ANDROID

====

B.C.A– SEMESTER-6

Ch-1

Compiled by: Gaurav Sardhara

3	**Android User Interface Design**	• Introducing Android Views and Layouts • TextView • EditText • Spinner Controls • Button • Check Boxes • Toggle Buttons • Using RadioGroups and RadioButtons • Date and Times from users • ProgressBar • Time Passage with the Chronometer • Menu • Dialogs • Tracing the Lifecycle of a Dialog • Defining a Dialog • Initializing a Dialog • Launching a Dialog • Dismissing a Dialog • Removing a Dialog from Use • Layout Classes • Frame layout • LinearLayout • RelativeLayout • TableLayout • Data-Driven Containers ListView GridView GalleryView ArrayAdapter	45 to 71
4	**Database Connectivity Using SQLite**	• Working with Application Preferences • Creating Private and Shared Preferences • Searching and Reading Preferences • Adding, Updating, and Deleting Preferences • Storing Structured Data Using SQLite Databases • Creating a SQLite Database Instance Using the Application Context • Configuring the SQLite Database Properties • Creating Tables and Other SQLite Schema Objects • Creating and updating database with SQLiteOpenHelper • SQLiteDatabase • Inserting Records • Updating Records • Deleting Records • Working with Transactions	72 to 96

		• Working with Cursors • Executing Simple Queries • Deleting Tables and Other SQLite Objects • Keeping Track of Database Field Names • Extending the SQLiteOpenHelper Class • Content provider with example • Writing a content provider • Defining URI • Content URIs • Sample application	
5	**Networking API**	• Overview of HTTP access on Android • Good practices for network access under Android • Java and HTTP access • Check the network availability • Proxy • Parsing XML Data • The Android Telephony package • Web View	97 to 106
6	**Notifications**	• What is Notification? • Notification Manager • Notifying wih the status bar • Vibrating the phone • Blinking the lights	107 to 110
7	**Services**	• Creating a Service • Services takes Two Forms • Service Overview • JSON • JSON – Elements • JSON – Parsing • Adavntages of JSON over XML • json object • json array	111 to 119
8	**Deployment of applications**	• Get Started with Publishing • *Set Up a Google Wallet Merchant Account* • Explore the Developer Console • Preparing for Release	120 To 123
9	**Introduction To iPhone**	• Introduction To X-Code (IDE) • iPhone SDK Core Components • Inside Xcode, Cocoa, and Objective-C • The iPhone Simulator • Layers of the iOS Architecture • The Core OS Layer • Apple XCode for C and C++ • Introduction	124

		• Xcode • Creating an Application • The main() Function • Build Your First iPhone App • Familiarize with Xcode Workspace • Cocoa Touch in relation to other iOS layers • Cocoa Touch FrameWorks • The Power of Objective-C • *Built on Objective-C*	to 145

UNIT – I

Introduction to Android

Importance of Mobile

- Commonplace problem solved with a one-button speed dial or a sample text message like “WRU?”
- Our mobile keep us safe and connected. Now We roam around freely and connected to friends, family.

Handset Manufacturers

- Before Android, mobile developers faced many roadblocks when it came to writing application. Building the better application, the unique application, the competing application, the hybrid application and incorporating many common tasks such as messaging and calling in a familiar way were often unrealistic goals.

First Generation Mobile

- These early phones were flawed, but they did something important – They changed the way people thought about communication. As mobile phone prices dropped, batteries improved, and reception areas grew, more and more people began carrying these handy devices.

WAP(Wireless Application Protocol)

- Commercializing WAP application was difficult and there was no built-in billing mechanism. Some of the most popular commercial WAP application that emerged during this time were simple wallpaper and ringtone catalogues that enabled users to personalize their phones for the first time. For example, a user browsed a WAP site and requeste a specific item. He filled out a simple order form with his phone number and his handset model. It was up to the content provider to deliver an image or audio file compatible with the given phone. Payment and verification were handled through various premium priced delivery mechanisms such as Short Message Service(SMS), Enhanced Messaging Service(EMS), Multimedia Messaging Service(MMS).

Different Platforms

- A variety of different proprietary platforms emerged—and developers are still actively creating applications for them.
    - Some smartphone devices ran Palm OS (now Garnet OS) and RIM BlackBerry OS.
    - Sun Microsystems took its popular Java platform and J2ME emerged (now known as Java Micro Edition [Java ME]).
    - Chipset maker Qualcomm developed and licensed its Binary Runtime Environment for Wireless (BREW).

- **Other plat- forms, such as Symbian OS, were developed by handset manufacturers such as Nokia,Sony Ericsson, Motorola, and Samsung.**
- **The Apple iPhone OS (OS X iPhone) joined the ranks in 2008.**

Forming the Open Handset Alliance

- The Open Handset Alliance(OHA) was formed in November 2007, to answer that very question. The OHA is a business alliance comprised of many of the largest and most successful mobile companies on the planet. Its members include chip makers, handset manufacturers, software developers, and service providers.The entire mobile supply chain is well represented.

Andy Rubin has been credited as the father of the Android platform. His company, Android Inc.,was **acquired(Purchased) by Google in 2005**. Working together, OHA members, including Google, began developing a non-proprietary open standard platform based upon technology developed at Android Inc. that would aim to alleviate(improve) the aforementioned problems hindering the mobile community.

The result is the Android project.To this day, most Android platform development is completed by Rubin’s team at Google, where he acts as VP of Engineering and manages the Android platform roadmap

Google hosts the Android open source project and provides online Android documentation, tools, forums, and the Software Development Kit (SDK) for developers. All major Android news originates at Google.

Android Platform Differences

Android is hailed as “the first complete, open, and free mobile platform”:

- ✓ **Complete:** The designers took a comprehensive approach when they developed the Android platform.They began with a secure operating system and built a robust software framework on top that allows for rich application development opportunities.
- ✓ **Open:** The Android platform is provided through open source licensing. Developers have unprecedented access to the handset features when developing applications.
- ✓ **Free:** Android applications are free to develop.There are no licensing or royalty fees to develop on the platform. No required membership fees. No required testing fees. No required signing or certification fees.Android applications can be distributed and commercialized in a variety of ways.

Latest version of Android

Figure 1.5 The Android mascot and logo.

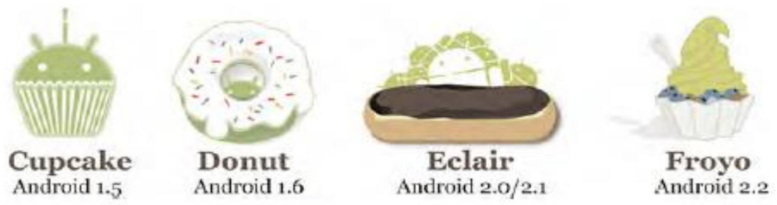


Figure 1.6 Some Android SDKs and their codenames.

Free and Open Source

Android is an open source platform. Neither developers nor handset manufacturers pay royalties or license fees to develop for the platform.

Familiar and Inexpensive Development Tools

Unlike some proprietary platforms that require developer registration fees, vetting, and expensive compilers, there are no upfront costs to developing Android applications.

Freely Available Software Development Kit

The Android SDK and tools are freely available. Developers can download the Android SDK from the Android website after agreeing to the terms of the Android Software Development Kit License Agreement.

Familiar Language, Familiar Development Environments

Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications. Android applications can be developed on the following operating systems:

- ✓ Windows XP (32-bit) or Vista (32-bit or 64-bit)
- ✓ Mac OS X 10.5.8 or later (x86 only)
- ✓ Linux (tested on Linux Ubuntu 8.04 LTS, Hardy Heron)

Reasonable Learning Curve for Developers

Android applications are written in a well-respected programming language: **Java**.

The Android application framework includes traditional programming constructs, such as threads and processes and specially designed data structures to encapsulate objects commonly used in mobile applications.

A "Free Market" for Applications

Android developers can distribute their applications to users in a variety of ways:

- ✓ Google developed the Android Market (see Figure 1.7), a generic Android application store with a revenue-sharing model.

- ✓ Handango.com added Android applications to its existing catalogue using their billing models and revenue-sharing model.
- ✓ Developers can come up with their own delivery and payment mechanisms.

The Android Platform

- ✓ Android is an operating system and a software platform upon which applications are developed.
- ✓ A core set of applications for everyday tasks, such as web browsing and email, are included on Android handsets.
- ✓ As a product of the OHA's vision for a robust and open source development environment for wireless, Android is an emerging mobile development platform.
- ✓ The platform was designed for the sole purpose of encouraging a free and open market that all mobile applications phone users might want to have and software developers might want to develope.

Android's Underlying Architecture

- ✓ The Android platform is designed to be more fault-tolerant than many of its predecessors.

- ✓ The handset runs a Linux operating system upon which Android applications are executed in a secure fashion. Each Android application runs in its own virtual machine

The Linux Operating System

The Linux 2.6 kernel handles core system services and acts as a hardware abstraction layer (HAL) between the physical hardware of the handset and the Android software stack.

Some of the core functions the kernel handles include

- ✓ Enforcement of application permissions and security
- ✓ Low-level memory management
- ✓ Process management and threading
- ✓ The network stack
- ✓ Display, keypad input, camera,Wi-Fi, Flash memory, audio, and binder (IPC) driver access

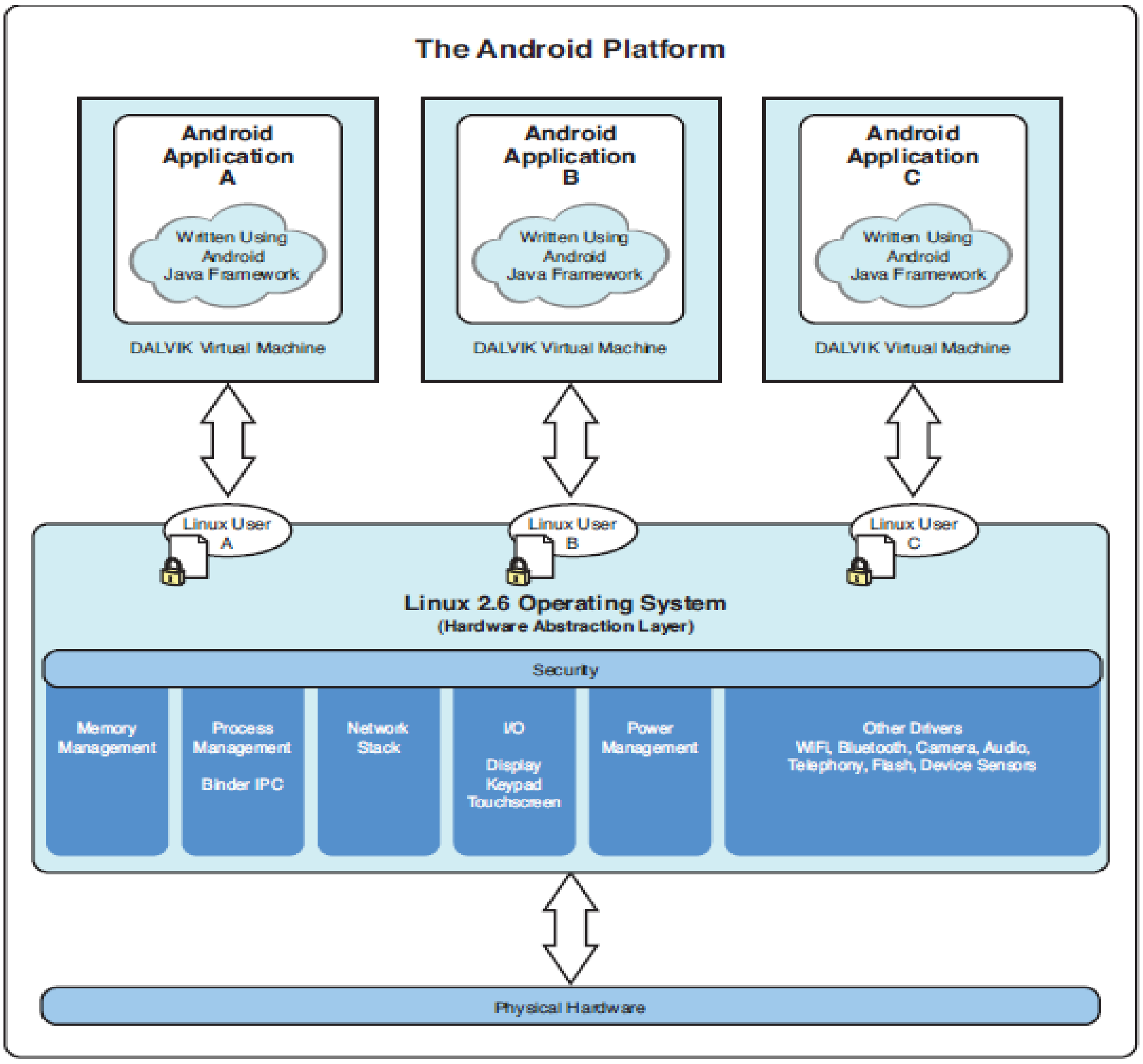


Diagram of Android platform architecture

Android Application Runtime Environment

- ✓ Each Android application runs in a separate process, with its own instance of the Dalvik virtual machine (VM). Based on the Java VM, the Dalvik design has been optimized for mobile devices.

- ✓ The Dalvik VM has a small memory footprint, and multiple instances of the Dalvik VM can run concurrently on the handset.

Security and Permissions

- ✓ The integrity of the Android platform is maintained through a variety of security measures.
- ✓ These measures help ensure that the user's data is secure

Applications as Operating System Users

- ✓ When an application is installed, the operating system creates a new user profile associated with the application. Each application runs as a different user, with its own private files on the file system, a user ID, and a secure operating environment.

Explicitly Defined Application Permissions

- ✓ Applications might also enforce their own permissions by declaring them for other applications to use.The application can declare any number of different permission types, such as read-only or read-write permissions

Application Signing for Trust Relationships

- ✓ All Android applications packages are signed with a certificate, so users know that the application is authentic.
- ✓ The private key for the certificate is held by the developer.

Marketplace Developer Registration

- ✓ To publish applications on the popular Android Market, developers must create a developer account.The Android Market is managed closely and no malware is tolerated.

Developing Android Applications

- ✓ The Android SDK provides an extensive set of application programming interfaces (APIs) that is both modern and robust.Android handset core system services are exposed and accessible to all applications.

Android Programming Language Choices

- ✓ Android applications are written in Java (see Figure 1.9). For now, the Java language is the developer's only choice on the Android platform.
- ✓ There has been some speculation that other programming languages, such as C++, might be added in future versions of Android.

Commonly Used Packages

- ✓ Developers use familiar class libraries exposed through Android's Java packages to perform common tasks such as graphics, database access, network access, secure communications, and utilities (such as XML parsing).

- ✓ The Android packages include support for
    - ❖ Common user interface widgets (Buttons, Spin Controls,Text Input)
    - ❖ User interface layout
    - ❖ Secure networking and web browsing features (SSL,WebKit)
    - ❖ Structured storage and relational databases (SQLite)
    - ❖ Powerful 2D and 3D graphics (including SGL and OpenGL ES)
    - ❖ Audio and visual media formats (MPEG4, MP3, Still Images)
    - ❖ Access to optional hardware such as location-based services (LBS),Wi-Fi, Bluetooth, and hardware sensors

Android Application Framework

- ✓ The Android application framework provides everything necessary to implement your average application.
- ✓ The Android application lifecycle involves the following key components:
    - ▪ Activities are functions the application performs.
    - ▪ Groups of views define the application's layout.
    - ▪ Intents inform the system about an application's plans.
    - ▪ Services allow for background processing without user interaction.
    - ▪ Notifications alert the user when something interesting happens.

- ✓ Android applications can interact with the operating system and underlying hardware using a collection of managers.

What is Android?

1.1. The Android operating system

Android is an operating system based on the Linux kernel. The project responsible for developing the Android system is called the *Android Open Source Project* (AOSP) and is primarily lead by Google.

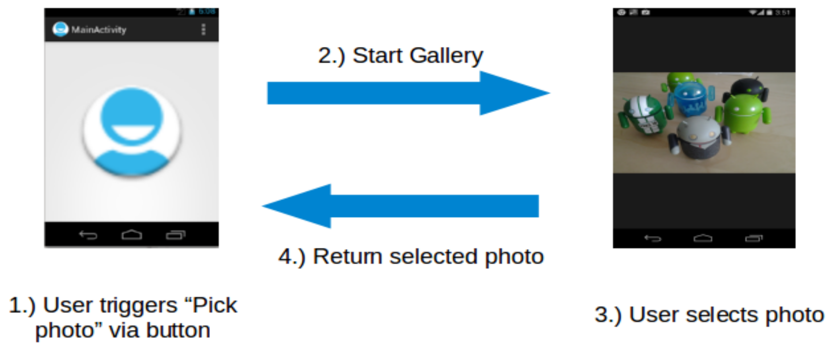
The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL-ES (short OpenGL) standard and grants access to the file system as well as an embedded SQLite database.

An Android application typically consists of different visual and non visual components and can reuse components of other applications.

1.2. Task

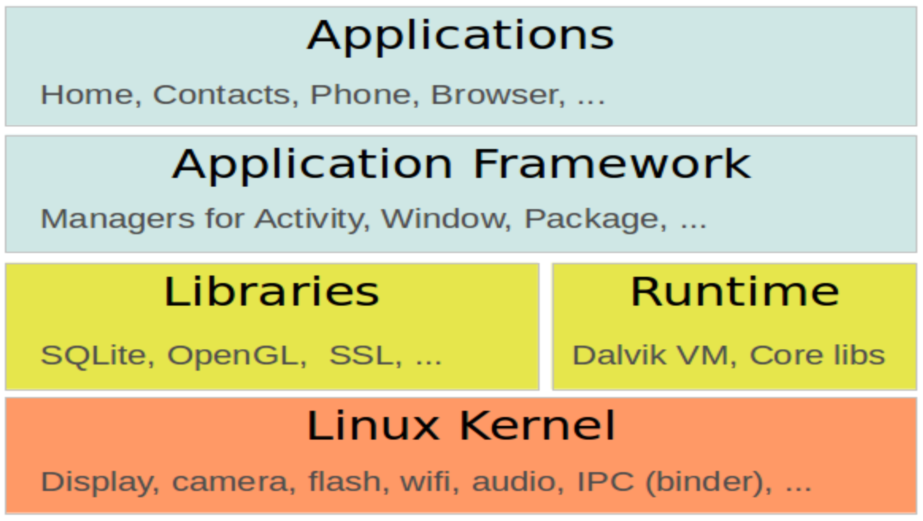
In Android the reuse of other application components is a concept known as *task*. An application can access other Android components to achieve a task. For example, from a component of your application you can trigger another component in the Android system, which manages photos, even if this component is not part of your application. In this component you select a photo and return to your application to use the selected photo.

Such a flow of events is depicted in the following graphic.


MainActivity
2.) Start Gallery
4.) Return selected photo
1.) User triggers "Pick photo" via button
3.) User selects photo

1.3. Android platform components

The Android system is a full software stack, which is typically divided into the four areas as depicted in the following graphic


Applications
Home, Contacts, Phone, Browser, ...
Application Framework
Managers for Activity, Window, Package, ...
Libraries
SQLite, OpenGL, SSL, ...
Runtime
Dalvik VM, Core libs
Linux Kernel
Display, camera, flash, wifi, audio, IPC (binder), ...

The levels can be described as:

- Applications - The Android Open Source Project contains several default application, like the Browser, Camera, Gallery, Music, Phone and more.
- Application framework - An API which allows high-level interactions with the Android system from Android applications.
- Libraries and runtime - The libraries for many common functions (e.g.: graphic rendering, data storage, web browsing, etc.) of the Application Framework and the Dalvik runtime, as well as the core Java libraries for running Android applications.
- Linux kernel - Communication layer for the underlying hardware.

The Linux kernel, the libraries and the runtime are encapsulated by the application framework. The Android application developer typically works with the two layers on top to create new Android applications.

1.4. Google Play

Google offers the Google Play service, a marketplace in which programmers can offer their Android applications to Android users. Customers use the *Google Play* application which allows them to buy and install applications from the Google Play service.

Google Play also offers an update service. If a programmer uploads a new version of his application to Google Play, this service notifies existing users that an update is available and allows them to install the update.

Google Play provides access to services and libraries for Android application programmers, too. For example, it provides a service to use and display Google Maps and another to synchronize the application state between different Android installations. Providing these services via Google Play has the advantage that they are available for older Android releases and can be updated by Google without the need for an update of the Android release on the phone.

2. Android Development Tools

2.1. Android SDK

The Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based. The primary way to develop Android applications is based on the Java programming language.

2.2. Android debug bridge (adb)

The Android SDK contains the Android debug bridge(adb), which is a tool that allows you to connect to a virtual or real Android device, for the purpose of managing the device or debugging your application.

2.3. Android Developer Tools and Android Studio

Google provides two integrated development environments (IDEs) to develop new applications.

The *Android Developer Tools* (ADT) are based on the Eclipse IDE. ADT is a set of components (plug-ins), which extend the Eclipse IDE with Android development capabilities.

Google also supports an IDE called *Android Studio* for creating Android applications. This IDE is based on the IntelliJ IDE.

Both IDEs contain all required functionality to create, compile, debug and deploy Android applications. They also allow the developer to create and start virtual Android devices for testing

2.4. Dalvik Virtual Machine

Currently Android versions use the Dalvik virtual machine. The latest Android versions introduced a new runtime the Android RunTime.

2.5. Android RunTime (ART)

With Android 4.4, Google introduced the *Android RunTime* (ART) as optional runtime for Android 4.4. It is uses as default runtime for all Android versions after 4.4.

ART uses Ahead Of Time compilation. During the deployment process of an application on an Android device, the application code is translated into machine code. This results in approx. 30% larger compile code, but allows faster execution from the beginning of the application.

This also saves battery life, as the compilation is only done once, during the first start of the application.

The dex2oat tool takes the .dex file created by the Android tool change and compiles that into an Executable and Linkable Format (ELF file). This file contains the dex code, compiled native code and meta-data. Keeping the .dex code allows that existing tools still work.

The garbage collection in ART has been optimized to reduce times in which the application freezes.

2.6. How to develop Android applications

Android applications are primarily written in the Java programming language.

During development the developer creates the Android specific configuration files and writes the application logic in the Java programming language.

The ADT or the Android Studio tools convert these application files, transparently to the user, into an Android application. When developers trigger the deployment in their IDE, the whole Android application is compiled, packaged, deployed and started.

2.7. Conversion process from source code to Android application

The Java source files are converted to Java class files by the Java compiler.

The Android SDK contains a tool called dx which converts Java class files into a .dex (Dalvik Executable) file. All class files of the application are placed in this .dex file. During this conversion process redundant information in the class files are optimized in the .dex file.

For example, if the same String is found in different class files, the .dex file contains only one reference of this String.

These .dex files are therefore much smaller in size than the corresponding class files.

The .dex file and the resources of an Android project, e.g., the images and XML files, are packed into an .apk (Android Package) file. The program *aapt* (Android Asset Packaging Tool) performs this step.

The resulting .apk file contains all necessary data to run the Android application and can be deployed to an Android device via the adb tool.

3. Security and permissions

3.1. Security concept in Android

The Android system installs every Android application with a unique user and group ID. Each application file is private to this generated user, e.g., other applications cannot access these files. In addition each Android application is started in its own process.

Therefore, by means of the underlying Linux kernel, every Android application is isolated from other running applications.

If data should be shared, the application must do this explicitly via an Android component which handles the sharing of the data, e.g., via a service or a content provider.

3.2. Permission concept in Android

Android contains a permission system and predefines permissions for certain tasks. Every application can request required permissions and also define new permissions. For example, an application may declare that it requires access to the Internet.

Permissions have different levels. Some permissions are automatically granted by the Android system, some are automatically rejected. In most cases the requested permissions are presented to the user before installing the application. The user needs to decide if these permissions shall be given to the application.

If the user denies a required permission, the related application cannot be installed. The check of the permission is only performed during installation, permissions cannot be denied or granted after the installation.

An Android application declares the required permissions in its AndroidManifest.xml configuration file. It can also define additional permissions which it can use to restrict access to certain components.

Prog-1

Bulding a Simple HelloWorld Application

1) New - >Project ->Android -> Android Project -> Next
2) Project Name = HelloWorld

   Package Name = com.androidbook.HelloWorld
   Target Name = Android 1.6
3) HelloWorldActivity

   System.out.println("Hello World!!!");

AndroidManifest.xml	Global application description file. It defines your application's capabilities and permissions and how it runs.
default.properties	Automatically created project file. It defines your application's build target and

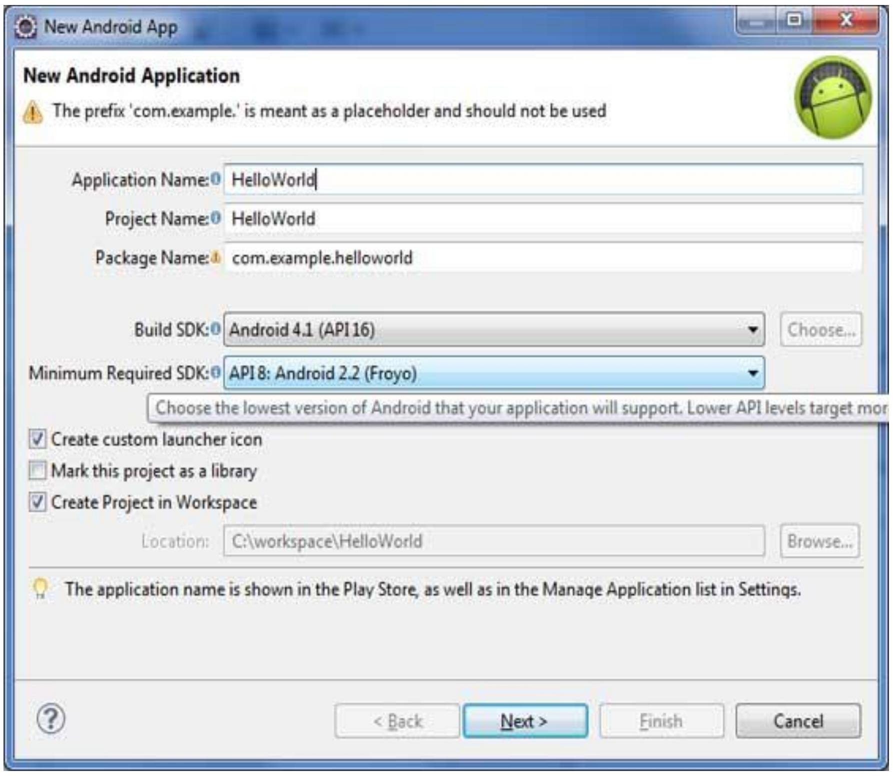
	other build system options, as required.
src Folder	Required folder where all source code for the application resides
src/com.androidbook.myfirstandroidapp/ MyFirstAndroidApp- Activity.java	Core source file that defines the entry point of your Android application.
gen Folder	Required folder where auto-generated resource files for the application reside
gen/com.androidbook.myfirstandroidapp/ R.java	Application resource management source file generated for you; it should not be edited
res Folder	Required folder where all application resources are managed. Application resources include animations, drawable image assets, layout files, XML files, data resources such as strings, and raw files.
es/drawable-*/icon.png	Resource folders that store different resolutions of the application icon.
res/layout/main.xml	Single screen layout file.
res/values/strings.xml	Application string resources.
assets Folder	Folder where all application assets are stored. Application assets are pieces of application data (files, directories) that you do

	not want managed as application resources

Prog-2

Step-1 Create Android Application

1) Follow the option **File -> New -> Project** and finally select **Android New Application** wizard from the wizard list.



New Android App

New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: HelloWorld

Project Name: HelloWorld

Package Name: com.example.helloworld

Build SDK: Android 4.1 (API 16)

Minimum Required SDK: API 8: Android 2.2 (Froyo)

Choose the lowest version of Android that your application will support. Lower API levels target mor

☑ Create custom launcher icon

☐ Mark this project as a library

☑ Create Project in Workspace

Location: C:\workspace\HelloWorld

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

1) **src**
   This contains the **.java** source files for your project. By default, it includes an *MainActivity.java* source file having an activity class that runs when your app is launched using the app icon.
2) **gen**
   This contains the **.R** file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
3) **bin**
   This folder contains the Android package files **.apk** built by the ADT during the build process and everything else needed to run an Android application.
4) **res/drawable-hdpi**
   This is a directory for drawable objects that are designed for high-density screens.
5) **res/layout**
   This is a directory for files that define your app's user interface.

6) **es/values**
   This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.
7) **AndroidManifest.xml**
   This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

The Main Activity File

```
package com.example.helloworld;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;

publicclassMainActivityextendsActivity{

@Override
publicvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu){
        getMenuInflater().inflate(R.menu.activity_main,
menu);
returntrue;
}
}
```

The Manifest File

```
<manifestxmlns:android="http://schemas.android.com/apk/re
s/android"
package="com.example.helloworld"
android:versionCode="1"
android:versionName="1.0">
<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="15"/>
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name=".MainActivity"
android:label="@string/title_activity_main">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
```

```
<categoryandroid:name="android.intent.category.LAUNCHER"/
>
</intent-filter>
</activity>
</application>
</manifest>
```

The Strings File

```
<resources>
<stringname="app_name">HelloWorld</string>
<stringname="hello_world">Hello world!</string>
<stringname="menu_settings">Settings</string>
<stringname="title_activity_main">MainActivity</string>
</resources>
```

The R File

```
package com.example.helloworld;

publicfinalclass R {
publicstaticfinalclass attr {
}
publicstaticfinalclass dimen {
publicstaticfinalint padding_large=0x7f040002;
publicstaticfinalint padding_medium=0x7f040001;
publicstaticfinalint padding_small=0x7f040000;
}
publicstaticfinalclass drawable {
publicstaticfinalint ic_action_search=0x7f020000;
publicstaticfinalint ic_launcher=0x7f020001;
}
publicstaticfinalclass id {
publicstaticfinalint menu_settings=0x7f080000;
}
publicstaticfinalclass layout {
publicstaticfinalint activity_main=0x7f030000;
}
publicstaticfinalclass menu {
publicstaticfinalint activity_main=0x7f070000;
}
publicstaticfinalclassstring{
publicstaticfinalint app_name=0x7f050000;
publicstaticfinalint hello_world=0x7f050001;
publicstaticfinalint menu_settings=0x7f050002;
publicstaticfinalint title_activity_main=0x7f050003;
}
publicstaticfinalclass style {
publicstaticfinalintAppTheme=0x7f060000;
}}
```

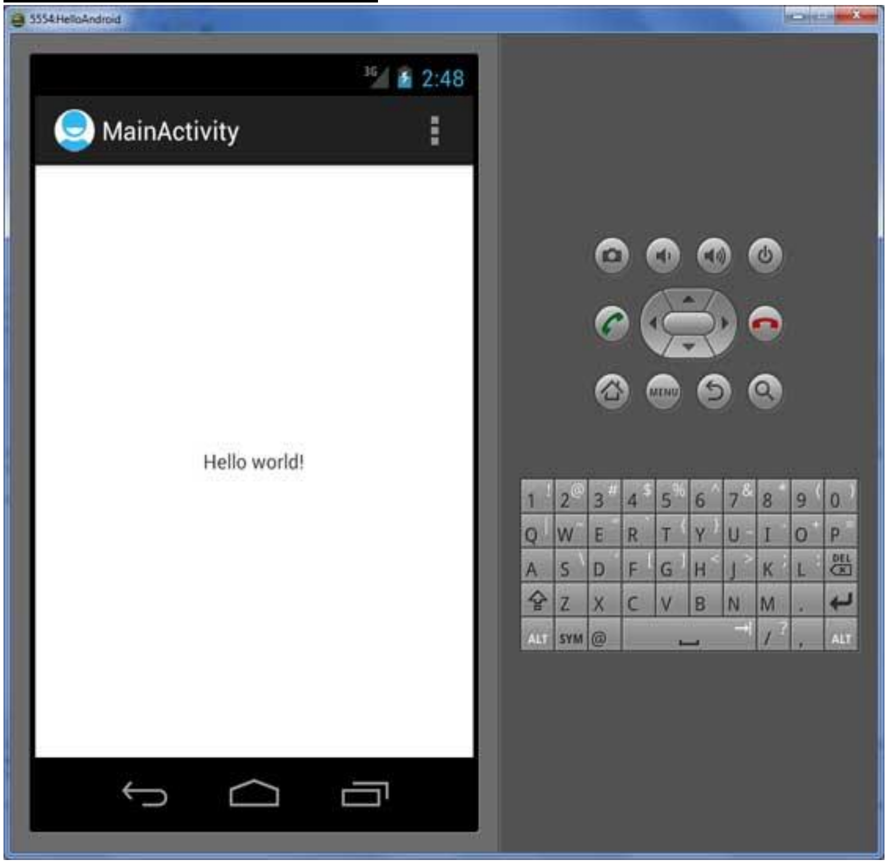
The Layout File

```
<RelativeLayoutxmlns: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">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_centerVertical="true"
android:padding="@dimen/padding_medium"
android:text="@string/hello_world"
tools:context=".MainActivity"/>

</RelativeLayout>
```

Running the Application



Hello world!