Summary

Section 2.2 Your First Program in Java: Printing a Line of Text

- A Java application (p. 35) executes when you use the `java` command to launch the JVM.
- Comments (p. 36) document programs and improve their readability. The compiler ignores them.
- A comment that begins with `//` is an end-of-line comment—it terminates at the end of the line on which it appears.
- Traditional comments (p. 36) can be spread over several lines and are delimited by `/*` and `*/`.
- Javadoc comments (p. 36), delimited by `/**` and `*/`, enable you to embed program documentation in your code. The `javadoc` utility program generates HTML pages based on these comments.
- A syntax error (p. 36; also called a compiler error, compile-time error or compilation error) occurs when the compiler encounters code that violates Java's language rules. It's similar to a grammar error in a natural language.
- Blank lines, space characters and tab characters are known as white space (p. 37). White space makes programs easier to read and is ignored by the compiler.
- Keywords (p. 37) are reserved for use by Java and are always spelled with all lowercase letters.
- Keyword `class` (p. 37) introduces a class declaration.
- By convention, all class names in Java begin with a capital letter and capitalize the first letter of each word they include (e.g., `SampleClassName`).
- A Java class name is an identifier—a series of characters consisting of letters, digits, underscores (_) and dollar signs ($) that does not begin with a digit and does not contain spaces.
- Java is case sensitive (p. 38)—that is, uppercase and lowercase letters are distinct.
- The body of every class declaration (p. 38) is delimited by braces, `{` and `}`.
- A `public` (p. 37) class declaration must be saved in a file with the same name as the class followed by the ".`java`" file-name extension.
- Method `main` (p. 38) is the starting point of every Java application and must begin with

```
public static void main(String[] args)
```

  otherwise, the JVM will not execute the application.
- Methods perform tasks and return information when they complete them. Keyword `void` (p. 38) indicates that a method will perform a task but return no information.
- Statements instruct the computer to perform actions.
- A string (p. 39) in double quotes is sometimes called a character string or a string literal.
- The standard output object (`System.out`; p. 39) displays characters in the command window.
- Method `System.out.println` (p. 39) displays its argument (p. 39) in the command window followed by a newline character to position the output cursor to the beginning of the next line.
- You compile a program with the command `javac`. If the program contains no syntax errors, a class file (p. 40) containing the Java bytecodes that represent the application is created. These bytecodes are interpreted by the JVM when you execute the program.
- To run an application, type `java` followed by the name of the class that contains the `main` method.

Section 2.3 Modifying Your First Java Program

- `System.out.print` (p. 42) displays its argument and positions the output cursor immediately after the last character displayed.
- A backslash (\) in a string is an escape character (p. 43). Java combines it with the next character to form an escape sequence (p. 43). The escape sequence `\n` (p. 43) represents the newline character.

Section 2.4 Displaying Text with `printf`

- `System.out.printf` method (p. 43; `f` means "formatted") displays formatted data.
- Method `printf`'s first argument is a format string (p. 44) containing fixed text and/or format specifiers. Each format specifier (p. 44) indicates the type of data to output and is a placeholder for a corresponding argument that appears after the format string.
- Format specifiers begin with a percent sign (`%`) and are followed by a character that represents the data type. The format specifier `%s` (p. 44) is a placeholder for a string.
- The `%n` format specifier (p. 44) is a portable line separator. You cannot use `%n` in the argument to `System.out.print` or `System.out.println`; however, the line separator output by `System.out.println` after it displays its argument is portable across operating systems.

Section 2.5 Another Application: Adding Integers

- An `import` declaration (p. 46) helps the compiler locate a class that's used in a program.
- Java's rich set of predefined classes are grouped into packages (p. 45)—named groups of classes. These are referred to as the Java class library (p. 46), or the Java Application Programming Interface (Java API).
- A variable (p. 46) is a location in the computer's memory where a value can be stored for use later in a program. All variables must be declared with a name and a type before they can be used.
- A variable's name enables the program to access the variable's value in memory.
- A `Scanner` (package `java.util`; p. 47) enables a program to read data that the program will use. Before a `Scanner` can be used, the program must create it and specify the source of the data.
- Variables should be initialized (p. 47) to prepare them for use in a program.
- The expression `new Scanner(System.in)` creates a `Scanner` that reads from the standard input object (`System.in`; p. 47)—normally the keyboard.
- Data type `int` (p. 47) is used to declare variables that will hold integer values. The range of values for an `int` is –2,147,483,648 to +2,147,483,647.
- Types `float` and `double` (p. 47) specify real numbers with decimal points, such as `3.4` and `-11.19`.
- Variables of type `char` (p. 47) represent individual characters, such as an uppercase letter (e.g., `A`), a digit (e.g., `7`), a special character (e.g., `*` or `%`) or an escape sequence (e.g., tab, `\t`).
- Types such as `int`, `float`, `double` and `char` are primitive types (p. 47). Primitive-type names are keywords; thus, they must appear in all lowercase letters.
- A prompt (p. 48) directs the user to take a specific action.
- `Scanner` method `nextInt` obtains an integer for use in a program.
- The assignment operator, `=` (p. 48), enables the program to give a value to a variable. It's called a binary operator (p. 48) because it has two operands.
- Portions of statements that have values are called expressions (p. 49).
- The format specifier `%d` (p. 49) is a placeholder for an `int` value.

Section 2.6 Memory Concepts

- Variable names (p. 50) correspond to locations in the computer's memory. Every variable has a name, a type, a size and a value.
- A value that's placed in a memory location replaces the location's previous value, which is lost.

Section 2.7 Arithmetic

- The arithmetic operators (p. 51) are `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division) and `%` (remainder).

- Integer division (p. 51) yields an integer quotient.
- The remainder operator, `%` (p. 51), yields the remainder after division.
- Arithmetic expressions must be written in straight-line form (p. 51).
- If an expression contains nested parentheses (p. 52), the innermost set is evaluated first.
- Java applies the operators in arithmetic expressions in a precise sequence determined by the rules of operator precedence (p. 52).
- When we say that operators are applied from left to right, we're referring to their associativity (p. 52). Some operators associate from right to left.
- Redundant parentheses (p. 53) can make an expression clearer.

Section 2.8 Decision Making: Equality and Relational Operators

- The `if` statement (p. 54) makes a decision based on a condition's value (true or false).
- Conditions in `if` statements can be formed by using the equality (`==` and `!=`) and relational (`>`, `<`, `>=` and `<=`) operators (p. 54).
- An `if` statement begins with keyword `if` followed by a condition in parentheses and expects one statement in its body.
- The empty statement (p. 57) is a statement that does not perform a task.

Self-Review Exercises

2.1 Fill in the blanks in each of the following statements:

a) A(n) _____ begins the body of every method, and a(n) _____ ends the body of every method.
b) You can use the _____ statement to make decisions.
c) _____ begins an end-of-line comment.
d) _____, _____ and _____ are called white space.
e) _____ are reserved for use by Java.
f) Java applications begin execution at method _____.
g) Methods _____, _____ and _____ display information in a command window.

2.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) Comments cause the computer to print the text after the `//` on the screen when the program executes.
b) All variables must be given a type when they're declared.
c) Java considers the variables `number` and `NuMbEr` to be identical.
d) The remainder operator (`%`) can be used only with integer operands.
e) The arithmetic operators `*`, `/`, `%`, `+` and `-` all have the same level of precedence.

2.3 Write statements to accomplish each of the following tasks:

a) Declare variables `c`, `thisIsAVariable`, `q76354` and `number` to be of type `int`.
b) Prompt the user to enter an integer.
c) Input an integer and assign the result to `int` variable `value`. Assume `Scanner` variable `input` can be used to read a value from the keyboard.
d) Print `"This is a Java program"` on one line in the command window. Use method `System.out.println`.
e) Print `"This is a Java program"` on two lines in the command window. The first line should end with `Java`. Use method `System.out.printf` and two `%s` format specifiers.
f) If the variable `number` is not equal to `7`, display `"The variable number is not equal to 7"`.

2.4 Identify and correct the errors in each of the following statements:

a)
```
if (c < 7);
   System.out.println("c is less than 7");
```

b)
```
if (c => 7)
   System.out.println("c is equal to or greater than 7");
```

2.5 Write declarations, statements or comments that accomplish each of the following tasks:

a) State that a program will calculate the product of three integers.
b) Create a `Scanner` called `input` that reads values from the standard input.
c) Declare the variables `x`, `y`, `z` and `result` to be of type `int`.
d) Prompt the user to enter the first integer.
e) Read the first integer from the user and store it in the variable `x`.
f) Prompt the user to enter the second integer.
g) Read the second integer from the user and store it in the variable `y`.
h) Prompt the user to enter the third integer.
i) Read the third integer from the user and store it in the variable `z`.
j) Compute the product of the three integers contained in variables `x`, `y` and `z`, and assign the result to the variable `result`.
k) Use `System.out.printf` to display the message `"Product is"` followed by the value of the variable `result`.

2.6 Using the statements you wrote in Exercise 2.5, write a complete program that calculates and prints the product of three integers.

Answers to Self-Review Exercises

2.1 a) left brace (`{`), right brace (`}`). b) `if`. c) `//`. d) Space characters, newlines and tabs. e) Keywords. f) `main`. g) `System.out.print`, `System.out.println` and `System.out.printf`.

2.2 a) False. Comments do not cause any action to be performed when the program executes. They're used to document programs and improve their readability.
b) True.
c) False. Java is case sensitive, so these variables are distinct.
d) False. The remainder operator can also be used with noninteger operands in Java.
e) False. The operators `*`, `/` and `%` are higher precedence than operators `+` and `-`.

2.3 a)
```
int c, thisIsAVariable, q76354, number;
```
or
```
int c;
int thisIsAVariable;
int q76354;
int number;
```

b)
```
System.out.print("Enter an integer: ");
```

c)
```
value = input.nextInt();
```

d)
```
System.out.println("This is a Java program");
```

e)
```
System.out.printf("%s%n%s%n", "This is a Java", "program");
```

f)
```
if (number != 7)
   System.out.println("The variable number is not equal to 7");
```

2.4 a) Error: Semicolon after the right parenthesis of the condition `(c < 7)` in the `if`. Correction: Remove the semicolon after the right parenthesis. [*Note:* As a result, the output statement will execute regardless of whether the condition in the `if` is true.]
b) Error: The relational operator `=>` is incorrect. Correction: Change `=>` to `>=`.

2.5 a)
```
// Calculate the product of three integers
```
b)
```
Scanner input = new Scanner(System.in);
```
c)
```
int x, y, z, result;
```
or
```
int x;
int y;
int z;
int result;
```
d)
```
System.out.print("Enter first integer: ");
```
e)
```
x = input.nextInt();
```
f)
```
System.out.print("Enter second integer: ");
```
g)
```
y = input.nextInt();
```
h)
```
System.out.print("Enter third integer: ");
```
i)
```
z = input.nextInt();
```
j)
```
result = x * y * z;
```
k)
```
System.out.printf("Product is %d%n", result);
```

2.6 The solution to Self-Review Exercise 2.6 is as follows:

```
1  // Ex. 2.6: Product.java
2  // Calculate the product of three integers.
3  import java.util.Scanner; // program uses Scanner
4
5  public class Product
6  {
7     public static void main(String[] args)
8     {
9        // create Scanner to obtain input from command window
10       Scanner input = new Scanner(System.in);
11
12       int x; // first number input by user
13       int y; // second number input by user
14       int z; // third number input by user
15       int result; // product of numbers
16
17       System.out.print("Enter first integer: "); // prompt for input
18       x = input.nextInt(); // read first integer
19
20       System.out.print("Enter second integer: "); // prompt for input
21       y = input.nextInt(); // read second integer
22
23       System.out.print("Enter third integer: "); // prompt for input
24       z = input.nextInt(); // read third integer
25
26       result = x * y * z; // calculate product of numbers
27
28       System.out.printf("Product is %d%n", result);
29    } // end method main
30 } // end class Product
```

```
Enter first integer: 10
Enter second integer: 20
Enter third integer: 30
Product is 6000
```

Exercises

2.7 Fill in the blanks in each of the following statements:

a) _____ are used to document a program and improve its readability.
b) A decision can be made in a Java program with a(n) _____.
c) Calculations are normally performed by _____ statements.
d) The arithmetic operators with the same precedence as multiplication are _____ and _____.
e) When parentheses in an arithmetic expression are nested, the _____ set of parentheses is evaluated first.
f) A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a(n) _____.

2.8 Write Java statements that accomplish each of the following tasks:

a) Display the message `"Enter an integer: "`, leaving the cursor on the same line.
b) Assign the product of variables `b` and `c` to variable `a`.
c) Use a comment to state that a program performs a sample payroll calculation.

2.9 State whether each of the following is *true* or *false*. If *false*, explain why.

a) Java operators are evaluated from left to right.
b) The following are all valid variable names: `_under_bar_`, `m928134`, `t5`, `j7`, `her_sales$`, `his_$account_total`, `a`, `b$`, `c`, `z` and `z2`.
c) A valid Java arithmetic expression with no parentheses is evaluated from left to right.
d) The following are all invalid variable names: `3g`, `87`, `67h2`, `h22` and `2h`.

2.10 Assuming that `x = 2` and `y = 3`, what does each of the following statements display?

a) `System.out.printf("x = %d%n", x);`
b) `System.out.printf("Value of %d + %d is %d%n", x, x, (x + x));`
c) `System.out.printf("x =");`
d) `System.out.printf("%d = %d%n", (x + y), (y + x));`

2.11 Which of the following Java statements contain variables whose values are modified?

a) `p = i + j + k + 7;`
b) `System.out.println("variables whose values are modified");`
c) `System.out.println("a = 5");`
d) `value = input.nextInt();`

2.12 Given that $y = ax^3 + 7$, which of the following are correct Java statements for this equation?

a) `y = a * x * x * x + 7;`
b) `y = a * x * x * (x + 7);`
c) `y = (a * x) * x * (x + 7);`
d) `y = (a * x) * x * x + 7;`
e) `y = a * (x * x * x) + 7;`
f) `y = a * x * (x * x + 7);`

2.13 State the order of evaluation of the operators in each of the following Java statements, and show the value of `x` after each statement is performed:

a) `x = 7 + 3 * 6 / 2 - 1;`
b) `x = 2 % 2 + 2 * 2 - 2 / 2;`
c) `x = (3 * 9 * (3 + (9 * 3 / (3))));`

2.14 Write an application that displays the numbers 1 to 4 on the same line, with each pair of adjacent numbers separated by one space. Use the following techniques:

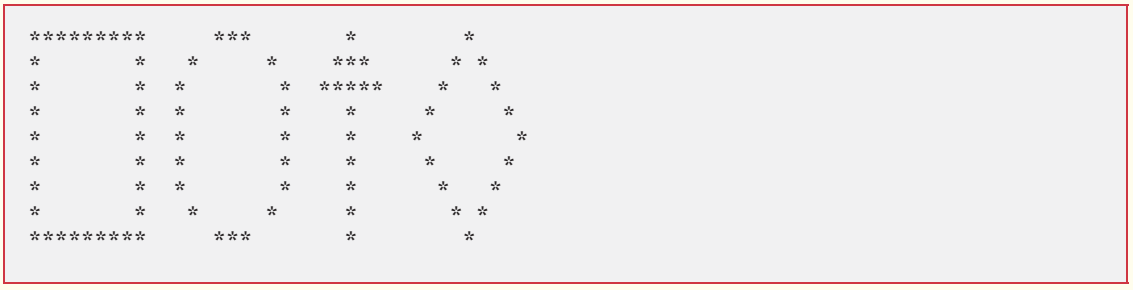
a) Use one `System.out.println` statement.
b) Use four `System.out.print` statements.
c) Use one `System.out.printf` statement.

2.15 *(Arithmetic)* Write an application that asks the user to enter two integers, obtains them from the user and prints their sum, product, difference and quotient (division). Use the techniques shown in Fig. 2.7.

2.16 *(Comparing Integers)* Write an application that asks the user to enter two integers, obtains them from the user and displays the larger number followed by the words `"is larger"`. If the numbers are equal, print the message `"These numbers are equal"`. Use the techniques shown in Fig. 2.15.

2.17 *(Arithmetic, Smallest and Largest)* Write an application that inputs three integers from the user and displays the sum, average, product, smallest and largest of the numbers. Use the techniques shown in Fig. 2.15. [*Note:* The calculation of the average in this exercise should result in an integer representation of the average. So, if the sum of the values is 7, the average should be 2, not 2.3333….]

2.18 *(Displaying Shapes with Asterisks)* Write an application that displays a box, an oval, an arrow and a diamond using asterisks (*), as follows:



2.19 What does the following code print?

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

2.20 What does the following code print?

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

2.21 What does the following code print?

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

2.22 What does the following code print?

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

2.23 What does the following code print?

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

2.24 *(Largest and Smallest Integers)* Write an application that reads five integers and determines and prints the largest and smallest integers in the group. Use only the programming techniques you learned in this chapter.

2.25 *(Odd or Even)* Write an application that reads an integer and determines and prints whether it's odd or even. [*Hint:* Use the remainder operator. An even number is a multiple of 2. Any multiple of 2 leaves a remainder of 0 when divided by 2.]

2.26 *(Multiples)* Write an application that reads two integers, determines whether the first is a multiple of the second and prints the result. [*Hint:* Use the remainder operator.]

2.27 *(Checkerboard Pattern of Asterisks)* Write an application that displays a checkerboard pattern, as follows:

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

2.28 *(Diameter, Circumference and Area of a Circle)* Here's a peek ahead. In this chapter, you learned about integers and the type `int`. Java can also represent floating-point numbers that contain decimal points, such as 3.14159. Write an application that inputs from the user the radius of a circle as an integer and prints the circle's diameter, circumference and area using the floating-point value 3.14159 for π. Use the techniques shown in Fig. 2.7. [*Note:* You may also use the predefined constant `Math.PI` for the value of π. This constant is more precise than the value 3.14159. Class `Math` is defined in package `java.lang`. Classes in that package are imported automatically, so you do not need to `import` class `Math` to use it.] Use the following formulas (*r* is the radius):

diameter = 2*r*
circumference = 2π*r*
area = πr^2

Do not store the results of each calculation in a variable. Rather, specify each calculation as the value that will be output in a `System.out.printf` statement. The values produced by the circumference and area calculations are floating-point numbers. Such values can be output with the format specifier `%f` in a `System.out.printf` statement. You'll learn more about floating-point numbers in Chapter 3.

2.29 *(Integer Value of a Character)* Here's another peek ahead. In this chapter, you learned about integers and the type `int`. Java can also represent uppercase letters, lowercase letters and a considerable variety of special symbols. Every character has a corresponding integer representation. The set of characters a computer uses together with the corresponding integer representations for those characters is called that computer's character set. You can indicate a character value in a program simply by enclosing that character in single quotes, as in `'A'`.

You can determine a character's integer equivalent by preceding that character with `(int)`, as in

```
(int) 'A'
```

An operator of this form is called a cast operator. (You'll learn about cast operators in Chapter 4.) The following statement outputs a character and its integer equivalent:

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

When the preceding statement executes, it displays the character `A` and the value `65` (from the Unicode® character set) as part of the string. The format specifier `%c` is a placeholder for a character (in this case, the character `'A'`).

Using statements similar to the one shown earlier in this exercise, write an application that displays the integer equivalents of some uppercase letters, lowercase letters, digits and special symbols. Display the integer equivalents of the following: `A B C a b c 0 1 2 $ * + /` and the blank character.

2.30 *(Separating the Digits in an Integer)* Write an application that inputs one number consisting of five digits from the user, separates the number into its individual digits and prints the digits separated from one another by three spaces each. For example, if the user types in the number `42339`, the program should print

```
4   2   3   3   9
```

Assume that the user enters the correct number of digits. What happens when you enter a number with more than five digits? What happens when you enter a number with fewer than five digits? [*Hint:* It's possible to do this exercise with the techniques you learned in this chapter. You'll need to use both division and remainder operations to "pick off" each digit.]

2.31 *(Table of Squares and Cubes)* Using only the programming techniques you learned in this chapter, write an application that calculates the squares and cubes of the numbers from 0 to 10 and prints the resulting values in table format, as shown below.

```
number  square  cube
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
```

2.32 *(Negative, Positive and Zero Values)* Write a program that inputs five numbers and determines and prints the number of negative numbers input, the number of positive numbers input and the number of zeros input.

Making a Difference

2.33 *(Body Mass Index Calculator)* We introduced the body mass index (BMI) calculator in Exercise 1.10. The formulas for calculating BMI are

$$BMI = \frac{weightInPounds \times 703}{heightInInches \times heightInInches}$$

or

$$BMI = \frac{weightInKilograms}{heightInMeters \times heightInMeters}$$

Create a BMI calculator that reads the user's weight in pounds and height in inches (or, if you prefer, the user's weight in kilograms and height in meters), then calculates and displays the user's body mass index. Also, display the following information from the Department of Health and Human Services/National Institutes of Health so the user can evaluate his/her BMI:

```
BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overweight:  between 25 and 29.9
Obese:       30 or greater
```

[*Note:* In this chapter, you learned to use the `int` type to represent whole numbers. The BMI calculations when done with `int` values will both produce whole-number results. In Chapter 3 you'll learn to use the `double` type to represent numbers with decimal points. When the BMI calculations are performed with `double`s, they'll both produce numbers with decimal points—these are called "floating-point" numbers.]

2.34 ***(World Population Growth Calculator)*** Use the web to determine the current world population and the annual world population growth rate. Write an application that inputs these values, then displays the estimated world population after one, two, three, four and five years.

2.35 ***(Car-Pool Savings Calculator)*** Research several car-pooling websites. Create an application that calculates your daily driving cost, so that you can estimate how much money could be saved by car pooling, which also has other advantages such as reducing carbon emissions and reducing traffic congestion. The application should input the following information and display the user's cost per day of driving to work:

a) Total miles driven per day.
b) Cost per gallon of gasoline.
c) Average miles per gallon.
d) Parking fees per day.
e) Tolls per day.