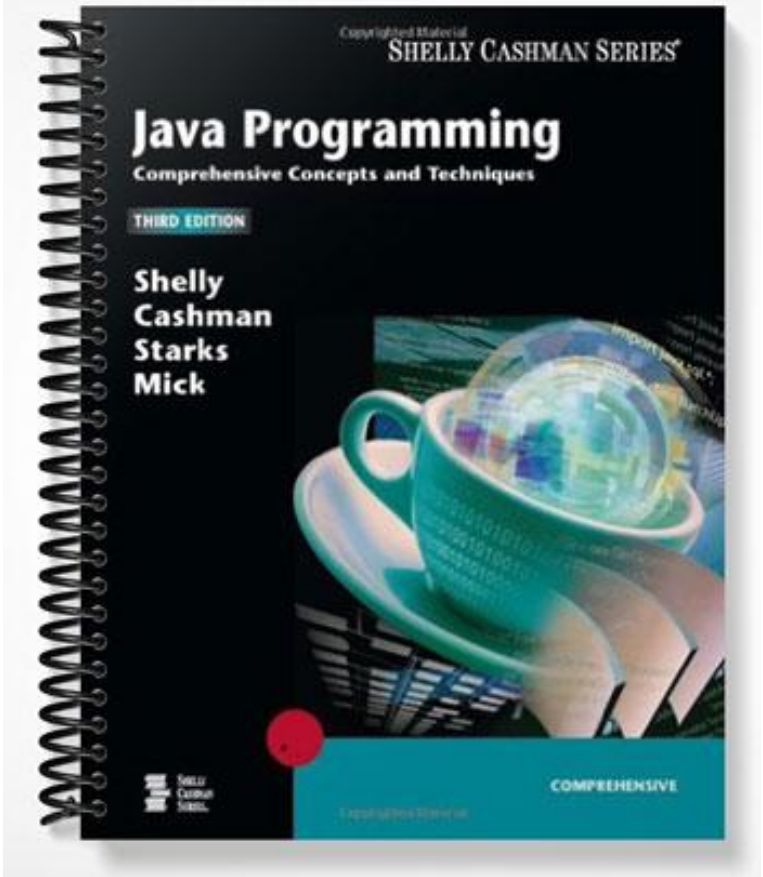
SOLUTIONS MANUAL



Java Programming, Third Edition

Instructor's Manual

CHAPTER TWO

CREATING A JAVA APPLICATION AND APPLET

OBJECTIVES

Students will have mastered the material in this chapter when they can:

- Write a simple Java application
- Use TextPad
- Understand the different types and uses of comments
- Use proper naming conventions for classes and files
- Identify the parts of a class header and method header
- Code output
- Use the println() method
- Compile a Java program
- Understand the common types of errors
- Run a Java program
- Edit Java source code to insert escape characters and a system date
- Print source code
- Differentiate between an application and an applet
- Create an applet from Java source code
- Write code to display a graphic, text, color, and the date in an applet
- Create an HTML host document
- Run a Java applet

CHAPTER OVERVIEW

In this chapter, students learn the basic form of a Java application and an applet. They learn how to use TextPad to enter comments as documentation, enter a class header and a method header, and use the println() method to display output in a console application. The println() method uses character strings, escape characters, and concatenated data to format output. J2SE 5.0 includes the printf() method that can embed data using a percent sign and a formatting character. After learning how to compile the source code and debug any errors, they learn how to run the Java application. Next, they learn how to edit source code in the TextPad window using the import statement to import packages, a Date constructor, and escape characters to format output. They learn how to edit existing source code to convert the application into an applet that can run on the Web. They learn how to import applet packages, change a class name, and extend the Applet class, and how to use the paint, drawString, and getImage methods to complete the applet code, Finally, they create an HTML host document to display the applet and run it using Applet Viewer.

INSTRUCTOR NOTES

Introduction, 46

LECTURE NOTES

Define **user interface** and review that the Java platform allows the programmer to create programs with different types of user interfaces. Mention that in this chapter, the students will create a user interface for a console application (a command-line interface) and an applet (graphical user interface displayed in a browser). Point out that detailed instructions will guide the student through the coding, compiling, and execution process.

DISCUSSION TOPICS

Discuss the uses of consoled applications (networking and server applications, testing, system configuration, mobile devices). Why are console applications better than GUIs for these applications (speed, display space)? Will the use of consoled applications diminish as GUIs become more sophisticated?

QUICK QUIZZES

What is a user interface? (Answer: the way in which a user enters data into and receives feedback from a computer)

TROUBLESHOOTING TIPS

Ask the class about any problems they encountered while installing the J2SE or TextPad. Encourage the students to share solutions to any issues.

Chapter Two — The Welcome to My Day Program, 46

LECTURE NOTES

Define **splash screen** and point out that in this chapter, students will develop a program called The Welcome to My Day program, which displays a welcome message, the user's name, and the system date on a splash screen. Review that the splash screen will be the first step towards creating an electronic calendar application, which will be available both as a console application and as an applet. Use Figure 2-1a to show what the welcome message will look like as a console application, and Figure 2-1b to show how the applet will display on the screen.

CLASSROOM ACTIVITIES

Ask the students to brainstorm some reasons to incorporate a splash screen in an application. Ask the students to point out the similarities and differences between the appearance of the console application and the applet display.

Program Development, 47

LECTURE NOTES

Use Table 2-1 to review the six phases of the development cycle and the tasks related to each phase.

CLASSROOM ACTIVITIES

Give each of six students a sign on which is written one of the phases of the development cycle from Table 2-1. Ask these six students to stand in different places in the room. Prepare slips of

paper with the tasks from column three of Table 2-1, and pass these out to the other students. Ask each student with a task slip to find the phase to which he or she belongs. Suggest that the students may use the textbook as necessary.

Analysis and Design**, **48
LECTURE NOTES
Remind the students that the first two phases in the development cycle involve analyzing the requirements document and then designing a solution. Call attention to the requirements document in Figure 2-2.

CLASSROOM ACTIVITIES
Ask the students to analyze the requirements document in Figure 2-2 and decide which design elements are specific in the requirements document and which are flexible.

QUICK QUIZZES
What are the six sections of the requirements document in Figure 2-2? (Answer: Date submitted, Submitted by, Purpose, Application title, Algorithms, and Notes)

Problem Analysis, 48
LECTURE NOTES
Define **prototype**. Point out that the purpose of The Welcome to My Day program is to build a prototype welcome splash screen. Review the two purposes of a splash screen. Review the text to be displayed. Define **system date**. Point out the additional requirements for the applet version of the program.

QUICK QUIZZES
What text is to be displayed by the program? (Answer: a welcome message, the user's name, and the system date)
What is the system date? (Answer: the current date and time generated by the operating system of a computer)

Design the Solution, 49
LECTURE NOTES
Use Figure 2-3a to illustrate a hand-drawn storyboard for the console version of the program. Use Figure 2-3b to illustrate a storyboard for the applet version of the program. Point out the margins, and remind the students that design experts recommend that top and left margins improve a display. Define **default**. Mention the requirement concerning closing the splash screen, and remind the students that they can rely on the default close features to satisfy this requirement. Point out the Tip on page 51.

DISCUSSION TOPICS
Point out the system date display in Figures 2-1a and 2-1b. Ask the students to decide if the system date should be displayed in a less cryptic form, and to defend their stand.

Program Design, 51
LECTURE NOTES

Remind the students that when the storyboards have been drawn and the details of the design have been worked out, the next step is to design the program logic. Point out the flowchart in Figure 2-4a and the event diagram in Figure 2-4b, and discuss how they illustrate the program logic.

DISCUSSION TOPICS
Ask the students to decide which is better for designing the logic of this program, a flowchart or an event diagram. Ask if using both is better, or if it is redundant. Ask them to defend their answers.

CLASSROOM ACTIVITIES
Divide the class into groups and ask each group to design a more detailed flowchart than that of Figure 2-4a. Remind them to use the requirements document as a guide.

PROJECTS TO ASSIGN
Ask the students to design their own interface based on the requirements document in Figure 2-2. Students must adhere to the requirements document, but encourage them to be creative with the flexible design elements.

Using TextPad, 52
LECTURE NOTES
Define **TextPad**. Remind the students of what a VATE is (from Chapter 1). Mention some of the advantages of using TextPad over using other text-editing programs.

QUICK QUIZZES
What is a VATE? (Answer: a value-added text editor)
What are some of the advantages of using TextPad? (Answer: TextPad displays line numbers and color coding, and contains many programmer-friendly tools, including commands to compile and execute both applications and applets.)

Starting TextPad, 52
LECTURE NOTES
Review Steps 1 and 2 on pages 53 and 54 to start TextPad, using Figures 2-5 and 2-6 to illustrate. Point out Other Ways to start TextPad.

TROUBLESHOOTING TIPS
Remind the students that their All Programs submenu may not look exactly like the one in Figure 2-5, but that the important thing is that they find the TextPad text and icon.
Mention the possibility that a Tip of the Day dialog box may display and that they should close it using its Close button.
Guide the students to Appendix C, and suggest that their lives will be easier if they change the settings on their TextPad to match the ones in the textbook.

The TextPad Window, 54
LECTURE NOTES

Using Figure 2-6, point out the **coding window**, the **Selector window**, and the **Clip Library window**, and explain the uses of each one. Review how Java programmers sometimes customize the display of the TextPad window.

CLASSROOM ACTIVITIES
Look at the characters in the Clip Library and discuss cases in which they would be applicable. Students new to programming will find a discussion of the different groups of characters helpful.

QUICK QUIZZES
How can you customize the display of the TextPad window? (Answer: drag, move, or close one or more of the windows)

PROJECTS TO ASSIGN
TextPad users frequently contribute additional functionality to TextPad through added utilities and libraries. Ask students to look for other utilities and clip libraries that would be helpful for Java programming at the TextPad Web site (textpad.com).

Displaying Line Numbers in the TextPad Window, 55
LECTURE NOTES
Define **line numbers**. Mention the advantages of displaying line numbers as you write code. Review Steps 1 and 2 to display line numbers in the TextPad window, using Figures 2-7 and 2-8 to illustrate. Mention Other Ways to display line numbers. Review how to make displaying line numbers the default setting.

TROUBLESHOOTING TIPS
Emphasize the importance of having line numbers for finding compiler errors in even small programs. This may be the most valuable function of a VATE! Help those students who have trouble making TextPad default to showing line numbers.

QUICK QUIZZES
What are the advantages of inserting line numbers in the source code? (Answer: to keep track of lines while entering code; to have a line number reference for possible errors in compilation)

Saving a TextPad Document, 56
LECTURE NOTES
Mention the reasons for saving a TextPad document before you begin coding. Review Steps 1 through 9 on pages 57 through 60 to save a TextPad document, using Figures 2-9 through 2-16 to illustrate. Mention Other Ways to save a TextPad document. Point out that the Save button on the Standard toolbar is enabled only when changes have been made to a TextPad document, and that when it is enabled, it is the most efficient way to save a document.

DISCUSSION TOPICS
Ask the students to discuss how often during the coding process a programmer should save the document. Ask them to defend their answers. Warn them that you will remind them of this discussion later in the semester!

QUICK QUIZZES
Why should a programmer save a document before beginning the coding process? (Answer: to identify the document as one that contains Java source code, thus enabling TextPad's color-coding capabilities related to Java; to display the document name on the title bar; to make subsequent saves more efficient; and to prevent lost data)

Coding the Program, 61
LECTURE NOTES
Review the task to be performed: display a welcome message, the user's name, and the system date on the screen.

CLASSROOM ACTIVITIES
If it is possible during the classroom time, keep TextPad open and code the Welcome to My Day program dynamically, displaying it to the class, while illustrating the material in this section.

Coding Comments as Documentation*, *61
LECTURE NOTES
Define **comments**. Define **block comment**, **doc comment**, and **line comment**. Use Table 2-2 to show the syntax for each type of comment, and the placement of each type. Point out the Tip on page 61, and define **comment header**. Emphasize the importance of comments in programming. Refer to Figure 2-17 to show the comment header that the students are to key in. Define **tab characters**, and explain where the students should use tabs in lines 1 through 9 of their programs. Review Step 1 on page 63 to code the comments, using Figure 2-18 to show how it looks when it has been typed in. Mention the Tip on page 64.

CLASSROOM ACTIVITIES
Ask the students to brainstorm all the reasons they can think of to add comments to a program. Ask them to mention places where comments should be added (at the beginning of a program, at the start of every major section of a program, to describe the intended meaning of a single line, anywhere that the code might be obscure or hard to understand, any others that they might name).

DISCUSSION TOPICS
Ask the students to choose one of the following statements, and be ready to defend their choice:
Too many comments are just as bad as too few comments.
You can never have too many comments.

QUICK QUIZZES
What are the three types of comments allowed in a Java program? What is the proper syntax for each type? (Answer: /* block*/, /** doc*/, and // line)
How do you enter a tab character? (Answer: by pressing the TAB key)
What color does TextPad display comments in? (Answer: green)

PROJECTS TO ASSIGN

Suggest that the students might want to design a template comment header for the top of a file that can be used for their programs in this class. This comment header might contain information that you as the instructor require.

TROUBLESHOOTING TIPS
Some students may be surprised at the automatic indent provided by TextPad; explain that this is one of the programmer helps provided by TextPad and other VATEs.
Make certain the students understand that they may need TWO tabs to make the columns line up.
Point out that the students must manually enter a line break in line 6, to make the purpose words display on lines 6 and 7 (that is, the text is not going to line wrap automatically).
Some students may have trouble de-indenting the symbols on line 8; suggest the use of the BACKSPACE key.

The Class Header*, *64
LECTURE NOTES
Define **class header** and explain what it does. Define **public** and **access modifier**. Review that an access modifier, also called a scope identifier, tells which objects can access the class. Point out the word class and the class name, Welcome, in the class header. Define **keywords**, or **reserved words**. Call attention to Table 2-3, and suggest that the students become familiar with the Java keywords. Review the Java class naming rules in Table 2-4. Point out that by convention, class names are uppercase and words within the class name are distinguished by an uppercase letter, although this is not mandatory. Define **case-sensitive**, and mention that Java is case-sensitive. Point out that the class name and the name of the Java source file should match exactly. Define **braces**, show where the braces are on the keyboard, and mention that all code entered after the class header is the body of the class and must be enclosed in braces {}. Review that in Java, code between braces forms a single, logical unit in the program. Review Step 1 on page 66 to code the class header, and remind the students that what they type in should look like the code in Figure 2-20.

CLASSROOM ACTIVITIES
Divide the class into groups. Ask each group to refer to Table 2-4, and to come up with 30 legal class names. Then, ask them to come up with 30 illegal class names, 10 for each of the rows of Table 2-4.

DISCUSSION TOPICS
Discuss the importance of "good programming practice" for writing code (readability of code). Often, the Java compiler will allow erratic spacing and use of case in the code, but common convention dictates a narrower style. For example, what are the conventions, as opposed to the requirements, for class names in the class header? What is the convention for object names or data items? What is the convention for the placement of braces? In the case of braces, the convention varies, so encourage the students to be consistent throughout the code with whatever style they choose. Finally, what is the convention for indentation of code?

QUICK QUIZZES
What does the access modifier, public, mean? (Answer: indicates that this code can be accessed by all objects in the program)

What does an access modifier do? (Answer: specifies the circumstances in which the class can be accessed)
What is a Java keyword? (Answer: a word with a special meaning, which is reserved for the use of the compiler and cannot be used as a class name)
How many Java keywords are there? (Answer: 61, as listed in Table 2-3)

The Method Header*, *67
LECTURE NOTES
Define **method header**, and mention that the method header is a message to the Java compiler, giving it information about the method. Use Table 2-5 to point out the general form and the purpose of a method header, and to show some examples of method headers. Define **main() method**, and review that every stand-alone Java application MUST have a main() method. Use Figure 2-21 to show the method header of the main() method in the program under discussion. Define **method modifier**. Point out that a method may have more than one modifier, as in the case of this main() method, which is both public and static. Define **static**. Review that a typical method header has three parts after the modifiers. Define **return value**. Define the keyword, **void**, and point out that since this main() method does not return a value, the keyword, void, is used instead of a data type. Define **parameter** and **argument.** Define **identifier** and **variable**. Point out that the same spelling restrictions that apply to class names also apply to variable names (identifiers). Define **data type** and **String[]**, and promise that the students will learn more about data types later. Define **body**, and explain that the body of a method contains the executable code. Review Step 1 on page 69 to code the method header, using Figure 2-22 to show how it should look. Mention the Tip on page 69.

CLASSROOM ACTIVITIES
The method header **public static void main(String[] args)** is the standard header for the method that starts the execution of a stand-alone Java program. Is the name of the parameter (args) a convention or a requirement? Discuss the difference between convention and requirement coding practices.

DISCUSSION TOPICS
Ask the students to discuss using a consistent style of indentation and placement of braces. Why is it important? Which is better, to put the braces on the line with code or on a line by itself? Discuss the practice of typing the close bracket on a line below the open bracket at the same time you type the open bracket. Is this a good idea? Why or why not?

QUICK QUIZZES
What are the three parts of a method header that typically follow the method modifiers? (Answer: the data type of the return value, the method name, and the parameter list)
What names can be used for the main() method in a Java program? (Answer: only the name main())
How can you recognize a method name in Java? (Answer: It is always followed by a set of parentheses.)
How would a Java programmer read this method header: "public static void main(String[] args)"? (Answer: This main method is public and static, accepts a String parameter named args, and returns void.)

PROJECTS TO ASSIGN
Ask the students to do some research to determine the relationship between an argument and a parameter.

TROUBLESHOOTING TIPS
This section contains several important topics including parameters, calling a method, and returning values from a method. Go slowly through the definitions of parameters, identifiers, variables, return value, and data types. Some students may not be familiar with the concept of passing data to and from methods.

Coding Output*, *70
LECTURE NOTES
Use Figure 2-23 to show the code that the students will enter to cause the program to display output. Define **class definition**, and explain the **System class**. Define **extends**, and explain how the System class extends, or inherits, methods from its superclass, Object. Mention the Tip on page 70. Define the word **out** in lines 14 through 18 of the program code. Define the **println() method**. Explain where the println() method comes from, and what it means to call a method. Point out the arguments of the method println() in lines 14 through 18 of the program code, and explain that println() with no arguments causes a blank line to be displayed. Define **literal**, and point out the literal arguments in lines 15, 16, and 17. Review Steps 1 through 4 on pages 71 and 72 to enter code to display output and save the program file, using Figures 2-24 through 2-26 to illustrate. Mention Other Ways to enter code and save a program file.

CLASSROOM ACTIVITIES
Discuss the difference between coding a method and calling a method.

DISCUSSION TOPICS
Discuss the elegance of coding small methods to serve a particular function, and then calling them from inside other methods. This encapsulates the code, making the program cleaner and easier to understand and debug. Ask the students to discuss what the optimal length of a method should be.

QUICK QUIZZES
What is the meaning of "ln" in println? (Answer: line)
What lines of Java code must end with a semicolon? (Answer: all lines except headers and lines containing only braces)

PROJECTS TO ASSIGN
Experiment with the literal arguments supplied to the System.out.println() method to see what happens to the output using various literals.

TROUBLESHOOTING TIPS
This section has many important details about syntax, such as the use of semicolons, periods, etc. Emphasize these points and remind the students that compiler errors frequently stem from small syntactic errors. Remind the students to use their own name in the appropriate literal.

Testing the Solution, 73
LECTURE NOTES
Recall that the fifth phase of the development cycle is to test the solution. Point out that with Java, testing involves (1) compiling the source code and then (2) executing the bytecode.

Compiling the Source Code, 73
LECTURE NOTES
Review that Java code must be compiled before it can be executed. Use Figure 2-27 to point out the Compile Java command on the Tools menu in TextPad. Describe the files that the compilation process creates, and tell where they are saved. Review Steps 1 and 2 to compile source code. Mention Other Ways to compile source code.

DISCUSSION TOPICS
Does one need the source file, executable files, or both to run a java program?

QUICK QUIZZES
What is the name of the compilation command in Java? (Answer: javac.exe)

TROUBLESHOOTING TIPS
The students should be familiar with compiling in both TextPad and the command prompt. Go through the steps for setting the classpath, changing directories, etc. to compile at the command prompt using the javac command.

Debugging the Solution, 74
LECTURE NOTES
Define **debugging**. Explain that if TextPad displays error messages when compilation is attempted, it will be necessary to find and correct the errors before proceeding.

TROUBLESHOOTING TIPS
This will undoubtedly be the most difficult part of programming for first-time programmers. Encourage your students to look first for the simple syntactic errors when fixing compiler errors, and remind them that fixing one error may in fact rid the program of several error messages.

System Errors, 74
LECTURE NOTES
Define **system error**, and tell when a system error usually occurs. Explain where system errors usually display. Use Figure 2-28 to explain one type of system error.

CLASSROOM ACTIVITIES
Ask the students if they have had any issues with compiling the sample program and getting system errors. Help them troubleshoot errors and encourage students to give tips from their own experience.

QUICK QUIZZES
When does a system error occur? (Answer: when a system command is not set properly, software is installed incorrectly, or the location of stored files has changed)

TROUBLESHOOTING TIPS
Most system errors can be corrected by setting up the system correctly. Refer students to Appendices B and D for explanations about setting up their systems.

Syntax Errors*, *75
LECTURE NOTES
Define **syntax error**. Define **Command Results window**, and use Figure 2-29 to show the Command Results window in TextPad. Use Table 2-6 to point out common syntax errors, the error messages they generate, and how to correct them.

CLASSROOM ACTIVITIES
If possible during the class period, omit one of the semicolons in the code for the main() method, attempt to compile the code, and point out what happens. Use Figure 2-29 to show the two syntax errors that are displayed because of a missing semicolon in line 15. Point out the line numbers in the error descriptions.

DISCUSSION TOPICS
Discuss with the students the best way to proceed when encountering a long list of errors. Why is it important to fix the error that occurs first?

QUICK QUIZZES
In which window does TextPad list syntax errors? (Answer: the Command Results window)
What are the most common mistakes that cause syntax errors? (Answer: mistakes in capitalization, mistakes in spelling, the use of incorrect special characters, and omission of correct punctuation)

TROUBLESHOOTING TIPS
Discuss other tips for debugging when fixing errors sequentially does not seem to do the trick. For example, sometimes commenting code out (placing comment marks on one or more lines) can help pinpoint errors.

Semantic Errors*, *76
LECTURE NOTES
Define **semantic error**, and explain what a semantic error looks like to the Java compiler. Use Figure 2-30 to show the semantic error caused by misspelling the println() method. Note that the misspelling in this case is to include a space before the parentheses.

CLASSROOM ACTIVITIES
If possible to display code during the class period, insert a spelling error in your code and compile to show the error to the class. Go through the error message to illustrate its components. Show how to display the code in the coding window by clicking the file name, and by double-clicking the first line of the error message. Show how to correct the error, and then re-compile.

DISCUSSION TOPICS
What is the difference between syntactic and semantic errors?

***Logic and Run-Time Errors**, 77*
LECTURE NOTES
Programs that compile successfully may still have errors that show up during the execution of the program. Define **logic error** and **run-time error**, also called an **exception**. Explain that errors may occur when the code cannot handle invalid input from the user. Point out that sometimes there are no error messages during execution, but the output is incorrect due to errors in the program.

CLASSROOM ACTIVITIES
If possible during the class period, show an example of a run-time exception by attempting to run the Welcome program with an incorrect filename.

DISCUSSION TOPICS
Catching invalid input from the user in a program is one of the most important issues in designing a program. The Welcome program does not take input and therefore is less prone to run-time errors. Discuss the scenario in which the user is required to enter a four-digit company password in order to continue in the program. What would the programmer need to include in the code, in general terms, for the program to be error-free during execution?

PROJECTS TO ASSIGN
Ask the students to look at the Exception class in the Java API and discuss in general terms the functionality of exceptions, such as catching invalid input.

QUICK QUIZZES
What is another name for a run-time error? (Answer: an exception)

Running the Application, 77
LECTURE NOTES
Explain that after a Java program is compiled into bytecode, and syntax and semantic errors are fixed, the program must be executed to test for logic and run-time errors.

DISCUSSION TOPICS
Discuss the relationship between compilation of a program and the testing phase of the program development cycle. Often, the programmer is relieved when the program compiles successfully, but the testing phase actually takes place after compilation is successful, using appropriate input data and checking the expected output.

***Running the Application**, 77*
LECTURE NOTES
Point out that an application can be executed either from the Run Java Application command on the TextPad Tools menu, or with the *java* command followed by the class filename at the command prompt. Review Steps 1 through 3 on page 78 to run the application, using Figures 2-31 and 2-32 to illustrate. Point out Other Ways to run the application. Mention the difference between the println() method and the print() method.

QUICK QUIZZES

When is the print() method, rather than the println(), method useful? (Answer: when you want the insertion point to display directly after a prompt to the user.)

PROJECTS TO ASSIGN
Ask students to use the Java API and language documentation on the Java Sun Microsystem Web site to research other methods supported by the System.out object.

TROUBLESHOOTING TIPS
The details of compiling and running code may be frustrating to the class if they are encountering problems. Make sure the students have a chance to share any issues they encounter with executing a program, as well as any solutions.

Editing the Source Code, 79
LECTURE NOTES
Define **edit**, and explain that whenever code is changed, it again must be saved, compiled and executed, and, if necessary, debugged. Review that the student's program needs to be edited to obtain the current date from the operating system and format and display the date according to the requirements documentation. Discuss Figure 2-33

PROJECTS TO ASSIGN
Suggest that the students investigate the editing functions available in TextPad, and refer them to Appendix C for more information.

Entering Code to Import Packages, 80
LECTURE NOTES
Define the **import statement**. Use Table 2-7 to show some of the more widely used Java packages that may be imported from the SDK. Explain that the import statement must be placed before the class header. Point out the meaning of an asterisk (*) after the package name. Mention the Tip on page 81. Review Steps 1 and 2 on pages 81 and 82 to code the import statement, using Figures 2-34 and 2-35 to illustrate.

DISCUSSION TOPICS
Ask the students to discuss the advantage of including just the individual class name in an import statement rather than the entire package? When is it better to include the whole package with the asterisk? Guide a discussion of where in the source code import statements be placed (i.e, whether they should come before or after the comment header).

QUICK QUIZZES
What is the only package that is imported automatically without an explicit command? (Answer: the java.lang package)
What is the meaning of an asterisk after a package name in the import statement? (Answer: The asterisk causes the program to load all the classes in the package that contains the named class.)
What are some of the classes that would be loaded along with the Date class if an asterisk were placed after the import statement in line 10? (Answer: Calendar, Timer, Locale, TimeZone)

PROJECTS TO ASSIGN

Ask the students to study Table 2-7 in order to become familiar with the packages listed there, along with their descriptions and examples of the classes in the packages.

Entering Code to Call a System Date Constructor, 82

LECTURE NOTES

Define **Date class**. Explain the purpose of a storage location. Refer to line 16 in Figure 2-33 to show the code to construct a storage location for the date. Define **constructor**. Define **declare**. Refer again to line 16 in Figure 2-33 to show how the variable currentDate is constructed and its type is declared. Point out the Tip on page 82, and define **deprecated**. Review Steps 1 and 2 on page 83 to code a call to a system date constructor, using Figures 2-36 and 2-37 to illustrate.

CLASSROOM ACTIVITIES

Show an example of constructor code in a class to illustrate the concept of class constructor as a form of initialization.

QUICK QUIZZES

What is the syntax of a constructor? (Answer: type-name variable-name = new)

TROUBLESHOOTING TIPS

Make certain the students understand that object variables can be declared from any available class, either from the SDK or programmer defined classes.

Formatting Output Using Escape Characters, 84

LECTURE NOTES

Define **escape characters.** Explain that escape characters are denoted by a character preceded (escaped) by a backward slash. Point out some of the Java escape characters in Table 2-8, and explain that they are used to format output. Review Steps 1 through 3 to code escape characters to format output, using Figures 2-38 through 2-40 to illustrate. Refer to Figure 2-33 and point out line 20, mentioning the fact that currentDate is a variable name, so it will not be printed out literally; rather, the value contained in currentDate will be printed out. Define **concatenate**, and mention the + symbol, which is used to concatenate strings (in this case, the string of escape characters and the string contained in currentDate). Review Step 1 on page 86 to enter code to concatenate String data. Define the **printf() method** and explain its functionality.

QUICK QUIZZES

What are the escape character codes for a horizontal tab, a backspace, a new line, and a carriage return? (Answer: \t, \b, \n, and \r)

TROUBLESHOOTING TIPS

Emphasize that escape characters are placed inside quotes just as are String literals.

Recompiling and Running the Application, 86

LECTURE NOTES

Remind the students that whenever source code is edited, the program must be recompiled in order to create the updated bytecode for execution. Review Steps 1 through 3 on page 87 to

recompile and run the application, using Figure 2-41 to show what the output should look like. Mention Other Ways to recompile and run the application.

Printing the Source Code, 87
LECTURE NOTES
Define **printout**, also called **hard copy**. Review Steps 1 and 2 on page 88 to print the source code, using Figure 2-40 to locate the Print button on the Standard toolbar and Figure 2-42 to show what the printout should look like. Mention Other Ways to print the source code.

Quitting TextPad, 88
LECTURE NOTES
Review Step 1 on page 89 to quit TextPad.

TROUBLESHOOTING TIPS
Remind the students that if TextPad asks them if they want to save their file, it means that they have made changes to the code since the last time it was saved.

Moving to the Web, 89
LECTURE NOTES
Mention that one useful feature of Java is the capability to develop programs that are machine-independent and can run on the Web. Recall from Chapter 1 that applets are small, simple programs that run as part of a Web page. Mention how applets differ from stand-alone applications. Point out that the students are to convert The Welcome to My Day program to an applet by importing two packages that implement graphical elements.

DISCUSSION TOPICS
Ask the students to discuss the differences between stand-alone applications and applets? When would one be preferred over another?

Opening an Existing File in TextPad*, *89
LECTURE NOTES
Review Steps 1 through 5 on pages 90 through 92 to start TextPad and open an existing file, using Figures 2-43 through 2-47 to illustrate.

DISCUSSION TOPICS
Discuss with the students the efficiency of opening an existing file for a new program and using key elements, such as the main() method, before saving it with a new filename. Would it be worth the time to create some template Java programs to start new applications or other program types?

Entering Code to Import Applet Packages*, *92
LECTURE NOTES
Point out that two packages need to be imported in order to create an applet. Define the **applet package**, and explain its function. Define the **Abstract Window Toolkit (AWT)**, and point out that it provides access to color, draw methods, and other GUI elements. Refer to program lines 11 and 12 in Figure 2-48 to show the two import statements that will be added to the program.

Note that because multiple methods will be used from these packages, a period followed by an asterisk is used to ask the computer to import all existing classes from both packages (java.applet.* and java.awt.*). Review Steps 1 and 2 on page 93 to code import statements.

PROJECTS TO ASSIGN
Suggest that the students look at the AWT and Applet documentation on the Java Sun Microsystems Web site. In particular, ask them to find several of the methods available in these classes.

QUICK QUIZZES
What packages should be imported when writing an applet? (Answer: java.applet.*, java.awt.*)

TROUBLESHOOTING TIPS
You might want to define the term "wildcard symbol" with regard to the asterisk, as students may not be familiar with regular expressions.

Changing the Class Name and Extending the Applet Class, 93
LECTURE NOTES
Review that the file name and the class name of the applet will need to be changed to differentiate it from the console application. Define the **extends command**, and refer to Figure 2-48 to show where the extends command should be added to the class header. Mention that this will allow the applet to inherit general applet characteristics from the Applet class. Discuss Figure 2-50. Review Steps 1 and 2 on page 94 to edit the class name and extend the Applet class, using Figures 2-49 and 2-50.

QUICK QUIZZES
What does the extends command do? (Answer: It creates a subclass that is an extension of the superclass whose name follows the extends command.)

The paint() Method, 95
LECTURE NOTES
Define the **init() method,** and explain that init() is one of the methods inherited from the Applet class. Recall that with an applet, Java does not look for a main() method as in stand-alone applications, rather the Applet class uses the init() method to load automatically the initial setup of the applet. Explain the **paint() method** that the students will be coding, which will graphically draw text and an image on the screen after the applet is initialized. Paint() will accept a parameter of type Graphics, which is conventionally identified by the variable name *g*, and will not return a value. Define **reference variable**, and point out that the variable *g* refers to an instance of a Graphics object, and therefore is called a **reference variable**. Refer to Figure 2-51 to show the source code for the beginning of the paint() method. Review Step 1 on page 96 to code the paint() method header, using Figure 2-52 to show the code in the TextPad window.

CLASSROOM ACTIVITIES
Ask the students to explain how the user starts executing an applet? Discuss whether or not the programmer needs to code the init() method. Discuss the reference variable that the paint() method creates and initializes?

DISCUSSION TOPICS
Ask the students what would happen if the programmer coded a method that was already in the superclass. For example, suppose the Applet class contains a paint() method, and the student codes a paint() method as well. What would happen? Introduce in general terms the concept of overriding methods that are in the superclass.

QUICK QUIZZES
What is a reference variable? (Answer: a variable that refers to an instance of an object)

PROJECTS TO ASSIGN
Ask the students to search the Java API and find some of the other methods in the Graphics package that use the reference variable *g* as a parameter.

The drawString() Method*, *96
LECTURE NOTES
Define the **drawString() method**, and recall that Applets use the drawString() method from the AWT package to display text output instead of the println() method. Mention that the drawString() method accepts three arguments as parameters inside the parentheses: the String data to display, the horizontal coordinate for the String, and the vertical coordinate for the String. Refer the students to Figure 2-51 to show three calls to the drawString() method. Point out the syntax of the call to the drawString() method. Mention the Tip on page 97, and define **pixel**. Define the **toString() method**, and explain why it must be used with the variable currentDate in the drawstring() method. Review Step 1 on page 97 to code the drawString() methods, using Figure 2-53 to show what the code should look like in TextPad.

CLASSROOM ACTIVITIES
Discuss with the students how a programmer can determine the desired horizontal and vertical coordinates of a String.

DISCUSSION TOPICS
Ask the students to explain what it means for the toString() method to be polymorphic.

QUICK QUIZZES
Where does the drawString() method come from? (Answer: It is taken from the awt package.)
What is a pixel? (Answer: the basic unit of programmable color on a computer display or in a computer image)
What must the first parameter for the drawString() method be? (Answer: a String)
What is the toString() method used for? (Answer: to convert any nonString object to a String)

PROJECTS TO ASSIGN
Look at the Java API and find some classes that implement the toString() method. Is it a method in the Object class?

Entering Code to Draw an Image and Set the Background Color, 98
LECTURE NOTES

Define the **Image object**, the **getImage() method**, the **getDocumentBase() method**, and the **drawImage() method**. Use Figure 2-54 to explain the syntax of these methods and how they are used in the paint() method. Point out the keyword, this, in line 24, and explain its use. Mention that the g. in front of the methods means that the Graphics object g is calling the methods. Define the **setBackground() method**, and call attention to line 25, pointing out that the setBackground() method takes a Color object and its attribute to set the background color of the applet window. Point out Table 2-9, where the colors that can be used with the Color object are listed. Review Step 1 to enter code to draw an image and set the background color, using Figure 2-55 to show what the code should look like in the TextPad window.

QUICK QUIZZES
What object is used to change the background color of an applet window? (Answer: the Color object)
What parameters does the drawImage() method take? (Answer: image, horizontal and vertical coordinates, "this")

PROJECTS TO ASSIGN
Encourage your students to experiment with the applet code, customizing it using different Image objects, and using different colors and fonts with the setBackground(), setForeground(), setFont(), and setSize() methods. They will need to look at the Java APIs to determine the functionality of some of the methods.

TROUBLESHOOTING TIPS
The meaning of the keyword, this, may not be intuitive. Emphasize the ability of objects to refer to themselves with the keyword, this.

Saving a Source Code File with a New Name, 100
LECTURE NOTES
Review Steps 1 through 3 to save the source code file with a new file name, using Figure 2-56 to illustrate. Mention the Other Ways to save the source code file.

Compiling the Applet, 101
LECTURE NOTES
Mention that during compilation, the compiler looks for the image file in the same directory as the source code file. Point out that possible errors that could be generated while compiling the applet include an incorrect location for the Java compiler, typing mistakes, omitting special characters, case-sensitive errors, and file name errors. Review Steps 1 through 2 to compile the applet. Mention Other Ways to compile the applet.

CLASSROOM ACTIVITIES
Ask students to share tips on debugging any errors they may have encountered during compilation.

Creating an HTML Host Document, 101
LECTURE NOTES

Define **host**, and point out that an applet is executed by a host, or reference program, such as a Web page. Review that the HTML in a Web page is primarily for displaying information, not user interaction, and that applets are useful for adding more functionality to a Web page.

Coding an HTML Host Document, 101

LECTURE NOTES

Define **tag**, also called markup, and tell how it is used in HTML. Define **start tag** and **end tag**, and explain their syntax. Use Figure 2-57 to show the code for the HTML host document for the WelcomeApplet. Review Steps 1 through 5 on pages 102 through 104 to code an HTML host document, using Figures 2-58 through 2-60 to illustrate.

QUICK QUIZZES

What should an APPLET tag contain in order to access the Java applet? (Answer: the name of the bytecode file and the width and height of the window in which to run the applet)

PROJECTS TO ASSIGN

Suggest that the students experiment with the width and height values for the window in the HTML document and observe the effect on the applet design.

TROUBLESHOOTING TIPS

HTML and its tags are generally not case-sensitive. The name of the referenced class much match exactly, however, because it is a Java class, and Java is case-sensitive.

Running an Applet, 104

LECTURE NOTES

Review that an applet is run by first running the HTML host document. Point out that this may be done from TextPad. Review Steps 1 through 3 on pages 105 and 106 to run an applet, using Figures 2-61 through 2-63 to illustrate. Mention Other Ways to run an applet.

DISCUSSION TOPICS

Ask the students to discuss the advantages of Applet Viewer over a browser for viewing an applet. Why, then, should an Applet ever be run in a browser?

PROJECTS TO ASSIGN

Refer the students to Appendix D for information on running an applet from the command prompt window.

Documenting the Applet and HTML Host Document, 106

LECTURE NOTES

Review that the final step in the program development cycle is to document — that is, print a hard copy of — the applet and the HTML host document code. Review Steps 1 through 7 on pages 107 through 109 to document the applet and HTML host document, using Figures 2-64 through 2-68 to illustrate. Mention Other Ways to print a hard copy of the applet and the HTML host document code. Mention the Tip on page 109.

DISCUSSION TOPICS

Discuss other ways to archive the code rather than hard copies (zip, tar, or jar files). Why should the code be archived electronically?

Quitting TextPad*, *110
LECTURE NOTES
Review Step 1 to quit TextPad. Point out Other Ways to quit TextPad.

Chapter Summary, 110
LECTURE NOTES
The students learned how to code a simple console application and applet in this chapter. In the process, they learned to use the text editor TextPad to compile, debug, and run a Java application and applet, and to create an HTML host document. The students also were introduced to basic Java syntax, as well as to concepts such as importing packages, using constructors, and extending classes.

What You Should Know, 111
LECTURE NOTES
Use these concepts to review the chapter with the students.

Key Terms, 112

Abstract Window Toolkit (AWT) (92)
access modifier (64)
applet package (92)
argument (68)
block comment (61)
body (68)
braces { } (66)
case-sensitive (66)
class definition (70)
class header (64)
Clip Library window (55)
coding window (54)
Command Results window (75)
comment header (61)
comments (61)
concatenate (86)
constructor (82)
data type (68)
Date class (82)
debugging (74)
declare (82)
default (50)
deprecated (82)
doc comment (62)
drawImage() method (98)
drawString() method (96)
edit (79)
end tag (101)
escape characters (84)
exception (77)
extends (70)
extends command (94)
getDocumentBase() method (98)
getImage() method (98)
hard copy (87)
host (101)
identifier (68)
Image object (98)
import statement (81)
init() method (95)
keywords (64)
line comment (62)
line numbers (55)
literal (71)
logic error (77)
main() method (67)
method header (67)
method modifier (67)
out (70)
paint() method (95)
parameter (67)
pixel (97)