exibir a mensagem “Enter an integer: “, deixando o cursor na mesma linha.

b) Atribuir o produto de variáveis b e c para a variável a.

c) Utilizar um comentário para afirmar que um programa executa um cálculo de exemplo de folha de pagamento.

**2.9** Determine se cada uma das seguintes afirmações é *verdadeira* ou *falsa*. Se *falsa*, explique por quê.

a) Operadores Java são avaliados da esquerda para a direita. V

b) Os seguintes nomes são todos de variável válidos: \_under\_bar\_, m928134, t5, j7, her\_sales$, his\_$account\_total, a, b$, c, z e z2. F

c) Uma expressão aritmética Java válida sem parênteses é avaliada da esquerda para a direita. V

d) Os seguintes nomes são todos de variável inválidos: 3g, 87, 67h2, h22 e 2h. F

**2.10** Supondo que x = 2 e y = 3, o que cada uma das instruções a seguir exibe?

a) System.out.printf(“x = %d%n”, x); x = 2

b) System.out.printf(“Value of %d + %d is %d%n”, x, x, (x + x)); Value of 2 + 2 is 4

c) System.out.printf(“x =”); x =

d) System.out.printf(“%d = %d%n”, (x + y), (y + x)); 5 = 5

**2.11** Quais instruções Java a seguir contêm variáveis cujos valores são modificados?

a) p = i + j + k + 7; V

b) System.out.println(“variables whose values are modified”); F

c) System.out.println(“a = 5”); F

d) value = input.nextInt(); V

**2.12** Dado que *y = ax*3 + 7, quais das seguintes alternativas são instruções Java corretas para essa equação?

a) y = a \* x \* x \* x + 7; V

b) y = a \* x \* x \* (x + 7); F

c) y = (a \* x) \* x \* (x + 7); F

d) y = (a \* x) \* x \* x + 7; V

e) y = a \* (x \* x \* x) + 7; V

f) y = a \* x \* (x \* x + 7); F

**2.13** Declare a ordem de avaliação dos operadores em cada uma das seguintes instruções Java e mostre o valor de x depois que cada instrução

é realizada:

a) x = 7 + 3 \* 6 / 2 - 1;

x = 7 + 3 \* 6 / 2 - 1

x = 7 + 18 / 2 - 1

x = 7 + 9 - 1

x = 7 + 9 - 1

x = 15

b) x = 2 % 2 + 2 \* 2 - 2 / 2;

x = 2 % 2 + 2 \* 2 - 2 / 2

x = 0 + 2 \* 2 - 2 / 2

x = 0 + 4 - 2 / 2

x = 0 + 4 - 1

x = 3

c) x = (3 \* 9 \* (3 + (9 \* 3 / (3))));

x = (3 \* 9 \* (3 + (9 \* 3 / (3))))

x = (3 \* 9 \* (3 + (9 \* 3 / 3)))

x = (3 \* 9 \* (3 + (27 / 3)))

x = (3 \* 9 \* (3 + 9))

x = (3 \* 9 \* 12)

x = 324

**2.14** Escreva um aplicativo que exiba os números 1 a 4 na mesma linha, com cada par de adjacentes separados por um espaço. Use as seguintes técnicas: Resolução: Ex02\_14.java

a) Uma instrução System.out.println.

b) Quatro instruções System.out.print.

c) Uma instrução System.out.printf.

**2.15** (*Aritmética*) Escreva um aplicativo que solicite ao usuário inserir dois inteiros, obtenha dele esses números e imprima sua soma, produto, diferença e quociente (divisão). Utilize as técnicas mostradas na Figura 2.7. Resolução: Ex02\_15.java

**2.16** (*Comparando inteiros*) Escreva um aplicativo que solicite ao usuário inserir dois inteiros, obtenha dele esses números e exiba o número maior seguido pelas palavras “is larger". Se os números forem iguais, imprima a mensagem “These numbers are equal". Utilize as técnicas mostradas na Figura 2.15. Resolução: Ex02\_16.java

**2.17** (*Aritmética, menor e maior*) Escreva um aplicativo que insira três inteiros digitados pelo usuário e exiba a soma, média, produto e os números menores e maiores. Utilize as técnicas mostradas na Figura 2.15. [*Observação:* o cálculo da média neste exercício deve resultar em uma representação de inteiro. Assim, se a soma dos valores for 7, a média deverá ser 2, não 2,3333...] Resolução: Ex02\_17.java

**2.18** (*Exibindo formas com asteriscos*) Escreva um aplicativo que exiba uma caixa, uma elipse, uma seta e um losango utilizando asteriscos (\*), como segue: Resolução: Ex02\_18.java

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

**2.19** O que o seguinte código imprime? Resolução: Ex02\_19.java

System.out.printf("\*%n\*\*%n\*\*\*%n\*\*\*\*%n\*\*\*\*\*%n");

**2.20** O que o seguinte código imprime? Resolução: Ex02\_20.java

System.out.println("\*");

System.out.println("\*\*\*");

System.out.println("\*\*\*\*\*");

System.out.println("\*\*\*\*");

System.out.println("\*\*");

**2.21** O que o seguinte código imprime? Resolução: Ex02\_21.java

System.out.print("\*");

System.out.print("\*\*\*");

System.out.print("\*\*\*\*\*");

System.out.print("\*\*\*\*");

System.out.println("\*\*");

**2.22** O que o seguinte código imprime? Resolução: Ex02\_22.java

System.out.print("\*");

System.out.println("\*\*\*");

System.out.println("\*\*\*\*\*");

System.out.print("\*\*\*\*");

System.out.println("\*\*");

**2.23** O que o seguinte código imprime? Resolução: Ex02\_23.java

System.out.printf("%s%n%s%n%s%n", "\*", "\*\*\*", "\*\*\*\*\*");

**2.24** (*Inteiros maiores e menores*) Escreva um aplicativo que leia cinco inteiros, além de determinar e imprimir o maior e o menor inteiro no grupo. Utilize somente as técnicas de programação que você aprendeu neste capítulo. Resolução: Ex02\_24.java

**2.25** (*Ímpar ou par*) Escreva um aplicativo que leia um inteiro, além de determinar e imprimir se ele é ímpar ou par. [*Dica:* utilize o operador de resto. Um número par é um múltiplo de 2. Qualquer múltiplo de 2 deixa um resto 0 quando dividido por 2.] Resolução: Ex02\_25.java

**2.26** (*Múltiplos*) Escreva um aplicativo que leia dois inteiros, além de determinar se o primeiro é um múltiplo do segundo e imprimir o resultado. [*Dica:* utilize o operador de resto.] Resolução: Ex02\_26.java

**2.27** (*Padrão de tabuleiro de damas de asteriscos*) Escreva um aplicativo que exiba um padrão de tabuleiro de damas, como mostrado a seguir: Resolução: Ex02\_27.java

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média**

**2.28** (*Diâmetro, circunferência e área de um círculo*) Eis uma prévia do que veremos mais adiante. Neste capítulo, você aprendeu sobre inteiros e o tipo int. O Java também pode representar números de pontos flutuantes que contêm pontos de fração decimal, como 3,14159. Escreva um aplicativo que leia a entrada a partir do usuário do raio de um círculo como um inteiro e imprima o diâmetro do círculo, circunferência e área utilizando o valor do ponto flutuante 3,14159 para π. Utilize as técnicas mostradas na Figura 2.7. [Observação: você também pode empregar a constante Math.PI predefinida para o valor de π. Essa constante é mais precisa que o valor 3,14159. A classe Math é definida no pacote java.lang. As classes nesse pacote são importadas automaticamente, portanto, você não precisa importar a classe Math para utilizá-la.] Adote as seguintes fórmulas (*r* é o raio):

*diâmetro* = 2*r*

*circunferência* = 2π*r*

*área* = π*r*2

Não armazene os resultados de cada cálculo em uma variável. Em vez disso, especifique cada cálculo como o valor de saída em uma instrução System.out.printf. Os valores produzidos pelos cálculos de circunferência e área são números de ponto flutuante. A saída desses valores pode ser gerada com o especificador de formato %f em uma instrução System.out.printf. Você aprenderá mais sobre números de pontos flutuantes no Capítulo 3. Resolução: Ex02\_27.java

**2.29** (*O valor inteiro de um caractere*) Eis outra prévia do que virá adiante. Neste capítulo, você aprendeu sobre inteiros e o tipo int. O Java também pode representar letras maiúsculas, minúsculas e uma variedade considerável de símbolos especiais. Cada caractere tem uma representação correspondente de inteiro. O conjunto de caracteres que um computador utiliza com as respectivas representações na forma de inteiro desses caracteres é chamado de conjunto de caracteres desse computador. Você pode indicar um valor de caractere em um programa simplesmente incluindo esse caractere entre aspas simples, como em ‘A'. Resolução: Ex02\_29.java

Você pode determinar o equivalente em inteiro de um caractere precedendo-o com (int), como em

(int) 'A'

Um operador dessa forma é chamado operador de coerção. (Você aprenderá sobre os operadores de coerção no Capítulo 4.) A instrução a seguir gera saída de um caractere e seu equivalente de inteiro:

System.out.printf("The character %c has the value %d%n", 'A', ((int) 'A'));

Quando a instrução precedente executa, ela exibe o caractere A e o valor 65 (do conjunto de caracteres Unicode®) como parte da string. O especificador de formato %c é um espaço reservado para um caractere (nesse caso, ‘A').

Utilizando instruções semelhantes àquela mostrada anteriormente neste exercício, escreva um aplicativo que exiba os equivalentes inteiros de algumas letras maiúsculas, minúsculas, dígitos e símbolos especiais. Mostre os equivalentes inteiros do seguinte: A B C a b c 0 1 2 $ \* + / e o caractere em branco.

**2.30** (*Separando os dígitos em um inteiro*) Escreva um aplicativo que insira um número consistindo em cinco dígitos a partir do usuário, separe o número em seus dígitos individuais e imprima os dígitos separados uns dos outros por três espaços. Por exemplo, se o usuário digitar o número 42339, o programa deve imprimir 4 2 3 3 9

Suponha que o usuário insira o número correto de dígitos. O que acontece quando você insere um número com mais de cinco dígitos?

O que acontece quando você insere um número com menos de cinco dígitos? [*Dica:* é possível fazer este exercício com as técnicas que aprendeu neste capítulo. Você precisará tanto das operações de divisão como das de resto para “selecionar” cada dígito.] Resolução: Ex02\_30.java

**2.31** (*Tabela de quadrados e cubos*) Utilizando apenas as técnicas de programação que aprendeu neste capítulo, escreva um aplicativo que calcule os quadrados e cubos dos números de 0 a 10 e imprima os valores resultantes em formato de tabela como a seguir: Resolução: Ex02\_31.java

**Linha do tempo

Descrição gerada automaticamente**

**2.32** (*Valores negativos, positivos e zero*) Escreva um programa que insira cinco números, além de determinar e imprimir quantos negativos, quantos positivos e quantos zeros foram inseridos. Resolução: Ex02\_32.java

**Fazendo a diferença**

**2.33** (*Calculadora de índice de massa corporal*) Introduzimos a calculadora de índice de massa corporal (IMC) no Exercício 1.10. As fórmulas para calcular o IMC são

Texto

Descrição gerada automaticamente com confiança baixa

Crie um aplicativo de calculadora IMC que leia o peso do usuário em libras e a altura em polegadas (ou, se preferir, o peso em quilogramas e a altura em metros) e, então, calcule e exiba o índice de massa corporal dele. Além disso, que exiba as seguintes informações do Department of Health and Human Services/National Institutes of Health, assim o usuário pode avaliar o seu IMC:

BMI VALUES

Underweight: less than 18.5

Normal: between 18.5 and 24.9

Overweight: between 25 and 29.9

Obese: 30 or greater

[*Nota:* neste capítulo, você aprendeu a utilizar o tipo int para representar números inteiros. Os cálculos de IMC, quando feitos com valores int, produzirão resultados com números inteiros. No Capítulo 3, você aprenderá a utilizar o tipo double para representar números com pontos decimais. Quando os cálculos de IMC são realizados com doubles, eles produzirão números com pontos decimais — esses são chamados de números de “ponto flutuante”.] Resolução: Ex02\_33.java

**2.34** (*Calculadora de crescimento demográfico mundial*) Utilize a internet para descobrir a população mundial atual e a taxa de crescimento demográfico mundial anual. Escreva um aplicativo que introduza esses valores e, então, que exiba a população mundial estimada depois de um, dois, três, quatro e cinco anos. Resolução: Ex02\_34.java

**2.35** (*Calculadora de economia da faixa solidária*) Pesquise vários sites sobre faixa solidária. Crie um aplicativo que calcule o custo diário de dirigir, para estimar quanto dinheiro pode ser economizado com o uso da faixa solidária, que também tem outras vantagens, como reduzir emissões de carbono e congestionamento de tráfego. O aplicativo deve introduzir as seguintes informações e exibir o custo por dia de dirigir para o trabalho do usuário: Resolução: Ex02\_35.java

a) Quilômetros totais dirigidos por dia.

b) Preço por litro de gasolina.

c) Quilômetros médios por litro.

d) Taxas de estacionamento por dia.

e) Pedágio por dia.

**Capítulo 03**

**Exercícios de revisão**

**3.1** Preencha as lacunas em cada uma das seguintes sentenças:

a) Toda declaração de classe que inicia com a palavra-chave public deve ser armazenada em um arquivo que tem exatamente o mesmo nome que a classe e terminar com a extensão de nome do arquivo .java.

b) A palavra-chave class em uma declaração de classe é imediatamente seguida pelo nome da classe.

c) A palavra-chave new solicita memória do sistema para armazenar um objeto, e então chama o construtor da classe correspondente para inicializar esse objeto.

d) Todo parâmetro deve especificar um(a) tipo e um(a) nome.

e) Por padrão, as classes que são compiladas no mesmo diretório são consideradas como estando no mesmo pacote, conhecido como pacote padrão.

f) O Java fornece dois tipos primitivos para armazenar números de ponto flutuante na memória: float e double.

g) As variáveis de tipo double representam números de ponto flutuante de precisão dupla.

h) O método scanner nextDouble retorna um valor double.

i) A palavra-chave public é um modificador de acesso.

j) O tipo de retorno void indica que um método não retornará um valor.

k) O método Scanner nextLine lê os caracteres até encontrar um caractere de nova linha, então retorna esses caracteres como uma String.

l) A classe String está no pacote java.lang.

m) Um(a) declaração import não é requerido(a) se você sempre referenciar uma classe por meio do seu nome completamente qualificado.

n) Um(a) número de ponto flutuante é um número com um ponto de fração decimal, como 7,33, 0,0975 ou 1000,12345.

o) As variáveis de tipo float representam simples números de ponto flutuante de dupla precisão.

p) O especificador de formato %f é utilizado para gerar saída de valores de tipo float ou double.

q) Os tipos no Java são divididos em duas categorias — tipo primitivo e tipo referência.

**3.2** Determine se cada uma das seguintes sentenças é *verdadeira* ou *falsa*. Se *falsa*, explique por quê.

a) Por convenção, os nomes de método são iniciados com letra maiúscula, e todas as palavras subsequentes a ele também começam com letra maiúscula. Falsa. Por convenção, os nomes de método são iniciados com letra minúscula e todas as palavras subsequentes começam com letra maiúscula.

b) Uma declaração import não é necessária quando uma classe em um pacote utiliza outra no mesmo pacote. Verdadeira.

c) Parênteses vazios que se seguem a um nome de método em uma declaração indicam que ele não requer nenhum parâmetro para realizar sua tarefa. Verdadeira.

d) Uma variável de tipo primitivo pode ser utilizada para invocar um método. Falsa. Uma variável de tipo primitivo não pode ser utilizada para invocar um método — uma referência a um objeto é necessária para que os métodos do objeto possam ser invocados.

e) As variáveis declaradas no corpo de um método particular são conhecidas como variáveis de instância e podem ser utilizadas em todos os métodos da classe. Falsa. Essas variáveis são chamadas variáveis locais e só podem ser utilizadas no método em que são declaradas.

f) O corpo de todos os métodos é delimitado pelas chaves esquerda e direita ({ e }). Verdadeira.

g) As variáveis locais de tipo primitivo são inicializadas por padrão. Falsa. As variáveis de instância de tipo primitivo são inicializadas por padrão. Deve-se atribuir um valor explicitamente a cada variável local.

h) As variáveis de instância de tipo por referência são inicializadas por padrão com o valor null. Verdadeira.

i) Qualquer classe que contém public static void main(String[] args) pode ser usada para executar um aplicativo. Verdadeira.

j) O número de argumentos na chamada de método deve corresponder ao de itens na lista de parâmetros da declaração desse método. Verdadeira.

k) Os valores de ponto flutuante que aparecem no código-fonte são conhecidos como literais de ponto flutuante e são tipos float por padrão. Falsa. Esses literais são de tipo double por padrão.

**3.3** Qual é a diferença entre uma variável local e uma variável de instância? Uma variável local é declarada no corpo de um método e só pode ser utilizada do ponto em que isso acontece até o fim da declaração do método. Uma variável de instância é declarada em uma classe, mas não no corpo de qualquer um dos métodos dessa classe. Além disso, as variáveis de instância são acessíveis a todos os métodos da classe. (Veremos uma exceção disso no Capítulo 8.)

**3.4** Explique o propósito de um parâmetro de método. Qual a diferença entre um parâmetro e um argumento? Um parâmetro representa informações adicionais que um método requer para realizar sua tarefa. Cada parâmetro requerido por um método é especificado na declaração do método. Um argumento é o valor real de um parâmetro de método. Quando um método é chamado, os valores de argumento são passados para os parâmetros correspondentes desse método para que ele possa realizar sua tarefa.

**Questões**

**3.5** (*Palavra-chave* ***new***) Qual é o objetivo da palavra-chave new? Explique o que acontece quando você a utiliza.

**3.6** (*Construtores padrão*) O que é um construtor padrão? Como as variáveis de instância de um objeto são inicializadas se uma classe tiver somente um construtor padrão?

**3.7** (*Variáveis de instância*) Explique o propósito de uma variável de instância.

**3.8** (*Usando classes sem importá-las*) A maioria das classes precisa ser importada antes de ser usada em um aplicativo. Por que cada aplicativo pode utilizar as classes System e String sem importá-las antes?

**3.9** (*Usando uma classe sem importá-la*) Explique como um programa pode usar a classe Scanner sem importá-la.

**3.10** (*Métodos* set *e* get) Explique por que uma classe pode fornecer um método *set* e um método *get* para uma variável de instância.

**3.11** (*Classe* ***Account*** *modificada*) Modifique a classe Account (Figura 3.8) para fornecer um método chamado withdraw que retira dinheiro de uma Account. Assegure que o valor de débito não exceda o saldo de Account. Se exceder, o saldo deve ser deixado inalterado e o método deve imprimir uma mensagem que indica "Withdrawal amount exceeded account balance" [Valor de débito excedeu o saldo da conta]. Modifique a classe AccountTest (Figura 3.9) para testar o método withdraw.

**3.12** (*Classe* ***Invoice***) Crie uma classe chamada Invoice para que uma loja de suprimentos de informática a utilize para representar uma fatura de um item vendido nela. Uma Invoice (fatura) deve incluir quatro partes das informações como variáveis de instância — o número (tipo String), a descrição (tipo String), a quantidade comprada de um item (tipo int) e o preço por item (double). Sua classe deve ter um construtor que inicializa as quatro variáveis de instância. Forneça um método *set* e um *get* para cada variável de instância.

Além disso, forneça um método chamado getInvoiceAmount que calcula o valor de fatura (isto é, multiplica a quantidade pelo

preço por item) e depois retorna esse valor como double. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo, ele deve ser configurado como 0.0. Escreva um aplicativo de teste chamado InvoiceTest que demonstra as capacidades da classe Invoice.

**3.13** (*Classe* ***Employee***) Crie uma classe chamada Employee que inclua três variáveis de instância — um primeiro nome (tipo String), um sobrenome (tipo String) e um salário mensal (double). Forneça um construtor que inicializa as três variáveis de instância. Forneça um método *set* e um *get* para cada variável de instância. Se o salário mensal não for positivo, não configure seu valor. Escreva um aplicativo de teste chamado EmployeeTest que demonstre as capacidades da classe Employee. Crie dois objetos Employee e exiba o salário *anual* de cada objeto. Então dê a cada Employee um aumento de 10% e exiba novamente o salário anual de cada Employee.

**3.14** (*Classe* ***Date***) Crie uma classe chamada Date que inclua três variáveis de instância — mês (tipo int), dia (tipo int) e ano (tipo int).

Forneça um construtor que inicializa as três variáveis de instância supondo que os valores fornecidos estejam corretos. Ofereça um método *set* e um *get* para cada variável de instância. Apresente um método displayDate que exiba mês, dia e ano separados por barras normais (/). Escreva um aplicativo de teste chamado DateTest que demonstre as capacidades da classe Date.

**3.15** (*Removendo código duplicado no método* main) Na classe AccountTest da Figura 3.9, o método main contém seis instruções (linhas 13 e 14, 15 e 16, 28 e 29, 30 e 31, 40 e 41 e 42 e 43) e cada uma exibe name e balance do objeto Account. Estude essas instruções e você perceberá que elas só diferem no objeto Account sendo manipulado — account1 ou account2. Neste exercício, você definirá um novo método displayAccount que contém *uma* cópia dessa instrução de saída. O parâmetro do método será um objeto

Account e o método irá gerar name e balance dele. Então você substituirá as seis instruções duplicadas em main por chamadas para displayAccount passando como argumento o objeto específico Account para saída.

Modifique a classe AccountTest da Figura 3.9 para declarar o seguinte método displayAccount *após* a chave direita de fechamento de main e *antes* da chave direita de fechamento da classe AccountTest:

public static void displayAccount(Account accountToDisplay)

{

// coloque aqui a instrução que exibe

// o name e o balance de accountToDisplay

}

Substitua o comentário no corpo do método por uma instrução que exiba name e balance de accountToDisplay.

Lembre-se de que main é um método static, assim pode ser chamado sem antes criar um objeto da classe em que é declarado. Também declaramos o método displayAccount como um método static. Quando main tem de chamar outro método na mesma classe sem antes criar um objeto dela, o outro método *também* deve ser declarado static.

Depois de concluir a declaração de displayAccount, modifique main para substituir as instruções que exibem name e balance de cada Account pelas chamadas para displayAccount — cada uma recebendo como seu argumento o objeto account1 ou account2, como apropriado. Então, teste a classe AccountTest atualizada para garantir que ela produz a mesma saída como mostrado na Figura

3.9.

**Fazendo a diferença**

**3.16** (*Calculadora de frequência cardíaca alvo*) Ao fazer exercícios físicos, você pode utilizar um monitor de frequência cardíaca para ver se sua frequência permanece dentro de um intervalo seguro sugerido pelos seus treinadores e médicos. Segundo a American Heart Association (AHA) (www.americanheart.org/presenter.jhtml?identifier=4736), a fórmula para calcular a *frequência cardíaca máxima* por minuto é 220 menos a idade em anos. Sua *frequência cardíaca alvo* é um intervalo entre 50-85% da sua frequência cardíaca máxima. [*Observação:* essas fórmulas são estimativas fornecidas pela AHA. As frequências cardíacas máximas e alvo podem variar com base na saúde, capacidade física e sexo da pessoa. Sempre consulte um médico ou profissional de saúde qualificado antes de começar ou modificar um programa de exercícios físicos.] Crie uma classe chamada HeartRates. Os atributos da classe devem incluir o nome, sobrenome e data de nascimento da pessoa (consistindo em atributos separados para mês, dia e ano de nascimento). Sua classe deve ter um construtor que receba esses dados como parâmetros. Para cada atributo forneça métodos *set* e *get*. A classe também deve incluir um método que calcule e retorne a idade (em anos), um que calcule e retorne a frequência cardíaca máxima e um que calcule e retorne a frequência cardíaca alvo da pessoa. Escreva um aplicativo Java que solicite as informações da pessoa, instancie um objeto da classe HeartRates e imprima as informações a partir desse objeto — incluindo nome, sobrenome e data de nascimento da pessoa — calcule e imprima a idade da pessoa (em anos), seu intervalo de frequência cardíaca máxima e sua frequência cardíaca alvo.

**3.17** (*Computadorização dos registros de saúde*) Uma questão relacionada à assistência médica discutida ultimamente nos veículos de comunicação é a computadorização dos registros de saúde. Essa possibilidade está sendo abordada de maneira cautelosa por causa de preocupações quanto à privacidade e à segurança de dados sigilosos, entre outros motivos. [Iremos discutir essas preocupações em exercícios posteriores.] A computadorização dos registros de saúde pode facilitar que pacientes compartilhem seus perfis e históricos de saúde entre vários profissionais de saúde. Isso talvez aprimore a qualidade da assistência médica, ajude a evitar conflitos e prescrições erradas de medicamentos, reduza custos em ambulatórios e salve vidas. Neste exercício, você projetará uma classe HealthProfile “inicial” para uma pessoa. Os atributos da classe devem incluir nome, sobrenome, sexo, data de nascimento (consistindo em atributos separados para mês, dia e ano de nascimento), altura (em metros) e peso (em quilogramas) da pessoa. Sua classe deve ter um construtor que receba esses dados. Para cada atributo, forneça métodos *set* e *get*. A classe também deve incluir métodos que calculem e retornem a idade do usuário em anos, intervalo de frequência cardíaca máxima e frequência cardíaca alvo (veja o Exercício 3.16), além de índice de massa corporal (IMC; veja o Exercício 2.33). Escreva um aplicativo Java que solicite as informações da pessoa, instancie um objeto da classe HealthProfile para ela e imprima as informações a partir desse objeto — incluindo nome, sobrenome, sexo, data de nascimento, altura e peso da pessoa ––, e então calcule e imprima a idade em anos, IMC, intervalo de frequência cardíaca máxima e frequência cardíaca alvo. Ele também deve exibir o gráfico de valores IMC do Exercício 2.33.