APPENDIX A

# Working with the Java Platform

In this appendix, you will:

- Configure Windows to work with the Java SE Development Kit
- Use Notepad to save and edit source code
- Use TextPad to work with Java

## Configuring Windows to Work with the Java SE Development Kit

Several versions of Java are available for free at the Java Web site (*www.oracle.com/technetwork/java/index.html*). The official name of the most recent version is Java Platform, Standard Edition 7, often called **Java SE 7** for short. Two version numbers (for example, 1.7.0 and 7) are used to identify this release of the Java Platform. Version 7 is the product version, and 1.7.0 is the developer version. The number 7 is used to reflect Java's evolving level of maturity. As updates to existing versions emerge or entirely new versions containing advanced features are released, you can download them. For example, a recent update as this book was being written is Java SE 7u7; *7u7* is short for *version 7, update 7*.

Java's Web site was *http://java.sun.com* before Java was purchased by Oracle. Now the Web site is *http://www.oracle.com/technetwork/java/index.html*. However, the shorter URL redirects you to the longer one, so you can use the shorter address if it is more convenient. Each new version of Java has a code name. The code name for version 7 is Dolphin. The name for version 5 was Tiger, and the name for version 6 was Mustang.

Over the years, Java has been inconsistent in numbering new versions. Before version 6, the standard editions were called JDK 1.0.3, JDK 1.1.2 through 1.1.8, J2SE 1.2.0 through 1.4.2, and J2SE 5.0. With versions 6 and 7, Java is attempting to simplify the name and number changes. Java sometimes adds a "half step" for minor revisions in a version, such as JDK 1.7.0_05.

The different names for Java versions are somewhat confusing and frequently misused. If you download Java to use with this book, you want to acquire the Java Standard Edition (SE) Development Kit, also known as the **JDK**. Java also supports the **Java Enterprise Edition** (EE), which includes all of the classes in the Java SE, plus a number of classes that are more useful to programs running on servers than on workstations. The Java EE Development Kit is known as **SDK**. The names of the development kits have changed frequently; originally, JDK meant "Java Development Kit," but that interpretation was used with the earliest Java versions and is no longer used officially.

The **Java Micro Edition** (ME) is another Java platform, which is used for small devices such as PDAs (personal digital assistants), cell phones, and other consumer appliances.

To configure your Windows operating system with the JDK, you must add the Java bin directory to the command path of your operating system (OS). That way, your OS will know where to look for the Java commands that you use.

One way to update the OS path for Windows is to edit or set the OS path in the autoexec.bat file. This file is automatically executed every time you start your computer. A simpler and less error-prone alternative is to type two commands at the OS prompt when you want to begin a session of working on Java programs. (These two commands are described later in this appendix.)

You do not need to be an operating system expert to issue operating system commands. Learning just a few commands allows you to create and run all the examples in this book.

## Finding the Command Prompt

To locate the command prompt on your Windows computer, click Start, point to Programs, point to Accessories, and then click Command Prompt.

In earlier versions of Windows, the console window was called the *MS-DOS (Microsoft Disk Operating System) prompt*, or more simply, the *DOS prompt*. Many people still use this term instead of *command prompt*.

## Command Prompt Anatomy

The Windows command prompt contains at least a disk drive name followed by a colon, a backslash, and a greater-than sign (for example, C:\>). You might also see folder or directory names within the command prompt just before the greater-than sign, as shown in the following examples:

C:\Documents and Settings>

C:\Documents and Settings\Administrator>

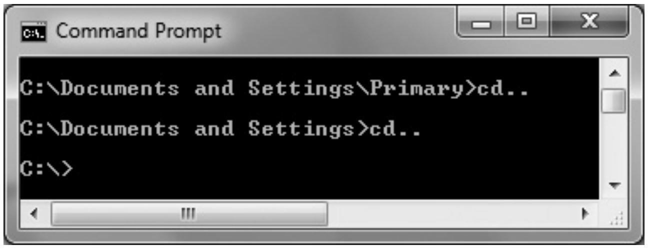
Each directory in the path is separated by a backslash.

## Changing Directories

You can back up one directory level by typing `cd` for "change directory," followed by two periods:

```
cd..
```

For example, if your OS prompt contains C:\Documents and Settings\Primary> and you type `cd..`, the command prompt changes to C:\Documents and Settings>. If you type `cd..` again, the prompt changes to C:\>, indicating the root directory. Figure A-1 shows this progression.



```
C:\Documents and Settings\Primary>cd..

C:\Documents and Settings>cd..

C:\>
```

**Figure A-1** Results following two `cd..` commands

When you have multiple directories to back through, it is easier to use the following command:

```
cd\
```

This takes you immediately to the root directory instead of backing up one level at a time.

At the command prompt, you can change to another disk drive by typing its name and a colon, then pressing Enter. For example, the following command changes the command prompt to refer to the A drive:

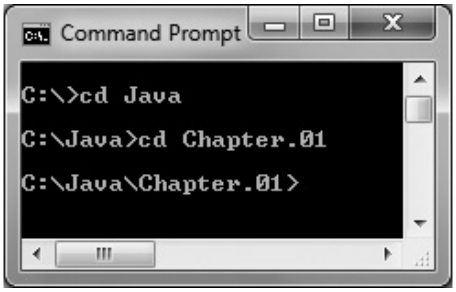
```
A:
```

You can change the directory by typing `cd` followed by the name of the directory. For example, if you have a folder named Java and it contains a folder named Chapter.01, you can change the command prompt to the Chapter.01 folder by backing up to the root directory and typing the following:

```
cd Java
cd Chapter.01
```

As shown in Figure A-2, the command prompt now reads C:\Java\Chapter.01>. When you compile and execute your Java programs, you should start from the command prompt where the files are stored.

When your command prompt display is filled with commands, it can look confusing. If you want, you can type `cls` (for Clear Screen) to remove old commands.



```
C:\>cd Java

C:\Java>cd Chapter.01

C:\Java\Chapter.01>
```

**Figure A-2** Changing to the Java\Chapter.01 directory from the root directory

## Setting the `class` and `classpath` Variables

When you start a Java session, you might need to set the `class` and `classpath` options. These settings tell the operating system where to find the Java compiler and your classes. If you or someone else has altered your autoexec.bat file to contain these commands, you do not need to type them. Otherwise, every time you want to compile and execute Java programs, you need to type statements similar to the following:

```
path = c:\program files\java\jdk1.7.0\bin
set classpath=.
```

After you have typed the `class` and `classpath` statements, you can compile and run as many Java programs as you want without typing these commands again. You must type them again if you close the Command Prompt window or restart your computer.

The first statement sets the path and allows the OS to recognize the `javac` command you use when compiling programs. Consider the following example:

```
path = c:\program files\java\jdk1.7.0\bin
```

This example assumes that you are using JDK 1.7.0 and that it is stored in the java folder in the program files folder. These are the defaults when you download Java from the Java Web site; if you installed Java in a different location, you need to alter the command accordingly.

The command `set classpath=.` tells Java to find your compiled classes in the current directory when you execute your applications and applets. There must be no space between `classpath` and the equal sign, or between the equal sign and the period.

After you set the path correctly, you should be able to use the `javac` command. If you attempt to compile a Java class and see an error message that `javac` is not a recognized command, either Java was not properly installed or the path command was incorrect. If classes compile successfully but do not execute, you might have entered the `classpath` command incorrectly.

## Changing a File's Name

When working through the examples in this book, you will often find it convenient to change the name of an existing file—for example, when you want to experiment with altering code without losing the original version, or if you find that when you previously saved a file, you mistyped a filename so that it did not match the class name within the .java file you created. You can take at least three approaches:

- Open the existing file using the appropriate software application (for example, Notepad), click File on the menu bar, and then click Save As. Select the folder you want, then type a new filename for the file. Now you have two versions—one with the old name and one with the new.
- In Windows, open My Computer and locate the misnamed file. Select the file and then click the filename. (Do not double-click the filename unless you want to open the file.) You can then edit the filename by using a combination of the Backspace, Delete, and character keys. Press Enter when the filename is correct.
- At the command prompt, use the `rename` command. You type `rename`, a space, the old filename, another space, and the new filename. For example, to change a file named xyz.java to abc.java, type the following at the command prompt for the directory containing the existing file:

  ```
  rename xyz.java abc.java
  ```

## Compiling and Executing a Java Program

At the command prompt, change from the default drive prompt to the drive where your application is stored. Then change the directory (or folder) to the directory that holds your application.

To compile an application or applet, you type the `javac` command to start the Java compiler, then type a space and the complete name of the .java file—for example, First.java. If the application doesn't compile successfully, the path might not be set correctly to the Java JDK bin directory where the javac.exe file is located. Also, you might have failed to use the same spelling as the Java filename.

When you compile a .java file correctly, the Java compiler creates a .class file that has the same filename as the .java file. Thus, a successful compilation of the First.java file creates a file named First.class. To run a Java application, you use the `java` command and the class name without the .class extension. For example, after an application named First.java is compiled, producing First.class, you execute the program using the following command:

```
java First
```

After the program executes, control is returned to the command prompt. If a program does not end on its own, or you want to end it prematurely, you can press Ctrl+C to return to the command prompt.

After you compile a Java program, you can execute it as many times as you want without recompiling. If you change the source code, you must save and compile again before you can see the changed results in an executed application.

When you are testing a Java program, you often issue the commands to compile and execute it many times before you are satisfied with the results. If you press the Up Arrow key at the command line, the previous commands appear in reverse succession. When you find the command you want to repeat, just press Enter.

## Using Notepad to Save and Edit Source Code

You can use the Windows Notepad text editor to save and edit the source code for Java programs. To start Notepad using Windows, click the Start menu, point to All Programs, point to Accessories, and click Notepad. After you start Notepad, you can enter and edit the code just as you would with any text editor.

Saving source code in Notepad requires that the Java source file be saved with a .java extension. Because Java is case sensitive, you must save a file with the proper capitalization. If the class name of the file and the filename do not match in both spelling and case, you receive an error when you attempt to execute compiled source code. The default extension for Notepad documents is .txt. To create a file with a .java extension, use Save As, locate the folder you want, and type the filename. In Windows 7 or Vista, you can simply type the filename. In older Windows operating systems, type the filename with double quotation marks around it, as in "First.java". This ensures that the file is not saved as "First.java.txt". Then click Save.

## Using TextPad to Work with Java

As an alternative to Notepad, you can use TextPad—a text editor that includes many features helpful to Java programmers. You can download a trial version from *www.textpad.com*. Unlike Notepad, TextPad is not included with Windows; to install TextPad, run the setup file after downloading it from the TextPad Web site, and then respond to the dialog box options. Because you download a trial version, you should purchase TextPad if you decide to use it beyond the trial period. Note that TextPad runs only under the Windows operating system. If you are not using Windows, you can use the text editor that comes with your operating system, or you can search the Web to find a text editor that better suits your needs.

To enter and edit source code in TextPad, you can use the same techniques that you use with any other Windows text editor. In short, you can use the standard Windows shortcut keystrokes and menus to enter, edit, and save your code. You can use the File menu to open and close files. You can use the Edit menu to cut, copy, and paste text, and you can use the Search menu to find and replace text. In addition, TextPad color-codes the source files so it is easier to recognize the Java syntax. TextPad also makes it easier to save Java files with the proper capitalization and extension. To compile the current source code, you can select the Compile Java command from the Tools menu. If the source code does not compile cleanly, TextPad displays a Command Results window, including line numbers that identify the source of problems. With TextPad, you can choose to display line numbers in your code.

## Key Terms

**Java SE 7** is the most recent version of Java. The full, official name is Java Platform, Standard Edition 7.

The **JDK** is the Java Standard Edition Development Kit.

The **Java Enterprise Edition** (EE) includes all of the classes in the Java SE, plus a number of classes that are more useful to programs running on servers.

The **SDK** is the Java EE Development Kit.

The **Java Micro Edition** (ME) is another Java platform, which is used for small devices such as PDAs, cell phones, and other consumer appliances.