Quickly build your first Android apps

Learn Android App Development

Wallace Jackson

Apress®

Contents at a Glance

Introduction

The Android OS is currently the most popular OS in the world today, running on everything from watches to HD smartphones to touchscreen tablets to eBook readers to interactive television sets.

Since there are billions of Android consumer electronics devices owned by people all over the world, it stands to reason that developing applications for these people might just be an extremely lucrative undertaking, given that you have the right concept and design. This book will go a long way toward helping you to learn how to go about creating an attractive Android application which spans multiple types of Android device types and supports multiple Android OS versions.

I wrote *Learn Android App Development* as the next level up from my *Android Apps for Absolute Beginners* title, targeting those readers who are more technically proficient, and who are familiar with computer programming concepts and techniques. That being said, this would be a good follow-on title to the *Android Apps for Absolute Beginners* title, and both cover the latest Android 4.2.2 Jelly Bean Plus Android operating system version.

I designed this book to be a far more comprehensive overview of the Android application development work process than most Android app development books. For this reason, this book covers the use of a wide variety of other open source software packages, such as GIMP and Audacity, for instance, and how their usage fits into the overall Android application development work process. This approach serves to set this book distinctly apart from the other Android application development titles currently on the market.

The book starts out with installing the latest Java and Android SDKs and the Eclipse IDE with the Android ADT Bundle, and then progresses through creating a basic Android application and then adding to that application with each chapter. We continue this process until all the major concepts are covered and implemented in one comprehensive Android application. This approach more closely parallels real world application development, where an application is continuously added to over time, making it more and more robust and feature filled as time goes on, while making sure each new feature does not cause the application to crash.

We look at Java objects and constructors, user interface design using XML mark-up, digital imaging and graphics design, digital video and animation, audio sampling and audio sequencing, and other advanced new media concepts and multimedia application features, as that is what is popular in Android application development today. We look at core Android OS areas, including Content Providers (SQLite Databases), Broadcast Receivers, Services, and using Events, Intents, and Activities, all in great detail.

We cover the foundational knowledge that you will need to be able to work in the more advanced areas that the Android OS encompasses. Some of these include digital image compositing, digital video optimization, procedural animation, database design, multi-screen resolution support, 3D rendering, and similar advanced topics regarding which an Android developer needs to know at least the basics in order to work intelligently within their application design and development work process.

If you want a comprehensive overview of Android, Eclipse, Java, XML, and the Android Developer Tools environment, as well as knowledge about how to optimally use these technologies with leading open source new media content design and development tools, then this book will be of great interest to you.

Chapter 1

Building an Android IDE for Version 4.2: Acquiring, Installing, and Configuring an Android Development Environment

The first thing that we need to do before we can learn Android Application Development is to put together a working Android Application Development Environment on our development workstation. Hopefully you have an entry-level quad-core AMD or Intel computer with 4GB or more of DDR memory and Windows 7 or Windows 8; the computer that I will be using for this book is a $398 ACER 64-bit quad-core AMD, running at 3.1GHz with 4GB of DDR3 memory and 1TB hard disk drive and Windows 7 that I picked up at Walmart.

Fortunately for us Android App developers, very powerful 64-bit computers are readily available for a few hundred dollars! If you have a 32-bit computer, that will also work just as well for Android Application Development, because the Android 4.2 Development Environment comes in both 32-bit and 64-bit flavors. Additionally, all the software that we will be using for app development in this book is free for commercial use, also known as open source, so the cost of starting up your own Android Application Software Development business is quite low these days indeed.

If for some reason, you want to use a development environment that predates Android 4.2.2 (which I would strongly recommend against), there is an Appendix at the end of this book that covers the much more involved work process for installing the Android 4.1.2 development environment.

Our Plan of Attack

In this chapter, we will make sure that our system has the very latest versions of the Oracle Java 6 Software Development Kit (Java SDK, also known as the JDK, or Java Development Kit) programming environment, as well as the Android Software Development Environment, which consists of the Google Android Software Development Kit (SDK), Android Development Tools (ADT) Plug-ins for Eclipse, and the Eclipse 4.2 Integrated Development Environment (IDE).

All these installed together at once are cumulatively known as the **Android ADT Bundle**, which, as of Android 4.2, you can now download all at once, in under 400MB, at the **Android Developer** website at `http://developer.android.com`.

Before Android 4.2 Jelly Bean+ (Android API Level 17), developers had to download and install each of these components individually, which was quite tedious. If you want to do it this way, or see what it would be like to have to do it this way (and gain a greater understanding of what is going on between Eclipse and Android SDK and Android ADT) you can see the long-version of the install in Appendix A of this book.

Once our JDK is downloaded and installed, we will then download and install an Android Integrated Development Environment (IDE) called the **Android ADT Bundle**, the foundation of which is the Eclipse 4.2 Juno for Java EE IDE. Eclipse makes developing Android Apps easier by providing us with a slick Graphical User Interface (GUI) with which we can write, run, test, and debug our Android application code.

Eclipse runs “on top of” the Java Runtime Environment (JRE), because Eclipse is written in Java, and thus it uses the Java Platform to run its own codebase, which makes up the Eclipse IDE user interface and feature set, which you will see (as you progress throughout this book) is quite extensive indeed.

This is the primary reason that we downloaded and installed the Java 6 JDK first, so that the Java SDK and JRE are in place on our workstation. In this way, once we get into installing the ADT Bundle, which is based on Eclipse, Eclipse can easily find the Java Runtime Environment (so that Eclipse can launch and run). Once Eclipse is able to find Java it can use the Java SDK to build the Java programming code foundation for our Android Development Environment, because Android APIs (SDK) are based on the Java 6 APIs (SDK).

Once we have the ADT Bundle downloaded and installed and working smoothly on top of Java 6, we essentially have installed, all in one bundle, the Google Android Software Development Kit (SDK), the Eclipse 4.2.2 IDE, and all the ADT plug-ins needed to develop for Android 4.2.2 API Level 17.

For a bird’s—eye view, if this process were formulated into an equation, it would look something like this:

JDK (Java 6 SDK) + ADT Bundle (Eclipse + Android SDK + ADT Eclipse Plug-Ins) = Custom Android IDE

As part of the Android Bundle installation and configuration for development usage work process (the second major part), we will install some **Android Virtual Device** (AVD) Emulators, which will live inside Eclipse 4.2, and which will allow us to test our applications on various Android Virtual Devices, such as a Virtual Nexus 7 Tablet, or a Virtual Nexus S Smartphone, or even a Virtual GoogleTV Set. So let’s get started with this process now, so we can get it over with, and start developing apps in the next chapter!

Foundation of Android Programming: Java 6

The foundation of Android Application Development, both from a programming as well as an Integrated Development Environment (IDE) standpoint, is Java 6. Android Applications are written using the Java 6 programming language (and using XML as well, which we'll get into in more detail in Chapter 2), and Android Apps are developed inside the Eclipse 4.2.2 IDE, which is also written in the Java 6 programming language, and which runs on top of the Java 6 Runtime Environment, also known as the JRE. To put it mildly, the exact order in which you set up the various software components that make up your Android Development Environment is very important, and is the reason for this first chapter.

So that we have both the Java programming language, which we gain access to via the JDK or Java Developer Kit, as well as the Java Runtime Environment (JRE), which is part of the JDK, go to the Oracle TechNetwork and download the latest JDK 6 installation software and install it on your machine. We do this first because Eclipse needs Java to run, that is, Eclipse can be said to run "on top of" the Java platform and language. Android also requires Java, as well as Eclipse, for its Android Developer Tools (ADT) plug-ins, so we install the Java Platform and Java Environment first, then the Android ADT Bundle.

Let's get started.

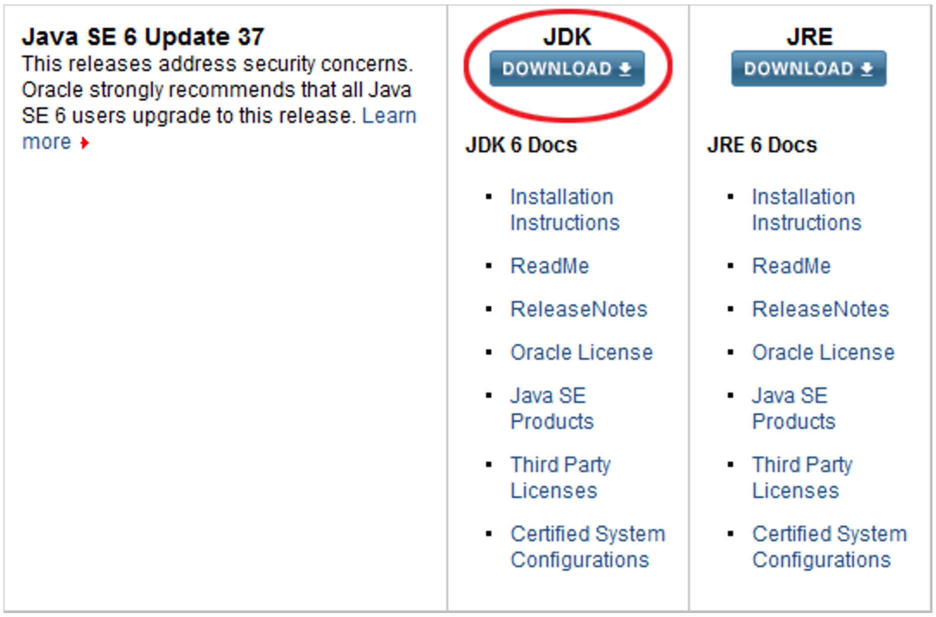
Installing the JDK

The first thing we must do is get to the Java SDK download page, and there are two ways to do this; one is generic, one is precise. The generic way, which will always work, even if Oracle changes the location of its Java SDK download page (which it probably won't), is to use Google Search with the keyword phrase **"Java SDK Download",** which should bring up the Oracle TechNetwork Java download URL. The second way is to type the URL for the page directly into the browser. Here is the URL:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

What this points to is the Internet (HTTP) and the Oracle website in their TechNetwork area (folder) in the Java area (sub-folder) for the Java SE or Standard Edition area (sub-sub-folder) in the Downloads area (sub-sub-sub-folder). There are three primary versions of Java: SE or Standard Edition for individual users, EE or Enterprise Edition for large collections of users, and ME or Micro Edition for older mobile flip-phones. Most modern smartphones use Android and Java SE, rather than Java ME. One of the really cool things about Android is that it uses the full Standard Edition of Java (known as Java SE) just like a PC does. This is because Android runs "on top of" a full version of the Linux OS Kernel, so an Android consumer electronics device is essentially a full-blown Linux computer, for all practical purposes.

Once you type in this URL, you arrive at the Java 6 JDK download page, and you need to find the Java 6 JDK download portion of the page, which looks like the (partial) page section shown in Figure 1-1.



Java SE 6 Update 37
This releases address security concerns. Oracle strongly recommends that all Java SE 6 users upgrade to this release. Learn more

JDK 6 Docs

- Installation Instructions
- ReadMe
- ReleaseNotes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

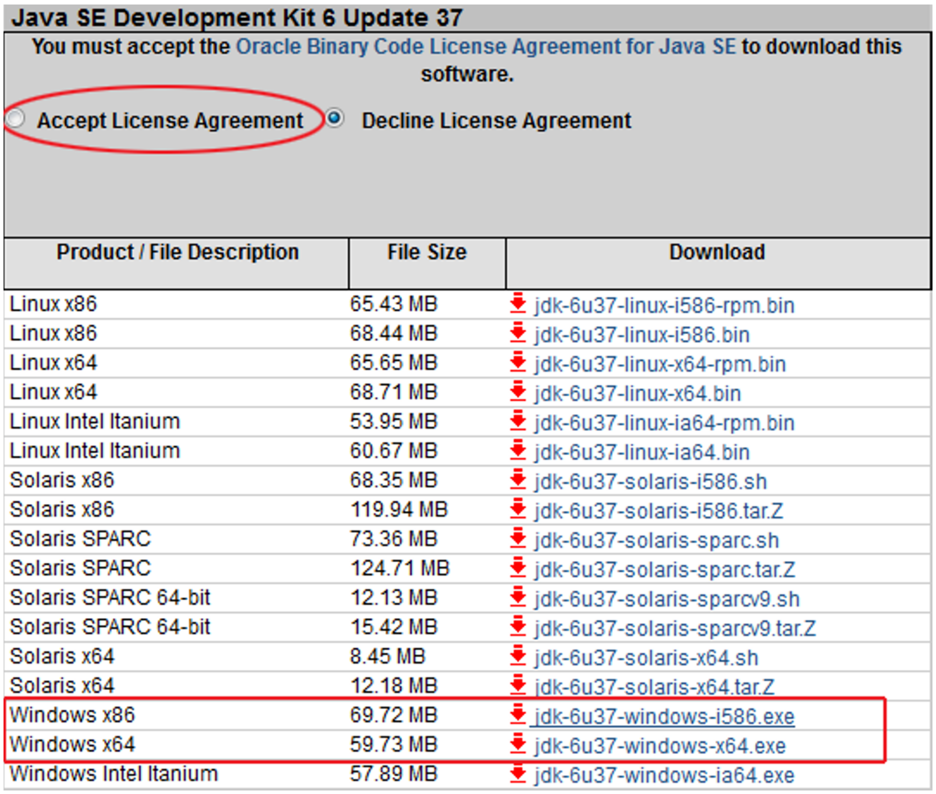
JRE 6 Docs

- Installation Instructions
- ReadMe
- ReleaseNotes
- Oracle License
- Java SE Products
- Third Party Licenses
- Certified System Configurations

Figure 1-1. *The Java SE 6 JDK download section of the Oracle TechNetwork Java SE webpage*

Scroll about halfway down the page and click the blue **DOWNLOAD** button under the JDK (remember, the JDK contains *both* the JDK and the JRE, so don't download the JRE at all), as shown in Figure 1-1.

This takes you to the Java 6 JDK download page shown in Figure 1-2, where you first accept the software licensing agreement, and then download either the Windows 32-bit version or the Windows 64-bit version of Java 6.



Java SE Development Kit 6 Update 37

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux x86	65.43 MB	jdk-6u37-linux-i586-rpm.bin
Linux x86	68.44 MB	jdk-6u37-linux-i586.bin
Linux x64	65.65 MB	jdk-6u37-linux-x64-rpm.bin
Linux x64	68.71 MB	jdk-6u37-linux-x64.bin
Linux Intel Itanium	53.95 MB	jdk-6u37-linux-ia64-rpm.bin
Linux Intel Itanium	60.67 MB	jdk-6u37-linux-ia64.bin
Solaris x86	68.35 MB	jdk-6u37-solaris-i586.sh
Solaris x86	119.94 MB	jdk-6u37-solaris-i586.tar.Z
Solaris SPARC	73.36 MB	jdk-6u37-solaris-sparc.sh
Solaris SPARC	124.71 MB	jdk-6u37-solaris-sparc.tar.Z
Solaris SPARC 64-bit	12.13 MB	jdk-6u37-solaris-sparcv9.sh
Solaris SPARC 64-bit	15.42 MB	jdk-6u37-solaris-sparcv9.tar.Z
Solaris x64	8.45 MB	jdk-6u37-solaris-x64.sh
Solaris x64	12.18 MB	jdk-6u37-solaris-x64.tar.Z
Windows x86	69.72 MB	jdk-6u37-windows-i586.exe
Windows x64	59.73 MB	jdk-6u37-windows-x64.exe
Windows Intel Itanium	57.89 MB	jdk-6u37-windows-ia64.exe

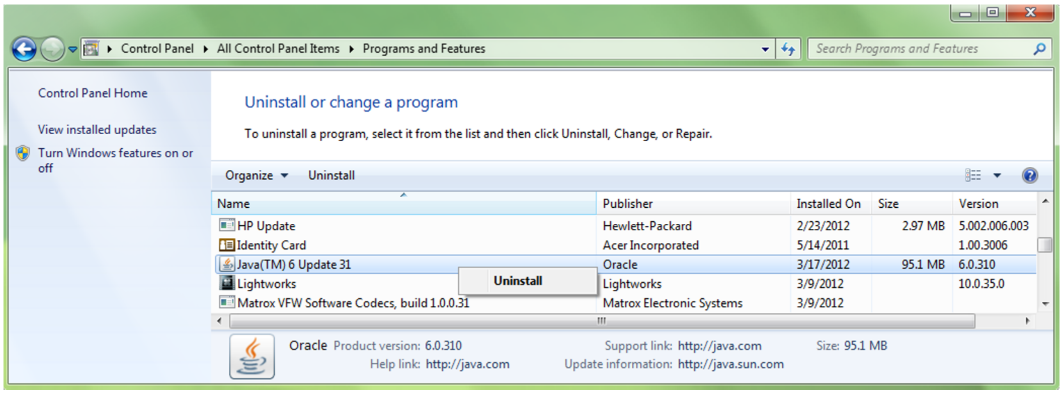
***Figure 1-2.** Accept the Java 6 License Agreement and download Windows x86 32-bit – or Windows x64 64-bit .exe file*

If you are using Linux OS or Solaris OS, you can find 32-bit and 64-bit versions for these operating systems here as well. Note that before you can download the Windows x86 32-bit .exe file or the Windows x64 64-bit .exe file, you must click the radio button selector next to the "Accept License Agreement" option at the top of the webpage section, as shown in Figure 1-2. Once you do this, the download links on the right become bold, and you can click them to download the installation file.

Click the **jdk-6u43-windows-i586.exe** link (or whatever that latest update revision happens to be) if you have a 32-bit Windows OS such as Windows XP, or the **jdk-6u43-windows-x64.exe** link (or whatever that latest update revision happens to be) if you have a 64-bit Windows OS such as Windows 7 or Windows 8, and then download the JDK 6 installer file to your Downloads folder on your workstation. Note that you do *not* need to download both files.

Removing Any Outdated JDKs

Before you install the current Java 6 JDK, you should check your system to make sure that no preexisting (older) versions of Java are already installed on your workstation. This is done in Windows via the Control Panel, which is accessed via the Windows Start Menu. Launch the Control Panel now, and find the icon labeled "Programs and Features" and launch the Programs and Features dialog shown in Figure 1-3.


Control Panel ▸ All Control Panel Items ▸ Programs and Features
Search Programs and Features
Control Panel Home
View installed updates
Turn Windows features on or off
Organize
Uninstall
Uninstall

Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Name	Publisher	Installed On	Size	Version
HP Update	Hewlett-Packard	2/23/2012	2.97 MB	5.002.006.003
Identity Card	Acer Incorporated	5/14/2011		1.00.3006
Java(TM) 6 Update 31	Oracle	3/17/2012	95.1 MB	6.0.310
Lightworks	Lightworks	3/9/2012		10.0.35.0
Matrox VFW Software Codecs, build 1.0.0.31	Matrox Electronic Systems	3/9/2012		

Oracle Product version: 6.0.310 Support link: http://java.com Size: 95.1 MB
Help link: http://java.com Update information: http://java.sun.com

***Figure 1-3.** Right-click the existing Java installation, and select Uninstall to remove it from your system*

Notice on my workstation that I already have 32-bit Java 6 Update 31 installed. Because I want to install the most recent 64-bit development software possible during this installation process, I am going to uninstall the previous Java 6 Update 31 installation, as well as any older versions of Eclipse and Android (if they exist, which they didn't on this particular workstation) using the exact same work process.

To do this, you need to select the older version of Java, and then right-click it, and select the **Uninstall** option, as shown in Figure 1-3. Windows then proceeds to remove the older version(s) of Java (and Eclipse and Android, if needed) from your system.

Next, go into your **Downloads** folder (or your desktop, or wherever you saved your file download), and double-click the **jdk-6u43-windows-i586.exe** or the **jdk-6u43-windows-x64.exe** file to start the Java 6 installation. Accept the default settings for the installation, which should proceed fairly quickly if you have a modern day workstation with a fast hard disk drive and at least 3GB of memory.

Once the Java 6 JDK installation is finished, you will be ready to download and install the Android ADT Bundle. Let's do that next, so that we have an Android IDE set-up to use for our future Android application development in the rest of the book.

The Android Developer Tools (ADT) Bundle

As of Android 4.2 Jelly Bean, the Android Software Development Kit, or Android SDK, is bundled with the Eclipse IDE and Android Development Tools (ADT) in one large 400MB download called the ADT Bundle.

Downloading the Android ADT IDE

The first thing we must do is get to the Android Developer webpage, and there are again two ways to do this; one generic, the other precise. The generic way, which will always work, even if Google changes the location of its Android Developer webpage (which it probably won't), is to use Google Search with the keyword phrase "Android Developer Website", which should bring up the `Developer.Android.com` website URL. The second way is to type the URL (I suggest that you memorize this URL) for the Android Developer webpage directly into the browser: `http://developer.android.com/index.html`.

Once you type in the URL, you arrive at the Android Developer website home page, and you need to find the **Get the SDK** button at the bottom of the page, which looks like the menu shown in Figure 1-4.

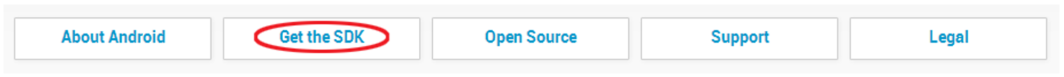
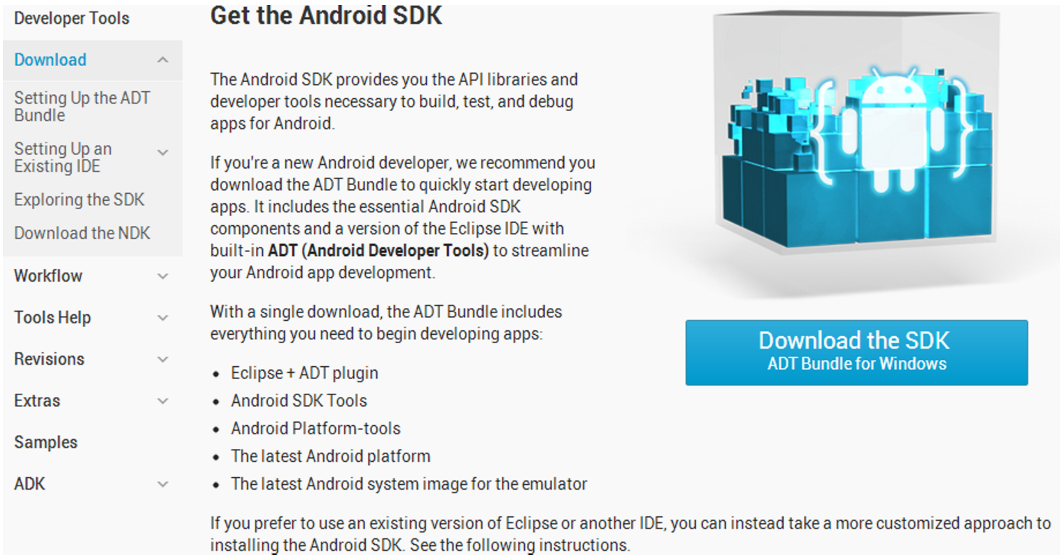


***Figure 1-4.** Selecting the Get the SDK button to go to the Android SDK download page on the `Developer.Android.com` site*

Once you click the **Get the SDK** button you are taken to the **Get the Android SDK** page, where you see a large blue button that says: **Download the SDK - ADT Bundle for Windows** (or it may say for Linux or for Macintosh, if you are currently using those operating systems; the website auto-detects your OS and gives you the proper download link). Click this blue button, as shown in Figure 1-5, to go to the downloads page to get the Android SDK ADT Bundle installer .zip file, which is now available in 32-bit and 64-bit versions.



Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

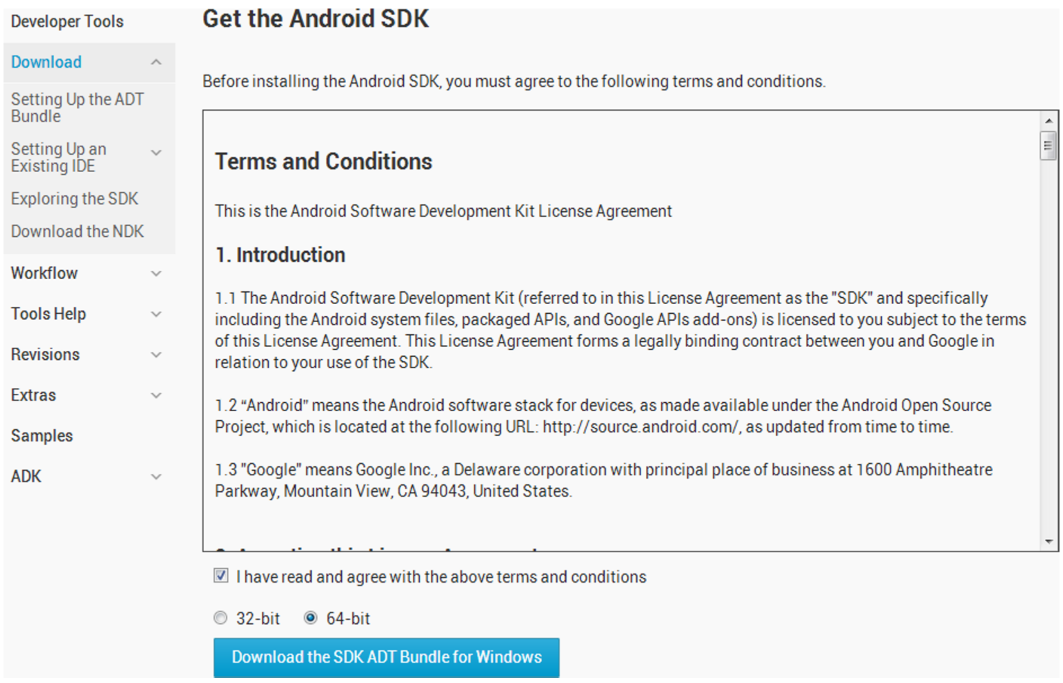
With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions.

***Figure 1-5.** The Download the SDK for Windows ADT Bundle button on the Get the Android SDK page on `Developer.Android.com`*

Once you click the **Download the SDK ADT Bundle** button, you are taken to the page shown in Figure 1-6 where you can read the licensing agreements for the software packages (Android 4.2.2 SDK, Eclipse 4.2.2, Android Developer Tools, etc.) that you are downloading in the bundled installation.



Get the Android SDK

Before installing the Android SDK, you must agree to the following terms and conditions.

Terms and Conditions

This is the Android Software Development Kit License Agreement

1. Introduction

1.1 The Android Software Development Kit (referred to in this License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of this License Agreement. This License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

[✓] I have read and agree with the above terms and conditions

32-bit [● 64-bit]

Download the SDK ADT Bundle for Windows

Figure 1-6. The Agree to Terms and Conditions and Download the SDK ADT Bundle for Windows page on `Developer.Android.com`

After you click the **"I have read and agree with the above terms and conditions"** checkbox, and have then selected the radio button for the type of OS (32-bit or 64-bit) that you are running, you can click the **Download the SDK ADT Bundle for Windows** button and download the appropriate zip file for your OS type.

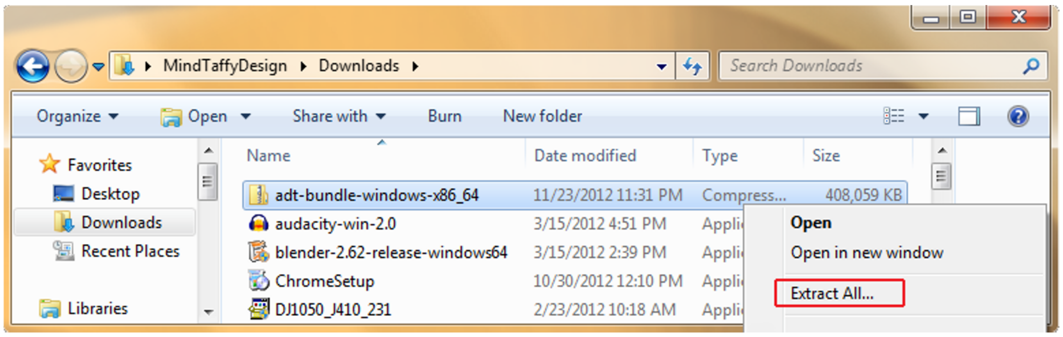
Because I have a Windows 7 64-bit Quad-Core AMD system, I selected the 64-bit SDK version, which was only recently available as of Android 4.2 API Level 17, also known as Jelly Bean Plus. Before Android 4.2 came out in late November 2012 (over the Thanksgiving holiday believe it or not), Android application developers had no choice but to use a 32-bit development software environment, even if they developed on a 64-bit workstation. So if you like to keep everything 64-bit clean on your 64-bit workstation, you're in luck! Let's start the download and get that 400MB file downloading and onto our workstation's hard disk drive now.

Installing the Android ADT IDE

Once the 400MB download has finished, open your Windows Explorer File Navigation and File Management Utility, and find the .zip file in your system's Downloads folder. If you can't find it, put the a partial name of the zip file using wildcard characters in the Search box at the upper right, and select your Computer or C:\ drive on the left, and Windows Explorer will find the file for you.

Here's the search string I use: `adt-bundle*.zip` (an asterisk wildcard expands to fit anything else in filename).

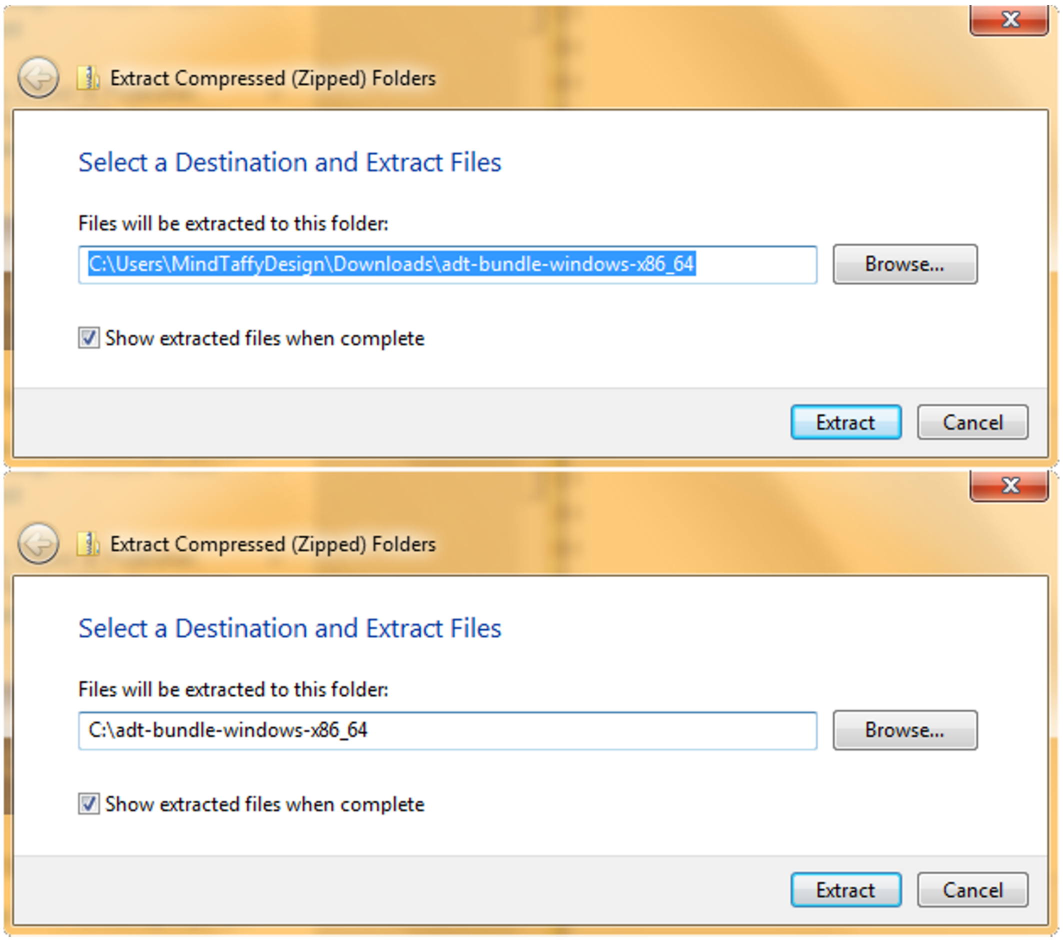
Once you find the **adt-bundle-windows** zip archive file, right-click it to get a context-sensitive menu of everything that can be done to or with that file, and choose the **Extract All…** option to extract all the files and folders inside of the zip archive onto your hard disk drive. This is shown in Figure 1-7 (circled in red).



Name	Date modified	Type	Size
adt-bundle-windows-x86_64	11/23/2012 11:31 PM	Compress...	408,059 KB
audacity-win-2.0	3/15/2012 4:51 PM	Appli	
blender-2.62-release-windows64	3/15/2012 2:39 PM	Appli	
ChromeSetup	10/30/2012 12:10 PM	Appli	
DJ1050_J410_231	2/23/2012 10:18 AM	Appli	

Figure 1-7. Right-clicking the adt-bundle zip file in your Downloads folder and selecting the Extract All… option

Next you will see an **Extract Compressed (Zipped) Folders** dialog, asking you to select your destination directory in which to extract the ADT Bundle files and folder hierarchy. As shown in the top dialog in Figure 1-8, the archive extraction utility places the current location of the file into your system Downloads folder as its destination path, which is not surprising, as it really doesn't know where you want to put these files, so it makes an assumption and puts itself right where the zip file is "standing" currently, so to speak. Let's change that location, and put the files in a more logical location on our workstation.


Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:
C:\Users\MindTaffyDesign\Downloads\adt-bundle-windows-x86_64

[✓] Show extracted files when complete

Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:
C:\adt-bundle-windows-x86_64

[✓] Show extracted files when complete

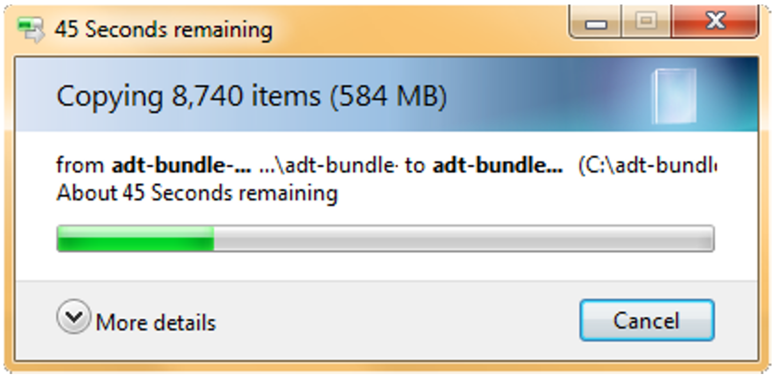
***Figure 1-8.** Changing the extract files destination folder from C:\Users\Username\Downloads to C:\adt-bundle-windows*

Because we don't really want our ADT Bundle installed into our Downloads folder (which we use primarily to hold our file downloads), the easiest work process is to move it up to the top or "root" of our C:\ primary hard disk drive, so let's click the cursor right in front of the adt-bundle in the filename, and then backspace over the Downloads, (*Your User Name Here*), and Users sections of the auto-generated pathname.

In this way, you end up with the C:\ hard drive specifier right in front of the `adt-bundle-windows-x86 .zip` filename that you just downloaded as your folder name. This installs the ADT Bundle in the Root (top) of your C:\ drive, on your primary hard disk partition, where you can easily see it and find it anytime that you need to. The resulting path is shown in the lower part of the screen shot in Figure 1-8.

Note that if you wanted to, you could also remove the `windows` and `x86` parts of the folder name and name the folder `C:\ADT` or `C:\ADT-Bundle` if you wanted to, or `C:\Android_Development_Environment` or something similar. Because you are going to find and set-up the Quick Launch Icon for this Application Executable, it doesn't matter what you call this top-level folder, as long as you can easily locate it.

Once you click the **Extract** button, you get a small progress dialog, shown in Figure 1-9, which shows you all the 8,740 items (584 megabytes) that are being inflated and copied onto your hard disk drive. After this process is complete, open your Windows Explorer file viewing utility, and find the **eclipse.exe,** which is the Eclipse 4.2.2 application's executable file.

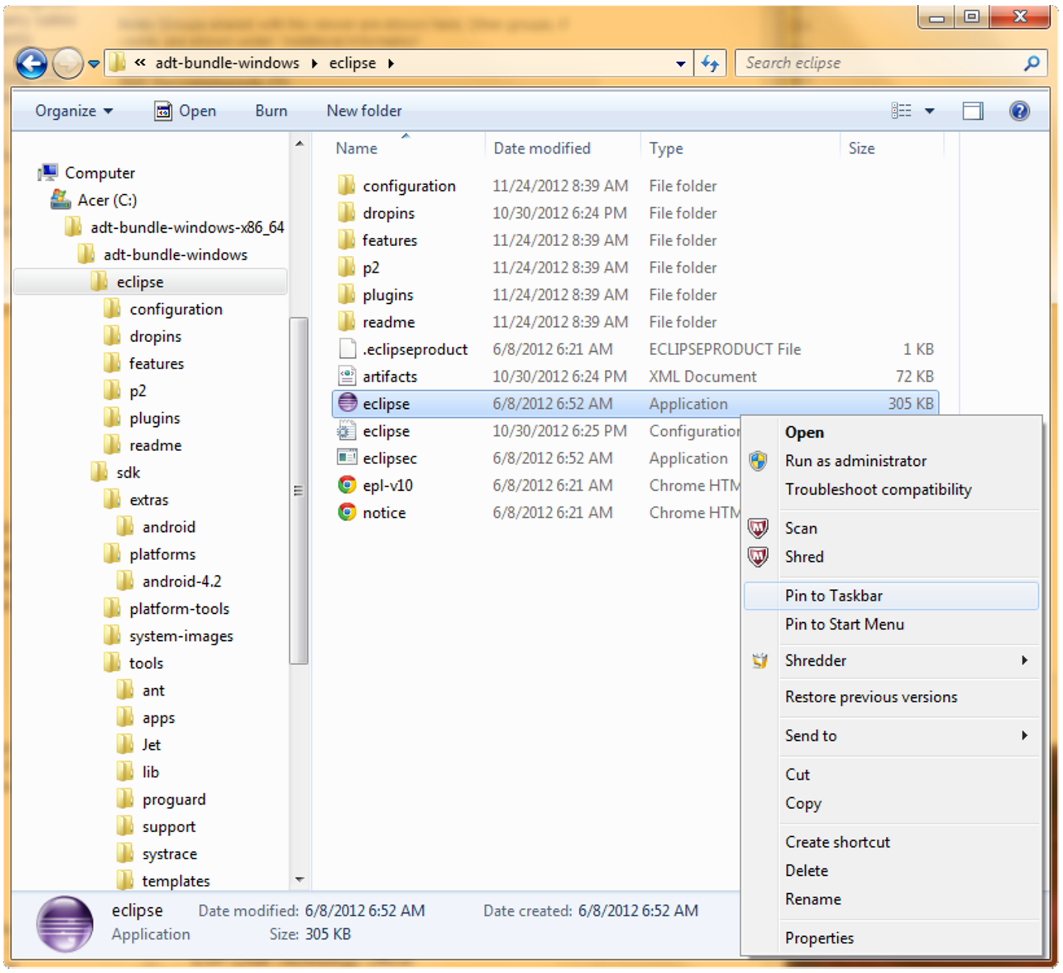

45 Seconds remaining

Copying 8,740 items (584 MB)

from **adt-bundle-...** ...\adt-bundle· to **adt-bundle...** (C:\adt-bundl
About 45 Seconds remaining

***Figure 1-9.** Extracting the 8,740 files totaling 584MB from the 400MB ADT Bundle zip file*

To do this, open the C: drive on the left navigation panel in the Windows Explorer utility, and then look for the **adt-bundle-windows-x86** filename that you downloaded, installed, and named, and under that you will find the eclipse folder and its sub-folders, as shown in Figure 1-10. If you used your own custom name, such as `ADT_Bundle`, then look for whatever name you decided to use for your ADT Bundle folder.



Name	Date modified	Type	Size
configuration	11/24/2012 8:39 AM	File folder	
dropins	10/30/2012 6:24 PM	File folder	
features	11/24/2012 8:39 AM	File folder	
p2	11/24/2012 8:39 AM	File folder	
plugins	11/24/2012 8:39 AM	File folder	
readme	11/24/2012 8:39 AM	File folder	
.eclipseproduct	6/8/2012 6:21 AM	ECLIPSEPRODUCT File	1 KB
artifacts	10/30/2012 6:24 PM	XML Document	72 KB
eclipse	6/8/2012 6:52 AM	Application	305 KB
eclipse	10/30/2012 6:25 PM	Configuratio	
eclipsec	6/8/2012 6:52 AM	Application	
epl-v10	6/8/2012 6:21 AM	Chrome HTM	
notice	6/8/2012 6:21 AM	Chrome HTM	

eclipse Date modified: 6/8/2012 6:52 AM Date created: 6/8/2012 6:52 AM
Application Size: 305 KB

Figure 1-10. Finding the Eclipse Application file in Windows Explorer and right-clicking to invoke the Pin to Taskbar feature

Setting Up a Quick Launch Icon on Your Taskbar

Look for a purple sphere next to the word **eclipse** with a file type of "application" and that is your Eclipse IDE software executable file. Next, right-click this **Eclipse application file** to get another context-sensitive menu of commands, and then choose the **Pin to Taskbar** command in the third section of this menu. This installs Eclipse as a Quick Launch Icon in your Taskbar Program launch area. This work process is also shown in Figure 1-10.

Once you select the **Pin to Taskbar** command from the right-click menu, the Eclipse 4.2 ADT Bundle purple sphere launch icon appears on your Windows Taskbar. This is usually located at the bottom of the OS screen, but can be docked (via dragging) onto the top, or even the left or the right of the screen. Figure 1-11 shows the Eclipse 4.2 ADT Bundle Launch Icon installed on my Win7 Taskbar along with other professional open source software packages (GIMP, Blender 3D, Audacity, Lightworks, OpenOffice, and Chrome, shown in that order), many of which we'll be installing later on in this chapter.

***Figure 1-11.** The Eclipse 4.2 ADT Bundle Launch Icon is shown on the far left side of the Windows Taskbar*

Click the Eclipse Quick Launch Icon and make sure Eclipse for ADT fires up; if it does, it means that you have installed Java 6 correctly, and that Eclipse ADT Bundle has successfully found Java, and is using it to run its integrated development environment. The start-up screen is shown in Figure 1-12.

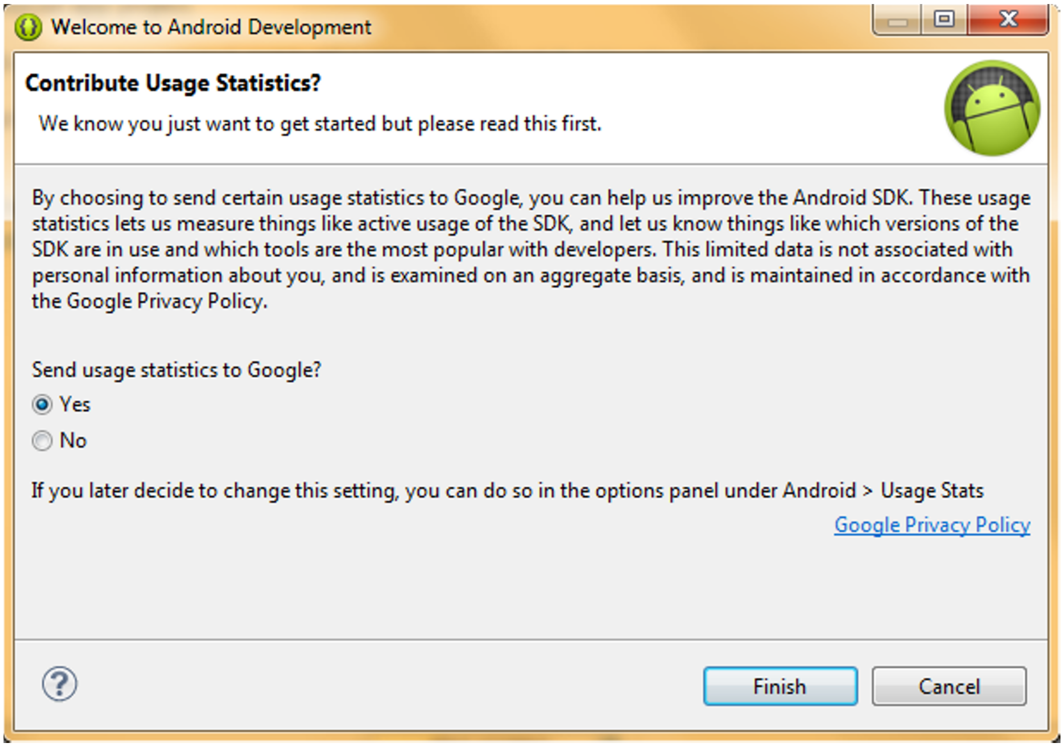
***Figure 1-12.** Launching the Android Developer Tools version of Eclipse 4.2.2 via your newly installed Quick Launch Icon*

It is important to note that as of Android 4.2 Jelly Bean+, that the Eclipse Juno 4.2 start-up screen is now gone, and the new ADT Bundle start-up screen shows that Eclipse is now an integrated part of the 64-bit **Android Developer Tools** software bundle that we have just installed.

Launching and Configuring the Android Developer Tools IDE

After you launch the Eclipse 4.2 for Android Developer Tools software, the first screen that you see pop up is the **Contribute Usage Statistics** dialog. You only see this screen the first time you launch the Eclipse ADT Bundle, so select the option that you want for usage statistic sharing now. You can always change your mind (the option chosen) later on, if you like, via the Android ➤ Usage Stats menu sequence.

I selected the **Send Usage Statistics to Google** (Yes) option, so that Google can see how I use Eclipse and ADT, and so they can make improvements to the product. You can also select the No option with zero negative repercussions. This dialog is shown in Figure 1-13.



Contribute Usage Statistics?

We know you just want to get started but please read this first.

By choosing to send certain usage statistics to Google, you can help us improve the Android SDK. These usage statistics lets us measure things like active usage of the SDK, and let us know things like which versions of the SDK are in use and which tools are the most popular with developers. This limited data is not associated with personal information about you, and is examined on an aggregate basis, and is maintained in accordance with the Google Privacy Policy.

Send usage statistics to Google?

[✓ Yes] No

If you later decide to change this setting, you can do so in the options panel under Android > Usage Stats

Figure 1-13. Contribute Usage Statistics dialog encountered on initial launch of Eclipse ADT software

Once you make your selection, click the **Finish** button, and the Eclipse ADT Bundle continues to launch the Android Integrated Development Environment (IDE).

When you launch the Android Developer Tools version of Eclipse 4.2 for the first time, you will see an **Android IDE Welcome** screen. Everything on that screen we will be covering in the first few chapters of this book, so read it all if you want, and then close it using the **X** in the right side of the **Android IDE Tab** at the top. Once you do this, an empty version of the Eclipse IDE then appears, as shown in Figure 1-14.

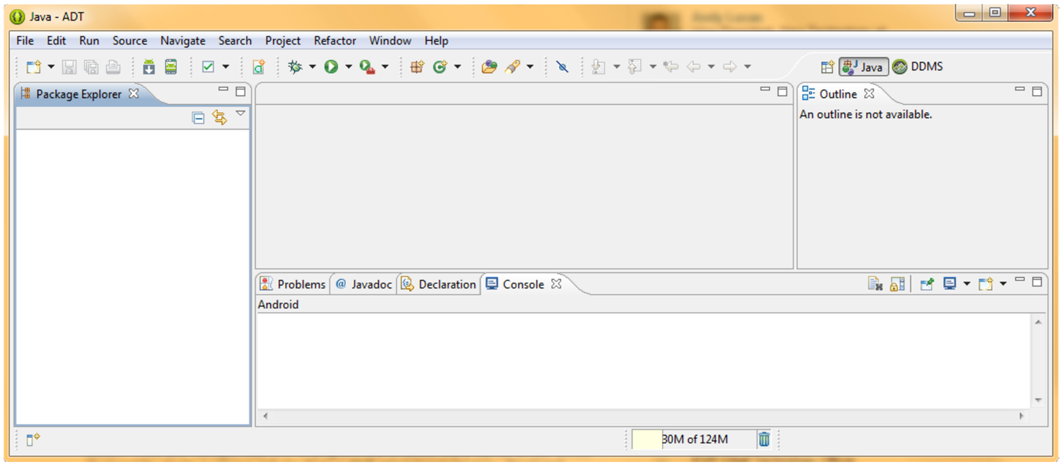
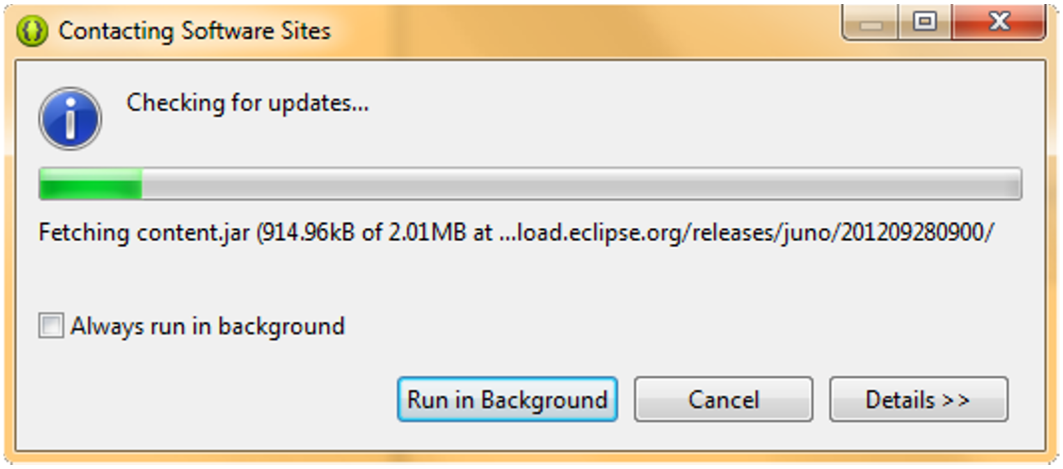

Java - ADT
File Edit Run Source Navigate Search Project Refactor Window Help
Java
DDMS
Package Explorer
Outline
An outline is not available.
Problems
Javadoc
Declaration
Console
Android
30M of 124M

***Figure 1-14.** Eclipse 4.2 for ADT on first launch showing empty Package Explorer, Editing, Console, and Outline panes*

The first thing that we want to do is to make sure the ADT Bundle that we just installed is completely up to date. We do this via the **Help** menu, and its **Check for Updates** command, located near the bottom. Once you have selected this option, you see the dialog shown in Figure 1-15. If there are updated versions of any of the 8,740 files that were installed during your ADT Bundle installation, you will see the **Checking for Updates…** function "fetching" them, and updating the older versions that are on your system currently.


Contacting Software Sites

Checking for updates...

Fetching content.jar (914.96kB of 2.01MB at ...load.eclipse.org/releases/juno/201209280900/

Always run in background

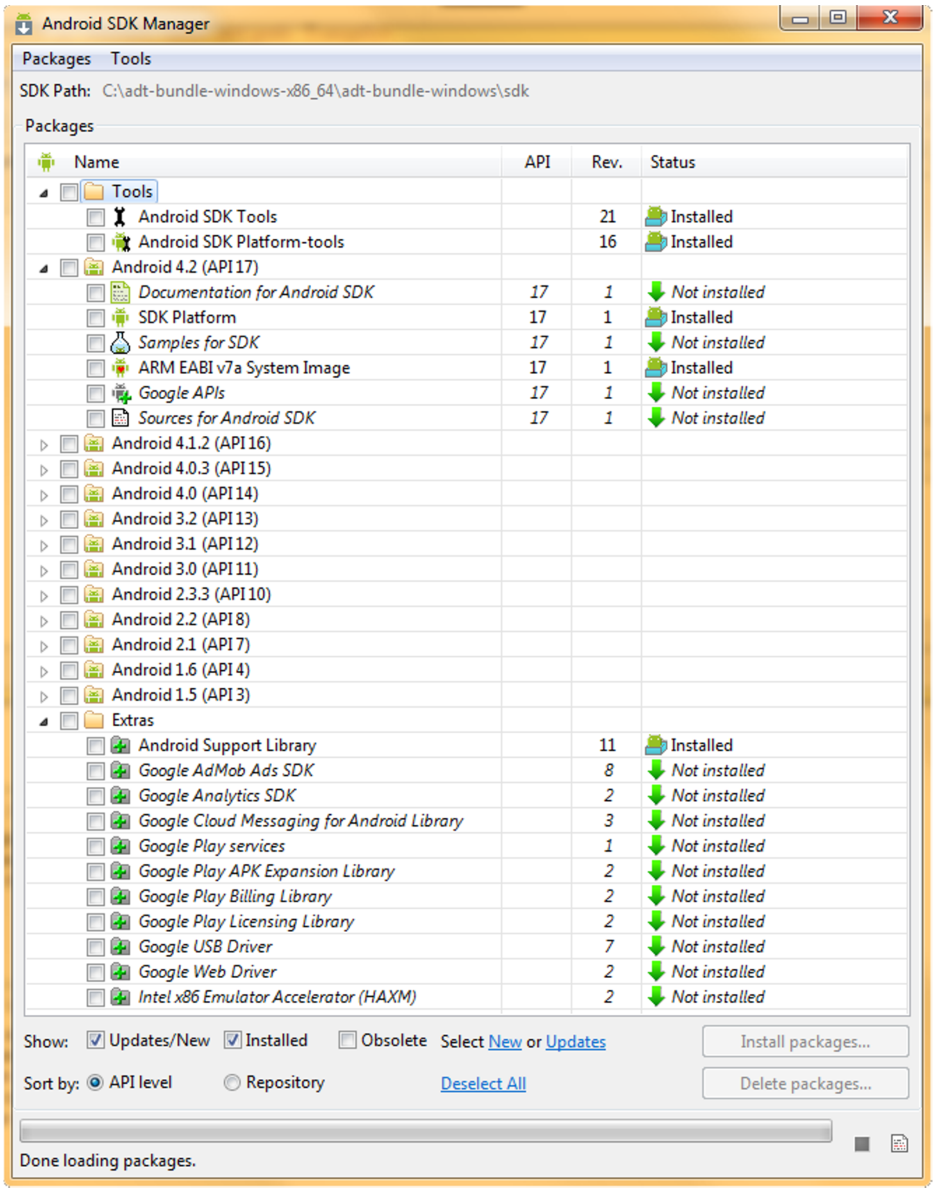
***Figure 1-15.** Checking for Android software updates online, using the software repositories at Eclipse and Google*

Now let's take a look at how to access the **Android SDK** portion of the Eclipse ADT Bundle at any time to see what Android API support level is currently installed inside of Eclipse.

The Android SDK Manager Tool

Because we downloaded the Android 4.2 ADT Bundle, we know that this is going to be at least Android 4.2 API Level 17, but let's now use the **Android SDK Manager** tool to dive deeper in, and to see what Android SDK support is actually afforded to us. We can access the Android SDK part of the Android IDE at any time via the Eclipse 4.2 **Window** menu, and its **Android SDK Manager** option, located in the bottom section of that menu. Clicking this option opens up an Android SDK Manager window, which shows which API Levels (Android OS Versions) we have currently installed (on our system) in our Android IDE. Let's do that now so that you can see what the ADT Bundle installation actually installed on your PC.

As you can see in Figure 1-16, this includes the Android SDK Tools, the Android SDK Platform Tools, the Android SDK 4.2 Platform, the ARM Processor system image (for testing in the Emulator we'll be setting up soon), and the Android Support Library. The **Android Support Library** allows previous versions of Android to be supported, so that we do not necessarily need to download all the other API Levels to support all the hardware devices out there. This is important to note, and is the primary reason why we are only going be to using Android API Level 17 installed during this process in the examples in this book and not download any of the other Android APIs shown in Android SDK Manager at this time.



Android SDK Manager

Packages Tools

SDK Path: C:\adt-bundle-windows-x86_64\adt-bundle-windows\sdk

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		21	Installed
Android SDK Platform-tools		16	Installed
Android 4.2 (API 17)			
Documentation for Android SDK	*17*	*1*	*Not installed*
SDK Platform	17	1	Installed
Samples for SDK	*17*	*1*	*Not installed*
ARM EABI v7a System Image	17	1	Installed
Google APIs	*17*	*1*	*Not installed*
Sources for Android SDK	*17*	*1*	*Not installed*
Android 4.1.2 (API 16)			
Android 4.0.3 (API 15)			
Android 4.0 (API 14)			
Android 3.2 (API 13)			
Android 3.1 (API 12)			
Android 3.0 (API 11)			
Android 2.3.3 (API 10)			
Android 2.2 (API 8)			
Android 2.1 (API 7)			
Android 1.6 (API 4)			
Android 1.5 (API 3)			
Extras			
Android Support Library		11	Installed
Google AdMob Ads SDK		*8*	*Not installed*
Google Analytics SDK		*2*	*Not installed*
Google Cloud Messaging for Android Library		*3*	*Not installed*
Google Play services		*1*	*Not installed*
Google Play APK Expansion Library		*2*	*Not installed*
Google Play Billing Library		*2*	*Not installed*
Google Play Licensing Library		*2*	*Not installed*
Google USB Driver		*7*	*Not installed*
Google Web Driver		*2*	*Not installed*
Intel x86 Emulator Accelerator (HAXM)		*2*	*Not installed*

Show: ☑ Updates/New ☑ Installed ☐ Obsolete Select New or Updates

Sort by: ● API level ○ Repository

Done loading packages.

***Figure 1-16.** Android SDK Manager dialog, showing the libraries and packages installed in the ADT Bundle*

In the future, if you absolutely need to develop for other versions of Android, such as 2.3.7 for Amazon Kindle Fire, or 4.0.4 for Amazon Kindle Fire HD, for instance, simply select those Android versions, and click on the **Install packages...** button at the bottom right of the window. I strongly suggest that you do this one API Level at a time, and only for those Android API Levels that you are absolutely sure that you need to develop for, as each involves downloading hundreds of megabytes of files and documentation.

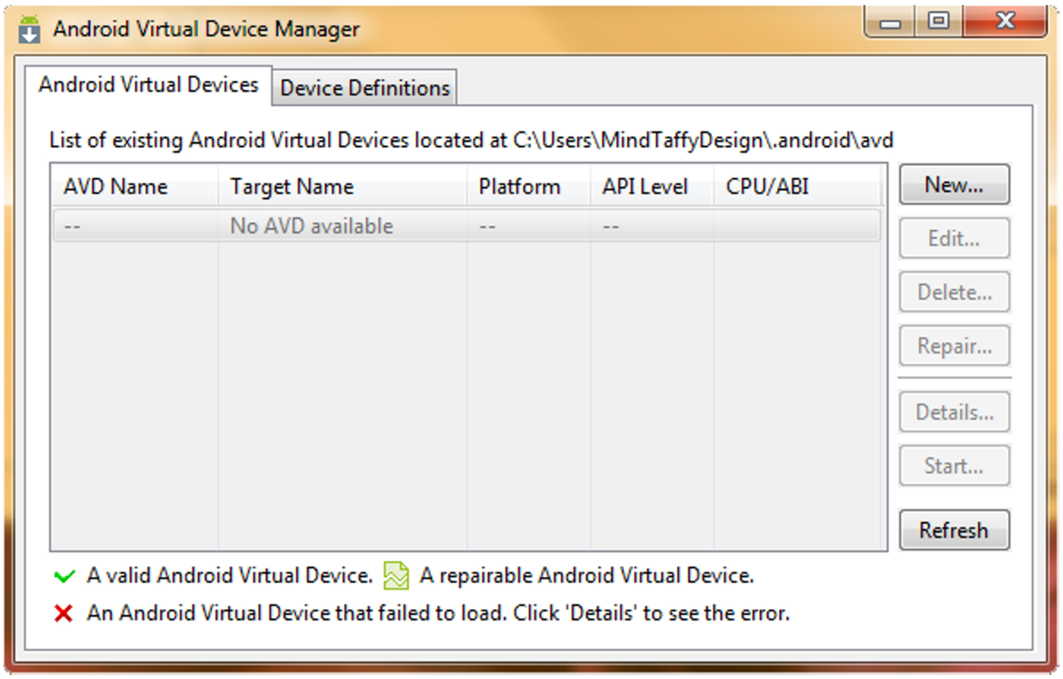
Because we are not going to do anything right now with the Android SDK Manager, click the red X at the top right of the window to close it, and let's get ready to set-up our emulators for the Android devices for which we are developing. In the next section, we'll learn how to set up AVD emulators for some of the more popular smartphone, tablet, and GoogleTV consumer electronics devices.

Configuring Android Virtual Devices (AVDs)

The last step in the process to get your Android Development Environment ready for the rest of this book is to set up your **Android Virtual Devices** (AVDs). An **AVD** is an **emulator** that mimics an Android smartphone, tablet, e-reader, iTV Set (GoogleTV) or set-top box consumer electronics product and allows you to test your Android Apps on your workstation.

Having an emulator is desirable because the work process of constantly uploading your application onto an actual physical Android device (hardware) is much more tedious than right-clicking your project and selecting Run As Android Application. That's not to say that you shouldn't test your app out once in a while on a real Android hardware device, but the AVD allows you to test as you develop, and with much greater speed and frequency. In this section, we'll create three such devices: a Nexus 7 tablet, a Nexus S smartphone, and an Android iTV Set.

Because you are already in Eclipse, go into the **Window** menu, and select the **Android Virtual Device Manager** option near the bottom. This opens up the **Android Virtual Device Manager** dialog, which lists your AVDs. You currently have no AVDs set-up, and it says: **No AVD available** within the central area of the dialog, as shown in Figure 1-17.



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

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

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

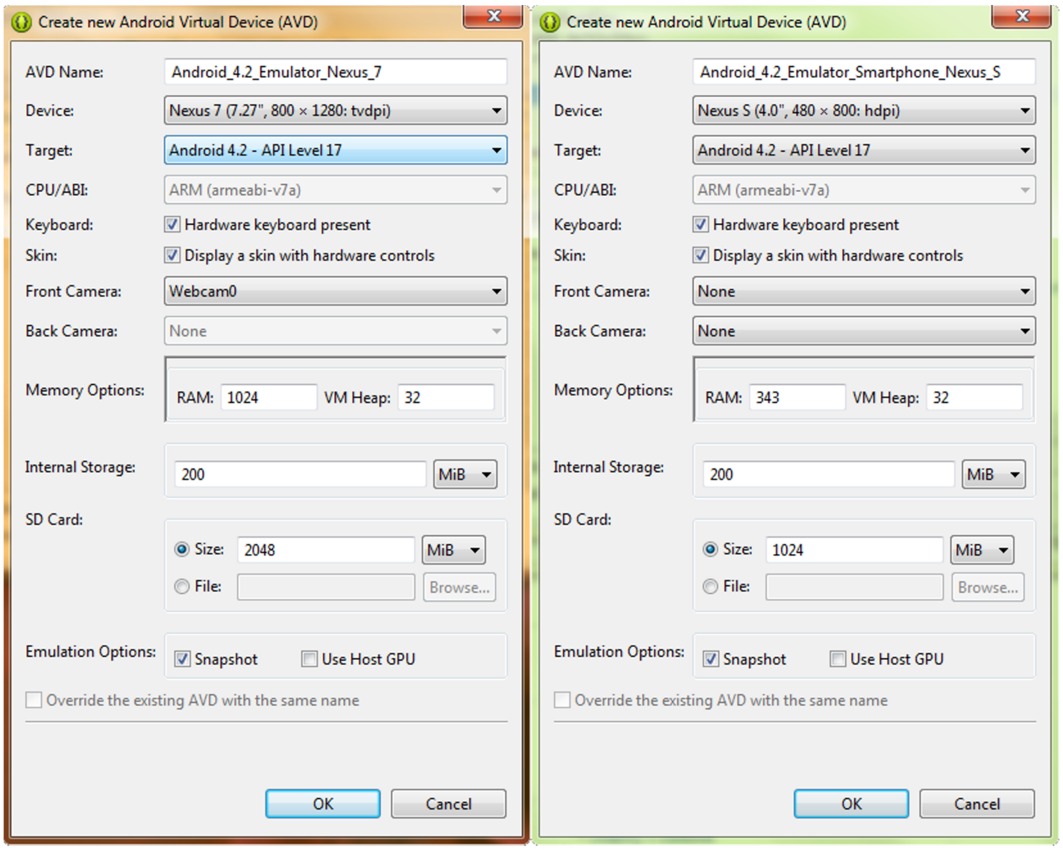
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

***Figure 1-17.** Android Virtual Device Manager dialog–click the New... button to add an AVD*

To add a new AVD to your Eclipse environment, click the **New…** button at the top right of the dialog, and open the **Create a New Android Virtual Device (AVD)** dialog.

Creating Your Nexus 7 Tablet Emulator

The first thing that we want to do is to name our emulator, so enter: **Android_4.2_Emulator_Nexus_7** in the **Name:** field, as shown in Figure 1-18. In the **Device:** field, use the drop-down menu (via the down arrow) to select the **Nexus 7** Tablet option. Then select the **Target:** for the emulator, which is the API Level for the emulator, which is Android 4.2, or API Level 17, which is the current latest revision of the Android OS.



Create new Android Virtual Device (AVD)

AVD Name: Android_4.2_Emulator_Nexus_7
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: Webcam0
Back Camera: None
Memory Options: RAM: 1024 VM Heap: 32
Internal Storage: 200 MiB
SD Card: ● Size: 2048 MiB ○ File: Browse...
Emulation Options: ✓ Snapshot ☐ Use Host GPU
☐ Override the existing AVD with the same name
OK Cancel

Create new Android Virtual Device (AVD)

AVD Name: Android_4.2_Emulator_Smartphone_Nexus_S
Device: Nexus S (4.0", 480 × 800: hdpi)
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: 343 VM Heap: 32
Internal Storage: 200 MiB
SD Card: ● Size: 1024 MiB ○ File: Browse...
Emulation Options: ✓ Snapshot ☐ Use Host GPU
☐ Override the existing AVD with the same name
OK Cancel

Figure 1-18. *Creating a New Android Device (AVD) for a Nexus 7 tablet and a Nexus S smartphone using the AVD Manager*

Because the Nexus 7 uses an ARM (armeabi-v7a) processor, this field is automatically set for you, and cannot be changed. Next, check **Hardware keyboard present** and **Display a skin with hardware controls** because those options fit the Nexus 7 for users who own the wireless keyboard, which most serious users do.

Set the **Front Camera** option drop-down to: **Webcam** if you have a webcam installed on your workstation, otherwise select **None**. This is a new Android AVD Emulator feature starting with Android 4.2. Note that before Android API Level 17, the Android Emulator (AVD) did not support camera testing of any kind.

The Nexus 7 AVD Profile sets the **Memory Options** and **Internal Storage** settings for you, based on the Nexus 7 manufacturer's hardware specifications. Next, set the **SD Card** to be a common **2048MB** SD card configuration, and check the **Snapshot** option.

Finally, click the **OK** button to create the first of three Android 4.2 AVD Emulators, this one for the Nexus 7 Tablet. If you are going to develop for some of the other popular API level tablets, utilize exactly the same work process and add other emulators named **Kindle Fire** (use Android 2.3.7 API Level 10 and HDPI) or **Kindle Fire HD** (use Android 4.0.4 API Level 15 and XXHDPI), for instance, if you need to test your applications for delivery on those other specific hardware platforms.

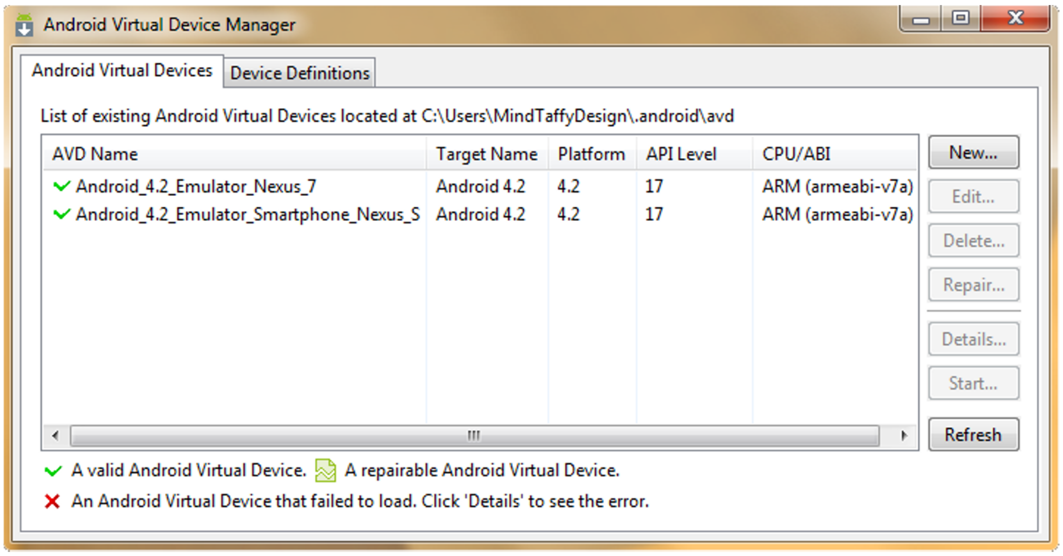
Next, let's create a Smartphone Emulator for the Nexus S, as shown on the right-side of Figure 1-18, so that we have a smartphone emulator as well as a tablet emulator.

Creating Your Nexus S Smartphone Emulator

The first thing we do is to name the emulator, so enter **Android_4.2_Emulator_Nexus_S** in the **Name** field. In the **Device** field, use the drop-down menu (via the down arrow) to select the **Nexus S** option. Then select the **Target** for the emulator, which is again Android 4.2 API Level 17.

Next check **Hardware keyboard present** and **Display a skin with hardware controls** because those options fit the Nexus S advanced user and make app testing easier. Set the **Front Camera** option drop-down to **Webcam** if you have a webcam on your workstation, otherwise select **None**.

The Nexus S AVD Profile sets the **Memory Options** and **Internal Storage** settings for you, based on the Nexus S hardware specs. Next, set the **SD Card** to be a common **1024MB** SD Card configuration, and check the **Snapshot** option. Finally, click the **OK** button to create your second AVD Emulator. We now have two emulators created in our Android Virtual Device Manager window, as shown in Figure 1-19.



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

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Android_4.2_Emulator_Nexus_7	Android 4.2	4.2	17	ARM (armeabi-v7a)
✓ Android_4.2_Emulator_Smartphone_Nexus_S	Android 4.2	4.2	17	ARM (armeabi-v7a)

✓ 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 1-19.** Android Virtual Device Manager showing two new emulators for the Nexus 7 Tablet and Nexus S Smartphone*

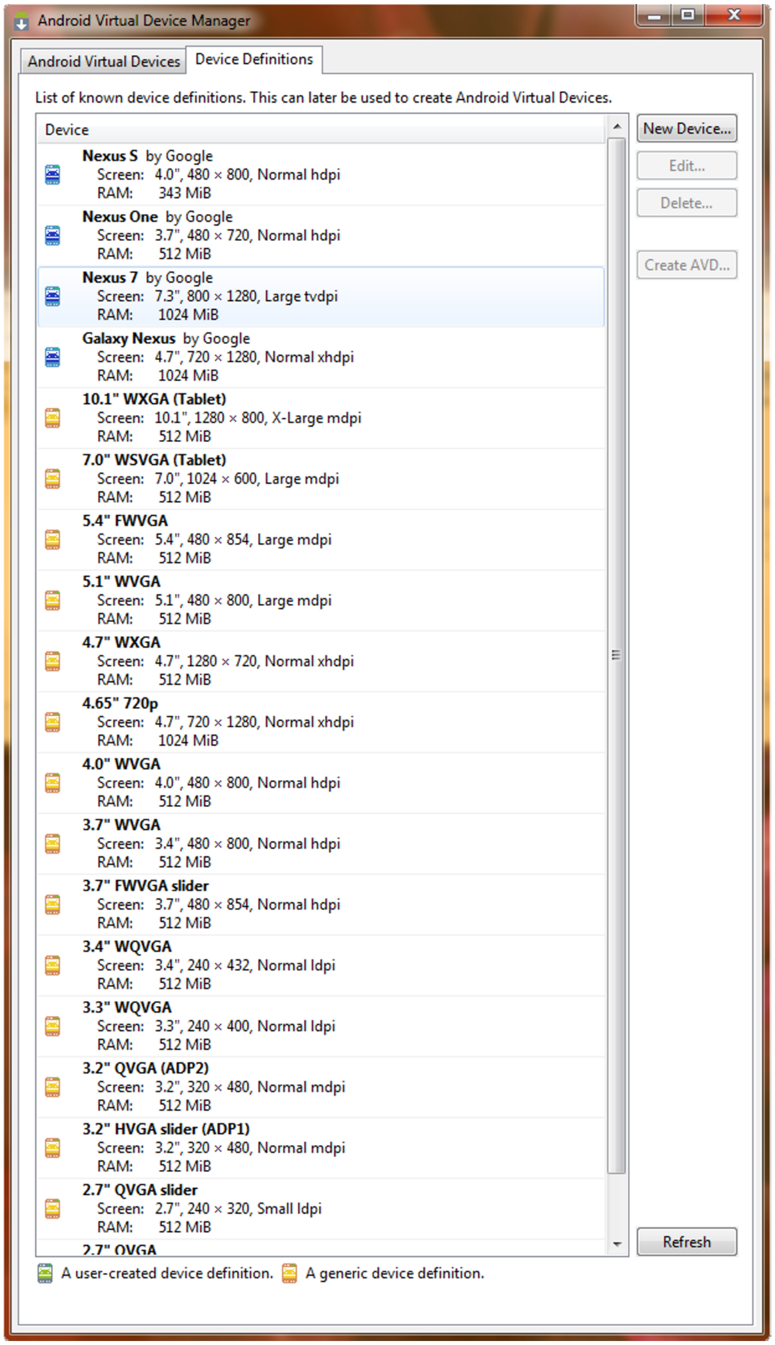
The only other type of Android virtual device that we should really create to test our content on in this book is the newest genre of Android device to hit the market—the iTV, or Interactive Television Set.

Creating Your Android iTV Emulator for GoogleTV Support

Android iTV sets run a branded version of the Android OS called GoogleTV and are available as iTV Sets or Set-Top Boxes (STBs) or Home Media Centers (HMCs) or even as USB sticks such as the Philips product.

Fortunately, I just got the new Philips GoogleTV USB stick product to use for an Android application development project for one of my clients. Thus I have the specifications for the product I'll need to create my own custom AVD device definition specification to use to create an AVD to use for testing.

The first thing I need to do is see whether it is on the list of preconfigured AVDs on the right-side tab labeled Device Definitions, as seen in Figure 1-20. However, the Philips GoogleTV USB stick is not one of them, so we need to go into yet another series of dialogs to add the Philips GoogleTV product to the list of products currently in the AVD Manager Device Definitions dialog. One of the reasons I am adding an iTV emulator to our tablet and smartphone emulators is specifically to show you this work process, as it is hidden even deeper in the AVD Manager than the AVD creation process is.



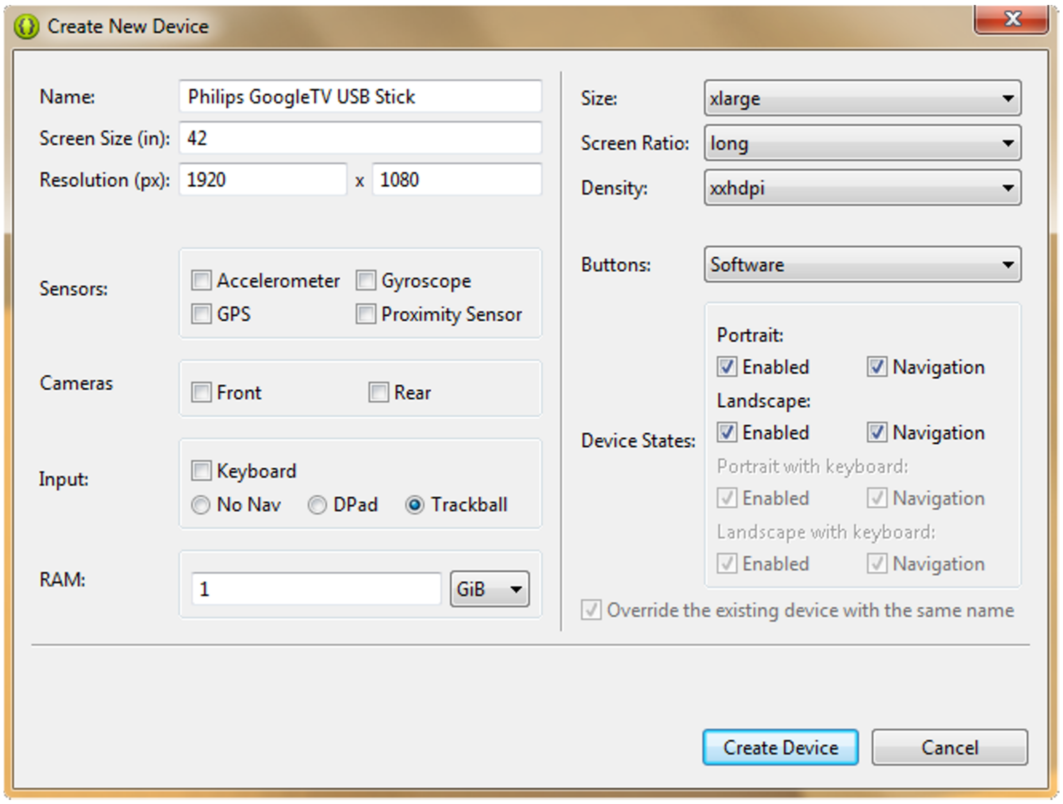
List of known device definitions. This can later be used to create Android Virtual Devices.

```
Device
Nexus S  by Google
   Screen:  4.0", 480 × 800, Normal hdpi
   RAM:     343 MiB
Nexus One  by Google
   Screen:  3.7", 480 × 720, Normal hdpi
   RAM:     512 MiB
Nexus 7  by Google
   Screen:  7.3", 800 × 1280, Large tvdpi
   RAM:     1024 MiB
Galaxy Nexus  by Google
   Screen:  4.7", 720 × 1280, Normal xhdpi
   RAM:     1024 MiB
10.1" WXGA (Tablet)
   Screen:  10.1", 1280 × 800, X-Large mdpi
   RAM:     512 MiB
7.0" WSVGA (Tablet)
   Screen:  7.0", 1024 × 600, Large mdpi
   RAM:     512 MiB
5.4" FWVGA
   Screen:  5.4", 480 × 854, Large mdpi
   RAM:     512 MiB
5.1" WVGA
   Screen:  5.1", 480 × 800, Large mdpi
   RAM:     512 MiB
4.7" WXGA
   Screen:  4.7", 1280 × 720, Normal xhdpi
   RAM:     512 MiB
4.65" 720p
   Screen:  4.7", 720 × 1280, Normal xhdpi
   RAM:     1024 MiB
4.0" WVGA
   Screen:  4.0", 480 × 800, Normal hdpi
   RAM:     512 MiB
3.7" WVGA
   Screen:  3.4", 480 × 800, Normal hdpi
   RAM:     512 MiB
3.7" FWVGA slider
   Screen:  3.7", 480 × 854, Normal hdpi
   RAM:     512 MiB
3.4" WQVGA
   Screen:  3.4", 240 × 432, Normal ldpi
   RAM:     512 MiB
3.3" WQVGA
   Screen:  3.3", 240 × 400, Normal ldpi
   RAM:     512 MiB
3.2" QVGA (ADP2)
   Screen:  3.2", 320 × 480, Normal mdpi
   RAM:     512 MiB
3.2" HVGA slider (ADP1)
   Screen:  3.2", 320 × 480, Normal mdpi
   RAM:     512 MiB
2.7" QVGA slider
   Screen:  2.7", 240 × 320, Small ldpi
   RAM:     512 MiB
2.7" QVGA
```

A user-created device definition. A generic device definition.

Figure 1-20. The AVD Manager Device Definitions tab, where you can select from predefined AVD devices or create your own

The key to unlocking this New Device Definition work process is, you guessed it, the **New Device...** button, located on the top right of the Android Virtual Device Manager's **Device Definitions** tab panel. Click the **New Device...** button, and you get the **Create New Device** dialog, shown in Figure 1-21.


Create New Device

Name: Philips GoogleTV USB Stick
Screen Size (in): 42
Resolution (px): 1920 x 1080
Sensors: ☐ Accelerometer ☐ Gyroscope ☐ GPS ☐ Proximity Sensor
Cameras ☐ Front ☐ Rear
Input: ☐ Keyboard
No Nav DPad [✓ Trackball]
RAM: 1 GiB

Size: xlarge
Screen Ratio: long
Density: xxhdpi
Buttons: Software
Device States:
Portrait: ☑ Enabled ☑ Navigation
Landscape: ☑ Enabled ☑ Navigation
Portrait with keyboard: ☑ Enabled ☑ Navigation
Landscape with keyboard: ☑ Enabled ☑ Navigation
☑ Override the existing device with the same name

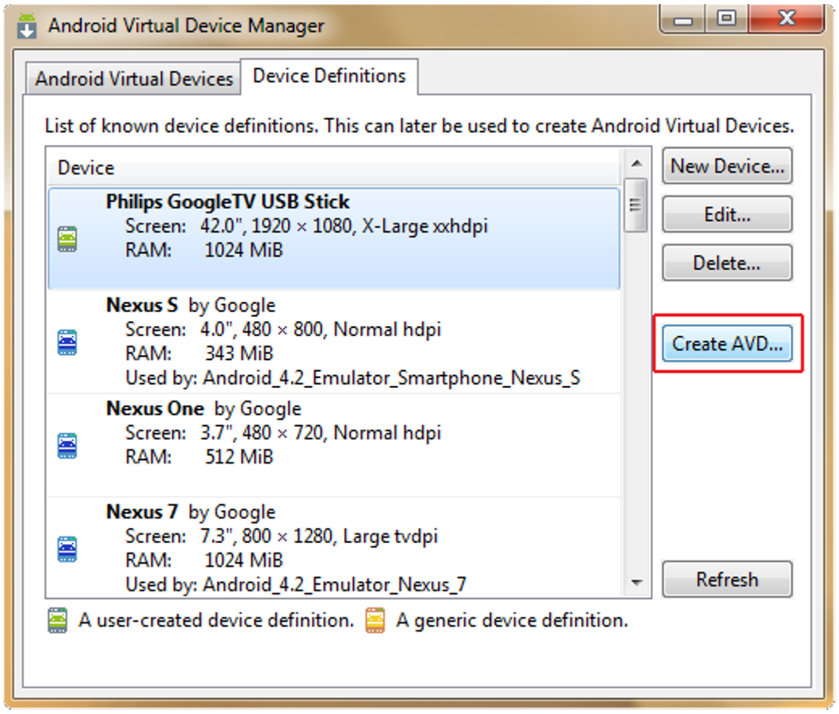
Create Device Cancel

Figure 1-21. The Create New Device dialog in the Device Definitions tab of the AVD Manager window

The Create New Device dialog lets us create a custom product device definition for the Device Definitions tab in the Android Virtual Device Manager window. Let's create one for the Philips GoogleTV USB stick now. In the **Name:** field, place the product name of Philips' GoogleTV USB stick, and also enter the most common **Screen Size:** of **42** inches, and its physical pixel resolution of HDTV **1920 x 1080** pixels. Because HDTVs don't have **sensors**, leave these four options unchecked, and because the GoogleTV USB stick doesn't have any **cameras**, leave these two options unchecked as well. The Philips GoogleTV USB stick product uses a mouse, so select the **Input:** section option of **Trackball**.

The product features 1GB of RAM, so enter a **1** in the **RAM:** field, and select the **GiB** drop-down menu selection. The product has an extra-large size, so select the **xlarge** value for the **Size:** option, and **long** for the **Screen Ratio:** option, because HDTVs are widescreen devices. The Pixel Density is 1920 by 1080 pixels regardless of screen size, so select **XXHDPI** for the **Density:** setting. Because this product is a USB stick, select **Software** for the **Buttons:** option and then enable the **Portrait** and **Landscape** mode checkboxes, as shown in Figure 1-21, because the product works well in both of these modes.

After all these new product specifications are entered correctly, you can click the **Create Device** button at the bottom of the dialog to add the new product definition to the Device Definitions tab of the AVD Manager window, as shown in Figure 1-22.



List of known device definitions. This can later be used to create Android Virtual Devices.

Device

Philips GoogleTV USB Stick
Screen: 42.0", 1920 × 1080, X-Large xxhdpi
RAM: 1024 MiB

Nexus S by Google
Screen: 4.0", 480 × 800, Normal hdpi
RAM: 343 MiB
Used by: Android_4.2_Emulator_Smartphone_Nexus_S

Nexus One by Google
Screen: 3.7", 480 × 720, Normal hdpi
RAM: 512 MiB

Nexus 7 by Google
Screen: 7.3", 800 × 1280, Large tvdpi
RAM: 1024 MiB
Used by: Android_4.2_Emulator_Nexus_7

A user-created device definition. A generic device definition.

***Figure 1-22.** Philips GoogleTV USB Stick added to the AVD Manager Device Definitions list*

Now that we have the Philips GoogleTV USB stick product in the Device Definitions tab, we can select it as shown in Figure 1-22 and create a New Android Virtual Device Emulator using this product definition by clicking on the Create AVD… button in the middle of the dialog.

This opens up the **Create new Android Virtual Device (AVD)** dialog, shown in Figure 1-23, with the proper device settings in the relevant areas of the dialog, much like we observed when we created our AVD Emulators for the Google Nexus products shown in Figure 1-18.