Chapter 2

Getting the Tools That You Need

In This Chapter

- Installing Java
- Downloading and installing the Android software tools
- Checking your Eclipse configuration
- Getting the code in this book's examples

Ergaliophile /ɜr gə li ə faɪ əl/ *noun* 1. A lover of tools. 2. A person who visits garage sales for rusty metal implements that might be useful someday but probably won't. 3. A person whose computer runs slowly because of the daily, indiscriminate installation of free software tools.

Several years ago, I found an enormous monkey wrench (more than a yard long and weighing 35 pounds) at a nearby garage sale. I wasn't a good plumber, and to this day any pipe that I fix starts leaking again immediately. But I couldn't resist buying this fine piece of hardware. The only problem was, my wife was sitting in the car about halfway down the street. She's much more sensible than I am about these matters, so I couldn't bring the wrench back to the car. "Put it aside and I'll come back for it later," I told the seller.

When I returned to the car empty-handed, my wife said, "I saw someone carrying the world's largest pipe wrench. I'm glad you weren't the one who bought it." And I agreed with her. "I don't need more junk like that."

So of course I returned later that day to buy the monkey wrench, and to this day the wrench sits in our attic, where no one ever sees it. If my wife ever reads this chapter, she'll be either amused or angry. I hope she's not angry, but I'm taking the risk because I enjoy the little drama. To add excitement to my life, I'm turning this trivial secret into a public announcement.

The Stuff You Need

This book tells you how to write Java programs, and before you can write them, you need some software tools. Here's a list of the tools you need:

- **A Java virtual machine**

  Cool people refer to this item as the *JVM* or simply as *Java*.
- **The Java code libraries**

  These code libraries are known affectionately as the *Java Runtime Environment* (JRE) or simply as *Java*.
- **An integrated development environment**

  You can create Java programs using geeky, keyboard-only tools, but eventually you'll tire of typing and retyping commands. An *integrated development environment* (IDE), on the other hand, is a little like a word processor: A word processor helps you compose documents (memos, poems, and other works of fine literature); in contrast, an IDE helps you compose computer programs.

  For composing Java programs, I recommend using the Eclipse IDE.

You should also gather these extra goodies:

- **Some sample Java programs to help you get started**

  All examples in this book are available for download from `www.allmycode.com/Java4Android`.
- **The Android Software Development Kit**

  The Android *Software Development Kit (SDK)* includes lots and lots of prewritten, reusable Android code and a bunch of software tools for running and testing Android apps.

  The prewritten Android code is the Android *Application Programming Interface (API).* The API comes in several versions — versions 9 and 10 (both code-named Gingerbread), versions 11, 12, and 13 (Honeycomb), versions 14 and 15 (Ice Cream Sandwich), and so on.
- **Android-oriented add-ons for the integrated development environment**

  By using add-ons, you customize the Eclipse IDE to help you compose, run, and test your Android apps. The set of Eclipse add-ons for working with Android apps is the *Android Development Toolkit (ADT).*

All these tools run on the *development computer* — the laptop or desktop computer you use to develop Java programs and Android apps. After you create an Android app, you copy the app's code from the development computer to a *target device* — a phone, a tablet, or (someday soon) a refrigerator that runs Android.

Here's good news: You can download from the web all the software you need to run this book's examples for free. The software is separated into three downloads:

- This book's website (`www.allmycode.com/Java4Android`) has a link to all code in the book.
- When you visit `www.java.com`, you can click a button to install the Java virtual machine.
- A button at the page `http://developer.android.com/sdk` gives you the big Android SDK download. In spite of its name, it includes more than simply the Android code libraries. The download includes all the ingredients you didn't already collect from `www.allmycode.com` or `www.java.com`.

REMEMBER

The websites I describe in this chapter are always changing. The software programs you download from these sites change, too. A specific instruction such as "Click the button in the upper-right corner" becomes obsolete (and even misleading) in no time at all. So in this chapter, I provide explicit steps, but I also describe the ideas behind them. Browse the suggested sites and look for ways to get the software I describe. When a website offers you several options, check the instructions in this chapter for hints on choosing the best option. If your computer's Eclipse window doesn't look quite like the one in this chapter's figures, scan your computer's window for whatever options I describe. If, after all that effort, you can't find the elements you're looking for, check this book's website (`www.allmycode.com/Java4Android`) or send an e-mail to me at `Java4Android@allmycode.com`.

If You Don't Like Reading Instructions . . .

I start this chapter with a brief (but useful) overview of the steps required in order to get the software you need. If you're an old hand at installing software, and if your computer isn't quirky, these steps will probably serve you well. If not, you can read the more detailed instructions in the next several sections.

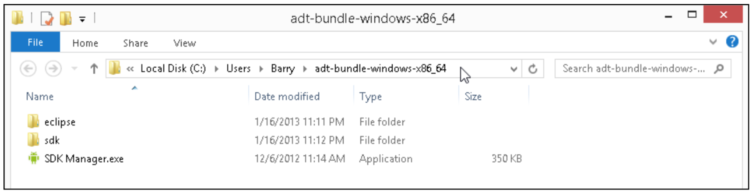
1. **Visit** `www.allmycode.com/Java4Android` **and download a file containing all the program examples in this book.**
2. **Visit** `www.java.com` **and download the Java Runtime Environment (if you don't already have a recent version of Java on your computer).**

   Choose a version of the software that matches your operating system (Windows, Macintosh, or whatever) and your operating system's word length (32-bit or 64-bit).
3. **Visit** `http://developer.android.com/sdk` **and download the Android Software Development Kit (SDK).**

   The downloaded bundle is a `.zip` archive file.

4. **Extract the contents of the downloaded archive file to your local hard drive.**

   On my Windows computer, I extract the `.zip` file's contents to a new folder, named `c:\Users\`*`MyUserName`*`\adt-bundle-windows-x86`. So I have the folders shown in Figure 2-1.



adt-bundle-windows-x86_64

File Home Share View

« Local Disk (C:) › Users › Barry › adt-bundle-windows-x86_64

Search adt-bundle-windows-...

Name	Date modified	Type	Size
eclipse	1/16/2013 11:11 PM	File folder	
sdk	1/16/2013 11:12 PM	File folder	
SDK Manager.exe	12/6/2012 11:14 AM	Application	350 KB

Figure 2-1: My Windows computer's `adt-bundle` folder.

   On my Mac, I extract the `.zip` file's contents into my existing `Applications` folder, as shown in Figure 2-2.

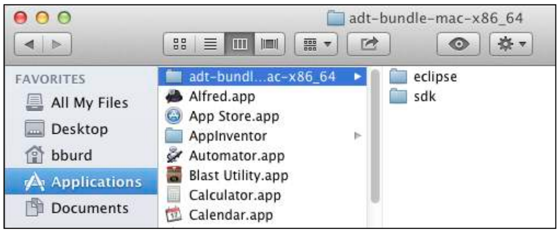


Figure 2-2: My Mac's `adt-bundle` folder.

   If the Android SDK `.zip` file contains more than one folder, don't separate the folders when you extract the `.zip` file's contents. Extract all content inside the `.zip` file to the same place on your hard drive.

5. **Launch the Eclipse app.**

   The first time you run a fresh, new copy of Eclipse, the Welcome screen appears.

6. **Dismiss the Welcome screen.**

   For most versions of Eclipse, you can dismiss the Welcome screen by clicking the little *x* icon that appears on a tab above the screen.

7. **Import the code that you downloaded in Step 1.**

In Eclipse, choose File➪Import➪Existing Projects into Workspace. Then browse for this book's sample code — the `.zip` file from Step 1. (If the web browser automatically expanded the `.zip` archive, browse for the folder containing the files that were in the archive.)

8. **Create an Android virtual device.**

   You can test Android programs on a phone or a tablet. But, for convenience, you might test on an *emulator* — a program that behaves like a phone or a tablet but runs on the development computer.

   To run an emulator, you need an *Android Virtual Device* (*AVD*), which is a set of specs for a device (processor type, screen size, screen resolution, and Android version, for example). In Eclipse, you create an AVD by choosing Window➪Android Virtual Device Manager and filling in the blanks. For more info, see the later section "Creating an Android Virtual Device."

For details about any of these topics, see the next several sections.

Those pesky filename extensions

The filenames displayed in My Computer or in a Finder window can be misleading. You may browse a directory and see the name `Mortgage`. The file's real name might be `Mortgage.java`, `Mortgage.class`, `Mortgage.`*`somethingElse`*, or plain old `Mortgage`. Filename endings such as `.zip`, `.java`, and `.class` are *filename extensions*.

The ugly truth is that, by default, Windows and Macs hide many filename extensions. This awful feature tends to confuse programmers. If you don't want to be confused, change your computer's systemwide settings. Here's how to do it:

- **In Windows XP:** Choose Start➪Control Panel➪Appearance and Themes➪Folder Options. Then skip to the fourth bullet.
- **In Windows 7:** Choose Start➪Control Panel➪Appearance and Personalization➪Folder Options. Then skip to the fourth bullet.
- **In Windows 8:** On the Charms bar, choose Settings➪Control Panel. In the Control Panel, choose Appearance and Personalization➪Folder Options. Then proceed to the following bullet.
- **In all versions of Windows (XP and newer):** Follow the instructions in one of the preceding bullets. Then, in the Folder Options dialog box, click the View tab. Look for the Hide File Extensions for Known File Types option. Make sure that this check box is *not* selected.
- **In Mac OS X:** In the Finder application's menu, select Preferences. In the resulting dialog box, select the Advanced tab and look for the Show All File Extensions option. Make sure that this check box *is* selected.

Getting This Book's Sample Programs

To get copies of this book's sample programs, visit www.allmycode.com/Java4Android and click the link to download the programs in this book. Save the download file (Java4Android_Programs.zip) to the computer's hard drive.

TIP

In some cases, you can click a download link all you want but the web browser doesn't offer you the option to save a file. If this happens to you, right-click the link (or control-click on a Mac). From the resulting contextual menu, select Save Target As, Save Link As, Download Linked File As, or a similarly labeled menu item.

Most web browsers save files to the Downloads directory on the computer's hard drive. But your browser may be configured a bit differently. One way or another, make note of the folder containing the downloaded Java4Android_Programs.zip file.

Compressed archive files

When you visit www.allmycode.com/Java4Android and you download this book's examples, you download a file named Java4Android_Programs.zip. A *zip* file is a single file that encodes a bunch of smaller files and folders. For example, my Java4Android_Programs.zip file encodes folders named 06-01, 06-02, and so on. The 06-02 folder contains subfolders, which in turn contain files. (The folder named 06-02 contains the code in Listing 6-2 — the second listing in Chapter 6.)

A .zip file is an example of a *compressed archive* file. Other examples of compressed archives include .tar.gz files, .rar files, and .cab files. When you *uncompress* a file, you extract the original files stored inside the larger archive file. (For a .zip file, another word for uncompressing is *unzipping*.) Uncompressing normally re-creates the folder structure encoded in the archive file. So after uncompressing my Java4Android_Programs.zip file, the hard drive has folders named 06-01, 06-02, with subfolders named src and bin, which in turn contain files named TypeDemo1.java, TypeDemo1.class, and so on.

When you download Java4Android_Programs.zip, the web browser may uncompress the file automatically for you. If not, you can see the .zip file's contents by double-clicking the file's icon. (In fact, you can copy the file's contents and do other file operations after double-clicking the file's icon.) One way or another, don't worry about uncompressing my Java4Android_Programs.zip file. When you follow this chapter's instructions, you can import the contents of the file into the Eclipse IDE. And behind the scenes, the Eclipse import process uncompresses the .zip file.

Gathering Information

For many people (including some inexperienced people), the installations of Java and the Android SDK are routine tasks. Visit a few websites, click some buttons, and then take a coffee break. But as you follow this chapter's instructions, you might have a question, experience a difficulty, or encounter a fork in the road. In that case, it helps to know your computer — which entails jotting down the answers to a few questions.

Are you running a 32-bit or 64-bit operating system?

In this chapter, you install Java and the Android SDK on your computer. Java comes in two flavors: 32-bit and 64-bit. The Android SDK comes in the same two flavors, and in order for the Android SDK to work with Java, the Java flavor must match the Android SDK flavor. In this section, you find out which flavor is best for your computer.

REMEMBER

The steps in this section are all optional. If you don't want to perform this section's fact-finding missions, try visiting `www.java.com` and `http://developer.android.com/sdk` to download whichever versions of Java and the Android SDK are offered to you by these two websites. If either site makes you choose between 32-bit and 64-bit software, be consistent. That is, get the 32-bit versions of both Java and the Android SDK, or get the 64-bit versions of both Java and the Android SDK. (For Windows, the 32-bit versions are the safest choice. For Mac, the 64-bit versions are the safest.)

For Windows 8, Windows 7, and Windows Vista:

1. **Press the Windows key.**

   In Windows 8, the Start screen appears. In Windows 7 and Windows Vista, the Start menu appears.

2. **In Windows 8, type the words** Control Panel**, and then press Enter. In Windows 7 or Windows Vista, click the Control Panel item on the Start menu.**

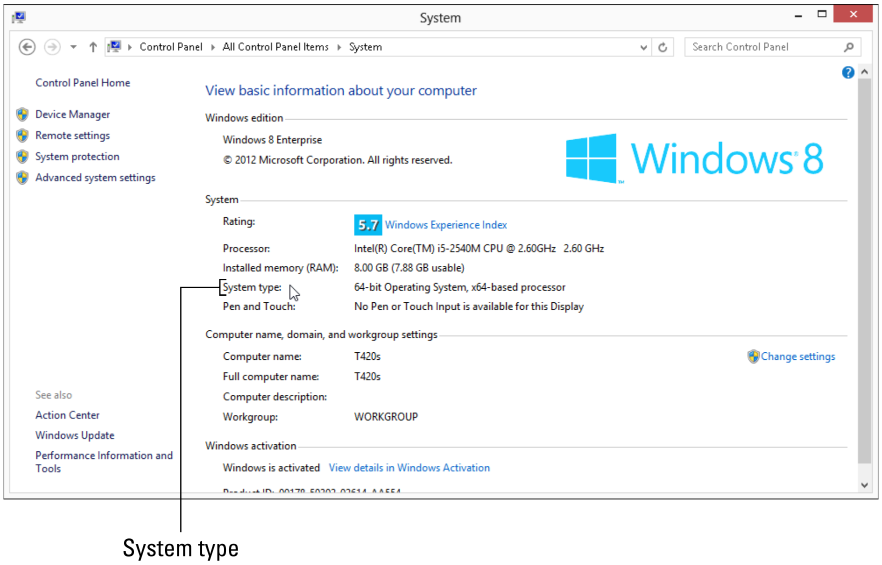
   The Control Panel appears.

3. **In the Control Panel, select System and Security (Windows 8 and Windows 7) or System and Maintenance (Windows Vista).**

The System window appears. To recognize the System window, look for the words `View basic information about your computer` near the top of the window.

4. **In the System window, look for the words** `System type`**.**

   The system type is either 32-bit or 64-bit, as shown in Figure 2-3.



View basic information about your computer

Windows edition

Windows 8 Enterprise

© 2012 Microsoft Corporation. All rights reserved.

System

Rating:	5.7 Windows Experience Index
Processor:	Intel(R) Core(TM) i5-2540M CPU @ 2.60GHz 2.60 GHz
Installed memory (RAM):	8.00 GB (7.88 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

Computer name:	T420s
Full computer name:	T420s
Computer description:	
Workgroup:	WORKGROUP

Windows activation

Windows is activated View details in Windows Activation

Figure 2-3: Determining the system type.

For Windows XP

1. **Press the Windows key.**

   The Start menu appears.

2. **Click the My Computer item on the Start menu.**

   Windows Explorer opens.

3. **In Windows Explorer, navigate to Drive C.**

4. **In Drive C, look for folders named** `Program Files` **and** `Program Files (x86)`**.**

   If you find `Program Files` but not `Program Files (x86)` folders, you're running 32-bit Windows. If you find both `Program Files` and `Program Files (x86)` folders, you're running 64-bit Windows.

For Macintosh OS X

1. **Choose Apple➪About This Mac.**

   The About This Mac window appears.

How many bits does your computer have?

As you follow this chapter's instructions, you may be prompted to choose between two versions of a piece of software — the 32-bit version and the 64-bit version. What's the difference, and why do you care?

A *bit* is the smallest piece of information that you can store on a computer. Most people think of a bit as either a zero or a one, and that depiction is quite useful. To represent almost any number, you pile several bits next to one another and do some fancy things with powers of two. The numbering system's details aren't showstoppers. The important thing to remember is that each piece of circuitry inside the computer stores the same number of bits. (Well, some circuits inside the computer are outliers with their own particular numbers of bits, but that's not a big deal.)

In an older computer, each piece of circuitry stores 32 bits. In a newer computer, each piece of circuitry stores 64 bits. This number of bits (either 32 or 64) is the computer's *word length*. In a newer computer, a word is 64 bits long.

"Great!" you say. "I bought my computer last week. It must be a 64-bit computer." Well, the story may not be that simple. In addition to a computer's circuitry having a word length, the operating system on it also has a word length. An operating system's instructions work with a particular number of bits. An operating system with 32-bit instructions can run on either a 32-bit computer or a 64-bit computer, but an operating system with 64-bit instructions can run only on a 64-bit computer. And to make things even more complicated, each program that you run (a web browser, a word processor, or one of your own Java programs) is either a 32-bit program or a 64-bit program. You may run a 32-bit web browser on a 64-bit operating system running on a 64-bit computer. Alternatively, you may run a 32-bit browser on a 32-bit operating system on a 64-bit computer. (See the figure that accompanies this sidebar.)

When a website makes you choose between 32-bit and 64-bit software versions, the main consideration is the word length of the operating system, not the word length of the computer's circuitry. You can run a 32-bit word processor on a 64-bit operating system, but you can't run a 64-bit word processor on a 32-bit operating system (no matter what word length the computer's circuitry has). Choosing 64-bit software has one primary advantage: 64-bit software can access more than 3 gigabytes of a computer's fast random access memory. And in my experience, more memory means faster processing.

How does all this information about word lengths affect Java and Android SDK downloads? Here's the story:

- If you run a 32-bit operating system, you run only 32-bit software.
- If you run a 64-bit operating system, you probably run some 32-bit software and some 64-bit software. Most 32-bit software runs fine on a 64-bit operating system.
- On a 64-bit operating system, you might have two versions of the same program. For example, on my Windows computer, I have two versions of Internet Explorer: a 32-bit version and a 64-bit version.

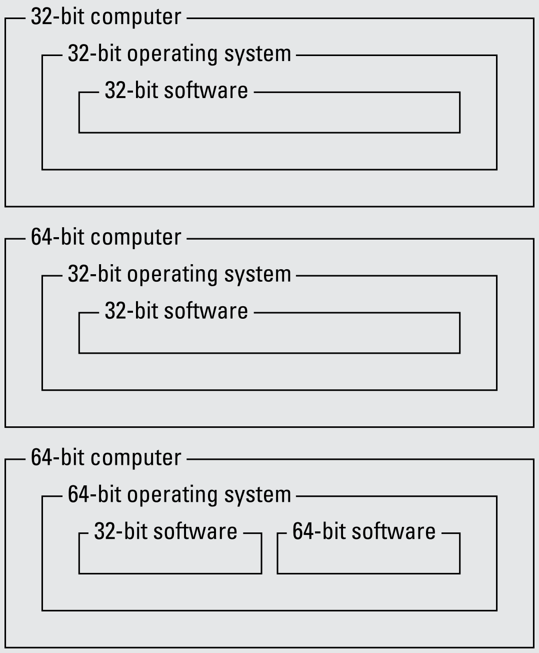
  Normally, Windows stores 32-bit programs in its `Program Files (x86)` directory and stores 64-bit programs in its `Program Files` directory.
- A chain of word lengths is as strong as its weakest link. For example, when I visit `www.java.com` and click the site's Do I Have Java? link, the answer depends on the match between my computer's Java

(continued)

(continued)

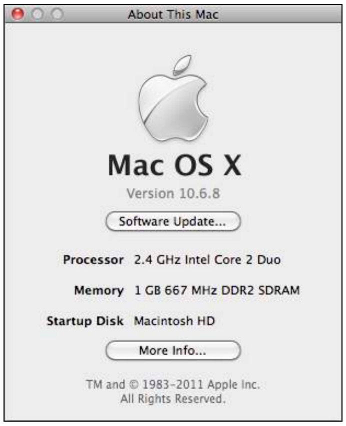
version and the web browser that I'm running. With only 64-bit Java installed on my computer, the Do I Have Java? link in my 32-bit Firefox browser answers, `No working Java was detected on your system`. But the same link in my 64-bit Internet Explorer answers, `You have the recommended Java installed`.

- Here's the most important thing to remember about word lengths: When you follow this chapter's instructions, you install Java software and Android SDK software on the computer. The Java software's word length must match the Android SDK's word length. In other words, 32-bit Android SDK software runs with 32-bit Java, and 64-bit Android SDK runs with 64-bit Java. I haven't tried all possible combinations, but when I try to run the 32-bit Android SDK with 64-bit Java, I see the misleading error message `No Java virtual machine was found`.



2. **In the About This Mac window, look for the word *Processor*.**

   If your processor is an Intel Core Solo or Intel Core Duo, you have a 32-bit Mac. All other Intel processors, including Intel Core 2 Duo, are 64-bit Macs. (See Figure 2-4.)



Mac OS X
Version 10.6.8

Processor 2.4 GHz Intel Core 2 Duo
Memory 1 GB 667 MHz DDR2 SDRAM
Startup Disk Macintosh HD

TM and © 1983-2011 Apple Inc.
All Rights Reserved.

Figure 2-4: Displaying the Mac processor type.

Here's an alternative (geeky) way to find out whether your Mac is a 32-bit or 64-bit operating system: In the Spotlight, type the word **Terminal**, and then press Enter. Then when the Terminal app opens, type **uname -a** and press Enter. If the Mac's response includes `i386`, you have a 32-bit system. If the Mac's response includes `x86_64` instead, you have a 64-bit system.

If you're a Mac user, which version of Mac OS X do you have?

To answer a burning question about the Macintosh operating system, follow these steps:

1. **Choose Apple⇨About This Mac.**

   The About This Mac window appears.

2. **In the About This Mac window, look for the word *Version*.**

   You see Version 10.8 (or something like that) in very light gray text. (Refer to Figure 2-4.)

The Android development software for the Mac requires OS X 10.5.8 or later, and an Intel processor. If the About This Mac window reports that you have a PowerPC processor or that your version of OS X is older than OS X 10.5, you'll have a hard time developing Android apps. (For versions such as OS X 10.5.1, you can try updating the system to version 10.5.8. For systems before OS X 10.5, and for systems running on PowerPC processors, you can search the web for hacks and workarounds. Of course, if you use hacks and workarounds, I make no promises.)

TIP

If you don't regularly apply software updates, choose Software Update from the Apple menu. In the resulting window, look for OS X updates and for items with the word *Java* in them. Select the relevant items, and then click the appropriate Install or Update button (or buttons). In addition, you can follow the instructions in the next section to find out whether the `www.java.com` website recommends updates.

Is a recent version of Java installed on your computer?

Android development requires Java 5.0 or later. Java 6 is recommended (but not absolutely required). Java 7 and beyond are overkill.

TECHNICAL STUFF

You might see *Java 1.5* and *Java 1.6* rather than *Java 5.0* and *Java 6.* Some people understand the differences these names make, but few people care. (If you're one of the people who care, see Chapter 1.)

Follow these steps to check for a recent version of Java on your computer:

1. **Visit** `www.java.com`**.**
2. **On the main page at** `www.java.com`**, click the Do I Have Java? link.**
3. **On the Do I Have Java? page, click the Verify Java Version button.**

After a brief pause, the java.com site reports that you have Java Version 7 Update 9, or something like that.

- If you have Java version 6 or higher, you're good to go. You don't have to install any other Java version. You can skip this chapter's later section "Setting Up Java."
- If the java.com site doesn't report that you have Java 6 or later, don't fret. The java.com site might be wrong!

  After all, a 32-bit web browser can't detect a 64-bit version of Java, and (as of early 2013) no browser running in Windows 8 mode can even detect Java. The potential pitfalls are endless.

  Anyway, if java.com doesn't report that you have Java 6 or later, I suggest following the instructions in the section "Setting Up Java." If you accidentally install a second version of Java (or a third or fourth version of Java), you'll probably be okay.

Setting Up Java

You can get the latest, greatest version of Java by visiting `www.java.com`. The site offers several alternatives.

- **(Recommended) Click the big Free Java Download button on the site's main page.**

  For most computers, clicking this Free Java Download button gives you all the Java you need for this book's examples. So if you're unsure what to do when you visit `www.java.com`, click the Free Java Download button and move to the section "Setting Up the Android SDK," later in this chapter.

  WARNING!

  If you're running Mac OS X 10.6 or earlier (or if you're running OS X 10.7 and you haven't upgraded to OS X 10.7.3 or later), clicking the Free Java Download button opens a "Sorry, Charlie!" page that tells you to download Java directly from Apple. Follow the instructions on that page to install Java on your computer.

- **(Optional) Follow the Do I Have Java? link.**

  When you follow this link, the web browser scans the computer for Java installations. For this book's examples, I recommend Java 6 (also known as Java 1.6) or later (Java 7, Java 8, or whatever). If your version of Java is older than Java 6 (or if the scan doesn't find Java on the computer), I recommend clicking one of the Download buttons at `www.java.com`.

- **(Optional) Pick and choose among Java versions.**

  If you click the All Java Downloads link at `www.java.com`, you can pick and choose from among several versions of Java — 32-bit and 64-bit versions for Windows, Mac, Linux, and Solaris computers.

  This alternative is useful for overriding the default Free Java Download button's choice. For example, you want the 64-bit version of Java even though the Free Java Download button gives you the 32-bit version. (See the sidebar "How many bits does your computer have?" earlier in this chapter.) Later, you might visit `www.java.com` with a Windows computer to download Java for your Macintosh.

- **(Optional) Cleanse your computer of all but the latest version of Java.**

  At `www.java.com`, the Remove Older Versions link promises to clean up any Java clutter you've collected over time. I've had some good luck and some bad luck in keeping multiple Java versions on a computer. In my opinion, this Remove Older Versions step is optional.

Visit the Remove Older Versions link if you're having trouble that you suspect is Java related. But if you've read several chapters of this book and the examples are running nicely, don't worry about an impending disaster from not having followed the Remove Older Versions link.

Setting Up the Android SDK

In this section, you get four useful tools in one download. Here's how:

1. **Visit** `http://developer.android.com/sdk`.
2. **Click the Download button on the web page.**
3. **Agree to all the legal mumbo-jumbo.**
4. **Choose between the 32-bit and 64-bit downloads.**

   For sage advice, see the earlier section "Are you running a 32-bit or 64-bit operating system?"

   After you make a choice, one last Download button appears. (At least, that's what happens early in 2013.)
5. **Click the last Download button and save the download to the local hard drive.**

   The downloaded file is one big `.zip` archive.
6. **Extract the contents of the downloaded archive file to the local hard drive.**

   On my Windows computer, I extract the `.zip` file's contents to the new folder `c:\Users\`*`MyUserName`*`\adt-bundle-windows-x86`. On my Mac, I extract the `.zip` file's contents to my existing `Applications` folder. (Refer to Figures 2-1 and 2-2.)

For help with archive files, see the earlier sidebar "Compressed archive files."

In Windows, the blank space in the name `Program Files` confuses some Java software. I don't think any of this book's software presents this problem, but I can't guarantee it. If you want, extract the `.zip` file's contents to the `C:\Program Files` or `C:\Program Files (x86)` folder. But make a mental note about your choice (in case you run into any trouble later).

The `.zip` archive that you download from `http://developer.android.com/sdk` contains these two components:

- **The eclipse component:** It contains a customized version of the popular Eclipse integrated development environment (IDE). You can compose, run, and debug Java applications in the Eclipse environment. This customized version of Eclipse includes the Android Development Toolkit (ADT) — extra plug-ins for working with Android apps.
- **The sdk component:** (Yes, only half of the large Android SDK download is the SDK component. If the names are misleading, don't blame me.) The SDK component contains the Android software library (one or more versions of the Android API). This component also contains a bunch of software tools for running and testing Android apps.

While you're still in the mood to follow my advice, note the location on the hard drive where the sdk component lands. (For example, in Figure 2-1, the SDK folder is `c:\Users\Barry\adk-bundle-windows-x86_64\sdk`.) I have a name for this location: the `ANDROID_HOME` folder.

Running Eclipse for the First Time

The first time you launch Eclipse, you perform a few extra steps. To get Eclipse running, follow these steps:

1. **Launch Eclipse.**

   In Windows, the Start menu may not have an Eclipse icon. In that case, look in Windows Explorer (it's File Explorer in Windows 8) for the folder containing the extracted Eclipse files. Double-click the icon representing the `eclipse.exe` file. (If you see an `eclipse` file but no `eclipse.exe` file, check the sidebar "Those pesky filename extensions," earlier in this chapter.)

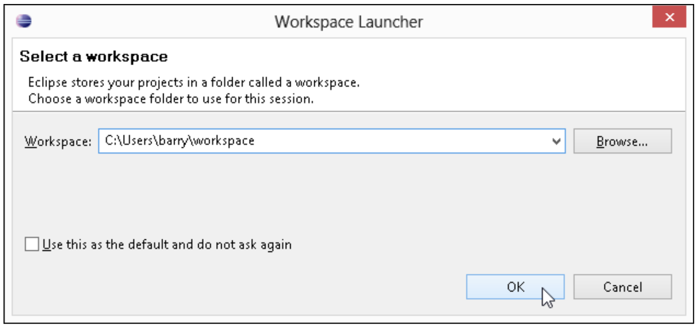
   On the Mac, go to the Spotlight and type **Eclipse** in the search field. When *Eclipse* appears as the Top Hit in the Spotlight's list, press Enter.

   When you launch Eclipse, you see the Workspace Launcher dialog box, as shown in Figure 2-5. The dialog box asks where, on the computer's hard drive, you want to store the code that you will create using Eclipse.

2. **In the Workspace Launcher dialog box, click OK to accept the default (or don't accept the default).**

   One way or another, it's no big deal.

   Because this is your first time using a particular Eclipse workspace, Eclipse starts with a Welcome screen, as shown in Figure 2-6.



Workspace Launcher

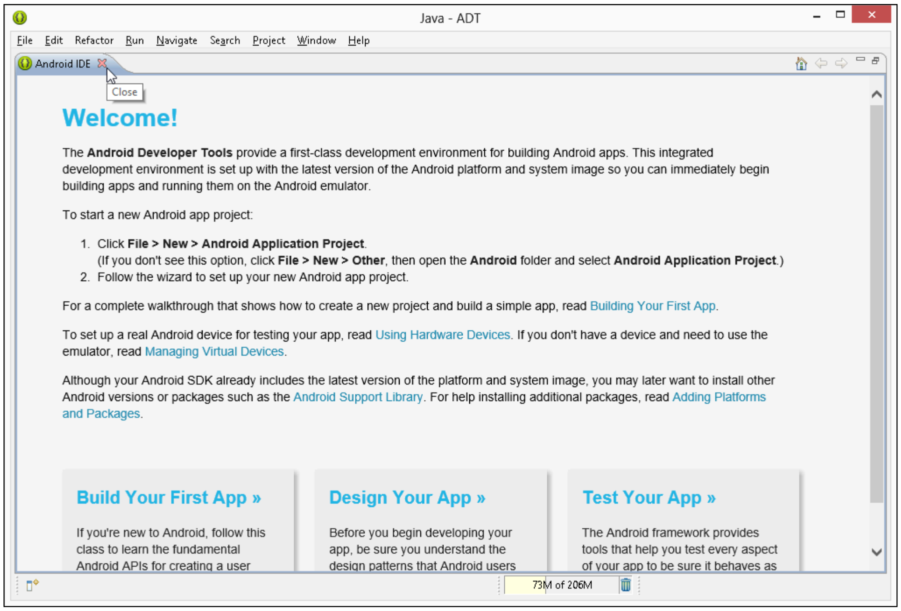
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\barry\workspace

☐ Use this as the default and do not ask again

Figure 2-5: The Eclipse Workspace Launcher.



Welcome!

The **Android Developer Tools** provide a first-class development environment for building Android apps. This integrated development environment is set up with the latest version of the Android platform and system image so you can immediately begin building apps and running them on the Android emulator.

To start a new Android app project:

1. Click **File > New > Android Application Project**.
   (If you don't see this option, click **File > New > Other**, then open the **Android** folder and select **Android Application Project**.)
2. Follow the wizard to set up your new Android app project.

For a complete walkthrough that shows how to create a new project and build a simple app, read Building Your First App.

To set up a real Android device for testing your app, read Using Hardware Devices. If you don't have a device and need to use the emulator, read Managing Virtual Devices.

Although your Android SDK already includes the latest version of the platform and system image, you may later want to install other Android versions or packages such as the Android Support Library. For help installing additional packages, read Adding Platforms and Packages.

Build Your First App »
If you're new to Android, follow this class to learn the fundamental Android APIs for creating a user

Design Your App »
Before you begin developing your app, be sure you understand the design patterns that Android users

Test Your App »
The Android framework provides tools that help you test every aspect of your app to be sure it behaves as

73M of 206M

Figure 2-6: The Welcome screen for Android's customized version of Eclipse.

3. Dismiss the Welcome screen.

In most versions of Eclipse, you can dismiss the Welcome screen by clicking the little *x* icon that appears on a tab above the screen.

A view of the main screen, after opening Eclipse with a brand-new workspace, is shown in Figure 2-7.

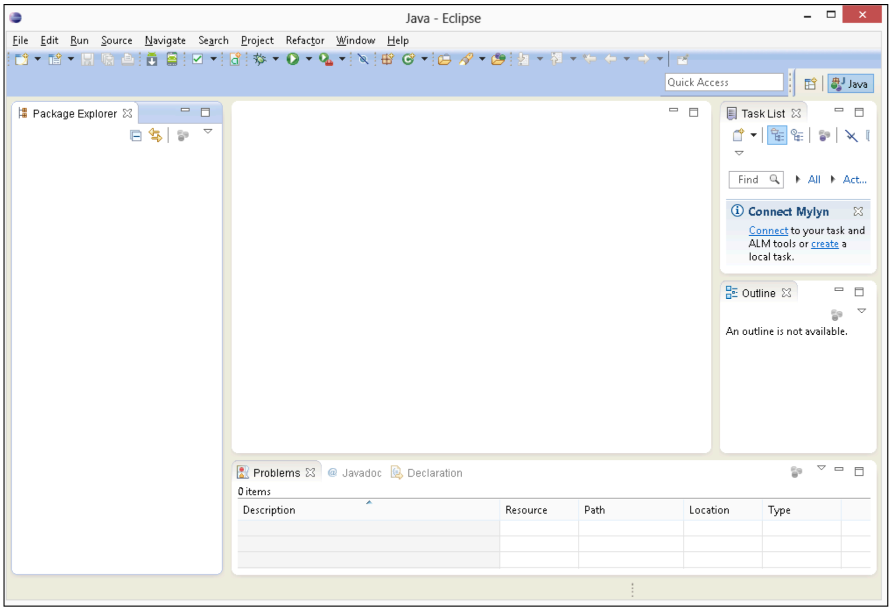


Figure 2-7: The Eclipse workbench with a brand-new workspace.

Dude, where's my Android SDK?

When you launch Eclipse, the Eclipse IDE looks on the hard drive for the prewritten, reusable Android code files. (After all, Eclipse uses these files to help you write and run Android apps.) If Eclipse has trouble finding these files, you see a nasty-looking Could Not Find SDK Folder message. To tell Eclipse where to install the Android SDK files, follow these steps:

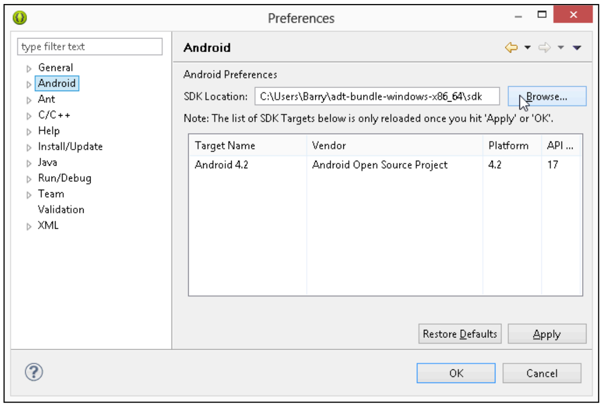
1. **In Windows, in the Eclipse main menu, choose Window➪Preferences. On the Mac, in the Eclipse main menu, choose Eclipse➪Preferences.**

   The Eclipse Preferences dialog box opens.

2. **In the tree list on the left side of the Preferences dialog box, select Android.**

   Don't expand the Android branch of the tree. Simply click the word *Android.*

   The SDK Location field appears in the main body of the Preferences dialog box, as shown in Figure 2-8.



SDK Location: C:\Users\Barry\adt-bundle-windows-x86_64\sdk

Note: The list of SDK Targets below is only reloaded once you hit 'Apply' or 'OK'.

Target Name	Vendor	Platform	API ...
Android 4.2	Android Open Source Project	4.2	17

Figure 2-8: Telling Eclipse about the location of the Android SDK.

3. **Click the Browse button and (of course) browse to the** `ANDROID_HOME` directory.

   For example, in Figure 2-1, the `ANDROID_HOME` directory is `c:\Users\Barry\adt-bundle-windows-x86_64\sdk`.

4. **Click Apply and OK, and all those good things to return to the main Eclipse workbench.**

Look again at Figure 2-8 and notice the text box in the window's upper-left corner — the box containing the words *type filter text*. The text box is for filtering the names of Eclipse preferences. Figure 2-8 displays only 11 preferences (such as General, Android, Ant, and C++). But this list of preferences expands to a tree with approximately 150 branches. Each branch refers to its own set of choices in the main body of the Preferences window. If you want to see a bunch of Eclipse preferences related to font (for example), type **font** in the little text box. Eclipse then displays only branches of the tree containing the word *font.*

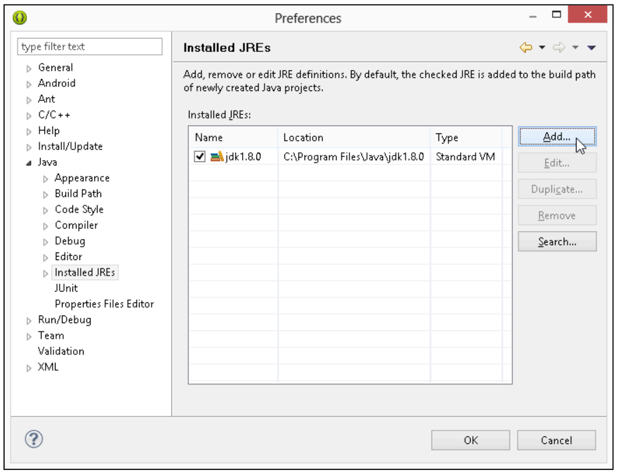
Eclipse, meet Java!

Eclipse normally looks on the computer for Java installations and selects an installed version of Java to use for running your Java programs. The computer may have more than one version of Java, so double-check Eclipse's Java version selection. The steps in this section show you how.

The steps in this section are optional. Follow them only if you suspect that Eclipse isn't using your computer's favorite version of Java.

1. ***In Windows:* From the Eclipse main menu, choose Window➪Preferences. *On the Mac:* From the Eclipse main menu, choose Eclipse➪Preferences.**

   As a result, the Eclipse Preferences dialog box appears. (You can follow along in Figure 2-9.)



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

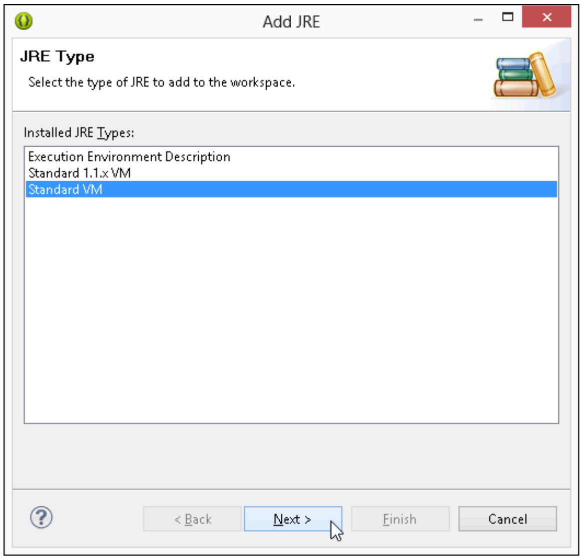
Name	Location	Type
✓ jdk1.8.0	C:\Program Files\Java\jdk1.8.0	Standard VM

Figure 2-9: The Installed JREs page of the Eclipse Preferences dialog box.

2. **In the tree on the left side of the Preferences dialog box, expand the Java branch.**
3. **Within the Java branch, select the Installed JREs subbranch.**
4. **Look at the list of Java versions (Installed JREs) in the main body of the Preferences dialog box.**

   In the list, each version of Java has a check box. Eclipse uses the version whose box is checked. If the checked version isn't your preferred version (for example, if it isn't version 6 or later), you have to make changes.
5. **If your preferred version of Java appears in the Installed JREs list, select that version's check box.**
6. **If your preferred version of Java doesn't appear in the Installed JREs list, click the Add button.**

   When you click the Add button, the JRE Type dialog box appears, as shown in Figure 2-10.


Add JRE

JRE Type

Select the type of JRE to add to the workspace.

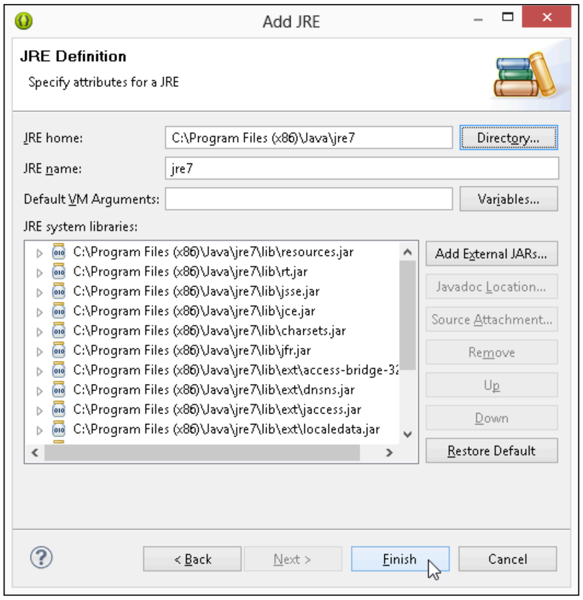
Installed JRE Types:

Execution Environment Description
Standard 1.1.x VM
Standard VM

Figure 2-10: The JRE Type dialog box.

7. **In the JRE Type dialog box, double-click Standard VM.**

   As a result, the JRE Definition dialog box appears, as shown in Figure 2-11. What you do next depends on a few different factors.


Add JRE

JRE Definition

Specify attributes for a JRE

JRE home: C:\Program Files (x86)\Java\jre7

JRE name: jre7

Default VM Arguments:

JRE system libraries:

```
C:\Program Files (x86)\Java\jre7\lib\resources.jar
C:\Program Files (x86)\Java\jre7\lib\rt.jar
C:\Program Files (x86)\Java\jre7\lib\jsse.jar
C:\Program Files (x86)\Java\jre7\lib\jce.jar
C:\Program Files (x86)\Java\jre7\lib\charsets.jar
C:\Program Files (x86)\Java\jre7\lib\jfr.jar
C:\Program Files (x86)\Java\jre7\lib\ext\access-bridge-32
C:\Program Files (x86)\Java\jre7\lib\ext\dnsns.jar
C:\Program Files (x86)\Java\jre7\lib\ext\jaccess.jar
C:\Program Files (x86)\Java\jre7\lib\ext\localedata.jar
```

Figure 2-11: The JRE Definition dialog box (after you've followed Steps 8 and 9).

8. **Fill in the JRE Home field in the dialog box.**

   How you do this depends on the operating system:

   • *In Windows:* Browse to the directory in which you've installed your preferred Java version. On my many Windows computers, the directory is either `C:\Program Files\Java\jre7`, `C:\Program Files\Java\jdk1.7.0`, `C:\Program Files (x86)\Java\jre8`, or something of that sort.

   • *On the Mac:* Use the Finder to browse to the directory in which you've installed your preferred Java version. Type the name of the directory in the dialog box's JRE home field.

     My Mac has one Java directory, named `/System/Library/Java/Java Virtual Machines/1.6.0jdk/Contents/Home`, and another Java directory named `/Library/Java/JavaVirtualMachines/JDK 1.7.0.jdk/Contents/Home`.

     Directories such as `/System` and `/Library` don't normally appear in the Mac's Finder window. To browse to one of these directories (to the `/Library` directory, for example) choose Go➪Go to Folder on the Finder's menu bar. In the resulting dialog box, type **/Library** and then press Go.

     As you navigate toward the directory containing your preferred Java version, you might encounter a `JDK 1.7.0.jdk` icon, or another item whose extension is `.jdk`. To see the contents of this item, control-click the item's icon and then select Show Package Contents.

   You might have one more thing to do back in the JRE Definition dialog box.

9. **Look at the JRE Name field in the JRE Definition dialog box; if Eclipse hasn't filled in a name automatically, type a name (almost any text) in the JRE Name field.**

10. **Dismiss the JRE Definition dialog box by clicking Finish.**

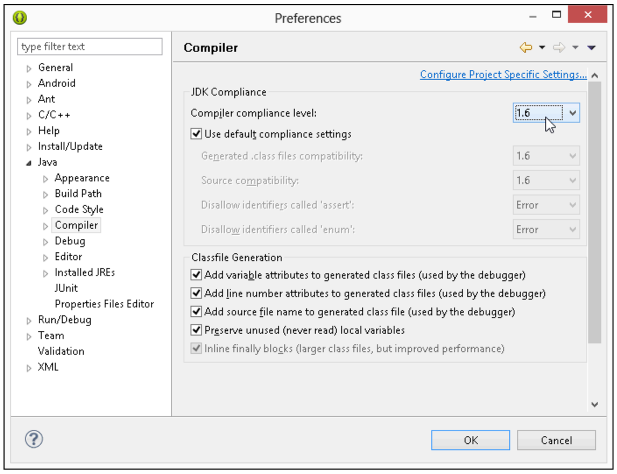
    The Preferences dialog box in Eclipse returns to the foreground. Its Installed JREs list contains the newly added version of Java.

11. **Select the check box next to the newly added version of Java.**

    You're almost done. (You have a few more steps to follow.)

12. **Within the Java branch on the left side of the Preferences dialog box, select the Compiler subbranch.**

    In the main body of the Preferences dialog box, you see the Compiler Compliance Level drop-down list, as shown in Figure 2-12.



Compiler

JDK Compliance

Compiler compliance level: 1.6

☑ Use default compliance settings

Generated .class files compatibility: 1.6

Source compatibility: 1.6

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

Figure 2-12: Setting the compiler compliance level.

13. **In the Compiler Compliance Level drop-down list, select 1.5 or 1.6.**

    Android works with only Java 1.5 or 1.6.

14. **Whew! Click the Preferences dialog box's OK button to return to the Eclipse workbench.**

Importing this book's sample programs

This import business can be tricky. As you move from one dialog box to the next, you see that many of the options have similar names. That's because Eclipse offers many different ways to import many different kinds of items. Anyway, if you follow these instructions, you'll be okay:

1. **Follow the steps in this chapter's earlier section "Getting This Book's Sample Programs."**
2. **On the Eclipse main menu, choose File⇨Import, as shown in Figure 2-13.**

   As a result, Eclipse displays the Import dialog box.
3. **In the tree in the Import dialog box, expand the General branch.**
4. **In the General branch, double-click the Existing Projects into Workspace subbranch, as shown in Figure 2-14.**

   As a result, the Import Projects dialog box appears.
5. **In the Import Projects dialog box, choose the Select Root Directory or the Select Archive File radio button, as shown in Figure 2-15.**

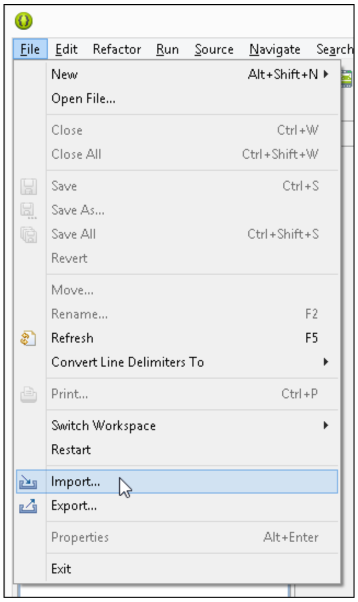
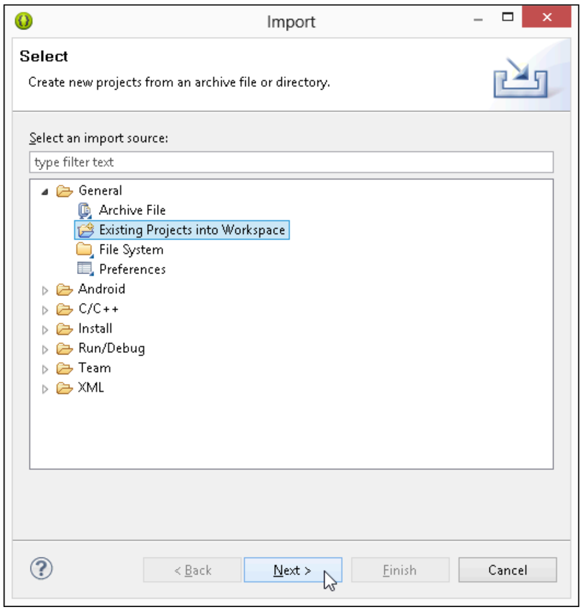


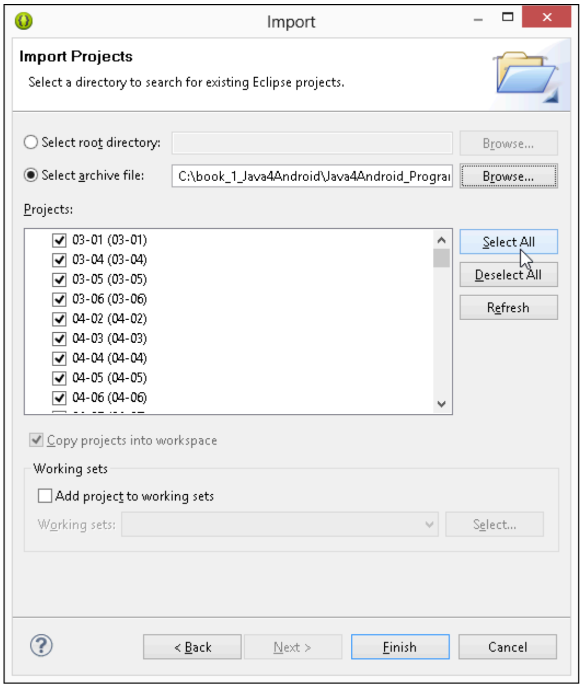
Figure 2-13: Starting to import this book's code.


Import

Select

Create new projects from an archive file or directory.

Figure 2-14: Among all the options, select Existing Projects into Workspace.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

( ) Select root directory: [ ] Browse...

(●) Select archive file: C:\book_1_Java4Android\Java4Android_Progra Browse...

Projects:

- ☑ 03-01 (03-01)
- ☑ 03-04 (03-04)
- ☑ 03-05 (03-05)
- ☑ 03-06 (03-06)
- ☑ 04-02 (04-02)
- ☑ 04-03 (04-03)
- ☑ 04-04 (04-04)
- ☑ 04-05 (04-05)
- ☑ 04-06 (04-06)

☑ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 2-15: The Import Projects dialog box.

This book's code lives in a folder named `Java4Android_Programs` or in an archive file named `Java4Android_Programs.zip`.

Safari on a Mac generally uncompresses `.zip` archives automatically, and Windows browsers (Internet Explorer, Firefox, Chrome, and others) do not uncompress `.zip` archives automatically. For the complete scoop on archive files, see the earlier sidebar "Compressed archive files."

6. **Click the Browse button to find the** `Java4Android_Programs.zip` **file or the** `Java4Android_Programs` **folder on the computer's hard drive.**

   If you're unsure where to find these items, look first in a folder named `Downloads`.

   After you find `Java4Android_Programs`, the Import Projects dialog box in Eclipse displays the names of the projects inside the file. (Refer to Figure 2-15.)

7. **Click the Select All button.**

   This book's examples are so exciting that you'll want to import all of them!

8. **Click the Finish button.**

   As a result, the main Eclipse workbench reappears. The left side of the workbench displays the names of this book's Java projects, as shown in Figure 2-16.

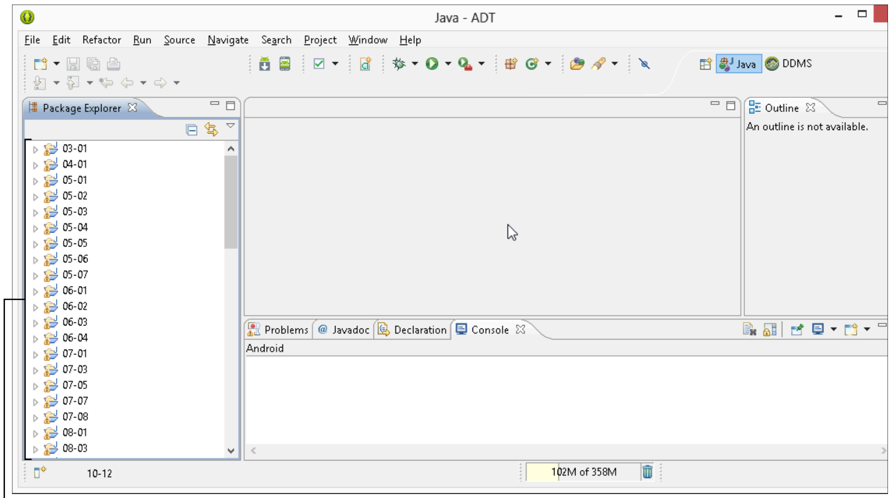


Figure 2-16: Eclipse displays a bunch of Java projects.

After importing the code from this book, you may see lots of red error markers indicating trouble with the book's projects. If you do, stay calm. The markers might disappear after several seconds. If they don't, check the lower area of the Eclipse workspace for a message similar to `Unable to resolve target 'android-15'`.

If you see such a message, it means that my book's code insists on an API level that you haven't installed on your computer. To fix the problem, do the following:

1. **On the Eclipse main menu, choose Window➪Android SDK Manager.**

   As a result, the computer displays the Android SDK Manager. (No surprise here!)

2. **Select the check box labeled Android 4.0.3 (API 15) or in whichever box is labeled with the missing API level number.**
3. **Click the Install button in the lower-right corner of the Android SDK Manager window.**
4. **Wait for installation to finish.**
5. **Close the Android SDK Manager.**
6. **Restart Eclipse.**

When Eclipse restarts, you see the red error markers for a few seconds. But after a brief (and possibly tense) waiting period, the error markers go away. You're ready to roll.

Creating an Android Virtual Device

You might be itching to run some code, but first you must have something that can run an Android program. By *something,* I mean either an Android device (a phone, a tablet, an Android-enabled toaster — whatever) or a virtual device. An *Android Virtual Device* (AVD) is a test bed for Android code on the development computer.

The Android SDK comes with its own *emulator* — a program that behaves like a phone or a tablet but runs on the development computer. The emulator translates Android code into code that the development computer can execute. But the emulator doesn't display a particular phone or tablet device on the screen. The emulator doesn't know what kind of device you want to display. Do you want a camera phone with 800-x-480-pixel resolution, or have you opted for a tablet device with its own built-in accelerometer and gyroscope? All these choices belong to a particular AVD. An AVD is simply a bunch of settings, telling the emulator all the details about the device to be emulated.

Before you can run Android apps on your computer, you must first create at least one AVD. In fact, you can create several AVDs and use one of them to run a particular Android app.

To create an AVD, follow these steps:

1. **In the Eclipse main menu, choose Window➪Android Virtual Device Manager.**

   The Android Virtual Device Manager window opens.

2. **In the Android Virtual Device Manager window, click New, as shown in Figure 2-17.**

   The Create New Android Virtual Device (AVD) window opens. That's nice!

3. **In the AVD Name field, type a new name for the virtual device.**

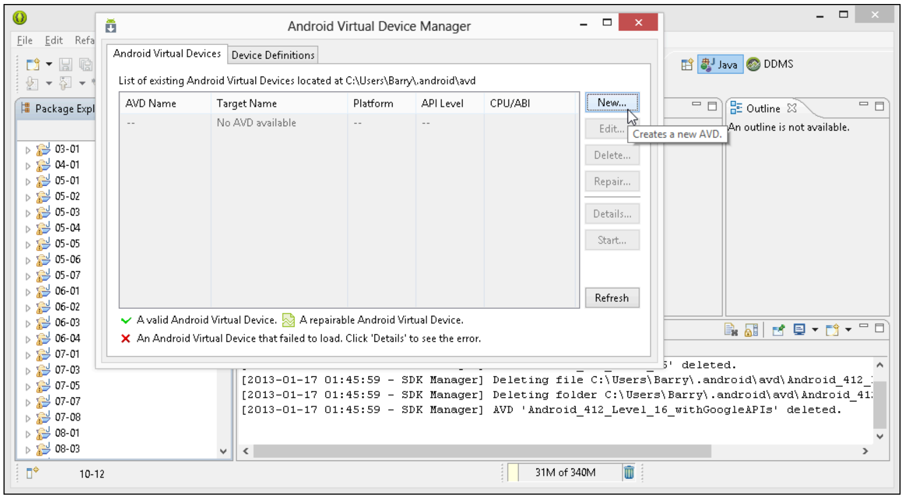
   You can name your device My Sweet Petunia, but in Figure 2-18, I name my device `Nexus7_Android4.2`. The name serves to remind me of this device's capabilities.

4. **In the Device drop-down menu, select a device type.**

   In Figure 2-18, I select Nexus 7 (7.27", 800 x 1280: tvdpi).

5. **Determine the kind of secure digital (SD) card your device has.**

   In Figure 2-18, I choose an SD card with a modest 1000 MiB, which is roughly 1 gigabyte. Alternatively, I could have selected the File radio button and specified the name of a file on my hard drive. That file would be storing information as though it were a real SD card on a real device.



List of existing Android Virtual Devices located at C:\Users\Barry\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

An outline is not available.

03-01, 04-01, 05-01, 05-02, 05-03, 05-04, 05-05, 05-06, 05-07, 06-01, 06-02, 06-03, 06-04, 07-01, 07-03, 07-05, 07-07, 07-08, 08-01, 08-03

```
5' deleted.
[2013-01-17 01:45:59 - SDK Manager] Deleting file C:\Users\Barry\.android\avd\Android_412_
[2013-01-17 01:45:59 - SDK Manager] Deleting folder C:\Users\Barry\.android\avd\Android_41
[2013-01-17 01:45:59 - SDK Manager] AVD 'Android_412_Level_16_withGoogleAPIs' deleted.
```

Figure 2-17: The Android Virtual Device Manager.

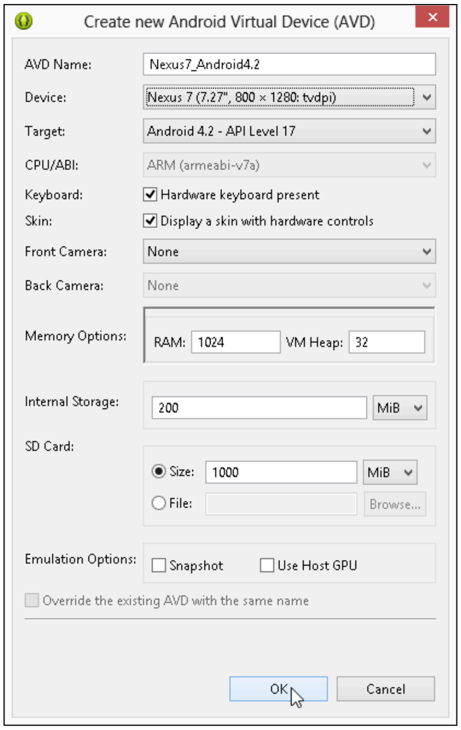
TECHNICAL STUFF

Recently, my department hired a new person. We offered a salary of $50K, which (we thought) meant $50,000 per year. Little did we know that the new person expected to be paid $51,200 each year. Computer scientists use the letter *K* (or the prefix *Kilo*) to mean 1,024 because 1,024 is a power of 2 (and powers of 2 are quite handy in computer science). The trouble is, the formal meaning of *Kilo* in the metric system is 1,000, not 1,024. To help clear things up (and to have fun creating new words), a commission of engineers created the *Kibibyte* (KiB) meaning 1,024 bytes, the *Mebibyte* (MiB) which is 1,048,576 bytes, and the *Gibibyte* (GiB), meaning 1,073,741,824 bytes. Most people (computer scientists included) don't know about KiBs or MiBs, and they don't worry about the difference between MiBs and ordinary megabytes. I'm surprised that the creators of the Android Virtual Device Manager thought about this issue.

6. **Leave the other choices at their defaults (or don't, if you don't want to) and click the Create AVD button.**

   The computer returns you to the Android Virtual Device Manager window, where you see a brand-new AVD in the list, as shown in Figure 2-19.

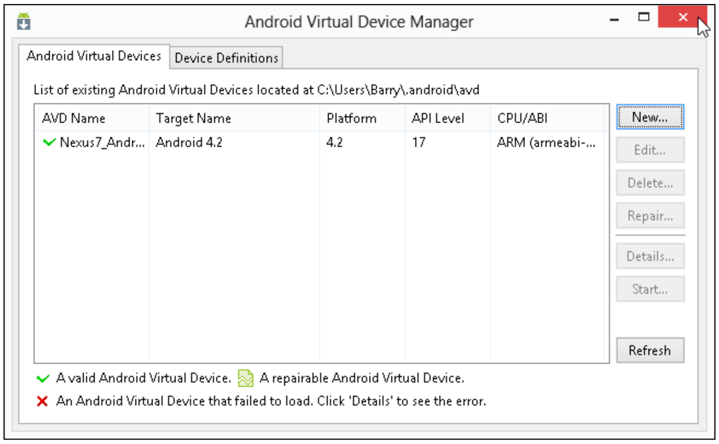
And that does it! You're ready to run your first Android app. I don't know about you, but I'm excited. (Sure, I'm not watching you read this book, but I'm excited on your behalf.) Chapter 3 guides you through the run of a standard Oracle Java program, and Chapter 4 does the same for an Android application. Go for it!



Create new Android Virtual Device (AVD)

AVD Name: Nexus7_Android4.2
Device: Nexus 7 (7.27", 800 × 1280: tvdpi)
Target: Android 4.2 - API Level 17
CPU/ABI: ARM (armeabi-v7a)
Keyboard: ☑ Hardware keyboard present
Skin: ☑ Display a skin with hardware controls
Front Camera: None
Back Camera: None
Memory Options: RAM: 1024 VM Heap: 32
Internal Storage: 200 MiB
SD Card: ● Size: 1000 MiB ○ File: Browse...
Emulation Options: ☐ Snapshot ☐ Use Host GPU
☐ Override the existing AVD with the same name

Figure 2-18: Creating a new Android virtual device.



Android Virtual Device Manager

List of existing Android Virtual Devices located at C:\Users\Barry\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Nexus7_Andr...	Android 4.2	4.2	17	ARM (armeabi-...

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 2-19: You've created an Android virtual device.