ANDROID PROGRAMMING IN A DAY

THE POWER GUIDE FOR BEGINNERS IN ANDROID APP PROGRAMMING!

2ND EDITION

SAM KEY

ANDROID AT WORK!

Android Programming In a Day!

BY SAM KEY

The Power Guide for Beginners In Android App Programming

Table of Contents

Introduction

I want to thank you and congratulate you for downloading the book, “Introduction to Android Programming in a Day – The Power Guide for Beginners in Android App Programming”.

This book contains proven steps and strategies on how to get started with Android app development.

This book will focus on preparing you with the fun and tiring world of Android app development. Take note that this book will teach you basic Android app programming and designing. Android development is a huge subject to tackle. And learning everything about it in just one slim book is nigh impossible. Nevertheless, using this book as a foundation of your journey in Android app development is crucial.

On the other hand, this book will mostly revolve around the familiarization of the Java programming, XML writing, Android operating system, Android SDK, Eclipse IDE, and Android Studio.

Why not focus on programming immediately? Unfortunately, the biggest reason many aspiring Android developers stop on learning this craft is due to the lack of wisdom on the tools and skills they need to make Android app development easier.

Sure, you can also make apps using other languages like Python and other IDEs on the market. However, you can expect that using those is much more difficult than learning what Google has recommended. As of now, Google recommends using the newest Android Studio and Android SDK to develop Android apps. Back then, Google and most Android app developers recommend using Eclipse and ADTs (Android Development Tools.

On a different note, you can use some online tools to develop your Android app for you. But where’s the fun in that? You will not learn if you use such tools. Although it does not mean that you should completely stay away from that option.

Anyway, the book will be split into two parts. The first part will prepare you and tell you the things you need before you develop apps and familiarize you with the skills and tools you need to develop Android apps. The second part will teach you in-depth knowledge about the Android operating system and Android

apps.

Also, this book will be sprinkled with tidbits about the basic concepts of Android app development, as well as various aspects – from the program structure to navigation. As you read along, you will have a concrete idea on what to do next.

Thanks again for downloading this book. I hope you enjoy it!

Part I: Preparations for Android App Development

Chapter 1: Introduction to Android

As you might already know, Android is an operating system for mobile devices such as smartphones and tablets. This operating system is based on Linux, which in turn is based on UNIX. It shares minor similarities to Apple’s iOS due to the fact that iOS is another UNIX-inspired operating system.

Mobile device application development has become easier with Android. Before, smartphone application development was difficult since most phones only used their respective firmware to operate. On the other hand, some ran on different types of operating systems.

With Android, developers can just create software for it and expect that their creations can run on all devices that use this operating system. By the way, the Open Handset Alliance created Android. The Open Handset Alliance is composed of Google and some other companies.

The first version of Android, which is Android 1.0, was launched in September 2008. Before that, the beta SDK (Software Development Kit) for Android was released in 2007. As of now, the latest version of Android is Android 5.0 Lollipop.

If you want to go in depth with Android, you can check its source code. Fiddling with Android's source code is okay, since it is free and open source. Android is under Apache License and GNU general Public License.

Android Features

Of course, for you to have a powerful application, you should know the capability of Android. Take note that some older versions of Android may not have some of the features mentioned here. On the other hand, some of the features may not be available in certain mobile devices.

- Support Multiple Connectivity Types: WiMAX, GSM, EDGE, CDMA, IDEN, EV-DO, Bluetooth, LTE, Wi-Fi, NFC, UMTC
- Wi-Fi Direct
- Storage Type: SQLite (a relational database commonly used everywhere)

- Messaging: MMS and SMS
- Media File Type Support: H.264, H.263, AMR, MPEG0-4, AMR-WB, HE-AAC, AAC, MP3, AAC 5.1, MIDI, WAV, OGG Vorbis, BMP, PNG, GIF, JPEG
- Built in Web Browser: Supports CSS3. HTML5, and JavaScript
- Other Features: Multi Touch, Multi Language, Widgets, GCM, Wi-Fi Direct, Android Beam

Android Application Development

Android application development requires basic knowledge of the Java language. In case that you are not familiar with Java, it is best that you step away from this book and learn that first. However, with basic knowledge of computer programming, you can get away without Java experience or expertise. Nevertheless, expect that you will face a very steep learning curve.

When you finish developing a program in Android, you can easily run it in your emulator or smartphone. In case that you have created a useful app, you can easily upload and share or sell it on Amazon's Appstore or Google Play.

Android application development is one of the most profitable ventures in the programming development industry. As of now, Android exists on millions of mobile devices and smart appliances in the world. Truth be told, it has already outnumbered the number of computers installed with Microsoft Windows. It is also not an exaggeration that, almost every day, a million of Android powered devices are activated.

In this book, you will learn how to create and package your own Android application. It will provide you with a step-by-step tutorial on preparing yourself and your computer for app development. And of course, the book will be sprinkled with some pro tips that can get you going in the programming industry fast.

Chapter 2: The Architecture of the Android Operating System

An Android operating system, regardless of version number, is composed of five software components. They are the Linux kernel, libraries, Android runtime, application framework, and applications. In case that you are familiar with operating systems, Android is based on Linux, which is also based on UNIX.

As an FYI, iOS is based on OS X, which is another operating system based on UNIX. Technically, you can consider iOS and Android as cousins.

The Linux Kernel

All operating systems have a kernel. The kernel is at the bottommost layer of the Android operating system. Without it, the rest of the components of the Android operating system will be useless.

The kernel's job is to communicate and manage the devices or hardware inside a computer or smartphone. It manages the resources that are used in the device. And it makes it possible for the CPU (Central Processing Unit) to communicate with other hardware components present in a device.

The kernel is capable of handling those components thanks to device drivers. A few of the usual components in an Android smartphone are the screen, camera, flash memory, keypad, Wi-Fi, speakers, and battery. All of those devices have device drivers that allow the kernel and the CPU to understand how to communicate with them.

The kernel is the core of an operating system. Without it, all the other core components in the operating system will be unusable. By the way, the Android operating system is currently using Linux 2.6 together with 115 or more patches. Due to its Linux roots, you can say that an Android device is as powerful as a computer. Heck, you can even set up an Android smartphone for use as a server in a matter of seconds.

Android Libraries

Just above the kernel are the Android libraries. In programming, libraries are reusable program components that make program development and computer operations easier. A few of the known libraries in Android are surface manager, media framework, SQLite, OpenGL, ES, FreeType, WebKit, SGL, SSL, and libc.

These libraries make things easier for programmers by preventing them from reinventing the wheel or recreating a component. For example, the WebKit is a library that contains procedures, classes, subroutines, and pre-written code to give a program the capability to browse the web.

In case that your program will have minor browsing capabilities or it will show ad banners from the web, you can just use the functions from the WebKit library. You do not need to develop your own web browser in your program, which is an inefficient task by itself.

On the other hand, the SSL library helps programs when it comes to establishing secure internet connections, while SQLite provides database capabilities to programs. You will learn the use of the other Android libraries as you advance in your path as an Android app developer.

Android Runtime

Just beside the Android libraries in the second layer in the Android platform is the Android runtime. It has two of the primary components that allow most applications on your smartphone or Android device to work. First are the core libraries. Second is the Dalvik Virtual Machine.

The two of them allows your Android devices to run programs written using Java. Primarily, the Dalvik Virtual Machine acts as a compiler for your Java applications. The Dalvik Virtual Machine reads and translates your Java programs to machine code, which allows your phone to execute them easily.

Aside from allowing Java programs to run in your phone, it also supports and performs multi threading and memory management. It means that it can allow your Java or Android programs to run simultaneously.

As an FYI, Java programs always require a virtual machine in order to run. Its primary function is to allow Java programs to run in any operating system. Normally, programs written in other languages need to be written precisely for the operating system it will run on.

Java programs, on the other hand, are needed to be programmed for the virtual machine. Instead of the programs being tailored for the operating systems, the virtual machines tailor itself to the operating system, which does a lot of favor to Java developers.

The Dalvik Virtual Machine was developed primarily for Android. Aside from allowing Java programs to run on mobile devices, it also makes them run optimally — in a sense that it will use less system resources and memory storage space. On the other hand, the core libraries are there to aid the Java programs to run.

Android Application Framework

The Android application framework is the third software layer in the Android operating system. It houses the much needed higher-level services and the basic functionalities of an Android device. All of the pieces of software in the Android application framework are written as Java classes, which make them dependent on the Android runtime component.

A few of the apps or services included in the Android application framework are the activity manager, window manager, content providers, view system, package manager, telephony manager, resource manager, location manager, and notification manager.

Android app developers need to work on these frameworks in order to make their applications seamlessly work with the Android operating system. Thankfully, Android developers can also use the framework to create much richer apps.

Android Applications

Finally, what is left is Android's application layer — the topmost one. Primarily, creating them is your goal. By default, the Android operating system comes with contacts, phone, browser, and home apps.

The complexity of the app depends on how much features you include. Take note that your applications will not work in case that one of the other components or layers of your operating system is damaged or missing.

On the other hand, you can save a lot of time if you take advantage of the libraries and frameworks in the Android operating system. Do remember that newer versions of Android have improved frameworks and libraries. And when creating an app, be mindful of the libraries and functions that you will use.

Some functions of libraries or frameworks in newer versions of Android are not present in older versions. In case a user runs an application that requires those functions but he is using an older version of Android, the app will not work.

Chapter 3: Preparation to Android App Programming

Android application development is not easy. You must have some decent background in program development. It is a plus if you know Visual Basic and Java. And it will be definitely a great advantage if you are familiar or have already used Eclipse's IDE (Integrated Development Environment). Also, being familiar with XML will help you.

You will need a couple of things before you can start developing apps.

Your Computer

First, you will need a high-end computer. It is common that other programming development kits do not need a powerful computer in order to create applications. However, creating programs for Android is a bit different. You will need more computing power for you to run Android emulators, which are programs that can allow you to test your programs in your computer.

Using a weak computer without a decent processor and a good amount of RAM will only make it difficult for you to run those emulators. If you were able to run it, it will run slowly.

Android Device

Second, you will need an Android device. That device will be your beta tester. With it, you will know how your program will behave in an Android device. When choosing the test device, make sure that it is at par with the devices of the market you are targeting for your app. If you are targeting tablet users, use a tablet. If you are targeting smartphones, then use a smartphone.

Android SDK

Third, you will need the Android SDK (Software Development Kit) from Google. The SDK is a set of files and programs that can allow you to create and compile your program's code. As of this writing, the latest Android SDK's file size is around 350mb. It will take you 15 – 30 minutes to download it. If you uncompressed the Android SDK file, it will take up around 450mb of your computer's disk space. The link to the download page is http://developer.android.com/sdk/index.html. By the way, when you get the SDK, you will also get Android Studio.

The SDK can run on Windows XP, Windows 7, Mac OSX 10.8.5 (or higher), and Linux distros that can run 32bit applications and has glibc (GNU C library) 2.11 or higher.

Once you have unpacked the contents of the file you downloaded, open the SDK Manager. That program is the development kit's update tool. To make sure you have the latest versions of the kit's components, run the manager once in a while and download those updates. Also, you can use the SDK Manager to download older versions of SDK. You must do that in case you want to make programs with devices with dated Android operating systems.

Eclipse and ADT (Android Development Tools)

Back then, Eclipse IDE was chosen as the preferred way to develop Android apps. After all, it was the primary IDE that most Java programmers use. And since Android apps are basically Java apps for the Android operating system, people have started using Eclipse.

However, Eclipse IDE alone is not capable of making life easier for Android app developers. Due to that, ADT or Android Development Tools for Eclipse IDE was developed. ADT is a plugin for Eclipse that makes the IDE more ready in developing Android applications.

If you plan to use Eclipse, it is essential that you use the latest version of Eclipse (3.7.2 Indigo or higher). As of this writing, the latest version of Eclipse is 4.4 Luna. To get Eclipse, go to this page: http://www.eclipse.org/mobile/. Eclipse is around 200MB to 300MB big.

JRE (Java Runtime Environment) and JDK (Java Development Kit)

Any programming venture with Java involved requires the Java Runtime Environment and Java Development Kit. JRE allows a computer to run Java programs, while JDK allows a programmer to develop Java programs. Eclipse and even Android Studio, which will be discussed later, highly depend on these two.

By default, if you are using a computer running on Microsoft Windows, it is probable that you already have JRE installed. However, please do note that it is a must that your JRE is up to date. You can update your Java by just going to your computer's control panel, opening the Java control panel by clicking Java, clicking on the Update tab on the Java's control panel, and clicking the Update Now button.

On the other hand, you must download JDK as it does not come preinstalled in computers. You can download it on Oracle's website: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Take note that these two programs alone can cost you 1GB to 2GB of disk space.

Android Studio

On the other hand, you have the option to use the new Android Studio. The Android Studio contains the Gradle, Google Cloud, JUnit, Maven, Live Preview, Android Studio IDE, Android SDK tools, Android 5.0 (Lollipop) platform, and Google APIs.

Most Android developers are shifting to this app development software package. As of now, it is considered the official IDE for Android apps. One of its main features is its intelligent code editor, which contains the following features: code analysis, refactoring, and advanced code/keyword completion.

Take note that just like the Android SDK, it will require a decent computer. Your computer must have 4GB RAM or more for optimal development experience. You must at least have 5GB free storage space. The installer alone requires 800MB. The Studio and Android SDK requires approximately 3GB or more. And the JDK requires 1GB. By the way that you might need to free up 2GB or more in case that you will need to add optional packages for your SDK.

By the way, despite being the primary development IDE for Android, you can still make do with Eclipse. Also, if you have already done any project in Eclipse, you can easily import it to Android Studio using its import function.

Installation of all those IDE is easy. Just follow the instructions. To get Android Studio, just go to the link mentioned in the Android SDK section.

Chapter 4: Basic Programming and Writing Course in Java and XML

Most of the time, you will be creating your app using the IDEs Android Studio and Eclipse. And most of the time, you will be dragging and dropping view elements, adjusting their properties, and adding some minor tweaks that can be done with a few clicks.

Unfortunately, those actions have their limits. And to be frank, you are just developing the tip of the iceberg — the app's design. Anyway, the point is that you are required to type in a few lines of code to make sure that your app will work just the way you want it.

To develop Android by manually writing source code, you must learn and get familiar with Java and XML. Java is a programming language. XML, on the other hand, is a markup language. Files written in Java ends with the .java extension, while files written in XML ends with the .xml extension.

Mostly, when dealing with Java in Android development, you are actually writing your program's code. Particularly, all the things that happen in your app will be dependent on your app's java code. In Android Studio, your app's main java file is located at app > src > main > java > com.example.admin.myapplication (or your app's reverse domain or package name) > MainActivity (or any name that you have assigned for your program). In case that you are going to use Eclipse, the main java file should be located at src > your app's package name > app name.java.

On the other hand, you will be primarily dealing with XML in case you are going to edit your app's activity and certain resources. For example, you will be editing the XML files for your activities saved in the app > src > main > res > layout. Also, you will be editing the XML file for the string values in your app in app > src > main > res > values > strings.xml.

Java Programming

Learning Java in one go can be a pain. However, you do not need to master Java in order to create a decent Android app. The important thing you should learn is how to write Java code properly by abiding to its syntax rules and familiarizing yourself with programming in general.

Syntax is a collection of rules in a language or programming languages. You can think of it as grammar rules in writing English sentences and paragraphs. Learning the syntax of a programming language is essential. Take note that one small syntax mistake in your source code's can ruin your app or make it behave the way you do not want to.

But even before you do learn the rules, you must learn the parts of the Java language. After all, learning the grammar rules in English is useless if you do not know where those rules apply — in English's case, those rules apply to the parts of the sentence.

Identifiers

Identifiers are names in programming languages. It can be used as a reference for an object, a name for a function, or a name for a variable. Identifiers make it easier for programmers to assign, invoke, and use other elements in the program.

When it comes to syntax, take note that Java is case sensitive. A variable with an identifier X is different from a variable with an identifier x.

Most Java programmers have naming conventions that they follow when creating identifiers. Writing letter case separated words (or CamelCase) is the most common naming convention for Java programmers.

When dealing with identifiers with multiple words, those words are separated with an uppercase first letter. For example: thisIsAVariable is a variable identifier that uses the letter case naming convention. By the way, it is common that the first letter of the identifier is in lower case.

Keywords

Keywords are identifiers that are reserved for special functions in a programming language. Since they are reserved, you cannot use them as identifiers for the other elements in your program. When using a source code editor or your IDE's text editor, keywords are highlighted with a different color — usually with blue — for you to easily identify them. In Java, there are around 50 keywords.

Literals

Literals are the raw data that you present to your source code. There are two common literal types that you will be using in your app's source code. The first literal type is numbers. The second type is strings. By the way, strings are text enclosed in single or double quotations. They are enclosed in quotations to prevent the compiler of Android Studio in mistaking your text as keywords, identifiers, or numbers.

Numbers can be represented in multiple ways in Java. It can be presented in decimal, octal, hexadecimal, or binary form. Other types of literals include Boolean values (truth values: true and false), null, and character.

Variables

Variables are identifiers that 'hold' data or literals. Whenever you assign literals to identifiers, they become variables. Unlike in other programming languages, you need to declare a variable and set the type of literal it can store. For example:

int thisVariableCanStoreIntegers; //This line of code declare thisVariableCanStoreIntegers as a variable that can hold integers. The line used the keyword int to tell Java that the variable's type is integer

thisVariableCanStoreIntegers = 23; //This line of code initialize the variable and assigns the integral number 23 to it

int anotherVariable = 14; //This line declares, initialize, and assign a value to the variable anotherVariable

int x, y, z; //You can declare multiple variables in one statement by using the comma (,) separator

int x = 2, y = 3, z = 4; //You can also do multiple assignments

Code Blocks

Code blocks are like paragraphs in the English language. A code block contains multiple sets of statements (or lines of codes), that work together to achieve a single goal. In some cases, the statements in code blocks are just put together to perform a batch of commands in one go.

Code blocks are used whenever you will need to create methods or classes —

two things that you will learn when you advance in Java or Android app development. To contain the statements within a code block, they are enclosed inside curly braces ({}) or separators. These braces are also used to signify a new scope. For example:

```
void thisIsARandomMethod() {
        int aVariable;
        aVariable = 2;

                                {

            int b;
            b = 3;

                                }

                                }
```

In the example, a new scope was created inside the code block. Take note that anything that you do in a different scope is independent from other parent or sibling scopes. For example, the variable b cannot be used outside its scope. However, child scopes can use the variables from parent scopes. You can use the aVariable inside the child scope.

By the way, indentation in code blocks is not necessary when programming using Java. However, they do make your source code neater and cleaner.

Comments

When typing source code, there will be times that you will need to place markers, explanations, or labels in your statements. This is when comments become handy. Comments in programming languages are lines of text that are ignored by the compiler or interpreter. They actually have no use for the computer or program, but they are indispensable tools for the programmer.

It is common in source code editors that comments are highlighted in green. However, in Android Studio, comments are highlighted in gray and italicized.

When programming in Java, you have three methods or types that you can use to create comments. They are documentation comments, end of line comments, and traditional comments.

Traditional comments are created by enclosing text using the opening /* and the closing */. For example:

```
/* Traditional comments are commonly used
when creating comments with multiple
lines of text.
Everything inside the opening and closing
symbols are ignored by the compiler
and the computer */
```

Documentation comments are just like traditional comments. However, unlike the traditional comments, they are treated specially Usually, they are treated as documentation text for your source code or file. Instead of the opening /*, documentation comments start with the opening /** and close with the closing /. *Also, each additional line must start with an asterisk ()*. For example:

```
/** This is a short documentation file.
* @author: The Author of this Book

*/
```

Lastly, and most probably the most used kind of comment, is the end of line comment. End of line comments are placed at the end of the line and starts with //. Unlike the previous types, end of line comments do not use any closing tag. Due to that, anything written after the two slashes are considered comments. Because of that, they are usually placed at the end of a statement or a physical line. On the other hand, they can be used a standalone physical line. For example:

```
int dog = 3; //This comment is ignored. And will not affect the statement before
it

// int cat = 4; The statement in this line will not work since it is considered a
comment.
```

Operators and Expressions

Programming is mostly about operations and evaluation of data. In Java, multiple operators are available for to use. Some of them deal with manipulating

numbers, including basic Mathematical operations. Some deals with strings. And some will provide you with program control flow.

On the other hand, take note of the term expressions. Expressions are combinations of literals, value-giving keywords or methods, operators, and variables that can be evaluated. For example:

int a = 1 + 1;

In the example, two operators were used. First is the assignment operator (=). The assignment operator allows associating a variable to a literal. Second is the addition operator (+). The addition operator adds the two numbers beside it. The expression in the statement is 1 + 1, which can be evaluated. The assignment operator will take the evaluated value of the expression to the variable.

Just like in Mathematics, operators in programming languages and Java have an order of precedence — an order which prioritizes the operators that should be evaluated first. Below is a table of most of the available operators in Java and their order of precedence.

Order of Precedence	Operator Symbol	Description / Purpose	Associativity
1	( )	Invokes methods, group variables, and literals	Left to Right
	[ ]	Initialize or access an array	
	.	Selects class member or methods	
2	++ --	Increment and decrement value of the literal or variable on its left by 1 if placed to the right (postfix)	
3	++ --	Increment and decrement value of the literal or variable on its	Right to Left

		right by 1 if placed to the left (prefix)	
	+ -	Unary minus and plus: indicates the sign of a numerical variable or literal if placed before the variable or literal (prefix)	
	!	Logical NOT Negates truth values. Returns 1 if value is 0. And returns 0 if value is not 0.	
	~	Bitwise NOT negates digits in a binary literal or variable	
4	* / %	Multiplication, division, modulo (division that returns the remainder instead of the quotient)	Left to Right
5	+ -	Addition, subtraction if placed in the middle of two variables or literals	
	+	Sting Concatenation if placed in the middle of two	

		variables or string literals (join strings together)	
6	>> << >>>	Bitwise signed right shift, left shift, unsigned right shift	
7	> >=	Less than, less than or equal to (usually used for relational comparison in conditional statements)	
	< <=	Greater than, Greater than or equal to (usually used for relational comparison in conditional statements)	
8	== !=	Is equal to, is not equal to (usually used for relational comparison in conditional statements)	
9	&	Logical and Bitwise AND	
10	^	Logical and Bitwise XOR	
11	|	Logical and Bitwise OR	
12	&&	Logical conditional AND	
13		Logical	

	||	conditional OR	
14	c ? t : f	Ternary conditional ?	Right to Left
15	=	Assignment operator	
	+= -=	Add and assign, subtract and assign	
	*= /= %=	Multiply and assign, divide and assign, modulo and assign	
	<<= >>= >>>=	Bitwise left shift and assign, signed right shift and assign, unsigned right shift and assign	
	&= ^= |=	AND and assign, XOR and assign, OR and assign	

Note that the lower the order of precedence, the higher the priority of the operator. Some operators were not included and some are not explained fully since you will not be needing them anytime soon. The operators that you will be mostly using in your apps are the mathematical, conditional, assignment, and relational/conditional operators. In some cases, you might need to use logical operators.

Statements, Physical Line, and Logical Line

A statement is composed of all or any of those parts combined. In literature, a statement is a sentence — an imperative one. Take note that it is essential that you separate the elements in your statements with spaces. It is not imperative, but usage of space in coding improves the readability of your source code.

Every statement in your source code will tell your computer or Android device

to do something. A simple declaration and assignment of a variable is a statement. Statements are also called logical lines.

In Java, a statement will be considered as such if it ends with the semicolon (;) separator. Whenever you finish writing a logical line, never forget to include a semicolon. If not, you will receive a syntax error.

On the other hand, a physical line is a line of code in programming. A physical line can or cannot be a statement. As you might have noticed in source code editors, every line in the editor is numbered. A line of text or code is a physical line.

Below is an example of the initial Java code of a new app made in Android Studio:

```
package com.example.admin.myapplication;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
```

```
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

XML Writing

Primarily, you will not be writing too much XML in your app if you are going to use IDEs. After all, most of the XML writing tasks you will do should be dedicated in designing your app's appearance. Since the IDEs will do the writing for you when you drag and drop the views that you want to appear, XML writing should not be a big issue.

Nevertheless, if you want full control of your app's properties and appearance, then it is essential that you edit your app's XML files on your source code editor. Do not worry; compared to Java programming, XML writing is much easier.

If you are familiar with HTML (HyperText Markup Language), then it will be easy for you to understand XML (eXtensible Markup Language). XML is technically similar to HTML. The main difference is that XML is much simpler and straight to the point. Machines and humans (those who are familiar with XML of course) can easily understand it.

Below is an example of an activity's XML file:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
   android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

   <TextView android:text="@string/hello_world"
android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</RelativeLayout>
```

Content and Markup

Content is the information that users will see once the XML file is processed. In Android, that's the activity screen and the elements you add to it. On the other hand, markups are words, keywords, and syntax that allow you to define content and let the machines know what to do with your content.

Elements, Opening, Closing, and Empty Element Tags

Tags are element names enclosed in chevrons (< > or less than and greater than signs). XML elements or tags depend on the machine that will read it. For example, XML files for Android apps cannot be processed the same way by other machines not running on Android. But it does not mean that it cannot be read. For example, other machines have no use or have no idea on how to process the TextView element.

A markup line always starts with an opening tag and ends with a closing tag. For example, in an activity in an Android app with a relative layout, the markup starts with <RelativeLayout> and ends with a </RelativeLayout>. Take note that the closing tag of an opening tag has the same tag with an additional slash (/) prefix.

By the way, having markup tags that require closing tags means that you can nest another markup construct inside. For example, the RelativeLayout tag is a ViewGroup that can act as a container for other View elements such as TextView (text field).

On the other hand, empty element markup tags cannot contain or nest another markup construct within it. Most empty elements in Android are View elements.

Attribute

An element in an XML file can contain attributes or properties. For example, a TextView element has an attribute or property of android:text. The attribute android:text holds the string or text that will be displayed on the screen or activity. Attributes are placed together with the opening tag that is enclosed in chevrons.

Usually, IDEs will auto populate an element’s attributes once it is placed on the Design view, which makes things a whole lot easier.

By the way, Android View elements and other elements in an Android app have multitudes of attributes. Most of the time, the IDE will not put all the attributes in the XML file — unless the default values of those attributes are changed. Also, you can edit the attributes of an element by using the properties box or window in Eclipse or Android Studio.

Chapter 5: Android’s Application Components

Every Android app needs four necessary components in order to be rich and decent. The components are attached in the AndroidManifest.xml — the app’s manifest file. The manifest file indicates how the components interact and how they are related. The four components are:

- Services: this component manages the processes in the background created by the app.
- Activities: this component is responsible for the display and UI (User Interface). It also is responsible for the actions that the user will perform within the app’s UI.
- Content Providers: this component manages data management and related issues.
- Broadcast Receivers: this component manages the communication between the app and the operating system — mainly with Android’s framework.

Services

The service component is a background office. Users are mainly unaware of services. They are pushed to the background since users do not need to know or see what happens when a service is running.

Usually, services handle boring and tedious operations needed for the app to do its thing or tasks. A good example is Facebook's Messenger app. Even if you switch to another app, its services will still run. One of its services is to fetch messages from the server in case that somebody messages you in Facebook.

When creating a service in your app, you will need to put it under the Service class as a subclass. For example:

```
public class ExampleService extends Service {
   <Insert the code here for ExampleService>
}
```

Activities

You can imagine an activity as a page of your app. In simple terms, an activity is a screen UI in your app. Every activity is loaded with UI elements in order for your user to complete a task or use the app. In Facebook's Messenger app, you can access multiple activities or pages.

For example, at the start of the Messenger app, you will see a login page. When you are logged in, the app will show the activity for your friends list. When you try to message a friend in the messenger, you will be transferred to the chat activity or page.

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

When creating an activity in your app, you will need to put it under the Activity class as a subclass. For example:

```
public class MainPage extends Activity {
   <Insert the code here for MainPage>

}
```

Content Providers

This component manages the flow of data from the application itself to others. The content provider can either store the data or information it has on the storage or file system, or put it in a database. Mostly, you will be using ContentResolver class methods when creating a content provider class.

When creating a content provider in your app, you will need to put it under the ContentProvider class as a subclass. For example:

```
public class ExampleContentProvider extends ContentProvider {
   <Insert the code here for ExampleContentProvider >

}
```

Broadcast Receivers

Broadcast receivers handle the messages that the system or other applications send or broadcast. Usually, broadcasts from other apps and the system tell applications about certain events such as a finished download. The broadcast receiver analyzes the message and performs the necessary action or response to the message.

When creating an activity in your app, you will need to put it under the BroadcastReceiver class as a subclass. The messages it receive are treated as Intent objects. For example:

```
public class ExampleBroadcastReceiver extends BroadcastReceiver {
   <Insert the code here for ExampleBroadcastReceiver>

}
```

Other Components

Of course, those are not the only components that your app may have. Some other components serve as ‘connectors’ and ‘enhancements’ to those. Using them wisely can get your app feature-rich and highly functional.

- Views: are the regular UI elements you typically see on activities such as lists, buttons, textboxes and etcetera.
- Fragments: unlike the views or UI elements, fragments are certain behaviors or responses on activities.
- Intents: their main function is to wire components together using messages.
- Layouts: handles and formats the appearance of views. These also manage the screen.
- Manifest: this is like the configuration or settings file of your application.
- Resources: are external data or files that are not coded within your application.

•

Chapter 6: Starting Your First Project Using Eclipse and Android SDK

To start creating programs, you will need to open Eclipse. The Eclipse application file can be found under the eclipse folder on the extracted files from the Android SDK. Whenever you run Eclipse, it will ask you where you want your Eclipse workspace will be stored. You can just use the default location and just toggle the don’t show checkbox.

New Project

To start a new Android application project, just click on the dropdown button of the New button on Eclipse's toolbar. A context menu will appear, and click on the Android application project.

The New Android Application project details window will appear. In there, you will need to input some information for your project. You must provide your program's application name, project name, and package name. Also, you can configure the minimum and target SDK where your program can run and the SDK that will be used to compile your code. And lastly, you can indicate the default theme that your program will use.

Application Name

The application name will be the name that will be displayed on the Google's Play Store when you post it there. The project name will be more of a file name for Eclipse. It will be the project's identifier. It should be unique for every project that you build in Eclipse. By default, Eclipse will generate a project and package name for your project when you type something in the Application Name text box.

Package Name

The package name is not usually displayed for users. Take note that in case you will develop a large program, you must remember that your package name should never be changed. On the other hand, it is common that package names are the reverse of your domain name plus your project's name. For example, if your website's domain name is www.mywebsite.com and your project's name is Hello World, a good package name for your project will be com.mywebsite.helloworld.

The package name should follow the Java package name convention. The naming convention is there to prevent users from having similar names, which could result to numerous conflicts. Some of the rules you need to follow for the package name are:

- Your package name should be all in lower caps. Though Eclipse will accept a package name with a capital letter, but it is still best to adhere to standard practice.
- The reverse domain naming convention is included as a standard practice.
- Avoid using special characters in the package name. Instead, you can replace it with underscores.
- Also, you should never use or include the default com.example in your package name. Google Play will not accept an app with a package name like that.

Minimum SDK

Minimum required SDK could be set to lower or the lowest version of Android. Anything between the latest and the set minimum required version can run your program. Setting it to the lowest, which is API 1 or Android 1.0, can make your target audience wider.

Setting it to Android 2.2 (Froyo) or API 8, can make your program run on almost 95% of all Android devices in the world. The drawback fn this is that the features you can include in your program will be limited. Adding new features will force your minimum required SDK to move higher since some of the new functions in Android is not available on lower versions of the API (Application Programming Interface).

Target SDK

The target SDK should be set to the version of Android that most of your target audience uses. It indicates that you have tested your program to that version. And it means that your program is fully functional if they use it on a device that runs the target Android version.

Whenever a new version of Android appears, you should also update the target SDK of your program. Of course, before you release it to the market again, make sure that you test it on an updated device.

If a device with the same version as your set target SDK runs your program, it will not do any compatibility behavior or adjust itself to run the program. By default, you should set it to the highest version to attract your potential app buyers. Setting a lower version for your target SDK would make your program old and dated. By the way, the target SDK should be always higher or equal with the minimum target SDK version.

Compile with

The compile with version should be set to the latest version of Android. This is to make sure that your program will run on almost all versions down to the minimum version you have indicated, and to take advantage of the newest features and optimization offered by the latest version of Android. By default, the Android SDK will only have one version available for this option, which is API 20 or Android 4.4 (KitKat Wear).

After setting those all up, it is time to click on the Next button. The new page in the screen will contain some options such as creating custom launcher icon and creating activity. As of now, you do not need to worry about those. Just leave the default values and check, and click the Next button once again.

Custom Launcher Icon

Since you have left the Create Custom Launcher option checked, the next page will bring you in the launcher icon customization page. In there, you will be given three options on how you would create your launcher. Those options are launcher icons made from an image, clipart, or text.

With the text and clipart method, you can easily create an icon you want without thinking about the size and quality of the launcher icon. With those two, you can just get a preset image from the SDK or Android to use as a launcher icon. The same goes with the text method since all you need is to type the letters you want to appear on the icon and the SDK will generate an icon based on that.

The launcher icon editor also allows you to change the background and foreground color of your icon. Also, you can scale the text and clipart by changing the value of the additional padding of the icon. And finally, you can add simple 3D shapes on your icon to make it appear more professional.

Bitmap Iconography Tips

When it comes to images, you need to take note of a few reminders. First, always make sure that you will use vector images. Unlike the typical bitmap images (pictures taken from cameras or images created using Paint), vector images provide accurate and sharp images. You can scale it multiple times, but its sharpness will not disappear and will not pixelate. After all, vector images do not contain information about pixels. It only has numbers and location of the colors and lines that will appear in it. When it is scaled, it does not perform antialiasing or stretching since its image will be mathematically rendered.

In case that you will be the one creating or designing the image that you will use for your program and you will be creating a bitmap image, make sure that you start with a large image. A large image is easier to create and design.

Also, since in Android, multiple sizes of your icon will be needed, a large icon can make it easier for you to make smaller ones. Take note that if you scale a big picture into a small one, some details will be lost, but it will be easier to edit and fix and it will still look crisp. On the other hand, if you scale a small image into a big one, it will pixelate and insert details that you do not intend to show such as jagged and blurred edges.

Nevertheless, even when scaling down a big image into a smaller one, do not forget to rework the image. Remember that a poor-looking icon makes people think that the app you are selling is low-quality. And again, if you do not want to go through all that, create a vector image instead.

Also, when you create an image, make sure that it will be visible in any background. Aside from that, it is advisable to make it appear uniform with other Android icons. To do that, make sure that your image has a distinct silhouette that will make it look like a 3D image. The icon should appear as if you were looking above it and as if the source of light is on top of the image. The topmost part of the icon should appear lighter and the bottom part should appear darker.

Activity

Once you are done with your icon, click on the Next button. The page will now show the Activity window. It will provide you with activity templates to work on. The window has a preview box where you can see what your app will look like for every activity template. Below the selection, there is a description box that will tell you what each template does. For now, select the Blank Activity and click Next. The next page will ask you some details regarding the activity. Leave it on its default values and click Finish.

Once you do that, Eclipse will setup your new project. It might take a lot of time, especially if you are using a dated computer. The next chapter will discuss the programming interface of Eclipse.

Chapter 7: Getting Familiar with Eclipse and Contents of an Android App

When Eclipse has finished its preparation, you will be able to start doing something to your program. But hold onto your horses; explore Eclipse first before you start fiddling with anything.

Editing Area

In the middle of the screen, you will see a preview of your program. In it, you will see your program's icon beside the title of your program. Just left of it is the palette window. It contains all the elements that you can place in your program.

Both of these windows are inside Eclipse's editing area. You will be spending most of your time here, especially if you are going to edit or view something in your code or layout.

The form widgets tab will be expanded in the palette by default. There you will see the regular things you see in an Android app such as buttons, radio buttons, progress bar (the circle icon that spins when something is loading in your device or the bar the fills up when your device is loading), seek bar, and the ratings bar (the stars you see in reviews).

Aside from the form widgets, there are other elements that you can check and use. Press the horizontal tabs or buttons and examine all the elements you can possibly use in your program.

To insert a widget in your program, you can just drag the element you want to include from the palette and drop it in your program's preview. Eclipse will provide you visual markers and grid snaps for you to place the widgets you want on the exact place you want. Easy, right?

Take note, some of the widgets on the palette may require higher-level APIs or versions of Android. For example, the Grid Layout from the Layouts section of the palette requires API 14 (Android 4.0 Ice Cream Sandwich) or higher. If you add it in your program, it will ask you if you want to install it. In case you did include and install it, remember that it will not be compatible for older versions or any device running on API 13 and lower. It is advisable that you do not include any element that asks for installation. It might result into errors.

Output Area, Status Bar, and Problem Browser

On the bottom part of Eclipse, the status bar, problem browser, and output area can be found. It will contain messages regarding to the state of your project. If Eclipse found errors in your program, it will be listed there. Always check the Problems bar for any issues. Take note that you cannot run or compile your program if Eclipse finds at least one error on your project.

Navigation Pane

On the leftmost part of your screen is the navigation pane that contains the package explorer. The package explorer lets you browse all the files that are included in your project. Three of the most important files that you should know where to look for are:

- activity_main.xml: This file is your program's main page or window. And it will be the initial file that will be opened when you create a new project. In case you accidentally close it on your editor window, you can find it at: YourProjectName > res > layout > activity_main.xml.
- MainActivity.java: As of now, you will not need to touch this file. However, it is important to know where it is since later in your Android development activities, you will need to understand it and its contents. It is located at: YourProjectName > src > YourPackageName > MainActivity.java.
- AndroidManifest.xml: It contains the essential information that you have set up a while ago when you were creating your project file in Eclipse. You can edit the minimum and target SDK in there. It is located at YourProjectName > AndroidManifest.xml.

Aside from those files, you should take note of the following directories:

- src/: This is where most of your program's source files will be placed. And your main activity file is locafile is located.
- res/: Most of the resources will be placed here. The resources are placed inside the subdirectories under this folder.
- res/drawable-hdpi/: Your high density bitmap files that you might show in your app will go in here.
- res/layout/: All the pages or interface in your app will be located here – including your activity_main.xml.
- res/values/: The values you will store and use in your program will be placed in this directory in form of XML files.

In the event that you will create multiple projects, remember that the directory for those other projects aside from the one you have opened will still be available in your package explorer. Because of that, you might get confused over the files you are working on. Thankfully, Eclipse's title bar indicates the location

and name of the file you are editing, which makes it easier to know what is currently active on the editing area.

Outline Box

Displays the current structure of the file you are editing. The outline panel will help you visualize the flow and design of your app. Also, it can help you find the widgets you want to edit.

Properties Box

Whenever you are editing a layout file, the properties box will appear below the outline box. With the properties box, you can edit certain characteristics of a widget. For example, if you click on the Hello World text on the preview of your main activity layout file, the contents of the properties box will be populated. In there, you can edit the properties of the text element that you have clicked. You can change the text, height, width, and even its font color.

Menu and Toolbar

The menu bar contains all the major functionalities of Eclipse. In case you do not know where the button of a certain tool is located, you can just invoke that tool's function on the menu bar. On the other hand, the tool bar houses all the major functions in Eclipse. The most notable buttons there are the New, Save, and Run.

As of now, look around Eclipse's interface. Also, do not do or change anything on the main activity file or any other file. The next chapter will discuss about how to run your program. As of now, the initial contents of your project are also valid as an android program. Do not change anything since you might produce an unexpected error. Nevertheless, if you really do want to change something, go ahead. You can just create another project for you to keep up with the next chapter.

Chapter 8: Testing Your Android App

By this time, even if you have not done anything yet to your program, you can already run and test it in your Android device or emulator. Why teach this first before the actual programming? Well, unlike typical computer program development, Android app development is a bit bothersome when it comes to testing.

First, the program that you are developing is intended for Android devices. You cannot actually run it normally in your computer without the help of an emulator. And you will actually do a lot of testing. Even with the first lines of code or changes in your program, you will surely want to test it.

Second, the Android emulator works slow. Even with good computers, the emulator that comes with the Android SDK is painstakingly sluggish. Alternatively, you can use BlueStacks. BlueStacks is a free Android emulator that works better than the SDK's emulator. It can even run games with it! However, it is buggy and does not work well (and does not even run sometimes) with every computer.

This chapter will focus on running your program into your Android device. You will need to have a USB data cable and connect your computer and Android. Also, you will need to have the right drivers for your device to work as a testing platform for the programs you will develop. Unfortunately, this is the preferred method for most beginners since running your app on Android emulators can bring a lot more trouble since it is super slow. And that might even discourage you to continue Android app development.

Why Android Emulators are Slow

Why are Android emulators slow? Computers can run virtual OSs without any problems, but why cannot the Android emulator work fine? Running virtual OSs is not something as resource-extensive anymore with today's computer standards. However, with Android, you will actually emulate an OS together with a mobile device. And nowadays, these mobile devices are as powerful as some of the dated computers back then. Regular computers will definitely have a hard time with that kind of payload from an Android emulator.

Luckily, there is a way for developers that have computers using Intel processors to ease up the burden of the Android SDK's emulator. And that is to download the Intel x86 Atom System Image and Intel x86 Atom System Image packages for the Android version they are going to use in the emulator. It can speed up the emulating process, but do expect that it may or may not be as fast as you would want it to be.

USB Debugging Mode

To run your program in an Android device, connect your Android to your computer. After that, set your Android into USB debugging mode. Depending on the version of the Android device you are using, the steps might change.

For 3.2 and older Android devices:

Go to Settings > Applications > Development

For 4.0 and newer Android devices:

Go to Settings > Developer Options

For 4.2 and newer Android devices with hidden Developer Options:

Go to Settings > About Phone. After that, tap the Build Number seven times. Go back to the previous screen. The Developer Options should be visible now.

Android Device Drivers

When USB debugging is enabled, your computer will install the right drivers for the Android device that you have. If your computer does not have the right drivers, you will not be able to run your program on your device. If that happens to you, visit this page: http://developer.android.com/tools/extras/oem-usb.html. It contains instructions on how you can install the right driver for your device and operating system.

Running an App in Your Android Device Using Eclipse

Once your device is already connected and you have the right drivers for it, you can now do a test run of your application. On your Eclipse window, click the Run button on the toolbar or in the menu bar.

If a Run As window appeared, select the Android Application option and click on the OK button. After that, a dialog box will appear. It will provide you with two options: running the program on an Android device or on an AVD (Android Virtual Device) or emulator.

If your computer properly identified your device, it will appear on the list. Click on your device's name and click OK. Eclipse will compile your Android app, install it on your device, and then run it. That is how simple it is.

Take note, there will be times that your device will appear offline on the list. In case that happens, there are two simple fixes that you can do to make it appear online again; 1. restart your device; 2. disable and enable the USB debugging function on your device.

Now, you can start placing widgets on your main activity file. However, always make sure that you do not place any widgets that require higher APIs.

Chapter 9: Starting a Project in Android Studio

This chapter will provide you with a step-by-step guide on how to start creating an app using Android Studio. Technically, it would be the same process with the Android SDK.

1. Open the Android Studio program. You will be greeted with a splash screen. It might take a minute or two before the studio opens up completely. Once it loads completely, a welcome page will appear.
2. In the welcome page, you will be provided with multiple options. To create a new app, double click on the Start a New Android Project option.
3. The New Project dialog box will open. It will require some information regarding you and your project. You will need to give your project a proper name in the Application Name textbox. Next, you will need to provide the company domain. In this case, just place the package name that you want —this was discussed a few chapters ago. Lastly, provide the project location — the directory in your storage device where you will save your project. Then click on the Next button.
4. After that, you will be provided with options regarding what Android devices your program will work on. Take note that you will need to choose the version of Android that your program will work on. Do not forget that lower versions of Android have older versions of libraries. Do note that according to recent Google Play statistics, Android 4.0.3 Ice Cream Sandwich is the most dominant version on the market. Almost 90% of people who download apps from Google Play use Android Ice Cream Sandwich.

   Also, TV, Wear, and Google Glass devices require higher Android versions. Smart TV commonly runs on Android Lollipop. And Wear requires KitKat or Lollipop (Android 4.4 and 5.0). Click on Next after you set the devices.
5. You will be redirected to the activity page. In this part, you can choose an activity template for your app. You can choose from none, blank, blank activity with fragment, full screen, Google AdMob Ads, Google Maps, Google Play Services, Login, Master/Detail Flow, Navigation Drawer, Settings, and Tabbed activity templates.

Of course, choosing any template aside from none will make things faster for you. If you want to explore and create an activity of your own, you can just choose the blank activity — despite being blank, it will come with an action bar. Click on next if you are done choosing. In case you have chosen no activity or Add No Activity, you can just click the Finish button for you to start creating your app in the IDE.

6. On the other hand, if you chose an activity, you will be redirected to a page where you can customize the activity you've chosen. By default, you will need to provide the activity's name, layout name, title, and menu resource name. If you chose an intricate activity, however, the wizard may ask you a few more things. Once you are done filling the form, hit the Finish button.

7. Hitting the Finish button will send you back to the initial screen — the welcome page. To start building your app, open the project you created by clicking Open an existing Android Studio project button. When opening a project, you must open the project folder where you saved your file.

   Once you open a project, Android Studio will open that same project whenever you open the program. In case you want to choose another project or create a new one, you must close the current project first. You can do that by clicking File and then Close Project.

Chapter 10: Fiddling with Android Studio

After creating a project in Android Studio, Gradle will create an app for you. It might take a minute or two depending on the speed of your computer. That app will contain all the default files and settings that a normal app has. The book has already discussed most of those default files and settings previously. And just like before, that new project can be ran on your test device or emulator.

Setting Up Your Test Device

Before you start creating your apps in Android Studio, you will want to edit and customize its settings first. As of this moment, you do not need to change any fancy configurations. You just need to customize the run and debugging options of Android Studio.

To do that, click on Run on the Menu bar. A context menu will appear; click on Edit Configurations. The Run/Debug Configurations window will pop up. Expand the Defaults item on the list menu on the left. Click on the Android Application item.

In that page, you can set up the target device that will be used to run or debug your app. You have three choices. You can either set Android Studio to use the Android device connected to your computer via USB, or use the Android emulator that you have created. On the other hand, you can opt to choose a chooser dialog instead. The chooser dialog will prompt you whenever you will run or debug your app. It will give you the option to run on your Android device or Android emulator.

To be honest, once Gradle generates all the important files in your project's app, you can run it almost immediately. Open an activity in your app and press the run button. If you are connected to your Android device, Android Studio will send, install, and run the app in your device.

On the other hand, if you are going to run your app in the emulator, set your virtual devices first. You can refer to the previous chapters on how to create a virtual Android device.

Updating the SDK

The next step is to update your SDK. While your program is open, click on Tools in the Menu bar. Then click on Android and click on SDK manager. When in the SDK manager window, make sure that the Updates/New checkbox is checked. On the right side, click on the Install <x> packages button.

Once you click on Install, another window will appear. All the packages will be crossed out. You need to accept the licenses that come with them in order to install them. You can do that by clicking on the Accept License radio button. After that, all the items will be checked. Afterwards, click on the Install button. Downloading the new packages might take a long time because they are typically large files.

Exploring Android Studio

Android Studio may overwhelm you at first. Some of its features will be relatively the same with those of Eclipse. But unlike Eclipse, Android Studio is geared towards Android app creation, so you might find some extra features. Those extra features are there to make your app development life easier.

Toolbar

Just below the Menu bar is the toolbar. It hosts shortcuts for file management, editing commands, run and debugging tools, and certain Android tools.

Tool Buttons

On the left, right, and bottom sides of Android Studio, you will see the Tool Buttons bar or Tool Window shortcuts bar. The tool buttons there will allow you to open or close tool windows such as the project, favorites, structure, captures, android, messages, todo, Maven projects, Gradle, and build variants windows.

By the default, the project tool window and messages window will be open beside the tool buttons bar.

Project Tool Window

The project tool window is much like your project's directory explorer. It will provide you with a list or directory of all the files included in your app. In the previous chapter, specifically on the Navigation pane in Eclipse, you were provided with information about certain directories in your Android app.

Since Android Studio comes with third party applications such as Gradle, you might see some unfamiliar directories such as the Gradle directory. Take note that, as of now, you should focus on your app's folder only, which is the folder named app. When tweaking your app's activities, you will be moving around app > src > main > res > layout. That is the directory where your activities' .xml files will be located.

Status Bar

The status bar is located on the bottommost part of Android Studio. It will provide you with various status messages of the program and your project.

Navigation Bar

Just below the toolbar, you will see the navigation bar. It will provide you the current location of the file opened in Android Studio. You can use the navigation bar to move around in your project folder.

Tab Bar

Below the navigation bar is the tab bar. The tab bar will allow you to switch the files you have opened in your project.

Android Studio's Workspace

In the middle of it all, you can find the workspace. Depending on the file type of the file you have opened, the elements in the workspace will change. For example, if you opened a .java file, the workspace will provide a source code editor. Opening an image file will provide you with an image viewer. You cannot perform any image editing in the image viewer. All you can do is inspect the image and use a color picker.

On the other hand, you will be provided with two modes of editing if you open an activity (an .xml from the layout folder). The first mode is Design. The second mode is Text (source code editing). You can switch modes by clicking on the tab buttons on the bottom part of the workspace.

Palette

With Design mode, you can design your app's activities by dragging and dropping from the palette window. The palette window will provide you with view elements. The elements are separated into multiple categories: Layouts, Widgets, Text Fields, Containers, Date and Time, Expert, and Custom.

Component Tree

In the component tree, you will be able to browse the view elements in your activity with ease. The component tree is similar to the outline view, window, or pane in Eclipse. The view elements will be ordered according to their structure level to make them easier to find. By just clicking the name of the view element, that element will be selected on the layout view window and on the properties window.

Context Menu

In Android Studio's context menu, you can perform multiple useful tasks with just a few clicks. By the way, the context menu is the small popup window that appears on the screen whenever you press the right mouse button.

With it, you can easily delete, cut, and copy the view elements that you want. Also, you can change the view element to a different but related view element within a few clicks. For example, if you placed an analog clock display on the activity. You can easily replace it with a digital clock by morphing it.

Another neat thing about the context menu in the workspace of Android Studio is that you can easily go to the physical line in the source code where the element was declared and open it up in the source code editor. Of course, these features are not only useable in the component tree. You can do that by clicking

the Go To Declaration menu item.

Layout View Window

For beginners, most of the application development process will be done on the Layout View window. In this window, a preview of your activity or app will be shown. Aside from seeing how it would look like on a device, you can interact with it by dragging view elements from the palette on it. Every change or addition in the Layout View will affect the source code.

You can zoom in and zoom out in the Layout view. Aside from that, you can create previews of your app in landscape mode (this is particularly useful for tablet devices). On the other hand, you can set a specific device where you will 'emulate' the display of your app. You can change it to an Android smartphone, Android Wear, or Android TV. By default, Studio Android's Layout View will be set to Nexus 4.

Aside from changing the device, you can change the theme of the phone to check if your design will adapt to any themes. You can also check previews of your app in different Android versions and different screen sizes in one go. However, do note that if your computer is not that powerful, it might take a while for the previews to render. Expect that it might take a minute or two, or Android Studio might crash on you.

Properties Window

The properties window of Android Studio is almost the same with the one in Eclipse. Whenever you highlight or select a view element, all its properties will be listed in the properties box. In there, you can manually edit or change the value of the highlighted view element. Take note that all the changes you make in the properties box will update the source code of your app.

Source Code / XML Editor

When you switch to text mode, the source code / XML editor of Android Studio will replace the palette, component tree, and properties windows. You will be left with the Layout view and the source code editor itself.

Even if the Layout View is present in text mode, you cannot drag and drop elements on the preview. In case that you double click any of the elements in the Layout View window, your workspace will revert to Design mode. Also, take note that any changes that you make in the source code will update the preview in the Layout View. If you remove a view element block in the source code, it will disappear from the Layout View window immediately.

Chapter 11: Designing Your First App in Android Studio

Currently, your app does not have anything on it except for the default TextView widget/object that contains the text, “Hello World!” Now, you will be taught how to add some elements on your main activity. But before that, you should know more details about View and Viewgroup objects.

View, as mentioned a while ago, is a typical element in an Android app. Most developers and users call them as widgets. Some of the View elements that you can add on your app are text fields, buttons, sliders, and etcetera.

On the other hand, Viewgroups are containers for View elements. They serve as the parent or root for View elements or as child for other Viewgroup objects. Keep in mind that View elements are stuck with being a child of a Viewgroup object.

Viewgroups house or group View objects together. Viewgroups make it easier to handle and organize the View elements in your activities. They are invisible to the users.

A Viewgroup can contain View objects such as text fields, buttons, and even another Viewgroup object. If you are familiar with HTML, Viewgroups are like div elements. They are invisible, but they do hold other elements within them.

Go and check your project’s component tree. As you can see, under Device Screen, you will see two other items. The first one is RelativeLayout. The second one is TextView. The RelativeLayout is a Viewgroup. And inside that Viewgroup is TextView, which is a View object.

In the Layout View window, you can see the TextView object that says, “Hello World!” On the other hand, you will not see any element other than that. When you click on the TextView item on the Component Tree, the Hello World text will be highlighted. But if you click on RelativeLayout, nothing will happen except for the deselection of any selected View element on the preview screen.

You can add any widget that you want for your app. When you add a view element, make sure that you check the changes on your activity’s xml file. Familiarize yourself with the attributes and the name of the elements when they are placed in an activity.

By the way, whenever you place a widget on the Design view, you might have noticed that there are some green lines and arrows in the widgets. Those are guiding lines. Those lines will help you set the location of your widgets relative

to the other elements on your app’s screen.

On the other hand, there are other layout modes available in Android. Each view has different behaviors and can affect the way Android places the elements that you place in the layout.

Anyway, you can run and test that app. As long as you did not change anything else, the app can be opened on your device or with an Android emulator.

Part II: Familiarity with the Android System

Chapter 1: Basic Program Structure

The Android application structure is quite rigid. In order for the elements to function properly, you have to put the files in their respective places. In this chapter, we will discuss the packages in Java, which are essentially folders where you can store your classes. We will also discuss the structure of an Android project as well as how to design a navigation system.

Java Packages

When you use packages in Java, you are able to write codes that are well-organized. Your related classes become well-structured and their specific purposes are defined. What does this mean exactly? Well, you may think that packages in Java are unfamiliar; but in reality, you have most likely already used them before.

You see, it is nearly impossible to create an application that does not use classes in various packages. For instance, Java outlines a high level package called Java. Inside this package, you can find other packages like lang. This package, in turn, contains the core classes such as util and String, which contain classes such as ArrayList.

Likewise, the Android API offers a high level Android package that holds packages such as graphics, widget, and view. If you want to use the ArrayList class for your code, you should use its entire path, which is java.util.ArrayList. Do not worry though. With Java, you do not need to type so many codes or words.

Java is actually preferred by a lot of programmers because there is not much typing involved. To write a program in Java, you just have to provide the import statement. For instance, you should write import java.util.ArrayList. By writing ArrayList, you are already referring to the class in the java.util package.

Java is also preferred by a lot of programmers because they can use a similar name on multiple classes as long as these classes are placed in dissimilar packages. Keep in mind that it is very important to use different names for different classes if you plan to put them in the same package. Otherwise, an overlap may occur.

When this happens, you can get very confused and some of your data might get lost. Also, you may have a problem if you write a code that makes use of a third-party library. You need to use different names because there is a high chance that these libraries have classes that have the same names.

How Does Android Use Packages

When it comes to Android, packages are used to arrange application codes as well as manage such applications. This operating system requires that each installed application contains a package identifier which is two levels deep or higher. Let us say that you have a package and you want to name it myniceapp. Well, this name is not appropriate and will not be recognized by the system.

So instead, you need to rename it to mycompany.myniceapp. Now, this name is more appropriate and acceptable. The primary package that stores the code of your application is meant to inimitably identify your application on your device. It is also meant to allow applications to communicate and exchange vital information.

Many Java developers actually create package structures with the reverse domain scheme utilized as basis. For instance, if you are an application developer who works at a certain company and you have an official website, you should use the appropriate codes and subpackages. Let us say that your company is called GreatSoftware and your official website is located at greatsoftware.com.

All the codes of your applications should be stored in the subpackages of com.greatsoftware. If you are creating an app called SmartApp, you should store all its code at com.greatsoftware.smartapp. You can also use the subpackages under it so you can break up your code further. You can do this by splitting the model and the view code and putting them in different packages.

The File Structure of an Android Project

Android projects made using Eclipse usually have a pre-made structure with codes and resources organized into numerous folders.

- Src/ - As mentioned, this contains the source code which is created for your applications.
- Gen/ - This contains the source code which the development tools of your application autogenerate. It is not a good idea to modify the source files contained in this folder. If you do, the changes you made will only be written over the ensuing time you create a project.
- Libs/ - This contains third-party libraries that have been pre-compiled. These JAR archives can be used for your application. For instance, if you are creating an application that obtains data from social networking feeds, you should use a social networking library that another programmer has already made for you. You should put this in the libs/ folder.
- Res/ - This contains folders that contain resources for the application. These include icons, GUI layouts, menus, *etc.*
- Assets/ contains media that might interest you for your application. Such media include audio files, video files, and image files that are not directly used in GUI layouts.

Application Resources

As stated earlier, the res folder contains a number of resources that you may need for your application. Such resources are kept in folders inside your res folders, depending on the resource type.

Menu/ - This stores XML files that define menus which are related to the screens in the application. You can make your application show up if you click the Menu button on your device.

Drawable-*dpi/ This contains images that are used in different parts of your application. These images include launch scree icons and images that are attached to the menus or buttons.

Your projects can have numerous drawable folders with names that are based from the resolution of your device. For instance, drawable-ldpi can contain images that are used on devices with low resolution. Drawable-hdpi can contain images that are used on devices with high resolution. And the list goes on. Through this, you can come up with images that are specifically designed for a variety of screens with different sizes. These images will look clear and crisp on these screens. They will not have any blurring or scaling.

Then again, if there was not any exact resolution, your Android operating system will search for the nearest match and then either gage it up or down so it could fit. Because of this, you can simply put all your images in the drawable-hdpi and no longer worry about different sizes.

Layout/ This stores XML files that give a description of the layout of widgets on the screen of your application, such as text fields and buttons. You do not have to directly write an XML though. The Android tools available in Eclipse have a drag and drop editor you can use to lay out screens.

Values/This contains certain values that are used in your application. These values include style definitions and text strings. The strings.xml file can be used to contain text strings that are useful for GUI layouts, such as buttons or labels. It is alright if you do not want to do this. However, if you decided to do so, you will be delighted to know that it can make applications easier and quicker to decode into different languages since all your strings will be conveniently stored in a single location. Hence, you only have to have this sole file translated.

You should take note that there are many other resources that you can also use.

Android Device Navigation

The navigation between applications and the other features on Android mobile phones is usually consistent. It does not matter what the manufacturer or form of the device is. Google has provided a set of controls which you can use to navigate on your device.

The most recent versions of the Android operating system have a requirement that all devices should have Back and Home buttons to support navigation among applications and screens. Older devices that run older Android versions usually have additional buttons, such as Search and Menu.

The most recent operating system menus have even been replaced by thinner toolbars and action bars. Such toolbars are visible all the time and cannot be hidden unless you requested so. On the other hand, backwards compatibility lets you use the menus of the older version in newer applications. It also allows you to upgrade your applications so you can use the new action bars. Actually, your emulator can still provide you with a Menu button in case you wanted to run legacy applications.

Designing Effective Navigation

When it comes to developing and designing an Android application, one of the most important steps that you need to do is to identify how users would use the application. Once you find out what types of data they are most likely to interact with in the application, you have to design the interactions that let users navigate into, back, and across a variety of content within the application.

Chapter 2: Application Navigation

Navigation with Up and Back

As an Android application developer, you need to make sure that the users are able to navigate consistently. Always remember that users tend to be frustrated from certain things, such as inconsistent basic navigation. Android 3.0 has made notable changes in the area of global navigation. Android 2.3 and older versions relied on the Back button for navigation support within the application.

Back vs. Up

The Up button is mainly used to navigate an application based on hierarchical relations between screens. Let us say that screen A shows a list of certain items. When you choose an item, screen B shows up and provides more detailed information about the item that you chose. Screen B also shows an Up button that you can click to go back to screen A. You should take note that if the screen is at the topmost portion of the application, it should not have an Up button.

On the other hand, the Back button is used to navigate an application in reverse order. It is primarily based on temporal relations between the screens, instead of the hierarchy of the application. If your previously viewed screen is a hierarchical parent of your current screen, you can press the Back button to achieve the same result as when you press the Up button. Unlike the Up button, however, the Back button does not ensure that the user stays within the application although it allows him to go back to the Home screen.

The Back button also allows the user to move on to a different application. In addition, it supports certain behaviors that are not directly connected to screen-to-screen navigation. It dismisses popups, dialogs, and other floating windows, as well as contextual action bars. It also removes the highlight of selected items and hides the IME or the onscreen keyboard.

Navigation within the Application

How to Navigate to Screens with Several Entry Points

There are instances in which a screen does not have a strict position within the hierarchy of the application and it can be reached from several entry points. For instance, you can reach a settings screen from a certain screen in your application. If you press the Up button, you should be able to go back to the referring screen. It should work just like the Back button.

How to Switch Views within a Screen

When you change the view options for your screen, the functionalities of your Up and Back buttons will not be affected. Your screen will remain in the same location within the hierarchy of your application and there would not be any new navigation history created.

You can change the view by using the left or right swipes, tabs, and collapsed tabs such as the dropdown. You can also change the view by sorting or filtering a list or changing the display characteristics, such as zooming in and zooming out.

How to Navigate Between Sibling Screens

If your application supports navigation from items so it can provide a more detailed view of these items, it is usually ideal to support direction navigation from such item to another one that either follows or precedes it. For instance, in Gmail, it is easy to swipe to the right or to the left from your conversation to read an older or newer one in your inbox. Likewise, when you change view within a screen, the navigation does not change the functionality of the Back button or the Up button.

Then again, one significant exception occurs when checking out detail views that are not connected by the referring list. For instance, when you browse in the Play Store and check out applications created by the same developer, you are able to add data in your history list when you follow their links. Likewise, you are able to create history when you click on different albums made by the same artist. The Back button is able to go through every previously viewed screen.

The Up button continues to bypass such screens and move onto the most recently viewed screen. Actually, you can improve the functionality of the Up button if you are knowledgeable on detail views. For instance, you can extend the Play Store and visualize its user to navigate from a Book option to a Movie adaption of the Book option. In this case, he can use the Up button so he can go to Movies, which he has not checked out before.

How to Navigate Into Your Application Using Home Screen Widgets and Notifications

If you wish to directly navigate to screens within the hierarchy of your application, you can use Home screen notifications or widgets. For instance, you can use the widget of Gmail Inbox as well as new message notifications so you can view conversations directly. Both of these widgets have the ability to bypass your Inbox screen.

So how should you use the Up button in these cases? Well, if your destination screen can be reached from a certain screen within your application, you should use the Up button to navigate to that screen. Otherwise, you should the Up button to go Home or the topmost portion of your application.

As for the Back button, you can have a more predictable navigation by inserting the upward navigation path into the back stack of the task. This upward navigation path is directed to the topmost portion of the screen of the application. This lets the user remember how he entered your application to navigate to the topmost portion of the screen before he exited. For instance, in Gmail, the home screen widget features a button for directly diving to the compose screen. Either the Up or the Back buttons from this screen can take users to the Inbox. From there, they can use the Back button to go back to Home.

Indirect Notifications

If your application needs to simultaneously show information regarding multiple events, you can program it to use a notification that directs users to the interstitial screen. Such screen summarizes these events as well as provides a path that they can use to dive deeper into the application. The notifications that direct users to such screen are known as indirect notifications.

However, unlike with standard notifications, you will be redirected to the point wherein the notification was activated from when you press the Back button from the interstitial screen of an indirect notification. There would not be any extra screens inserted into your back stack. When the users proceed into the application from their interstitial screen, the Up and Back buttons act like standard notifications.

They navigate within the application instead of going back to the interstitial screen. For instance, in Gmail, users receive indirect notifications from Calendar. When they touch this notification, an interstitial screen will open up. This screen displays important reminders for a variety of events. When they touch the Back button from this screen, they will be directed back to Gmail.

Conversely, if they touch a certain event, they will be taken away from this screen and brought to the full Calendar application in which more details about the event are shown. They can use the Up and the Back buttons to navigate to the topmost portion of the Calendar.

How About Popup Notifications?

Popup notifications have the ability to bypass the notification drawer. They generally show up in front of the user. However, they are not very frequently used. They are also meant to be reserved for events in which timely responses are necessary. They can also be used whenever the context of the user has to be interrupted.

For instance, Talk makes use of this style to alert users of invitations for video chat. Each time a user wants another user to video chat, a Talk notification appears. If after several seconds, the other user does not response, the notification expires automatically. When it comes to navigation, popup notifications generally follow the behaviors of the interstitial screens of indirect notifications.

The Back button gets rid of the popup notification. When the user navigates from his popup into a notifying application, the Up and the Back buttons abide by the rules of the standard notifications for navigating within the application.

Navigating Between Applications

Among the most significant strengths of the Android operating system is the ability of the applications to activate one another. Because of this, the user is able to directly navigate from a certain application to another.

For instance, if you want to take a picture, you can use the Camera application. Through this application, you can capture your desired image and wait for that image to be returned to the referring application. This can be any application, but Gallery is most commonly used to store the captured images.

This is truly great for the application developer and the user. As a developer, you can easily leverage your codes from your other applications. Your users, on the other hand, can enjoy a smooth and hassle-free experience of picture taking.

In order for you to understand application to application navigation, you need to understand the framework behavior of the Android operating system. Let us discuss the activities, intents, and tasks.

In Android, activity refers to a component of application that defines a screen of information. It also includes all the associated actions that the user can make. Your application is a collection of activities that consist of the activities that you create and the activities that you re-use from other applications.

Task refers to the activity sequence that users follow in order to achieve a goal. In fact, a single task can use different activities from a single application. It may also use activities from numerous applications.

Intent refers to the mechanism that an application uses to signal another application that it needs assistance in executing a particular action. The activities of an application indicate which intents they are allowed to respond to. Share and other common intents, for instance, may be easily executed through a variety of applications.

Sharing Between Applications

In order for you to have a better understanding on how tasks, intents, and activities work together, you should consider how a particular application lets users share their content by using another application.

For instance, when a user launches the Play Store application from his Home screen, a new task will be started. Once he navigates through the Play Store application and touches a certain item, he will remain in the same task and extend it by doing more activities.

Let us say that he takes interest in one of the endorsed books in the Play Store, he can hover and touch this option to see more details about it. After that, he has the option to either stay or leave. If he decides to share his activities, he can use the Share button. Once he triggered this action, he will be prompted with a dialog box.

This dialog box contains a list of his activities from a variety of applications, and all of these applications are allowed to handle his Share intent. If he wants to share his actions through Gmail, the compose activity of Gmail will be added to the list as a continuation of his first task. A new task will not be created. If Gmail already runs a task in the background, it will simply ignore the new task.

Chapter 3: Action Bars and Navigation Drawers

ction Bar

The action bar is located at the topmost portion of the screen of the application. Some of its primary functions include making significant actions accessible and prominent in a predictable pattern, such as Search or New; viewing switching within applications and supporting consistent navigation; reducing clutter by executing action overflow for actions that are not frequently used; and providing a special space for giving an application its identity.

If you have just started creating applications for the Android operating system, you should take note that the action bar is among the most significant design elements you can have.

The action bar is generally categorized into four functional sections. Most applications have these four sections: the application icon, view control, action buttons, and action overflow.

The application icon establishes the identity of your application. You can replace it with different brands or logos if you want. Keep in mind that if your application does not display the top level screen, you should place the Up caret to the left portion of your application icon so your users can easily navigate up the hierarchy.

The view control refers to how users can view the data in your application. For instance, if your application shows data in multiple ways, this particular section of your action bar will let your users change views. Popular examples of view switching controls include tab controls and drop down menus.

If your application does not support different views, you can use this section to show non-interactive content, such as long details about branding or the title of the application.

The action buttons display the most crucial actions of your applications. The actions that do not fit in your action bar are automatically transferred to the action overflow. If you want to view the name of your actions, you can press a certain icon for a few seconds.

The action overflow, on the other hand, moves actions that are not frequently used to the action overflow.

Adjusting to Screen Rotation and Different Sizes of Screens

When it comes to creating an application, see to it that you consider how you can adjust to screen rotation and different sizes of screens. You can adapt to these changes by making use of split action bars that will let you distribute their content across a variety of bars located below your screen or primary action bar.

What to Consider for Split Action Bars

If you wish to split up your content across different action bars, you can choose from three different locations. These are the main action bar, the top bar, and the bottom bar. If you want your users to be able to quickly switch between views, you can put a spinner or a tab in the top bar of your application. If you want to display the action overflow and the actions, you should use the bottom bar.

Action Buttons

The action buttons on the action bar of your application displays activities. Hence, you should consider which of your buttons are most often used and then organize them accordingly. The Android operating system usually displays the most important actions as action buttons and moves the others to the action overflow. Your action bar must only display actions that are available to users. If a certain action is not available, you should hide it. You should not let others see that it is disabled.

How to Prioritize Actions

If you are having a hard time prioritizing your actions, you can use the FIT scheme. FIT actually stands for Frequent, Important, and Typical.

An action can be considered Frequent if the users use it at least seven out of ten times whenever they visit the screen. It can also be considered Frequent if such users use it a few times in a row.

An action can be considered Important if the users find this action to be cool or interesting. It can also be labeled as Important if it does not require much effort to be executed.

An action can be considered Typical if it is presented as a first-class action in other similar applications. Likewise, it can be considered Typical if users become surprised when it is buried in the action overflow.

Once you have classified your actions to be Frequent, Important, or Typical, you should place them in your action bar. Else, you should place them in the action overflow.

Action Bar

The action bar is located at the topmost portion of the screen of the application. Some of its primary functions include making significant actions accessible and prominent in a predictable pattern, such as Search or New; viewing switching within applications and supporting consistent navigation; reducing clutter by executing action overflow for actions that are not frequently used; and providing a special space for giving an application its identity.

If you have just started creating applications for the Android operating system, you should take note that the action bar is among the most significant design elements you can have.

The action bar is generally categorized into four functional sections. Most applications have these four sections: the application icon, view control, action buttons, and action overflow.

The application icon establishes the identity of your application. You can replace it with different brands or logos if you want. Keep in mind that if your application does not display the top level screen, you should place the Up caret to the left portion of your application icon so your users can easily navigate up the hierarchy.

The view control refers to how users can view the data in your application. For instance, if your application shows data in multiple ways, this particular section of your action bar will let your users change views. Popular examples of view switching controls include tab controls and drop down menus.

If your application does not support different views, you can use this section to show non-interactive content, such as long details about branding or the title of the application.

The action buttons display the most crucial actions of your applications. The actions that do not fit in your action bar are automatically transferred to the action overflow. If you want to view the name of your actions, you can press a certain icon for a few seconds.

The action overflow, on the other hand, moves actions that are not frequently used to the action overflow.

Adjusting to Screen Rotation and Different Sizes of Screens

When it comes to creating an application, see to it that you consider how you can adjust to screen rotation and different sizes of screens. You can adapt to these changes by making use of split action bars that will let you distribute their content across a variety of bars located below your screen or primary action bar.

What to Consider for Split Action Bars

If you wish to split up your content across different action bars, you can choose from three different locations. These are the main action bar, the top bar, and the bottom bar. If you want your users to be able to quickly switch between views, you can put a spinner or a tab in the top bar of your application. If you want to display the action overflow and the actions, you should use the bottom bar.

Action Buttons

The action buttons on the action bar of your application displays activities. Hence, you should consider which of your buttons are most often used and then organize them accordingly. The Android operating system usually displays the most important actions as action buttons and moves the others to the action overflow. Your action bar must only display actions that are available to users. If a certain action is not available, you should hide it. You should not let others see that it is disabled.

How to Prioritize Actions

If you are having a hard time prioritizing your actions, you can use the FIT scheme. FIT actually stands for Frequent, Important, and Typical.

An action can be considered Frequent if the users use it at least seven out of ten times whenever they visit the screen. It can also be considered Frequent if such users use it a few times in a row.

An action can be considered Important if the users find this action to be cool or interesting. It can also be labeled as Important if it does not require much effort to be executed.

An action can be considered Typical if it is presented as a first-class action in other similar applications. Likewise, it can be considered Typical if users become surprised when it is buried in the action overflow.

Once you have classified your actions to be Frequent, Important, or Typical, you should place them in your action bar. Else, you should place them in the action overflow.

Navigation Drawer

The navigation drawer is basically a panel that shows up at the left portion of the screen to display the primary navigation options of the application. Your user can make the navigation drawer appear either by swiping from the left side of the screen towards the right or pressing the icon of the application on the action bar.

When the navigation drawer expands, the content is showed. When it is extended fully, the action bar adjusts accordingly so that its contents are shown. It does this by replacing the existing action bar title with the name of the application and getting rid of any contextual actions that run in the background. As for the overflow menu for Help and Settings, it stays visible.

The drawer panel can be opened by pressing the navigator drawer indicator. Navigation drawers are transparent, which is why they make views more organized. They can even be used at deeper navigation hierarchy levels. Users can switch to the important screens in their application from anywhere in such application. If they decided that they want to dismiss the navigation drawer, they can simply touch the content outside it.

They can also swipe to the left side of their screen, touch the icon or title of the application in the action bar, or press the Back button. Then again, you should take note that the navigation drawer must not be used as a general replacement for the top level navigation. You should use the structure of your application as a guideline on how to choose patterns for top level switching.

Where Should You Use Navigation Drawers?

Well, you can use navigation drawers if you have at least three unique top level views. A navigation drawer is efficient in concurrently showing large numbers of navigation targets.

You can also use them if your application has to cross navigate from a lower level. The navigation drawer can be accessed from anywhere in your application, which is why it can navigate from lower level screens to a different location in the application.

In addition, you can use navigation drawers if you have deep navigation branches. As you know, navigating to the top level of the application can become cumbersome and repetitive.

Hence, you need to find a way to navigate upwards more efficiently and quickly. Navigation drawers are perfect for this because they can be accessed from any location in the application.

Navigation Hubs

Keep in mind that the navigation drawer reflects the structure of your application as well as shows its primary navigation hubs. Navigation hubs are where users go to most often. They are also frequently used as jumping off points to other locations in the application. The navigation hubs correspond to the major areas of the application which is why they get the top level views.

If your application has a deep structure, you can use low level screens that users are most likely to visit. These screens can also serve as navigation hubs. In order to expedite access to navigation drawers, all your screens that correspond to your navigation drawer entries must display the indicator of your navigation drawer. It should be shown near the icon of the application in your action bar.

You can touch the icon of your application to make your navigation drawer slide in from the left part of your screen. All of the other low level screens should display the Up button near the icon of your application. You should still be able to access the navigation drawer when you swipe your finger across the screen, although it should not show up in your action bar.

Navigation Drawer Content

It is important for you to keep the contents of your navigation drawer focused on your application navigation. You should expose your navigation hubs as list items. There should be one item for every row.

You can organize your navigation targets by including titles. There is no need for you to make them interactive. You simply have to organize your navigation targets into topics that are functional. In case you have a lot of navigation targets, you can use titles to guide your users within the navigation drawer.

Your navigation targets may have leading icons and training counters. These elements are optional, however. So it is alright if you do not use them. Nonetheless, you should use your navigation counters to tell users whenever a state of data has been changed in a certain view.

What if you have a lot of views with subordinates? If this is the case, you can collapse them into a single expandable item. This will allow you to save space. Your navigation drawer parent will turn into a split item and its left side will allow users to navigate to the view of the parent item. The right side, on the other hand, will expand or collapse the list of its child items.

It is up to your discretion how you want the initial state of your collapsible items to be. Then again, there is a general rule that the top level view entries of navigation drawers have to be visible. So if you have a lot of collapsible items, you may want to collapse every item to let your users see the entire top level views.

When your users expand your navigation drawers, the task focus switches to choosing an item from your navigation drawer. Since the navigation drawers do not overlay your action bar, your users may not immediately notice that the items in your action bar do not necessarily affect the navigation drawers.

In order to lessen your confusion, you have to adjust the contents of your action bar, such as the icon and the name of your application. You should also remove any action from your action bar that is contextual to your underlying view. If you wish to retain actions using global scope, you may do so.

What’s more, you should adjust your overflow menu with your navigation targets like Help and Settings. It is not advisable to place your actions in your navigation drawer. Remember that actions are meant to be placed in the action bar. Your users expect to see such actions there.

Not every application uses navigation drawers though. You may feel tempted to

expose all the capabilities of your application in just one location. However, you should be careful with what you do. You should only put your actions in a location where all applications show them. The same thing applies to navigation targets, including Help and Settings.

There are times wherein users see a contextual action bar or CAB in place of the action bar of the application. This usually occurs when the users choose multiple items or texts after a press and hold gesture. Even though the contextual action bar is visible, make sure that you still let the users open the navigation drawers by edge swiping.

You should replace the contextual action bar with a regular action bar while your navigation drawer stays open. If your users dismiss the navigation drawer, then now is the time for you to display your contextual action bar again.

How Can You Introduce Your Users to the Navigation Drawer?

Once you launch your application, see to it that you introduce your users to the navigation drawers by opening them immediately. When they see these navigation drawers, they will be prompted to explore the contents of your application and know more about its structure.

You should continue launching your application with your navigation drawers open until you are confident that your users completely understand how to use it. Then, you can launch your application with your navigation drawers closed.

Moreover, see to it that you let your users take a quick look at your navigation drawer each time their fingers make contact with the screen. This will allow them to discover the navigation drawer as well as encourage them to provide feedback.

If you open your navigation drawer from a screen represented inside it, you should highlight such entry inside the navigation drawer. Likewise, if you open your navigation drawer from a screen not listed in it, you should not highlight any of the items inside the navigation drawer.

How Does a Navigation Drawer Impact the Overall Navigation of an Application?

You should take note that the navigation drawer is practically a substitute to other top level navigation patterns. If you want your applications with navigation drawers to consistently work with applications that use spinner patterns or tabs, you should consider several important factors.

For instance, keep in mind that touching the System Back at the top portion of your application does not open the navigation drawer. The System Back only functions according to the rules of navigation for the top level, such as navigating towards the Home screen or towards a previous application within the task.

You should also remember that when your users navigate to a lower hierarchy screen from your navigation drawer with such screen having a direct parent, the Back stack option resets and the Back option points to the parent of the target screen. The same Back behavior can be observed when users navigate into applications from a notification.

Finally, you should take note that the width of your navigation drawer essentially depends on the contents that you want to show. However, such contents should only be between 240 dp and 320 dp. You should not use line items that have a height falling below 48 dp. See to it that you also choose a navigation drawer background that best matches the theme of your application.

Chapter 4: Applications with User Information and Location

User Information

The Contacts Provider is basically the central repository of the contact information of the user. This includes data from social networking and contacts applications. In such applications, you can find Contacts Provider either by sending intents to a contacts application or calling ContentResolver.

How to Retrieve a Contact List

If you want to create an application that displays the information and location of the user, you should know how to retrieve lists of contacts with data that partly or completely match a search string. You can use a variety of techniques for this.

When matching contact names, you should get a list of contacts. You can do this by matching the search string with all or part of your contact name data. In order to return a list of matches, you should use the Contacts Provider. It allows for multiple instances of a similar name.

When matching a particular data type, such as phone numbers, you should get a list of contacts as well. You can do this by matching the search string with a certain type of data, such as e-mail addresses. Through this technique, you can list every one of your contacts who have an e-mail address that matches your search string.

When matching any type of data, you should once again get a list of contacts. You can do this matching your search string to any detail data, such as names, phone numbers, street addresses, and e-mail addresses among others. This technique will enable you to accept all types of data for a search string and list all the contacts that match it.

How to Retrieve Contact Details

It is not unusual for Android users to want to retrieve contact details, including phone numbers and e-mail addresses. As an application developer, you can either provide them with all the contact details or just show the details of a certain type of data, such as e-mail address.

Anyway, if you want to retrieve all of the contact details, you can search ContactsContract.Data for rows that contain the LOOKUP KEY of the contact. You can find this column in ContactsContract.Data. You should take note that retrieving all of the contact details tends to reduce the performance of devices.

This is because doing so requires the retrieval of all the columns in ContactsContrast.Data. So before you do this procedure, see to it that you consider the performance of your Android device.

On the other hand, if you are only going to retrieve certain contact data types, you should do the same process as with retrieving all of the contact details. However, you need to make a few changes with the projection, selection, and sort order.

Using Intents to Modify Contacts

Intents are very useful when it comes to modifying contact data even though it does not directly access the Contacts Provider. It starts the contacts application, which runs the necessary Activity.

You can use an Intent to upgrade or insert a contact. This will allow you to save time and energy in developing a code and a user interface. It will also let you avoid errors that have been caused by certain modifications that do not abide by the rules of the Contacts Providers.

Furthermore, it will allow you to reduce the amount of permissions that you need to make requests. Your applications will no longer need any permission to contact the Contacts Provider.

Inserting a New Contact with an Intent

As an application developer, you may want to let your users insert new contacts when the application receives new data. Let us say that your Android application is designed to review restaurants. You may want your users to add restaurants as their contacts while they review them. This is such a great idea since they can contact the restaurant easily in case they want to go back.

So what should you do to make this happen? Well, you need to build the intent by using as much data as you can. Afterwards, you should send your intent to the contacts application. When you use the contacts application to insert a contact, you are also able to insert a new raw contact into the ContactsContract.RawContacts table of your Contacts Provider.

Editing Existing Contacts with an Intent

An intent is also useful in allowing users to re-write or edit their existing data. Let us say that your application can store contacts with postal addresses. If your users want to add their postal codes, you should give them the option to search for the appropriate code and add it to their data.

In order to edit existing contact data using the intent, you should the same technique you used when inserting contact data. You should build the intent but do not forget to add the Contacts.CONTENT_LOOKUP_URI of the contact as well as its MIME type. If you wish to edit existing contact details, you can place them in the extended data of the intent. Take note that some of the name columns cannot be edited with an Intent.

Finally, you should send the intent. After this, you can expect the contacts application to display an edit screen. Once your users finish editing and saving their new contact data, the contacts application will display a contact list. Your application will be displayed when your users taps on the Back button.

Adding a Quick Contact Badge to the User Interface

QuickContactBadge is a widget that appears in the form of a thumbnail. Even though it is alright for you to use any Bitmap for your thumbnails, it is more ideal to use a Bitmap that has been decoded from the photo thumbnail of the contact. This tiny image will serve as a control whenever a user clicks on it.

Anyway, QuickContactBadge tends to expand into dialogs that contain large images and icons of applications. The large images are usually related to the contact. If there is no available image, you can use a placeholder graphic.

As for the icons of applications, they are usually handled by built-in applications. For instance, if one of a user’s contacts has more than one e-mail address, an e-mail icon will show up. Once he clicks on this icon, all the e-mail addresses of his particular contact will appear. This will allow him to choose which e-mail address to contact.

User Location

Specific longitude and latitude coordinates can show the location of a user. As an application developer, it is crucial for your application to show the geographic coordinates of the location of users so they can know what street they are on or what landmark is nearby.

Location Awareness

Mobile applications typically feature the location of the user. As you know, users often take their mobile phones with them wherever they go. Hence, adding location awareness to your application will provide them with a more contextual experience. However, if you are still using Android framework location APIs, you may want to consider switching to Google Play services location APIs.

You can use Google Play services location APIs to request for the last known location of a particular device. Usually, the current location of the user is registered as the last known location of his device. Nonetheless, you can also use the fused location provider to obtain its last known location. It is among the location APIs of Google Play. It manages location technologies, provides APIs, and optimizes batter power usage.

Conclusion

Thank you again for downloading this book!

I hope this book was able to help you get started with Android Programming in a Day!

The next step is to study the following:

Relative, Linear, and Table Layout: When it comes to designing your app, you need to know the different types of layouts. In later versions of Android, you can use other versions of layouts, but of course, the API requirements will go up if you use them. Master these, and you will be able to design faster and cleaner.

Adding Activities or Interface: Of course, you would not want your program to contain one page only. You need more. You must let your app customers to see more content and functions. In order to do that, you will need to learn adding activities to your program. This is the part when developing your Android app will be tricky. You will not be able to rely completely on the drag and drop function and graphical layout view of Eclipse. You will need to start typing some code into your program.

Adding the Action Bar: The action bar is one of the most useful elements in Android apps. It provides the best location for the most used functions in your program. And it also aid your users when switching views, tabs, or drop down list. Chapter 7 discusses more about action bars.

Learning More about Programming: The programming course in this book is not enough to let you make ‘good’ programs. You should learn more about flow control statements, iteration statements, and basic creation of methods in your app. Surely, you will gain more power in creating your apps once you have a better grasp in programming in Android by writing.

Adding Event Listeners: Event listeners are there to detect if the user of your app interacted to any of the element in your app’s screen. Detecting those ‘events’ can allow your app to react or do something in response. For example, an event that you can listen into is onClick. When your user clicks something (or clicks on a button), you can assign your program to do something like pushing a popup message on the screen.

Of course, adding event listeners will require you to program or write in your app’s java file. Due to that, it is essential that you familiarize yourself with programming with Java or with the keywords and intricacies of Android apps.

Also, you should also start researching on how to import and use libraries and classes in your app. Those libraries and classes will give you access to more functions, bringing you greater freedom in your app development pursuit.

Once you have gain knowledge on those things, you will be able to launch a decent app on the market. The last thing you might want to do is to learn how to make your program support other Android devices.

You must know very well that Android devices come in all shapes and form. An Android device can be a tablet, a smartphone, or even a television. Also, they come with different screen sizes. You cannot just expect that all your customers will be using a 4-inch display smartphone. Also, you should think about the versions of Android they are using. Lastly, you must also add language options to your programs. Even though English is fine, some users will appreciate if your program caters to the primary language that they use.

And that is about it for this book. Make sure you do not stop learning Android app development.

Finally, if you enjoyed this book, please take the time to share your thoughts and post a review on Amazon. We do our best to reach out to readers and provide the best value we can. Your positive review will help us achieve that. It'd be greatly appreciated!

Thank you and good luck!

Click here to leave a review on amazon.com

Preview Of ‘C Programming Success in a Day: Beginners’ Guide To Fast, Easy And Efficient Learning Of C Programming’

Chapter 1: Hello World – the Basics

When coding a C program, you must start your code with the function ‘main’. By the way, a function is a collection of action that aims to achieve one or more goals. For example, a vegetable peeler has one function, which is to remove a skin of a vegetable. The peeler is composed of parts (such as the blade and handle) that will aid you to perform its function. A C function is also composed of such components and they are the lines of codes within it.

Also, take note that in order to make your coding life easier, you will need to include some prebuilt headers or functions from your compiler.

To give you an idea on what C code looks like, check the sample below:

```
#include <stdio.h>
int main()
                                        {
    printf( "Hello World!\n" );
    getchar();
    return 0;
                                        }
```

As you can see in the first line, the code used the #include directive to include the stdio.h in the program. In this case, the stdio.h will provide you with access to functions such as printf and getchar.

Main Declaration

After that, the second line contains int main(). This line tells the compiler that there exist a function named main. The int in the line indicates that the function main will return an integer or number.

Curly Braces

The next line contains a curly brace. In C programming, curly braces indicate the start and end of a code block or a function. A code block is a series of codes joined together in a series. When a function is called by the program, all the line of codes inside it will be executed.

Printf()

The printf function, which follows the opening curly brace is the first line of code in your main function or code block. Like the function main, the printf also have a code block within it, which is already created and included since you included <stdio.h> in your program. The function of printf is to print text into your program's display window.

Beside printf is the value or text that you want to print. It should be enclosed in parentheses to abide standard practice. The value that the code want to print is Hello World!. To make sure that printf to recognize that you want to print a string and display the text properly, it should be enclosed inside double quotation marks.

By the way, in programming, a single character is called a character while a sequence of characters is called a string.

Check Out My Other Books

Below you'll find some of my other popular books that are popular on Amazon and Kindle as well. Simply click on the links below to check them out. Alternatively, you can visit my author page on Amazon to see other work done by me.

Click here to check out the rest of C Programming Success in a Day on Amazon.

If the links do not work, for whatever reason, you can simply search for these titles on the Amazon website to find them.