Building cool scripts, apps, and games
for Android Smartphones

# Practical Android Projects

Lucas Jordan | Pieter Greyling

Apress®

# Practical Android Projects

Lucas Jordan
Pieter Greyling

Apress®

**Practical Android Projects**

ISBN 978-1-4302-3243-8

ISBN 978-1-4302-3244-5 (eBook)

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/info/bulksales`.

The source code for this book is available to readers at `www.apress.com`. You will need to answer questions pertaining to this book in order to successfully download the code.

*To Sandy Pond.*

*—Lucas Jordan*

*To Paula and Guilhem for their love during the good times and the bad times. To precious Caitlin and Aaron, who are my guiding stars. To my relentlessly supportive and loyal mother Christina, my brother Cornelius, and my sister Hester. I could not have done this without all of you. Thank you.*

*—Pieter Greyling*

# Contents at a Glance

# Contents

# About the Authors

**Lucas Jordan** (`www.lucasjordan.com`) is a lifelong computer enthusiast and has worked for more than 13 years as a Java developer. He worked at the Children's Hospital Boston for a multidisciplinary applied research and education program called CHIP. After leaving Boston, Lucas settled in Rochester, New York and now works for EffectiveUI as a lead developer. He has contributed to his local Java User's Group (`RJUG.org`), presenting on JavaFX and GWT. In his free time, Lucas is starting a company called ClayWare Games, LLC with his wife Debra Lewis. ClayWare Games, LLC makes accessories and apps for mobile touch devices.

**Pieter Greyling** (`www.pietergreyling.com`) is an information technology expert and software architect with two and half decades of software development experience. He has worked on distributed software engineering projects with teams on several continents for many years. Pieter enjoys software programming and sees smartphone mobile technology as a wonderful way to add an extra element of fun into computing. In his copious free time, he likes to play console video games with his family and take a far too occasional bicycle ride.

# About the Technical Reviewer

**Tony Hillerson** is a software architect for EffectiveUI. He graduated from Ambassador University with a BA in MIS. On any given day, Tony might be working with Android, Rails, Objective-C, Java, Flex, or shell scripts. He has been interested in developing for Android since early betas. Tony has created Android screencasts, tech reviewed Android books, and spoken on Android at conferences. He also sometimes gets to write some Android code.

Tony is interested in all levels of usability and experience design, from the database to the server to the glass.

In his free time, Tony enjoys playing the bass, playing *World of Warcraft*, and making electronic music. Tony lives outside Denver, Colorado with his wife and two sons.

# Acknowledgments

We would like to acknowledge the excellent staff at Apress who managed to get this book completed on time after numerous delays. Good job, everyone!

A very special thank you goes to our coordinating editor, Corbin Collins, and Douglas Pundick, our developmental editor.

We also thank Frank Pohlmann, the lead editor, and Assistant Editorial Director Steve Anglin.

Thank you to our technical reviewer, Tony Hillerson, for giving us really valuable input and feedback.

# Preface

Android is a well-thought-out platform for developing mobile applications. Google has done a wonderful job of providing third-party developers with a world-class development environment. The ease of development combined with the enormous user base makes Android a very compelling platform for developers.

## What this Book Is

When you're building an Android application, many things are straightforward; however, there are facets to the Android platform for which the voice of experience is an invaluable guide. Each chapter in this book explores one of these facets and aims to guide the reader to a better understanding of the topic. By presenting a concrete example project and the steps required to make it work, the reader will gain insight into Android and avoid some pitfalls along the way.

In addition, we have tried to show alternative ways to develop with the Android SDK Tools and IDEs. There are projects here that not only cover programming Android applications with Java but also get you started working with other languages such as JavaScript and Lua.

## What You Will Need

Chapters 1 and 2 cover the groundwork of the Android development environment in detail. These chapters provide full instructions for creating and working with the Android SDK Tools and other development software such as IDEs and plugins.

In summary, to work with the projects in this book you will need the following:

- **A desktop computer running Windows, Linux, or Mac OS X**
  The book projects were developed using a mix of Windows XP, Ubuntu Linux, and Mac OS X. All the projects were tested for compatibility on these platforms.
- **Java SDK**
  The book uses the Java JDK 1.6.0_18 and later.
- **Apache Ant**
  We have found a stand-alone installation of Apache Ant to be very convenient and useful when working with the Android SDK terminal command-line tools. Full coverage of this aspect is given in the first two chapters of the book.
- **Google Android SDK**
  All the projects in this book were developed and built using the Android 2.3 "Gingerbread" SDK.

- **Integrated development environment (IDE)**
  We have used the following IDEs for the projects in the book: Eclipse, NetBeans, and IntelliJ IDEA Community Edition. All the IDE projects have been tested for compatibility, so you are free to choose your own IDE. In fact, we provide enough coverage using only the Android terminal command-line tools and Apache Ant for you to choose to forego an IDE altogether.

As we mentioned, Chapters 1 and 2 cover working with the core Android SDK and the installation and configuration of an Android development environment suited to your tastes.

All other chapters describe the setup and installation of any extra tools and software dependencies required for the content of that particular chapter.

## What You Need to Know

We expect you to be proficient in the Java programming language and perhaps JavaScript, plus another scripting language such as Python, Lua, Ruby, or Perl.

Chapter 1

# Android Fundamentals

The Android platform is a very exciting yet relatively new player in today's mobile device market. Beyond rating very highly in the number of cool features per device, Android-enabled smartphones are currently enjoying the highest percentage sales growth rate in the mobile industry.

According to Gartner Research,[1] worldwide sales of Android-based smartphones to end users have jumped from the number 6 spot in 2009 to number 4 by the end of the first quarter of 2010. This level of growth is expected to continue. In fact, Gartner has predicted that Android will become the number 2 worldwide mobile operating system in 2010 and will challenge Symbian for the number 1 position by 2014.[2]

We want to share with you some of the enthusiasm we have for this truly remarkable development platform. Throughout the course of this book, we will attempt to do this by showing the wide range of opportunities available at your fingertips when you choose to develop Android applications.

Perhaps you are reading this book in order to gain more background understanding of the Android platform. Perhaps you plan to roll up your sleeves and join us in running and playing with the projects in the emulator or your own device. We want to get you up and running quickly and provide you with sufficient understanding of the Android platform and Android Development Kit (ADK) development environment to have success with your goals.

With those goals in mind, this chapter aims to be as practical an introduction to Android development as possible. It also strives to cover a broad spectrum of required conceptual and theoretical background material in a concise and to-the-point manner.

We will start with a short description of the Android platform and then jump straight into coverage of the installation of the Android SDK and supporting development tools. To fully round out our SDK setup study, we embark on a step-by-step test drive that

[1] From Gartner press release: `http://www.gartner.com/it/page.jsp?id=1372013`

[2] From the Gartner press release: `http://www.gartner.com/it/page.jsp?id=1434613`

involves generating a bare-bones Android project and getting the resulting skeleton Android application up and running in the Android emulator.

The next order of business will be a tour of the Android platform architecture. Here we will describe the Android platform stack; Android component architecture; and Dalvik, the Android runtime. With this knowledge in hand, we then cover working with the Java IDEs Eclipse, NetBeans, and IntelliJ IDEA Community Edition; plus spend some time learning how to equip them with Android programming capabilities via plugins.

This means we have a lot of ground to cover, so let's get started.

## What Is Android?

In a nutshell, Android is an operating system targeted at mobile hardware such as phones and other constrained computing devices such as netbooks and tablet computers.

The concept and platform was the brainchild of Android Inc., a small startup company from Palo Alto, California, that was acquired by Google in 2005. Its stated goal was to create a small, stable, flexible, and easily upgraded operating system for handsets that would be highly attractive for device manufacturers and telephony carriers.

Android platform releases 1.x through 2.x are aimed primarily at smartphone devices, whereas it is reported that Android release 3.x will be the first operating platform specifically designed with high-end support for tablet computers.

The Android platform was originally unveiled in November 2007. The unveiling coincided with the announcement of the formation of the Open Handset Alliance, a group of companies that share the goal of promoting open standards for mobile device platforms such as Android.

In October 2008, Android was released under the Apache 2.0 open-source license.[3] This and the flexible component-based design of the platform present innovative and cost-effective opportunities for manufacturers and software developers alike. We aim to showcase some of these distinguishing platform capabilities during the course of this book.

## Installing the Android SDK

We will start by installing the core Android SDK and tools. Our aim is to get the Android emulator with our own simple application up and running on an Android Virtual Device (AVD) as soon as possible. The experience gained will then serve as a basis for further discussion.

[3] `http://source.android.com/source/licenses.html`

The examples and commands you will be shown were run on a mixture of Ubuntu GNU/Linux, Microsoft Windows, and Apple Mac OS X systems. All the tools, including the JDK and the Android SDK toolset, behave in a similar, if not identical, manner across the major supported computing platforms.

## Java Development Kit (JDK)

To begin with, you should have a recent version of the Java SDK (JDK) installed on your particular system. It can be obtained either from your operating system distribution package install manager application or directly downloaded from the Internet.[4] We assume that we do not need to go into the details for doing this. Suffice it to say that JDK5 or upward should be fine. This writing is based on JDK6.

**CHECKING THE JDK VERSION:** To confirm that a compatible version of the JDK is installed and available to the environment, we usually do a quick check on the command line or console terminal, as follows:

```
$ java -version
java version "1.6.0_18"
OpenJDK Runtime Environment (IcedTea6 1.8.1) (6b18-1.8.1-0ubuntu1)
OpenJDK Server VM (build 16.0-b13, mixed mode)
$ javac -version
javac 1.6.0_18
```

If something goes wrong, you should consult the JDK configuration documentation for your particular platform. We will not cover debugging Java installations here.

## Android SDK and Target Platforms

Assuming that our Java platform is ready, we now need to download the Android SDK starter package and use it to install our target Android platforms.

The Android SDK starter package can be downloaded from the official Google Android SDK download site.[5] Select the download appropriate for your development platform. The supported platforms currently include Windows, Mac OS X (Intel), and Linux (i386).

In the case of having downloaded an SDK starter package archive for Linux or Mac OS X, unpack the downloaded archive into a directory of your choice.

[4] https://jdk6.dev.java.net/

[5] http://developer.android.com/sdk/

In the case of having downloaded the Windows installer (.exe file), run the installer and install into a directory of your choice.

You could call this directory anything you like, but we recommend something similar to the following:

Linux or Mac OS X system: `~/android-sdk-linux_x86`

Windows system: `C:\android-sdk-windows`

Make a note of this directory path name for later use.

Within the root of the unpacked directory structure there should be a text file with a name like `SDK Readme.txt`. This has specific instructions for each platform. What is important to note here is that the downloaded archive does not include the complete SDK. The following note contains an extract from the readme shipped with the latest Android SDK as of this writing.[6]

**READ THE SDK README!** The Android SDK archive only contains the tools. It no longer comes populated with a specific Android platform or Google add-on. Instead, you use the SDK Manager to install or update SDK components such as platforms, tools, add-ons, and documentation. In order to start developing applications, you must install at least one version of the Android platform using the SDK Manager. This requires an Internet connection, so if you plan to use the SDK offline, please make sure to download the necessary components while online.

At this point, it is recommended to add the Android SDK tools directory to the development environment system `PATH` variable. The tools directory can be found under the preceding unpacked root directory: `<sdk>/tools/`.

Having the binaries and tools on the path will make it a lot more convenient to issue Android SDK commands from anywhere on the terminal console of your development system.

As an example, after adding the appropriate entries to the shell user login script for my GNU/Linux development system, we receive the following output from listing it with the Linux `cat` command:

```
$ cat ~/.bashrc
.....
#-- google android dev tools --
export PATH="$PATH: ~/android-sdk-linux_86/tools"
export PATH="$PATH: ~/android-sdk-linux_86/platform-tools"
.....
```

[6] Android SDK release 8, Android 2.3 platform

**SETTING THE PATH ON WINDOWS:** From the desktop, right-click `My Computer` and click `Properties`. Alternatively, from `Control Panel`, double-click `System`. Both options open the `System Properties` dialog box. Now click the `Advanced` tab. In the Advanced section, click the `Environment Variables` button. In the Environment Variables window, select the `PATH` variable in the User- or System Variable section, depending on whether you want the setting applied for all users or just yourself. Click the `Edit` button. Add or modify the path. Directories are separated by a semicolon. Click OK when done.

For confirmation, issuing the following command on your development system will print the current value of the system `PATH` variable to the terminal console window.

Linux and Mac OS X:

```
echo $PATH
```

Windows:

```
echo %PATH%
```

## Android Platform API Levels

The API level targeted by your application is very important for reasons of device compatibility and the software development- and maintenance lifetime of your codebase. If it is not managed properly, the maintenance of your application could potentially become a nightmare, especially if it is deployed to multiple Android devices and operating platforms.

It is also a good idea to become familiar with the folder structures of the Android SDK once it is installed. Again, this is especially valid if your applications will be built for multiple Android hardware targets.

For a better understanding of the subject of API levels, it is well worth the effort of reviewing the documentation found on the official developer's web site for Android API levels.[7] The tie-in between API level numbers and their corresponding platforms are clarified in Table 1–1, which was current at the time of writing.

[7] `http://developer.android.com/guide/appendix/api-levels.html`

**Table 1–1.** *Android Platform Versions and API Levels*

| Platform Version | API Level |
|---|---|
| Android 2.3 | 9 |
| Android 2.2 | 8 |
| Android 2.1 | 7 |
| Android 2.0.1 | 6 |
| Android 2.0 | 5 |
| Android 1.6 | 4 |
| Android 1.5 | 3 |
| Android 1.1 | 2 |
| Android 1.0 | 1 |

## Android Platform Setup

Here is a short list of dependencies for proceeding with the setup of SDK platforms:

- Android SDK starter package downloaded and unpacked.
- The JDK, ADK, and Ant tools are accessible on the environment path.
- We have a basic understanding of Android platform versions and API levels.
- Last but not least, we should be connected to the Internet.

We can now install the SDK platform components using the Android SDK and AVD Manager programs.

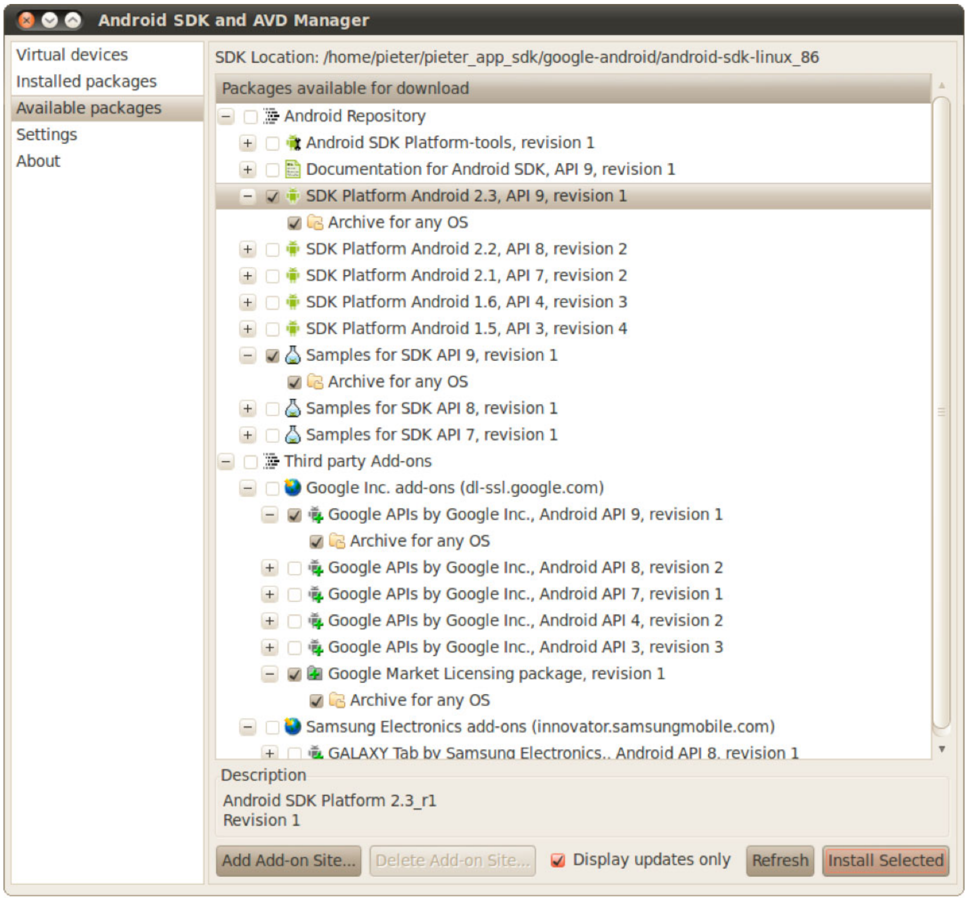
To start the SDK Manager on Linux or Mac OS X, execute the following command:

```
$ android
```

To start the SDK Manager on Windows, run the following program:

```
SDK Manager.exe
```

The main user interface of the Android SDK Manager on Linux should appear as in Figure 1–1.



SDK Location: /home/pieter/pieter_app_sdk/google-android/android-sdk-linux_86

Packages available for download

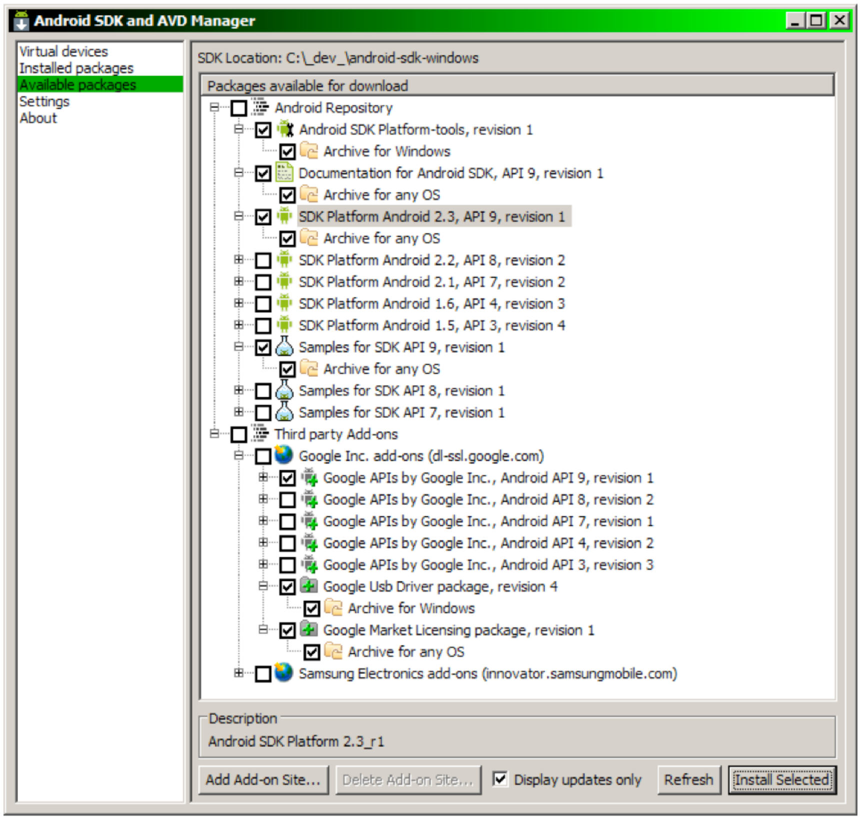
- [ ] Android Repository
  - [ ] Android SDK Platform-tools, revision 1
  - [ ] Documentation for Android SDK, API 9, revision 1
  - [x] SDK Platform Android 2.3, API 9, revision 1
    - [x] Archive for any OS
  - [ ] SDK Platform Android 2.2, API 8, revision 2
  - [ ] SDK Platform Android 2.1, API 7, revision 2
  - [ ] SDK Platform Android 1.6, API 4, revision 3
  - [ ] SDK Platform Android 1.5, API 3, revision 4
  - [x] Samples for SDK API 9, revision 1
    - [x] Archive for any OS
  - [ ] Samples for SDK API 8, revision 1
  - [ ] Samples for SDK API 7, revision 1
- [ ] Third party Add-ons
  - [ ] Google Inc. add-ons (dl-ssl.google.com)
    - [x] Google APIs by Google Inc., Android API 9, revision 1
      - [x] Archive for any OS
    - [ ] Google APIs by Google Inc., Android API 8, revision 2
    - [ ] Google APIs by Google Inc., Android API 7, revision 1
    - [ ] Google APIs by Google Inc., Android API 4, revision 2
    - [ ] Google APIs by Google Inc., Android API 3, revision 3
    - [x] Google Market Licensing package, revision 1
      - [x] Archive for any OS
  - [ ] Samsung Electronics add-ons (innovator.samsungmobile.com)
    - [ ] GALAXY Tab by Samsung Electronics., Android API 8, revision 1

Description
Android SDK Platform 2.3_r1
Revision 1

[✓] Display updates only

**Figure 1–1.** *The Android SDK and AVD Manager during initial SDK setup on Linux*

**WINDOWS USB DRIVER FOR ANDROID DEVICES:** It is worth showing the equivalent Android SDK and AVD Manager for the Windows platform (see Figure 1–2). It contains an important addition, the Windows USB Driver package for Android devices. This will become necessary when you develop, debug, and deploy directly in conjunction with a physical Android phone or other Android hardware device attached via USB cable to a Windows computer.



Android SDK and AVD Manager

Virtual devices
Installed packages
Available packages
Settings
About

SDK Location: C:_dev_\android-sdk-windows

Packages available for download

```
⊟ ☐ Android Repository
   ⊟ ☑ Android SDK Platform-tools, revision 1
       ☑ Archive for Windows
   ⊟ ☑ Documentation for Android SDK, API 9, revision 1
       ☑ Archive for any OS
   ⊟ ☑ SDK Platform Android 2.3, API 9, revision 1
       ☑ Archive for any OS
   ⊞ ☐ SDK Platform Android 2.2, API 8, revision 2
   ⊞ ☐ SDK Platform Android 2.1, API 7, revision 2
   ⊞ ☐ SDK Platform Android 1.6, API 4, revision 3
   ⊞ ☐ SDK Platform Android 1.5, API 3, revision 4
   ⊟ ☑ Samples for SDK API 9, revision 1
       ☑ Archive for any OS
   ⊞ ☐ Samples for SDK API 8, revision 1
   ⊞ ☐ Samples for SDK API 7, revision 1
⊟ ☐ Third party Add-ons
   ⊟ ☐ Google Inc. add-ons (dl-ssl.google.com)
       ⊞ ☑ Google APIs by Google Inc., Android API 9, revision 1
       ⊞ ☐ Google APIs by Google Inc., Android API 8, revision 2
       ⊞ ☐ Google APIs by Google Inc., Android API 7, revision 1
       ⊞ ☐ Google APIs by Google Inc., Android API 4, revision 2
       ⊞ ☐ Google APIs by Google Inc., Android API 3, revision 3
       ⊟ ☑ Google Usb Driver package, revision 4
           ☑ Archive for Windows
       ⊟ ☑ Google Market Licensing package, revision 1
           ☑ Archive for any OS
   ⊞ ☐ Samsung Electronics add-ons (innovator.samsungmobile.com)
```

Description
Android SDK Platform 2.3_r1

Add Add-on Site... | Delete Add-on Site... | ☑ Display updates only | Refresh | Install Selected

**Figure 1–2.** *The Android SDK and AVD Manager during initial SDK setup on Windows*

Note that in both cases we have selected the Android 2.3 platform, API level 9, plus the relevant additions such as documentation and SDK samples. Now click `Install Selected`. The appropriate SDK resource bundles will now be downloaded and installed into the SDK directory structure where we unpacked the SDK starter archive.

In order to maintain and update your SDK over time, an update session can be directly initiated from the command line by executing the following commands:

- In a terminal session on Linux/Mac OS X:

```
$ android update sdk
```

- Besides the option of simply running `SDK Manager.exe` again, the same can be achieved from the Windows command prompt with the following:

```
C:\> android.bat update sdk
```

Again, we assume that the Android tools can be found on the system path. Further information about managing your Android SDK installation can be found on the Android Developers “Adding SDK Components” page.[8]

### Extra Tools: Apache Ant

There are some development tools that no Java developer should do without. One such an indispensable utility is Apache Ant, which is a build tool that is Java's rough equivalent to `make`. `make` is traditionally used in C/C++ development environments. Ant also differs from `make` in that it uses XML to specify build steps and actions.

The Android SDK extensively uses Ant for its compilation, build, and deployment infrastructure. We will use it to test drive our core tools in the next section. So if it is not already installed on your system, we recommend you grab a copy and install it. If necessary, you can find installation instructions and more information about Ant on the official Ant web site.[9]

**SOME IDES ALREADY CONTAIN ANT:** If you will be using an IDE exclusively, installing a stand-alone instance of Apache Ant might not be necessary. IDEs such as Eclipse and NetBeans come packaged with an Ant distribution that they invoke behind the scenes during the build process.

If you are planning to work through the examples that follow, ensure that Ant is on the system environment path once it is installed.

## Android SDK Test Drive

We will now take our SDK and platform installation for a comprehensive test drive to complete the installation of runtime components and to confirm that everything was set up correctly. We will also get to know the environment better. This is a central part of this chapter and will form the basis of further subjects covered.

Initially, we will do the work from the terminal console, command line, or command prompt, whichever terminology is appropriate for your system or personal preference.

1. Create an application project directory to work in and call it `HelloAndroidSdk`. From within a parent- or home directory of your choice somewhere on your system, issue the following commands:

   On Linux or Mac OS X:

```
$ mkdir HelloAndroidSdk
$ cd HelloAndroidSdk
```

[8] http://developer.android.com/sdk/adding-components.html

[9] http://ant.apache.org/

On Windows:

```
C:\> md HelloAndroidSdk
C:\> cd HelloAndroidSdk
```

2. Next we will create a bare-bones Android application using the SDK tools, but before we do that, let's check the available platform targets. From now on, we will only show the GNU/Linux `bash` shell version of the command because the equivalents for the other platforms are identical in syntax. Issue the following command:

```
$ android list targets
```

Based on the SDK selections installed earlier, the output should be similar to this listing:

```
 Available Android targets:
id: 1 or "android-9"
     Name: Android 2.3
     Type: Platform
     API level: 9
     Revision: 2
     Skins: HVGA (default), QVGA, WQVGA400, WQVGA432, WVGA800, WVGA854
```

3. Now we will use the SDK tools to create a skeleton Android application targeting the previous platform within this folder. Enter the following command code as a single command line on the console:

```
$ android create project --target "android-9" --name MyAndroidSdkApp
--path ./MyAndroidSdkAppProject --activity MyAndroidSdkAppActivity
--package com.example.myandroid
```

**NOTE:** The `--target "android-9"` argument could also have read as follows: `--target 1`.

The successful completion of the command should result in output similar to this:

```
Created project directory: ./MyAndroidSdkAppProject
Created directory ./MyAndroidSdkAppProject/src/com/example/myandroid
Added file ./MyAndroidSdkAppProject/src/com/example/myandroid/➥
MyAndroidSdkAppActivity.java
Created directory ./MyAndroidSdkAppProject/res
Created directory ./MyAndroidSdkAppProject/bin
Created directory ./MyAndroidSdkAppProject/libs
Created directory ./MyAndroidSdkAppProject/res/values
Added file ./MyAndroidSdkAppProject/res/values/strings.xml
Created directory ./MyAndroidSdkAppProject/res/layout
Added file ./MyAndroidSdkAppProject/res/layout/main.xml
Added file ./MyAndroidSdkAppProject/AndroidManifest.xml
Added file ./MyAndroidSdkAppProject/build.xml
```

The Android SDK has now generated the full source code and resource files to build a complete and functional Android application.

A listing is shown in Figure 1–3 of the Java source code of one of the files, `MyAndroidSdkAppActivity.java`, that was generated. This is the application's main entry point, a class that extends the Activity class.

**ABOUT THE CODE:** We will not go into the detailed coding aspects of Android programming in this chapter. This chapter serves as the diving board used by the rest of the book to dive into the details of coding Android applications.

4. Next, we want to build the generated source code into an executable application. To do this, first enter the following into the new application directory:

```
$ cd MyAndroidSdkAppProject
```

Now issue the following command to instruct `ant` to build a debugging release of the application project:

```
$ ant debug
```

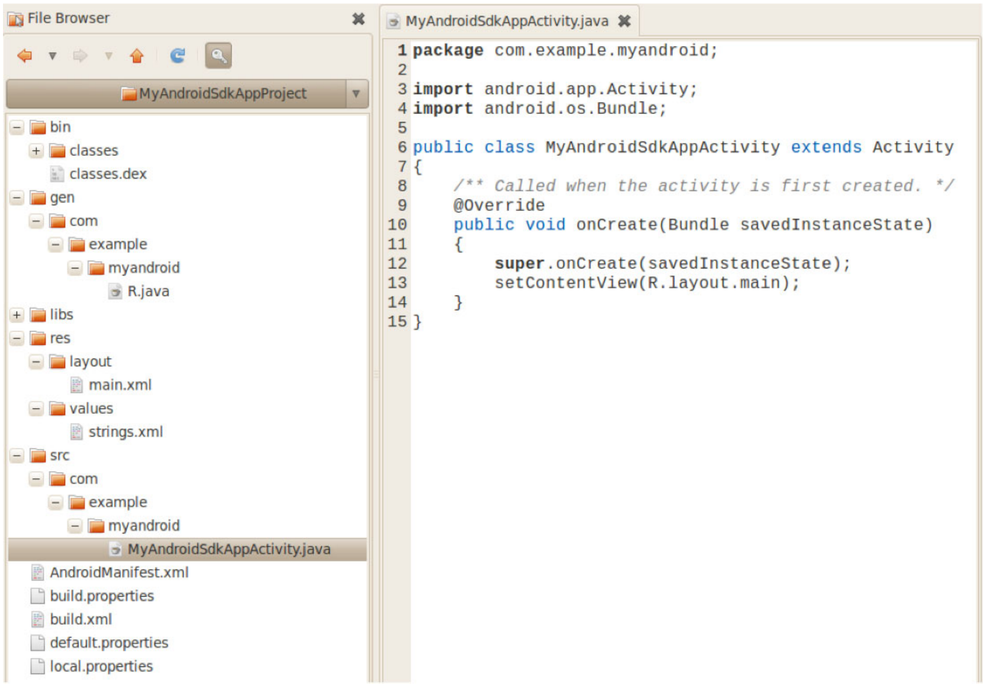
This should result in ample output similar to the following:

```
Buildfile: /HelloAndroidSdk/MyAndroidSdkAppProject/build.xml
    [setup] Android SDK Tools Revision 8
    [setup] Project Target: Android 2.3
    [setup] API level: 9 [setup] ...
BUILD SUCCESSFUL
Total time: 5 seconds
```

Assuming a successful build (as indicated by the message at the end of the listing) the `/MyAndroidSdkAppProject/bin` directory should now be populated with executable binaries. It should also contain debug versions of the application in the form of Dalvik Virtual Machine (DVM)–compatible classes (`classes.dex`) and Android application packages (`MyAndroidSdkApp-debug.apk`). We will cover them in more detail later on in the chapter.

The project directory should look similar to Figure 1–3. Feel free to investigate the project folder structures and the files that were created.

**THE MANIFEST FILE: ANDROIDMANIFEST.XML:** Another of the files that were generated in the root of the project is called the `AndroidManifest.xml` file. This is a very special file in that it defines and binds the application together. It is used by the Android SDK to declare essential information about the application for the benefit of the Android runtime system. Among other items, it identifies the application's Java package that serves as its unique name to the system, required permissions, components consumed and implemented, libraries to link against, and so on. Also see the Android Developers site for the Manifest File.[10]



```java
package com.example.myandroid;

import android.app.Activity;
import android.os.Bundle;

public class MyAndroidSdkAppActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

**Figure 1–3.** *Generated application directory and files*

**5.** Of course, we are eager to launch our new application, but first we need a device for it to run on. Because we will generally not use a physical device for ongoing development, we require a virtual machine on which to run an emulation of the Android runtime platform. The Android SDK takes care of both requirements.

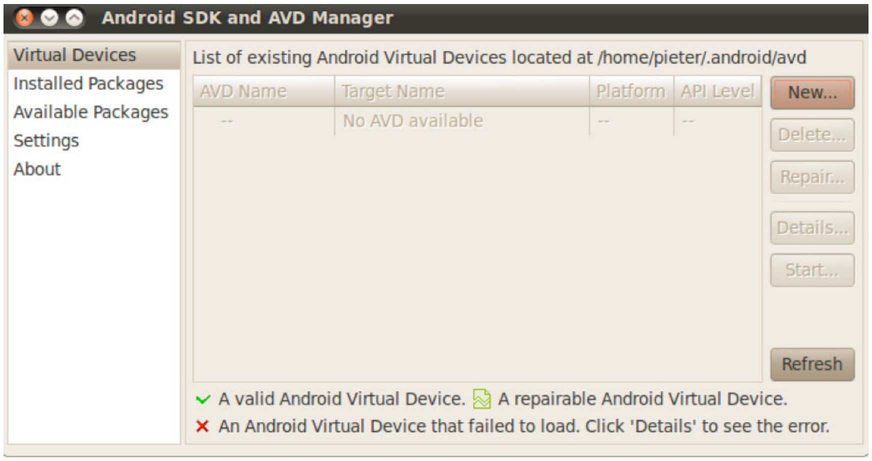
[10] `http://developer.android.com/guide/topics/manifest/manifest-intro.html`

- An Android virtual machine is called an *Android Virtual Device (AVD)*, and multiple AVDs can be configured using the AVD Manager to model your test- and production target device configurations. Reference material can be found on the Android Virtual Devices web site.[11]
- The Android runtime platform emulation is provided in the Android SDK and is simply called the *Android emulator*. The emulator is the platform that will run our application. Complete information is available Android emulator web site.[12]

6. To create an AVD, we will start the AVD Manager on the terminal command line by issuing the following command:

```
$ android
```

This will launch the familiar `Android SDK and AVD Manager` (see Figure 1–4).



Android SDK and AVD Manager

List of existing Android Virtual Devices located at /home/pieter/.android/avd

| AVD Name | Target Name | Platform | API Level |
|---|---|---|---|
| -- | No AVD available | -- | -- |

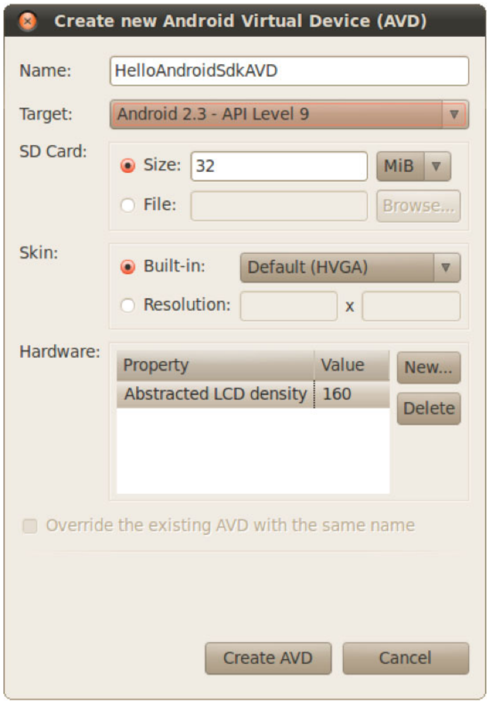
✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

**Figure 1–4.** *The Android SDK and AVD Manager with no AVDs*

7. Our next task is to create an AVD. Clicking the `New` button opens the `Create new Android Virtual Device (AVD)` form (see Figure 1–5).

[11] `http://developer.android.com/guide/developing/tools/avd.html`

[12] `http://developer.android.com/guide/developing/tools/emulator.html`



Create new Android Virtual Device (AVD)

| | |
|---|---|
| Name: | HelloAndroidSdkAVD |
| Target: | Android 2.3 - API Level 9 |
| SD Card: | [✓ Size: 32 MiB] File: |
| Skin: | [✓ Built-in: Default (HVGA)] Resolution: x |

Hardware:

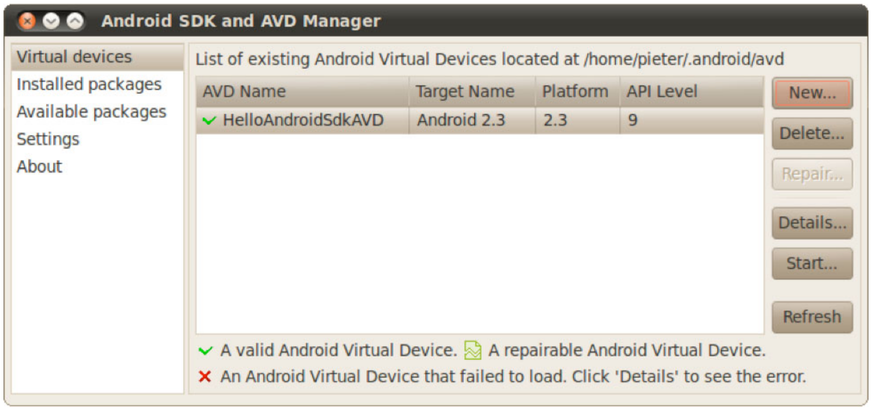
| Property | Value |
|---|---|
| Abstracted LCD density | 160 |

Override the existing AVD with the same name

**Figure 1–5.** *Creating a new AVD with the AVD Manager*

Fill out the text fields on the form to create a new AVD called `HelloAndroidSdkAVD` with a virtual SD card of 32MB in size. Then click the `Create AVD` button.

8. After an informational dialog telling us that the AVD was created successfully, we should be taken back to the main `Android SDK and AVD Manager` form (see Figure 1–6). Here we should now see our new `HelloAndroidSdkAVD` in the list of AVDs available to this instance of the Android SDK.



Android SDK and AVD Manager

List of existing Android Virtual Devices located at /home/pieter/.android/avd

| AVD Name | Target Name | Platform | API Level |
|---|---|---|---|
| ✓ HelloAndroidSdkAVD | Android 2.3 | 2.3 | 9 |

✓ 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 1–6.** *The Android SDK and AVD Manager listing the new Virtual Device*

9. Now that we have created our AVD, we can launch the emulator from the terminal and instruct it to run on top of our `HelloAndroidSdkAVD` virtual AVD. Issue this command on the console:

```
$ emulator -avd HelloAndroidSdkAVD
```

Because this is the first time we launch the emulator with our brand-new AVD, it can take a little while for the startup to complete.

**ANOTHER WAY TO LAUNCH THE EMULATOR/AVD COMBINATION:** Launching the emulator with our AVD can also be achieved directly from the AVD Manager graphical user interface (GUI) application by selecting the AVD in the `Virtual Devices` list and clicking the `Start` button.

Once the emulator is up and running, we should see the Android platform startup screen (see Figure 1–7). We now have a device to run our test application on. This device is essentially a full implementation of the Android platform stack including the DVM that, along with the AVD, provides us with a complete virtual mobile device. Leave the emulator running or restart it for the next section.

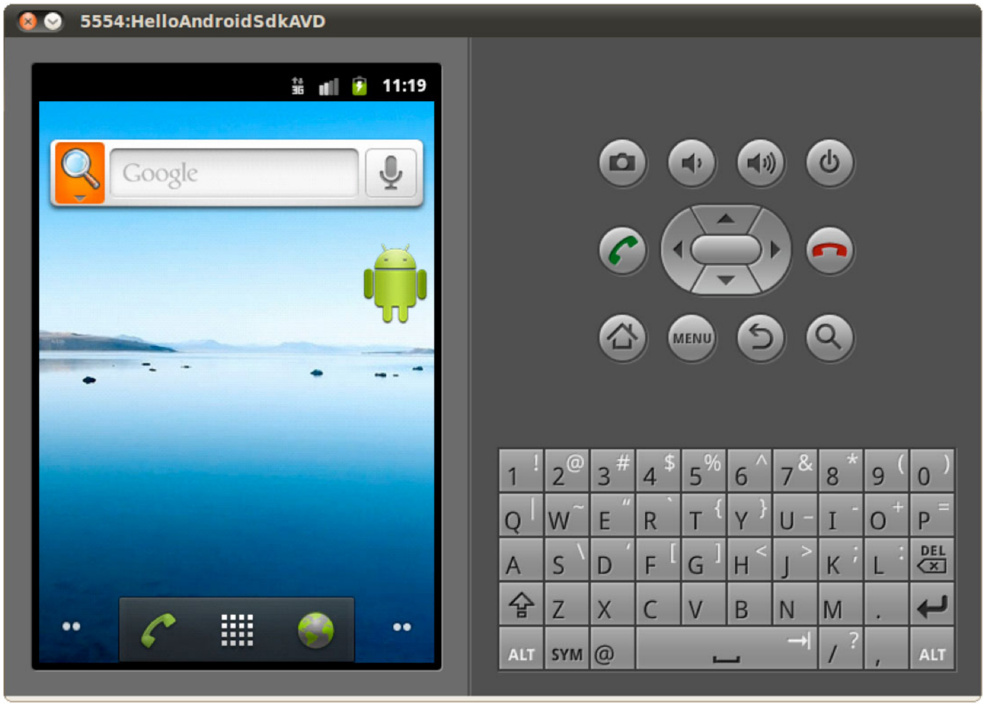


**Figure 1–7.** *The Android emulator running the new AVD*