ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

PG DIPLOMA IN COMPUTER APPLICATIONS

PGDCA

Introduction to Android Programming & Python

CSP-46

Block

2

Android Development Environment

Unit -1
Android Development Platform

Unit -2
Installation and configuration of ADE

Unit -3
First application and development environment

ଓଡ଼ିଶା ରାଜ୍ୟ ମୁକ୍ତ ବିଶ୍ୱବିଦ୍ୟାଳୟ, ସମ୍ବଲପୁର, ଓଡ଼ିଶା
Odisha State Open University, Sambalpur, Odisha
Established by an Act of Government of Odisha.

About this Course Material for OPEN DISTANCE LEARNING

Welcome to the course on “Fundamentals of Android Programming”.

The book ‘**Introduction to Android**’ has been produced by The Open University of Sri Lanka. The course material titled “**Android Programming**” has been produced by GRAPHIC ERA HILL UNIVERSITY, Dehradun, India as a part of the OER for Skills Development initiative of Commonwealth of Learning (COL). All the study material produced by Odisha State Open University is structured in the same way, as outlined below.

UNIT-01

Android Development Platform

Learning Objective

After Completion of this Unit we will learn about:

1) What is ADE Environment?
2) Know about Android development Platforms.
3) The background, and platform versions, system features, Android tools for the development environment
4) Reasons for Android Development
5) System Requirements for Android Development

Unit Structure

Unit 1

Android Development Platform

Introduction

In this unit you will be able to get familiar with available Android development platforms. You need to watch the video and install the required tools to start developing your first Android application.

Upon completion of this unit you should be able to:

Outcomes

- explain development platforms and distinguish each against their features and capabilities.
- describe the background, and platform versions, system features, Android tools for the development environment.
- configure the Android development environment in a computer.

Terminology

command line:	commands entered as inputs without IDE
SDK:	Software Development Kit
JDK:	Java Development Kit
Android Studio:	Official Android platform to develop Android apps

1.1 Reasons for Android Development

Today, mobile telephones have fundamentally changed the way of people interact. It is evident that mobile applications will be the future of handheld devices, Television and Automobile. Moreover, developers have started embedding Android in home appliances and other devices.

There are many reasons for the popularity of Android apps, such as:

- Google provides one window solution, as Play Store, to upload and download the application either free or with minimal charges. For uploading and distributing the app, developers have no need of any approval of someone.

- Developer is the owner of his / her app and has the total control on product. However, Google has rights to unpublish any Android application in play store, if it is not complying with Google's licenses. For instance, if application contains malicious code or violate license, Google has right to unpublish the application.
- Android has open source operating system, open source software development kit (SDK) and good documentation.
- Android applications are not limited to mobile devices (Phones & Tabs), but can be run on TVs, wearable devices, vehicles and even refrigerators.
- (Source: First Thrust Towards Android, Android Programming, Course Material for Open Distance Learning, Commonwealth of Learning 2016)

1.2 Android Development Platforms, Features and Tools

In unit 2, you have learned Android architecture and major components of the Android platform. Let's look at the Android platform and the features they are providing.

Android Studio is the official IDE for Android development, and with a single download it includes everything you need to begin developing Android apps as you can see below

- IntelliJ IDE + Android Studio plugin
- Android SDK Tools
- Android Platform-tools
- A version of the Android platform
- Android Emulator with an Android system image including Google Play Services

Android Studio provides tools for building apps on every type of Android device. Code editing, debugging, performance tooling, a flexible build system, and an instant build or deploy system are included in Android studio. Let's see what are the systems requirements to install Android studio in different operating systems.

System Requirements

System requirements for Windows, Mac OS and Linux are given below.

<u>Windows - Microsoft® Windows® 7/8/10 (32- or 64-bit)</u>

- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,

- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality

Mac - Mac® OS X® 10.10 (Yosemite) or higher, up to 10.12 (macOS Sierra)

- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

Linux - GNOME or KDE desktop

- Tested on Ubuntu® 12.04, Precise Pangolin (64-bit distribution capable of running 32-bit applications)
- 64-bit distribution capable of running 32-bit applications
- GNU C Library (glibc) 2.19 or later
- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution

For accelerated emulator: Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality, or AMD processor with support for AMD Virtualization™ (AMD-V™).

(Source: https://source.android.com/source/requirements.html, CC:BY: 2.5)

Command Line Tools

The Android SDK tools available from the SDK Manager provide additional command-line tools to help you during your Android development. The tools are classified into two groups: SDK tools and platform tools. SDK tools are platform independent and are required no matter which Android platform you are developing on. Platform tools are customized to support the features of the latest Android platform.

Additional Command Line Tools

The Android SDK tools available from the SDK Manager provide additional command-line tools to help you during your Android development. The tools are classified into two groups: SDK tools and platform tools. SDK tools are platform independent and are required no matter which Android platform you are developing on. Platform tools are customized to support the features of the latest Android platform.

SDK Tools

The SDK tools are installed with the SDK starter package and are periodically updated. The SDK tools are required if you are developing Android applications. The most important SDK tools include the Android SDK Manager (Android sdk), the AVD Manager (Android AVD) the emulator (emulator), and the Dalvik Debug Monitor Server (DDMS). A short summary of some frequently-used SDK tools is provided below.

Virtual Device Tools

a) Android Virtual Device Manager

The AVD Manager provides a graphical user interface in which you can create and manage Android Virtual Devices (AVDs) that run in the Android Emulator.

b) Android Emulator (emulator)

A QEMU(short for quick emulator) based device emulation tool that you can use to debug and test your applications in an actual Android run-time environment.

c) mksdcard

Helps you create a disk image that you can use with the emulator, to simulate the presence of an external storage card (such as an SD card).

Development Tools

Hierarchy Viewer (hierarchyviewer) - Provides a visual representation of the layout's View hierarchy with performance information for each node in the layout, and a magnified view of the display to closely examine the pixels in your layout.

SDK Manager

SDK Manager lets you manage SDK packages, such as installed platforms and system images.

sqlite3 - Lets you access the SQLite data files created and used by Android applications.

Debugging Tools

The debugging tools are further explained in the later units of this material.

a) Android Monitor

Android Monitor is integrated into Android Studio and provides logcat, memory, CPU, GPU, and network monitors for app debugging and analysis.

b) adb

Android Debug Bridge (adb) is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It also provides access to the device shell.

c) Dalvik Debug Monitor Server (DDMS)

DDMS lets you debug Android apps.

d) Device Monitor

Android Device Monitor is a stand-alone tool that provides a graphical user interface for several Android application debugging and analysis tools.

e) Systrace

This tool lets you analyze the execution of your application in the context of system processes, to help diagnose display and performance issues.

f) traceview

Provides a graphical viewer for execution logs saved by your application.

g) Tracer for OpenGL ES

Allows you to capture OpenGL ES(the standard for Embedded Accelerated 3D Graphics) commands and frame-by-frame images to help you understand how your app is executing graphics commands.

Build Tools

a) apksigner

Signs APKs and checks whether APK signatures will be verified successfully on all platform versions that a given APK supports.

b) JOBB

Allows you to build encrypted and unencrypted APK expansion files in Opaque Binary Blob (OBB) format.

c) ProGuard

Shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names.

d) zipalign

Optimizes APK files by ensuring that all uncompressed data starts with a particular alignment relative to the start of the file.

Image Tools

a) Draw 9-patch

Allows you to easily create a NinePatch (class permits drawing a bitmap in nine or more sections) graphic using a WYSIWYG (What You See Is What You Get) editor. It also previews stretched versions of the image, and highlights the area in which content is allowed.

b) Etc1tool

A command line utility that lets you encode PNG images to the ETC1 compression standard and decode ETC1 compressed images back to PNG.

Platform Tools

The platform tools are typically updated every time you install a new SDK platform. Each update of the platform tools is backward compatible with older platforms. Usually, you directly use only one of the platform tools—the Android Debug Bridge (adb). Android Debug Bridge is a versatile tool that lets you manage the state of an emulator instance or Android-powered device. You can also use it to install an Android application (APK) file on a device.

The other platform tools, such as aidl, aapt, dexdump, and dx, are typically called by the Android build tools, so you rarely need to invoke these tools directly. As a general rule, you should rely on the build tools to call them as needed.

Note: The Android SDK provides additional shell tools that can be accessed through adb, such as bmgr and logcat.

a) bmgr

A shell tool you can use to interact with the Backup Manager on Android devices supporting API Level 8 or greater.

b) logcat

Provides a mechanism for collecting and viewing system debug output.

(Source: https://developer.android.com/studio/command-line/index.html CC:BY: 2.5)

Now, you have and learnt the systems requirements (hardware/software features) to set up the Android development platform and the Android command line tools. It is vital to determine the specific features of each Android version and how it has been developed to performing better. In unit 1 we learnt the history of Android and how each version of Android evolved. Next, we will summarize the Android platform versions with their unique features.

Android platform versions and specific features

There are rapid developments and new versions for the Android and Table 4.1 summarize the specific features of different Android platform versions up to now.

Table 4.1 summary the specific features of different Android platform versions

Android Platform version	**Specific Features**
Android 1.6 - Donut	Quick search box Screen size diversity Android market
Android 2.1 - Eclair	Google maps navigation Home screen customization Speech-to- Text
Android 2.2- Froyo	Voice actions Portable Hotspot Performance
Android 2.3- Gingerbread	Gaming APIs Near Field Communication (NFC) Battery Management
Android 3.0- Honeycomb	Tablet-friendly design System bar Quick settings

Android 4.0- Ice cream Sandwich	Custom home screen Data usage control Android Beam
Android 4.1- Jelly Bean	Google now Actionable Notifications Account switching
Android 4.4- Kitkat	Voice: OK Google Immersive Design Smart Dialer
Android 5.0- Lollipop	Material Design Multiscreen Notifications
Android 6.0- Marshmallow	Now on tap Permissions Battery works smarter
Android 7.0 - Nougat	Multi Locale language settings Multi-window view Quick switch between apps Data Saver Notification Controls Display Size

You can read more information online about the relative numbers of devices that are running different versions in the following link.

https://developer.android.com/about/dashboards/index.html

Thus, creating apps in Android for various mobile devices are increasing day by day. Since developing native apps is expensive, the demand for cross platform app development tools is also increasing. It is essential to know cross platforms and tools for mobile application development to develop apps for enhancing the market capacity.

You can watch the online presentation given in the link below to get familiar with the cross platforms for mobile application development.
bit.ly/XPlatformMobileDev

1.3 Python for Android Mobile App Development

Python is a high-level programming language that is widely used in web development, app development, analyzing and computing scientific and numeric data, creating desktop GUIs, and for software development.

Python is the most taught programming language at the school and college level for the fact that it has several applications in real life.

Python is a powerful high-level language that can be used to create android and desktop apps from scratch. Just to give you a hint of how powerful this language is, Dropbox is created in Python.

Python is used in a wide variety of application domains as it can easily be connected with C, Objective-C, Java, or FORTRAN. It runs on all major operating systems, like Windows, Linux/Unix, OS/2, Mac, Amiga, etc. With the help of Python, we can create any type of mobile applications, like Calibre, OpenStack, Ubuntu Software Center, World of Tanks, Quora, BitTorrent, Reddit, Spotify, YouTube, Instagram, and many more.

1.4 Why Python for Mobile App Development?

1. Python runs on all major operating systems such as Windows, Linux/Unix, OS/2, Mac, Amiga, etc.
2. The language provide constructs intended to enable clear programs on both a small and a large scale.
3. Python provides more tools for both the developers and the system administrators.
4. Many companies like GOOGLE, Yahoo!, and IBM use this Python, because it's a fun and a dynamic language.
5. It provides faster development and portability allows for the same application to run across platforms.
6. Python is packed with rich libraries and many add-on packages to tackle specific tasks.

The features of Python is natural expression of procedural code, strong introspection capabilities, very clear, readable syntax, intuitive object orientation, very high level dynamic data types, extensions and modules easily written in C, C++ (or Java for

Jython, or .NET languages for IronPython), extensive standard libraries and third party modules for virtually every task, full modularity, supporting hierarchical packages, exception-based error handling and embeddable within applications as a scripting interface. Python is widely used in Web application development. It is also used in a wide range of non-scripting contexts. Python plays well with COM, .NET, CORBA, Java and it often compared to Tcl, Perl, Ruby, Scheme or Java.

1.5 Role of Python in Mobile Applications?

There are a variety of applications in various fields including business, entertainment, utilities, hospitality sector, games and much more, these apps are made sure to fit to various screen sizes be it iPad, iPhone, iPod Touch, laptop, palmtop etc. Mobile Application Development has become a leading business sector due to its increasing scope. For every person in the world, it is quite impossible to live without mobile. It plays an important role in all life. There are various software systems used by the designers for the mobile phones such as Symbian, J2Me, Android, Flash light, Python, Lazarus, BREW, etc.

This is a nice cross platform python framework which works for Android, Win7, Linux, Mac. It is a great tool for writing both simple scripts and complex, multi-threaded applications. The great thing about having Python on Android is the opportunity to use the untold thousands of lines of code already written and freely available. iPhone or Android App Developers take a number of cross platform development techniques in order to provide a great mobile application to the customers. The top tools used for cross-formatting mobile application development are RhoMobile, PhoneGap, Appcelerator, Sencha Touch, MoSync, Whoop, WidgetPad, GENWI, AppMakr, Mippin, SwebApps, MobiCart, etc.

Appcelerator – This platform uses web technologies and develops great apps for desktop, mobile and tablet applications. The languages which are used to develop apps are HTML, Python, PHP and few others.

MoSync SDK – It is used to build mobile applications iOS, Android, Windows Phone 7, Symbian, Windows Mobile, javaME and Blackberry. It uses common programming languages such as PHP, JavaScript, Ruby, & Python for cross platform Mobile Application Development.

1.6 Mobile Application written in Python

1. **Aarlogic C05/3** – Ready to use GSM /GPS tracking PCB with Python development on board with support of test server based on Google Maps.
2. **AppBackup** – an app for jailbroken iOS devices that lets one back-up and restore settings and data from App Store apps.
3. **Pyroute** – a GPS-capable mapping/routing application for mobile devices.

Android Google provides Android Scripting Environment (ASE) which allows scripting languages (Python included) to run on Android. The excellent features of Python allows it to plays an important role in mobile applications.

Activity

Activity 1.1

Explore most popular cross platforms and name three major cross platforms.

Briefly describe one of the major cross platform with the features and uses of them.

Hint: Check your answers with Answer guide at the end of this unit.

Now you are aware of the native and cross platform for mobile application development. We will now explore how the development environment is set up and configured.

Android.

Unit summary

The first step to start on Android based application development is to set up the development environment. Therefore, it is essential to know how to configure development environment and installing tools for Android application development.

In this unit, you have learned Android development platforms, tools and cross platforms. At the last section of the unit, you learnt about setting up the Android development environment by installing the latest JDK and Android Studio. It is also important to keep the Android studio IDE and Android SDK tools up to date at the development environment.

UNIT-02
Installation and configuration of ADE

Learning Objective

After Completion of this Unit we will learn about:

1) How to download and install JDK
2) Know about installation process of Eclipse and Android studio
3) Use of the ADT Plug-in for Eclipse
4) Learn about Android SDK
5) and how to deal with backward compatibility

Unit Structure

Unit 2

Installation and configuration of ADE

2.1 Configuring Android development environment

For developing the Android application, first you have to setup the Android development environment. There can be three different ways to do so.

1. Download and install the Eclipse + Java + ADT Bundle.
2. Download and install the Android Studio + Java.
3. Download and install every individual tool (like Java, SDK Manger, DDMS tool, AVD Manager, etc.) without any IDE and operate from command line and code in a text editor such as notepad or other.

In this course you can choose either Eclipse or Android Studio to develop the application. Eclipse and Android Studio, both are the good IDE for Android development. Android Studio is a native Android IDE that is fully dedicated to Android development. Eclipse is an open source Java IDE that is compatible for several platforms; ADT Plugin needs to be installed to make it ready for Android development.

2.1.1 Downloading and installing the JDK

Most of the programming of Android is done using the Java programming language. To download latest Java Development Kit, follow the following link:

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

Now, extract the downloaded zip file and double click on the .exe and follow the instructions.

2.2 Downloading and installing the Eclipse

Eclipse is an open source IDE which is very popular for Java development. You will use the Eclipse IDE for Android development because it provides a tight integration of many Android building and debugging tools. It is available for many platforms, included Windows, Linux and Mac OS. It has many versions for each platform. But for Android development you required Eclipse version 3.5(Galileo) or above. If you download version below than Galileo, then you cannot configure it for Android programing by using the ADT plug-in.

To set up the Eclipse you will use the latest versions of the following tools:

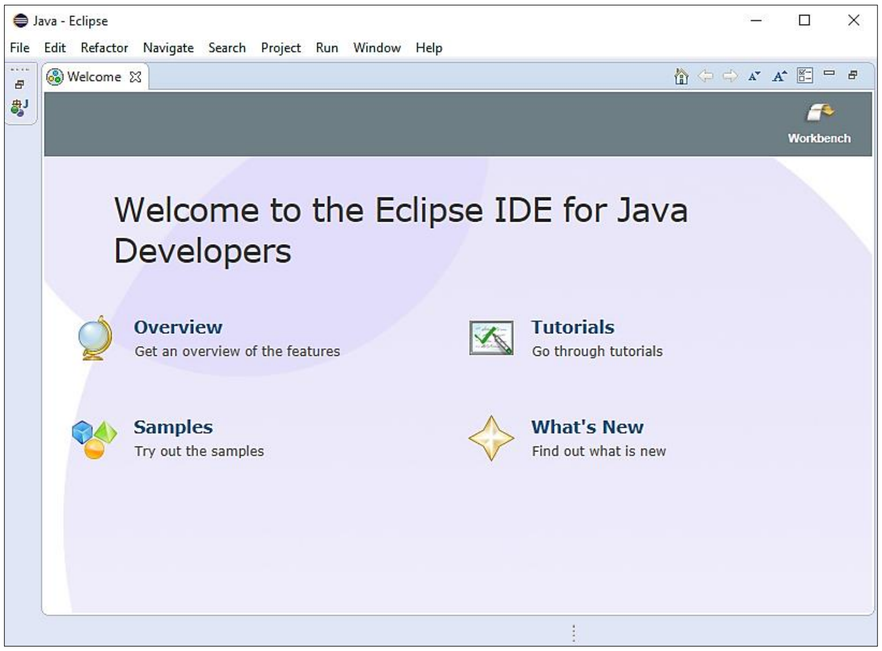
1. JDK: J2SE 8
2. Eclipse: Luna (Eclipse 4.4.2)
3. SDK: Marshmallow (Android 6.0 - API 23)

For downloading the tools, you need a high speed internet connection.

Use the following link to download the Eclipse IDE for Java Developer:

https://eclipse.org/downloads/

Now unzip the downloaded file in a new folder and double click on the file eclipse.exe. First of all you will need to create a workspace for your projects. A workspace is a folder where all projects and work done will be saved. In future you can create a new workspace for your new projects. First time when you run the Eclipse, the welcome window will look as Figure 1.3.



Welcome to the Eclipse IDE for Java Developers

Overview
Get an overview of the features

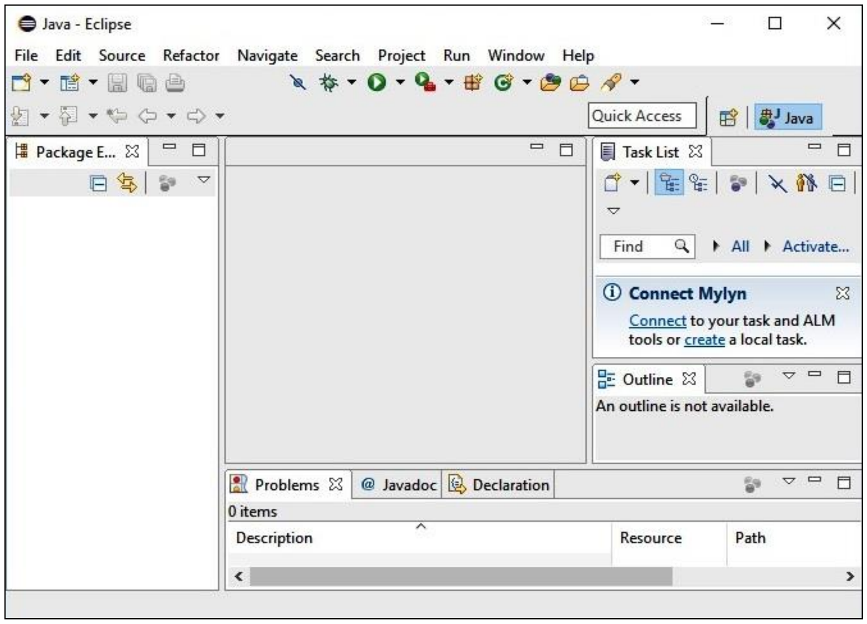
Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

Figure 1.3: Welcome Screen of Eclipse

And work window in Java perspective will look as Figure 1.4.



Connect to your task and ALM tools or create a local task.

An outline is not available.

0 items

Description	Resource	Path

Figure 1.4: Working window in Java Perspective

Till now, you have installed JDK and Eclipse. But still this environment is not ready for the Android development

2.2.1 Using the ADT Plug-in for Eclipse

For embedding the Android development tools in the Eclipse, you need to add ADT (Android Developer Tools) plug-in to the Eclipse. For adding the ADT plug-in to Eclipse there are two ways: first is by "Eclipse Marketplace" and second is by Install Software. You will use the first way. Steps are as follows:

1. Click on the "Eclipse Marketplace..."in the "Help" menu. (Figure 1.5)

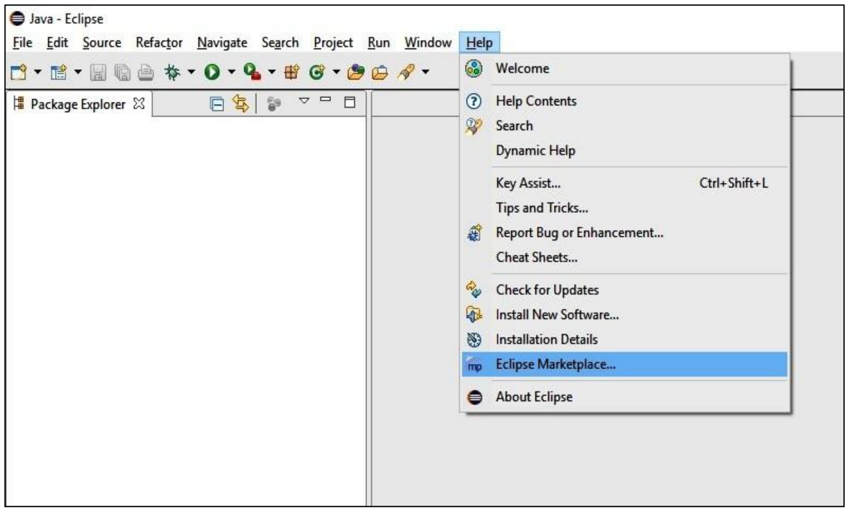
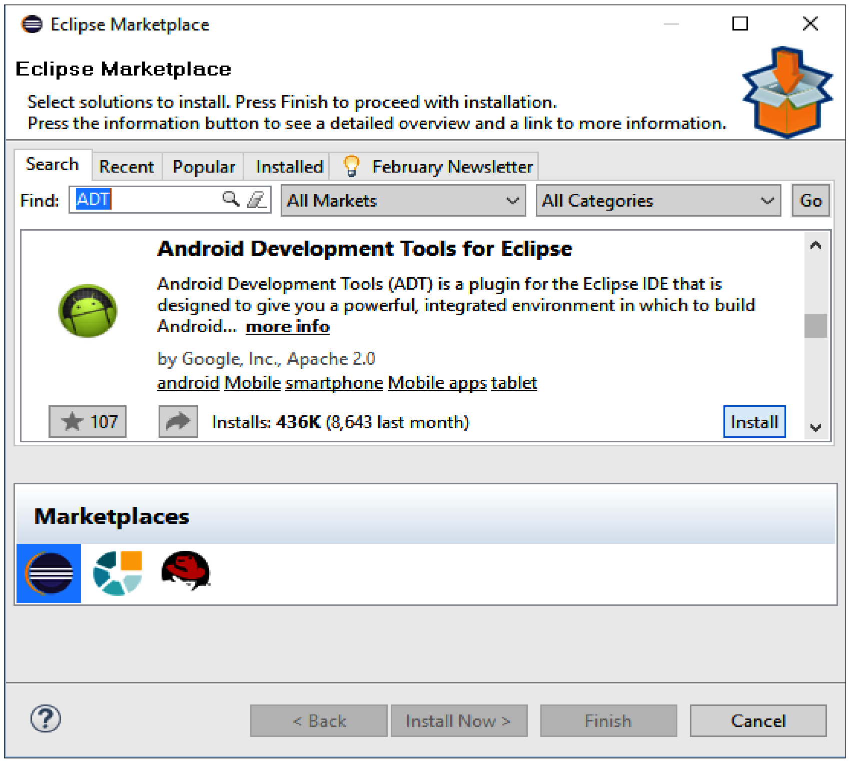


Figure 1.5: Selecting the Eclipse Marketplace from Help menu

2. Now type ADT in search box "Find" in Eclipse Marketplace window. Search the plug-in "Android Development Tools for Eclipse" in the list of plug-in and click on the Install button. (Figure 1.6)


Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.

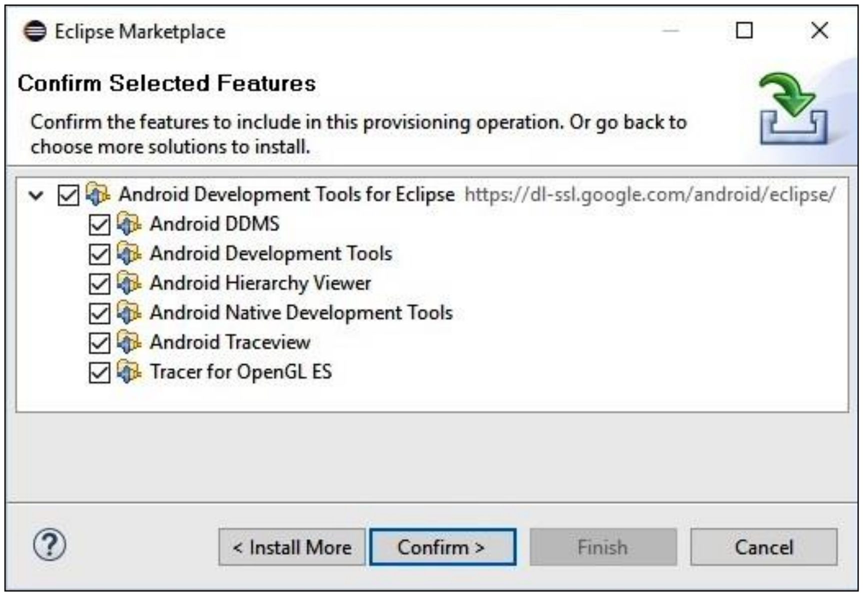
Android Development Tools for Eclipse

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android... more info

by Google, Inc., Apache 2.0
android Mobile smartphone Mobile apps tablet

Figure 1.6: Selecting the ADT plug-in in Marketplace

3. Now select the required features of ADT and click on the Confirm button as shown in Figure 1.7.


Eclipse Marketplace

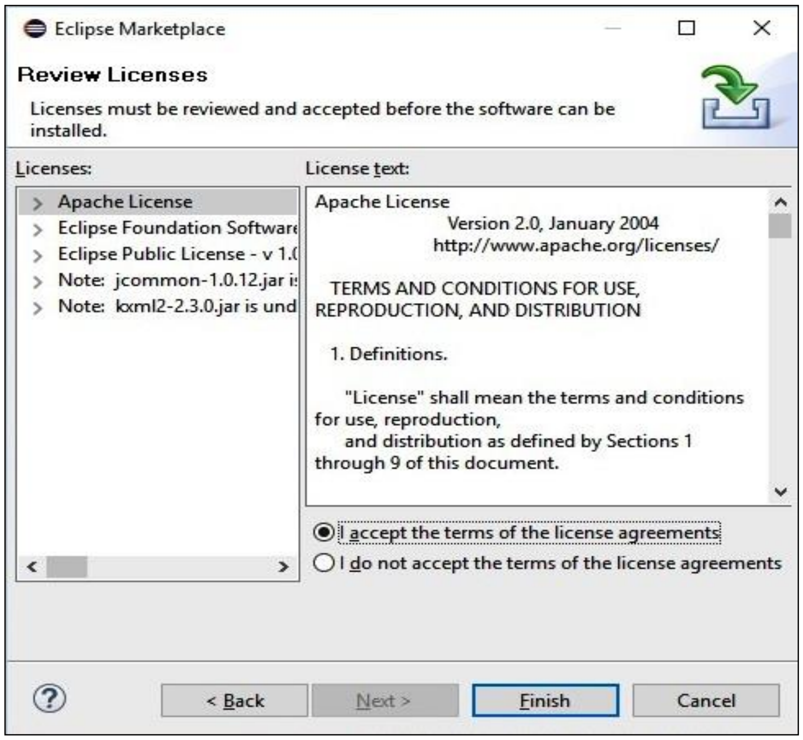
Confirm Selected Features

Confirm the features to include in this provisioning operation. Or go back to choose more solutions to install.

- [✓] Android Development Tools for Eclipse https://dl-ssl.google.com/android/eclipse/
  - [✓] Android DDMS
  - [✓] Android Development Tools
  - [✓] Android Hierarchy Viewer
  - [✓] Android Native Development Tools
  - [✓] Android Traceview
  - [✓] Tracer for OpenGL ES

Figure 1.7: Confirm the selected ADT features

4. Accept the license in Review Licenses window and click the Finish button. This process can take some time to download and install the ADT Bundle as per the internet speed. (Figure 1.8)



Eclipse Marketplace

Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

- Apache License
- Eclipse Foundation Software
- Eclipse Public License - v 1.0
- Note: jcommon-1.0.12.jar is
- Note: kxml2-2.3.0.jar is und

License text:

```
Apache License
                    Version 2.0, January 2004
                 http://www.apache.org/licenses/

  TERMS AND CONDITIONS FOR USE,
REPRODUCTION, AND DISTRIBUTION

  1. Definitions.

     "License" shall mean the terms and conditions
for use, reproduction,
     and distribution as defined by Sections 1
through 9 of this document.
```

[✓ I accept the terms of the license agreements]

I do not accept the terms of the license agreements

Figure 1.8: Accepting the license agreement

After adding the ADT Plug-in in Eclipse, all required tools (such as Android SDK Manager, AVD Manager, DDMS, and ADB etc.) that you have selected during ADT Plug-in installation, becomes available in Eclipse. All specified tools are described in the UNIT 2. The working window in DDMS (*Dalvik Debug Monitor Server*) prospective will look as the Figure 1.9.

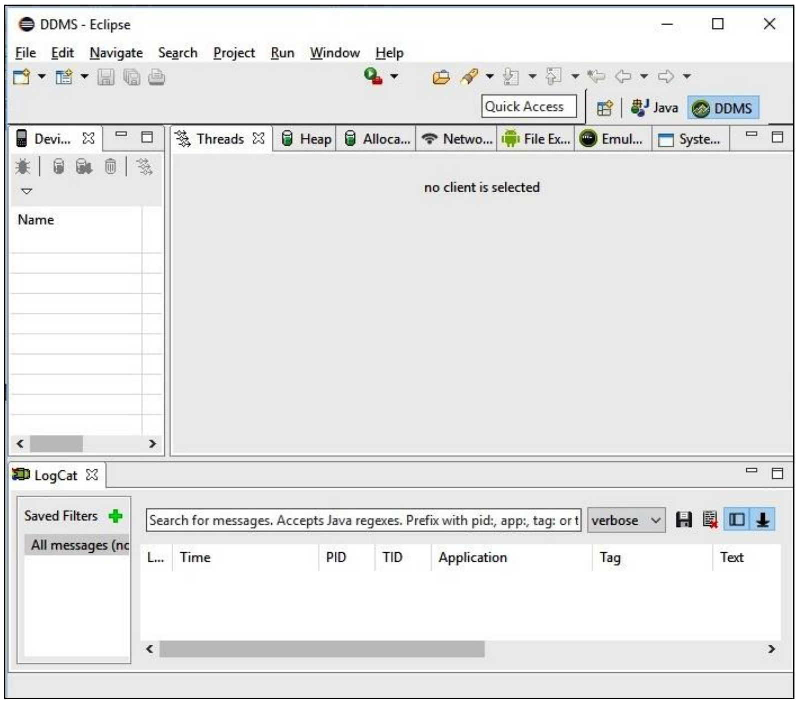

DDMS - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java
DDMS
Devi...
Threads
Heap
Alloca...
Netwo...
File Ex...
Emul...
Syste...
Name
no client is selected
LogCat
Saved Filters
All messages (nc
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or t
verbose
L... Time PID TID Application Tag Text

Figure 1.9: DDMS Prospective

Reading

Explore the DDMS prospective from Eclipse and try to learn its features. For more details of DDMS tool follow the following link:

http://developer.android.com/tools/debugging/ddms.html

Activity

Try another method to install the ADT plug-in using “Install New Software...” option of “Help” menu and write the steps.

Answer: ---

Hint: After clicking on the Add button of the “Install New Software” window a dialog box “Add Repository” will pop up. Enter name field value as “ADT Plugin” and location field value as a URL given below:

https://dl-ssl.google.com/android/eclipse

Now follow the instructions.

2.3 Downloading and installing the Android Studio

You can download Android Studio 2.1 Stable from the following link:

http://developer.android.com/sdk/index.html

Download the executable file from the above mentioned link, double click on the file and follow the installation instructions. To install the "Android Studio 2.1 Stable" you need to install Java SE Development Kit 8 first. You can pursue the following link to grab the installation instructions for Android Studio.

http://developer.android.com/sdk/installing/index.html

After installing when you will start the Android Studio first time, the very first window will look like Figure 1.10.

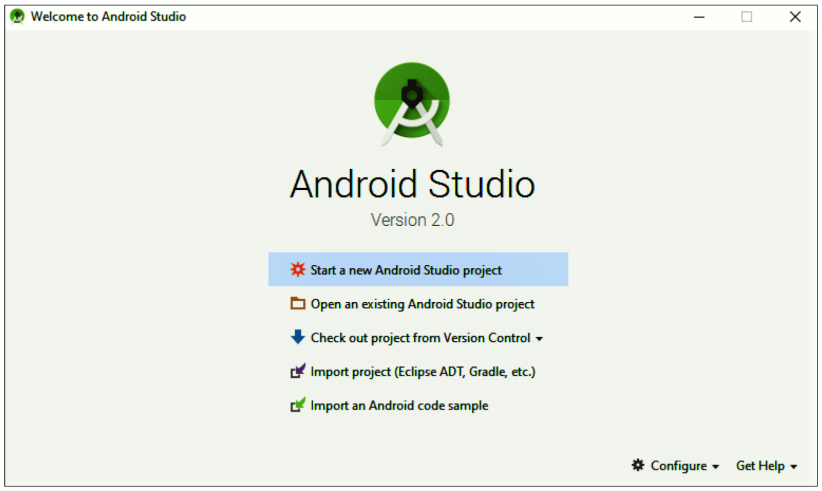

Welcome to Android Studio
Android Studio
Version 2.0
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

Figure 1.10: First time opening window of Android Studio

When you click on the first option to start a new Android Studio Project, multiple windows will pop up one after another to setup a new project.

Android Studio has numerous features. In this course, you are going to explore and use many features of Android Studio. For the prior reading and to understand the Android studio pursue the following web link:

http://developer.android.com/tools/studio/index.html

Android Studio has all required tools inbuilt for Android development. Only tool that you need to install is the JDK; and update the SDK if required which you can do with the help of Android SDK Manager.

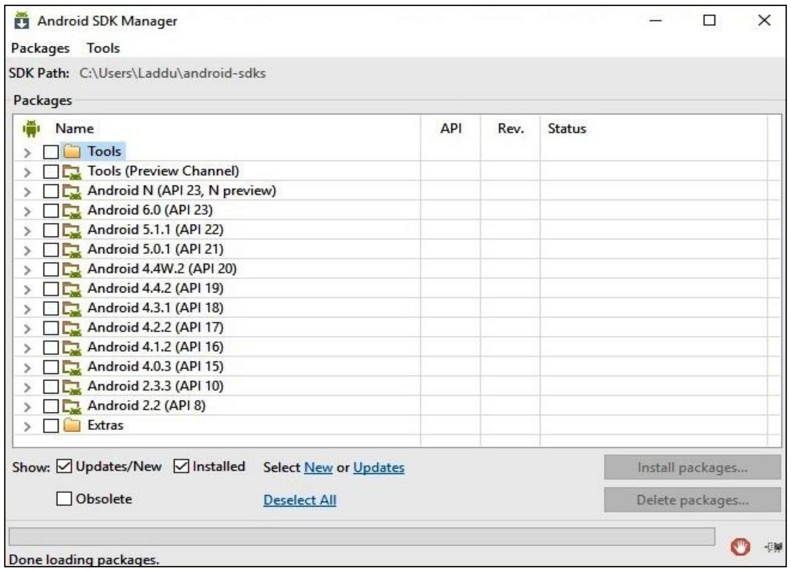
2.3.1 Downloading and updating the Android SDK

After setting up the either IDE, still one task is remaining; that is installing or updating the Android SDK. SDK is collection of different libraries and tools that enables you to program for Android.

If you are using the Android Studio, then you have no need to install the Android SDK because it already has it. But if you want to update or delete any library then you can use the Android SDK Manager tool.

If you are using the Eclipse, then there are two ways of installing the Android SDK: first by using Android SDK Manager and second by downloading the Android SDK separately and add it to the IDE using Preferences.

Use the following steps to open the SDK Manager Tool:
-In Android Studio, click on the “Tools” menu and select the “Android SDK Manager” in the “Android” sub menu. Default settings window will pop up. Now, to open the SDK Manager, click on the “Launch Standalone SDK Manager” link in the bottom of the window.
-In Eclipse, click on the menu “Window” and select “Android SDK Manager”.



Name	API	Rev.	Status
☐ Tools			
☐ Tools (Preview Channel)			
☐ Android N (API 23, N preview)			
☐ Android 6.0 (API 23)			
☐ Android 5.1.1 (API 22)			
☐ Android 5.0.1 (API 21)			
☐ Android 4.4W.2 (API 20)			
☐ Android 4.4.2 (API 19)			
☐ Android 4.3.1 (API 18)			
☐ Android 4.2.2 (API 17)			
☐ Android 4.1.2 (API 16)			
☐ Android 4.0.3 (API 15)			
☐ Android 2.3.3 (API 10)			
☐ Android 2.2 (API 8)			
☐ Extras			

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

Figure 1.11: Android SDK Manager

The SDK Manager will look as in the Figure 1.11.Now select the required API and tools, and click on Install Packages button. This can take some time depending on the internet speed.

Now, you are ready to go for making an Android App.

Note it!

Although both IDEs (Whether it is Eclipse or Android Studio) have some differences like the project structures, look ‘n’ feel, tools window arrangements, etc. but both IDEs uses same build tools and same coding style. Even project files and intermediate files created in Android are same, so choose either IDE. Android Studio has some advanced features like installing every missing file, tool, setting, etc. is just a click away; only you need a fast internet connection. But still, Android Studio is growing and it will take time to take over the other features of Eclipse. This course will help you to develop your app in both IDE’s; and this is the beauty of this course.

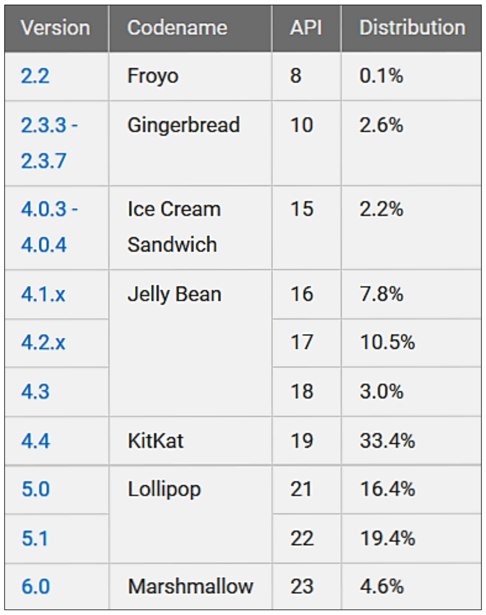
his video will show you how to install and configure the Android evelopment environment. What is the role of the ADT Plug-in in clipse?

https://www.youtube.com/watch?v=GDFEebaU_y4&index=3&list=PLXN-JCVb8z8BiyRhLO6HRNzfn5-TaTI_L

Video

2.4 Dealing with backward compatibility

When new version of Android comes into the market, older versions do not become obsolete so early. So, it becomes necessary to support old versions to target large number devices. You can take an idea of distribution of different Android versions from following distribution table (Figure 1.12.1) and pie chart (Figure 1.12.2):



Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.2%
4.1.x	Jelly Bean	16	7.8%
4.2.x		17	10.5%
4.3		18	3.0%
4.4	KitKat	19	33.4%
5.0	Lollipop	21	16.4%
5.1		22	19.4%
6.0	Marshmallow	23	4.6%

Figure 1.12.1 Android Version Distribution Table
(Source: http://developer.android.com/about/dashboards/index.html)

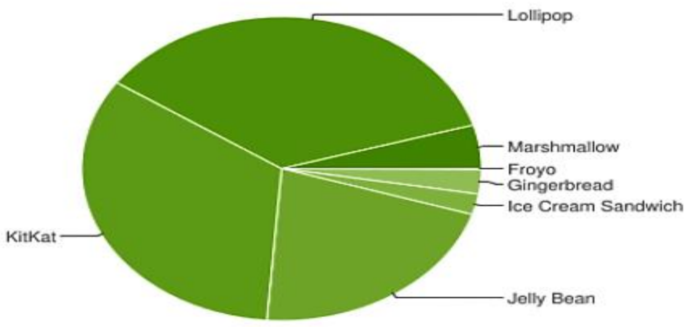


Figure 1.12.2 Android Version Distribution Pie-chart
(Source: http://developer.android.com/about/dashboards/index.html)

Note: This data is collected during a 7-day period ending on April 4, 2016. Any versions with less than 0.1% distribution are not shown.

Reading

For more Android statistics follow the following link:

http://developer.android.com/about/dashboards/index.html

To make your application is backward compatible to previous versions with best features and functionality; you must use Android Support Library in your applications. To install the Android support library to Android SDK follow the below mentioned steps:

1. Open the “SDK Manager” (Figure 1.11) and select “Support Libraries” from “Extras” section.
2. Now click on the “Install packages...” button.

Reading

To understand all features of Android Support Library follow the following link:

http://developer.android.com/tools/support-library/index.html

Video-V6: Install Android for Windows 10

URL:
https://storage.googleapis.com/androiddevelopers/videos/studio-install-windows.mp4

This video will show you how to install and configure the Android development environment using Android Studio in Windows 10. Furthermore, the following link has videos that shows each step of the recommended setup procedure for Mac and Linux.

https://developer.android.com/studio/install.html

2.5 Unit summary

Summary

In this unit you learned about the Android and its capabilities. You also explored the history of Android, different API levels of Android, Architecture of Android. You have also learned about setting up the Android development environment by installing the JDK, Eclipse, and ADT Bundle. In the last, you have learned about downloading and installing the Android SDK, support library and build tools using Android SDK Manager.

Assignment

2.6 Assignment

Q1. What is Android?

Answer: Android is an open source Linux based operating system which is a product of Google Inc.(after acquiring from Android Inc.). Initially it was developed for touch screen mobile phones, but now it is also available for tablets, smart watch and Android Auto too. It is written in C, C++ and Java. For developing the Android applications Android SDK is available with all supportive tools. You can also state that, android is a system that includes an open source operating system, an open source development platform and devices that run the operating system and applications created for it. It has become the multi-billion dollar industry since its inception, so there is lots of space for earning for developers and users.

Q2. Explain the role of Linux Kernel in Android.

Answer: Linux Kernel is the foundation component of Android platform. It is there to handle the hardware; means it helps the software part of the Android System to interact with the hardware. Because all hardware drivers (display driver, keypad driver, camera driver, Wi-Fi driver, Bluetooth driver, etc.) are inbuilt in the kernel, the android runtime does not need to worry about the hardware handling. It is the lowest layer of Android architecture and it serves as the abstraction layer to other layers

Q3. Describe the role of Dalvik Virtual Machine and Android Runtime.

Answer: Dalvik Virtual Machine (DVM) is a register based virtual machine that is used by Android system to run the Dalvik executable code (.dex file) which is a compiled code of Android. It used in Android System similarly as JVM works for Java to execute the byte code (**.**class file). DVM doesn't work with **.**class files. One thing that you must know is that implicitly .dex file is generated by the **.**class file after highly optimizing the **.**class file for low memory and least processing power. It gives the power to a device to become an Android device.

Android Runtime (ART) is the successor of Dalvik Virtual Machine (DVM). It has some advantage over DVM including ahead-of-time compilation (AOT), improved garbage collection and other development and debugging enhancements.

Q4. Enlist the different features of Android operating system.

Answer: Android has many features that make it different from other platforms. Those features are as follows:

- It is an open source platform, so you don't need any license or permission.
- It has an open marketplace Play Store for distributing your applications.
- Google play provides free and upfront purchase of your applications.
- There is no need of any special approval for app distribution.
-You don't need any special certificate to become an Android developer. Only you need basic knowledge of Java and little knowledge of XML and SQL.
- Developer is the sole proprietor of his / her application and has full control on the app.

Q5. Explain the role of ADT Plug-in in Eclipse.

Answer: Android Development Tools (ADT) plug-in is a plug-infor the Eclipse IDE that enhances the capabilities of Eclipse to quickly set up and manage the Android project. It provides a set of tools to integrate with

Eclipse IDE, so it is also called ADT bundle. It provides the GUI access to many command line tools for rapid development of Android application. It provides the following capability:

-Integration Android project creation, building, installation, packaging and debugging
-SDK tools integration
-Java programming language and XML editors
-Integrated documentation for Android framework APIs

2.7 Self-Assessment problem

Assessment

Problem 1. Why should you be the developer of Android?

Problem 2. Describe the role Android Support Library.

Problem 3. What are the similarities and differences in Eclipse and Android Studio IDEs? Explain.

Problem 4. Analyse the both IDEs (Eclipse and Android Studio). After analysing, which IDE will you prefer to choose for Android application development? Justify your answer.

Problem 5. What are the different native libraries in android architecture?

2.8 Online Test

Study Skills

Use the following URL to attempt the online quiz to test your skills after completing the unit 2:

http://tinyurl.com/colandroidtest1

UNIT-03
First application and development environment

Learning Objective

After Completion of this Unit we will learn about:

1) How to download and install JDK
2) Know about installation process of Eclipse and Android studio
3) Use of the ADT Plug-in for Eclipse
4) Learn about Android SDK
5) and how to deal with backward compatibility

Unit Structure

Unit 3

First application and development environment

3.1 Introduction

In previous chapter, you have learnt about the Android history and Android SDK installation. In this chapter you will learn about the basic constructs of the Android application as well as the steps to create and run an Android project. For creating an Android application you need some basic knowledge of Java and XML programming languages. You need Java programming for creating the logic part and XML programming for creating the design part (User Interface). However, you can code both logic and design part in Java, but it becomes very easy to deal with logic and design separately. You can also program in native languages C and C++ using NDK (Native Development Kit). You will code in Java programming and can use either Eclipse or Android Studio as an IDE (Integrated Development Environment). In this chapter you will also learn about creating an AVD (Android Virtual Device) and running your application on the created AVD.

Upon completion of this unit you will be able to:

Outcomes

- *Create* First Android application.
- *Configure* the AVD.
- *Save* Launch configurations.
- *Debug* Android project.
- *Run* Android application.
- *Understand* tools used in Android application development.
- *Have* Knowledge of different files created in the development process.
- *Add* Permissions to the Manifest file.

Terminology

Activity:	An Activity is a user interface provided as a screen to interact with the application.
Resource:	Resources are the supplementary files and contents that your program uses, such as bitmap images, layouts, UI strings, animation, etc.
Prospective:	Every working window of an IDE contains at least one perspective. It contains a set of editors, and

	other windows, those have a set of functionality to accomplish a specific task.
Package:	A package is a method of organizing the classes into a single name (or folder) that provides the concept of modular programming. It can contain distinct class files or JAR files (Java Archive File that contains multiple compressed class files under a single name).
Debugging:	It is the procedure of finding the defects in a source code and removing them.
Workspace:	Workspace is a folder created by the Eclipse or Android Studio where all projects and projects related data get stored.
Graphical Layout:	Graphical layout is an arrangement of visual elements (e.g. Buttons, Labels, and Text Fields etc.) on a page or a screen.
APK File:	Android application package (APK) is the file format that packages the .dex file and different resource files.It is used by the Android OS for installation an application in the Android devices.
NDK:	The Native Development Kit (NDK) is a tool set that enables you to implement the parts of your applications by means of native-code languages like C and C++.

3.2 Creating first Android application

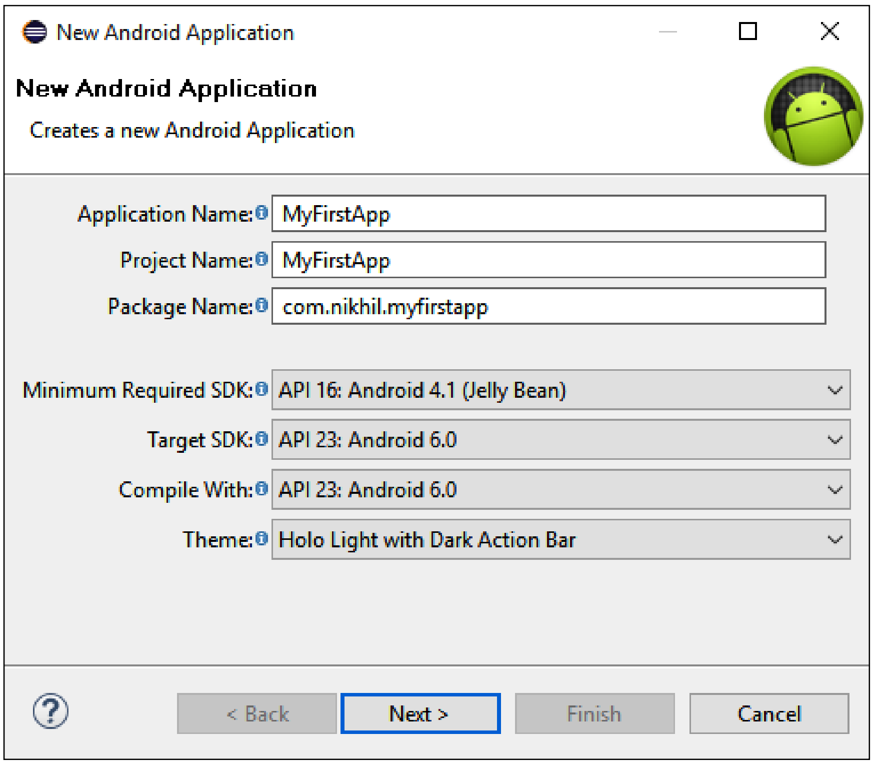
Now, after setting up the development environment, this is the time to move towards the Android application creation. Following sections will enable you to create a new Android project in both IDEs (Eclipse and Android Studio):

3.2.1 Creating a new Android project in Eclipse

First open the Eclipse IDE (that is configured for Android app development) and follow the following steps to create a new Android project using “New Android Application” wizard:

1. In “File” menu, select the “Android Application Project” option in the “New” submenu. After the click,the “New Android Application” window will pop up as shown in Figure 2.1. In this window you will fill the details of new application. Details of the fields are as follows:

   i. *Application Name:*It is the name of the application that appears on the Play Store and on the device of the user after installation. This must start with a capital letter.


New Android Application

New Android Application

Creates a new Android Application

Application Name: MyFirstApp

Project Name: MyFirstApp

Package Name: com.nikhil.myfirstapp

Minimum Required SDK: API 16: Android 4.1 (Jelly Bean)

Target SDK: API 23: Android 6.0

Compile With: API 23: Android 6.0

Theme: Holo Light with Dark Action Bar

Figure 2.1: Create a new Android application

ii. *Project Name:* It is used by the Eclipse and it must be unique in the workspace. Generally, application name and project name remains same; but this is not necessary.

iii. *Package Name:* It must be any unique identifier for your application to identify your app on the Play Store and user's device. It always remains same for your application even for multiple versions of your application because multiple versions of an application are considered as same application. Generally, for keeping it unique, we add domain name in reverse order with one or many unique java identifier. In addition, it must be a valid java package name.

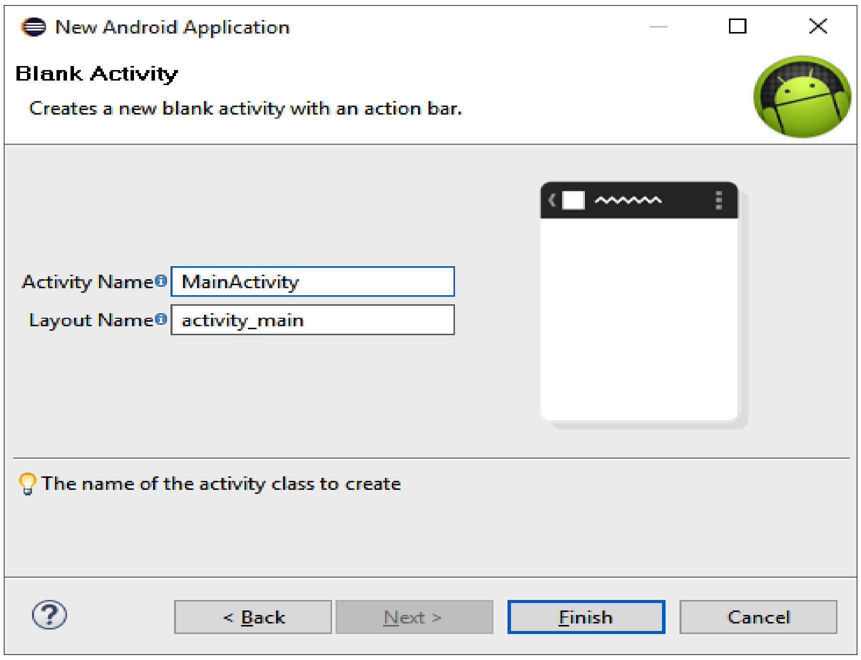
iv. *Minimum Required SDK:*The new versions of Android often provide best APIs for your application but you must continue to support previous versions of Android until older version devices get updated. So, during development process select the lowest version from the drop down list. As lower the version you select, more the devices it supports.

v. *Target SDK:* Select the highest API level that the application is going to work with. Since the application does not have forward compatibility issue for target API level. It is recommended to set highest level of API for Target SDK. Your application can still work with older versions up to Minimum SDK level that is set by you.

vi. *Compile With:* You should select the most recent version of installed SDK to compile the project.

vii. *Theme:* It specifies the user interface style to apply to your application. For now, just leave it as it is and click on the next button.

2. The “Configure Project” window will pop up. If not necessary, just read the options and click on the next button with default values.
3. Now another window, “Configure launcher icon”, will pop up. In this window, you can set the launcher icon for your application with additional attributes. You can select default icon, image, clipart, or any text as an icon. You can change the colour, shape and surroundings of the selected icon.
4. After setting the icon attributes, click on the next button. The “Create Activity” window will pop up on the screen which has different Activity templates. If you don’t want any Activity class in your code, uncheck the “Create Activity” check box and click on the next button. But for now, select the “Blank Activity” template that will create a blank Activity with action bar and click on next button.
5. When you click on the next button, the “Blank Activity” window will pop out on the screen (as shown in Figure 2.2). This window has two text fields: one is the name of the Activity and another is the name of the layout. A layout file is a XML file which contains the graphical layout of an activity.


New Android Application

Blank Activity

Creates a new blank activity with an action bar.

Activity Name: MainActivity

Layout Name: activity_main

The name of the activity class to create

Figure 2.2: Setting the name of Activity file and Layout file

Note it!

Name of the activity must start with a capital letter because it is going to be a class in a Java file (for example, MainActivity.java).

Name of the layout file must be in small case because it is going to be a resource file (for example, activity_name.xml).

6. Click on the Finish button. Now your project (with logic, design and all other support files and resources files) has been created. In Java Perspective, this will look as shown in Figure 2.3.

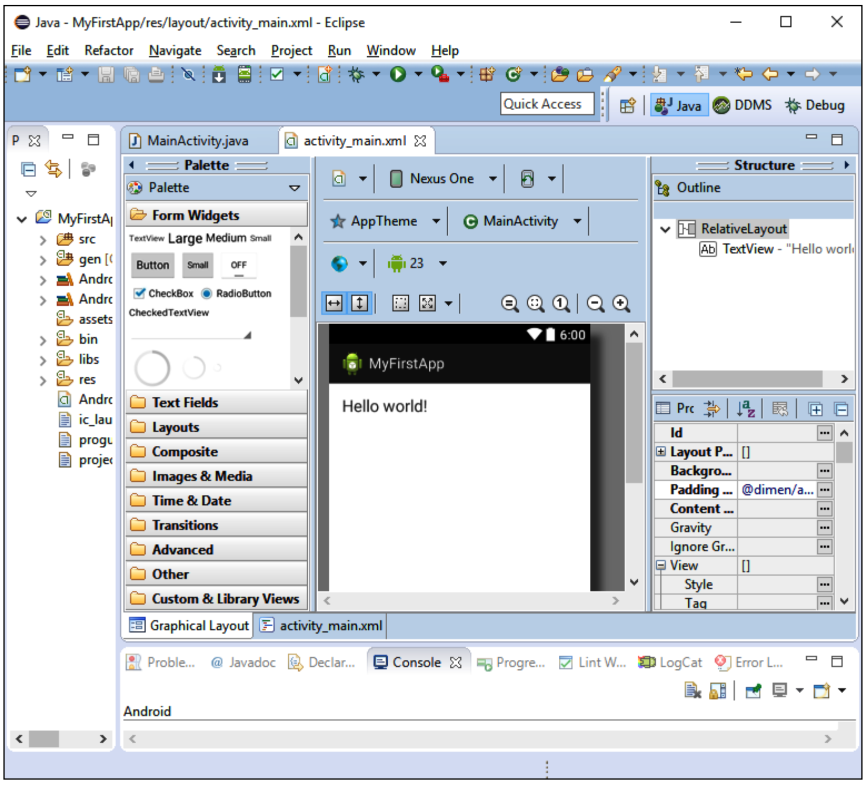


Figure 2.3: First look of the project in Java Perspective

After completing above mentioned steps, a new project will be created with all necessary files and a default activity displaying the text Hello world! Now you need to test this project (or default app) for multiple versions of Android.

3.2.2 Creating a new Android project in Android Studio

First thing that you must know is that in Android studio every project has its own separate instance i.e. each project has its own window of Android studio. So when you start Android Studio first time, it wouldn't allow you to enter without creating a new project (shown in Figure 1.10 of section 1.6.3 of UNIT 1). When you click on the "Start a new Android Studio project" option, you will be switched to the "New project" window as shown in Figure 2.5.

If you are creating a new project from while working on another project, then the steps are as follows:

1. Select the "New Project…" from "New" submenu of "File" menu as shown in Figure 2.4.

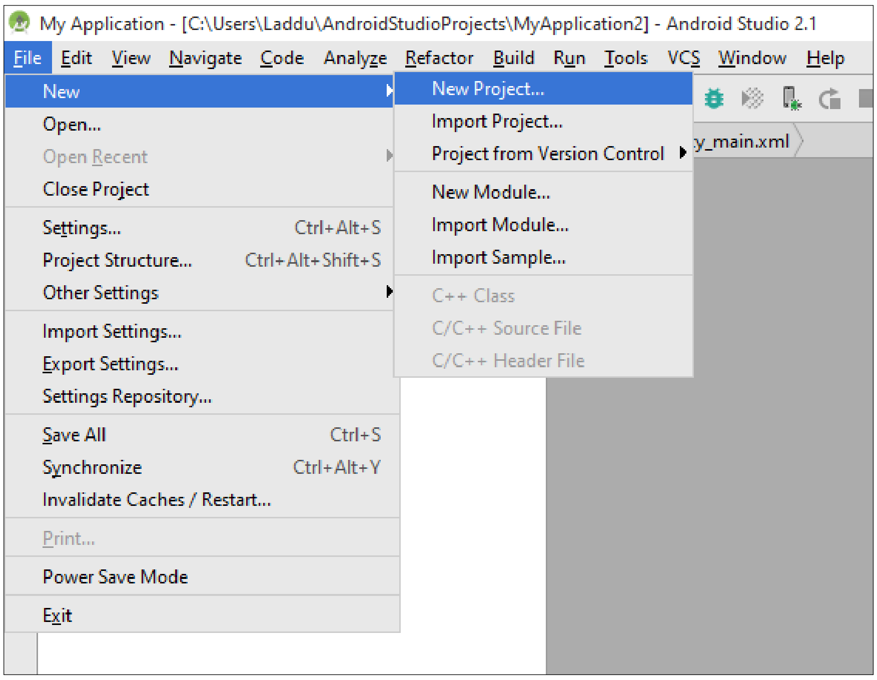
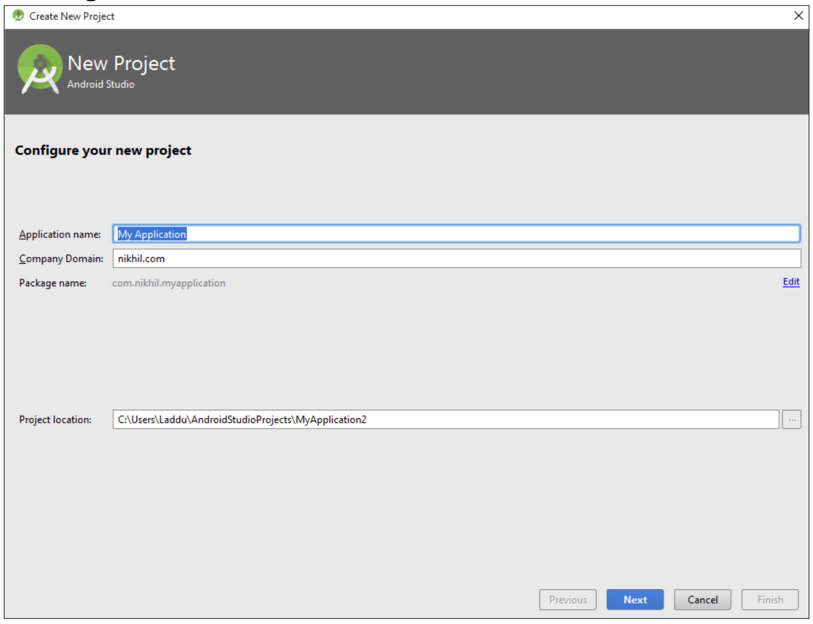


Figure 2.4: Creating a new project in Android Studio

2. First window for project setup will ask you to enter the name of the application and your company domain. Company domain name is used to assign the unique package name to your application because package name is going to identify your app in the application store uniquely. Here you will assign a project location (working directory) of your project where all the project work will be saved. It is shown in Figure 2.5.

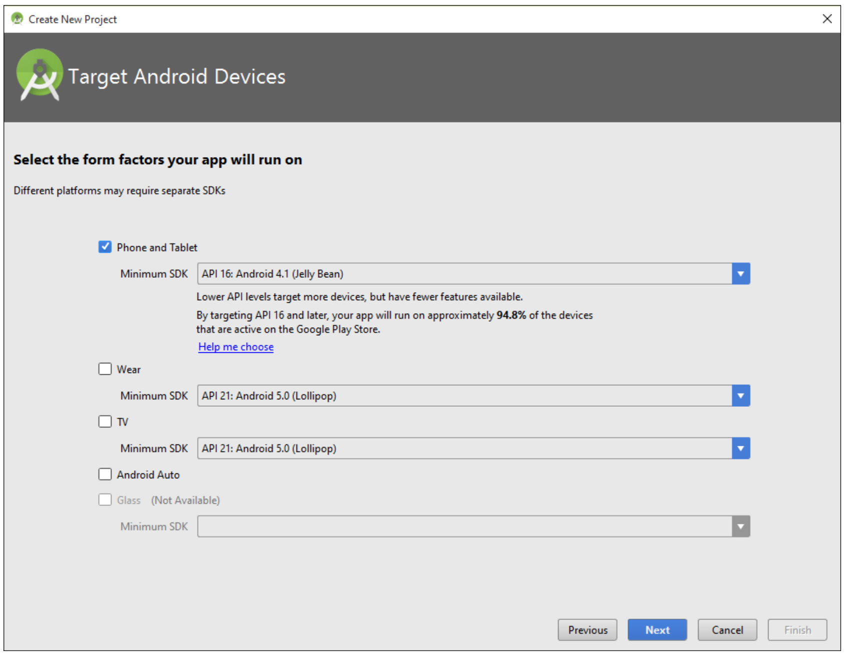


Configure your new project

Application name:	My Application
Company Domain:	nikhil.com
Package name:	com.nikhil.myapplication
Project location:	C:\Users\Laddu\AndroidStudioProjects\MyApplication2

Figure 2.5: Defining the name and the package of your new application

3. When you click on the next button, following window (as shown in Figure 2.6) will appear on the screen:


Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

- [x] Phone and Tablet
  Minimum SDK: API 16: Android 4.1 (Jelly Bean)
  Lower API levels target more devices, but have fewer features available.
  By targeting API 16 and later, your app will run on approximately **94.8%** of the devices that are active on the Google Play Store.
  Help me choose
- [ ] Wear
  Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] TV
  Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] Android Auto
- [ ] Glass (Not Available)
  Minimum SDK:

Figure 2.6: Selecting the platform and minimum SDK for your application

In this window (Figure 2.6), select the minimum SDK for targeted Android devices. Minimum SDK specifies the minimum Android version to which your application is going to support.

4. When you click on the next button, a window will pop up with different Activity templates. Select the appropriate Activity template and click on the next button (as shown in Figure 2.7)

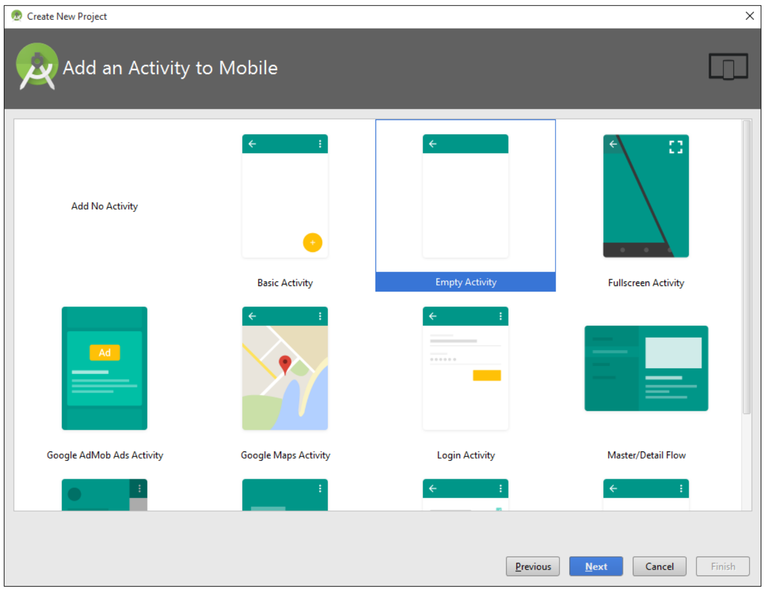
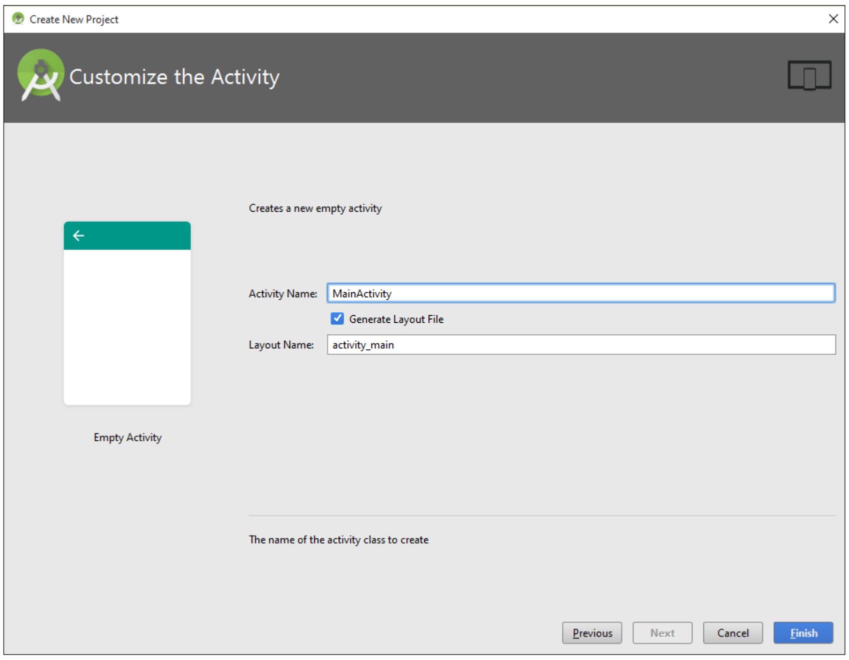


Figure 2.7: Selecting a default Activity template for your home screen

5. When you click on the next button, the “Customize the Activity” window will pop up. This will ask you to enter Activity file name (A Java file that is having the logic part of the Activity) and Layout

file name (An XML file that is having the design part of the Activity). It is shown in Figure 2.8.



Creates a new empty activity

Activity Name: MainActivity

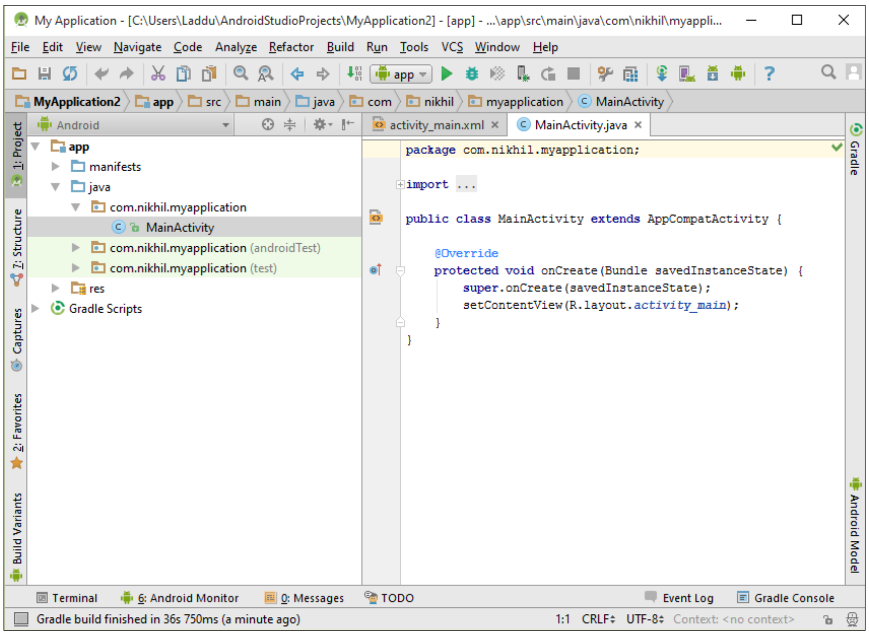
[✓] Generate Layout File

Layout Name: activity_main

The name of the activity class to create

Figure 2.8: Assigning the name to the Activity and its layout file

6. When you click on the finish button, a working window with text editor and different tools will be displaying on the screen (as shown in Figure 2.9).


My Application - [C:\Users\Laddu\AndroidStudioProjects\MyApplication2] - [app] - ...\app\src\main\java\com\nikhil\myappli...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication2 app src main java com nikhil myapplication MainActivity
Android
activity_main.xml
MainActivity.java
app
manifests
java
com.nikhil.myapplication
MainActivity
com.nikhil.myapplication (androidTest)
com.nikhil.myapplication (test)
res
Gradle Scripts
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model
Terminal
6: Android Monitor
0: Messages
TODO
Event Log
Gradle Console

```
package com.nikhil.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

Gradle build finished in 36s 750ms (a minute ago)   1:1 CRLF÷ UTF-8÷ Context: <no context>

Figure 2.9: working window of Android Studio

Video

This video will demonstrate step-by-step the process of creating your very first Android application in Google's Android Studio. What are different basic components of an Android application?

https://www.youtube.com/watch?v=uz25mf9T21I&index=4&list=PLXN-JCVb8z8BiyRhLO6HRNzfn5-TaTI_L

Testing of App on multiple physical devices at one place is not possible. Therefore, Android SDK provides an emulator to test your application against to all versions of Android. An emulator provides virtual environment to test your applications. It eliminates the requirement of a real physical device. This emulator uses an Android Virtual Device (AVD) to run, so it can be used only when an AVD has been created. Following sections explains how to create and configure an AVD.

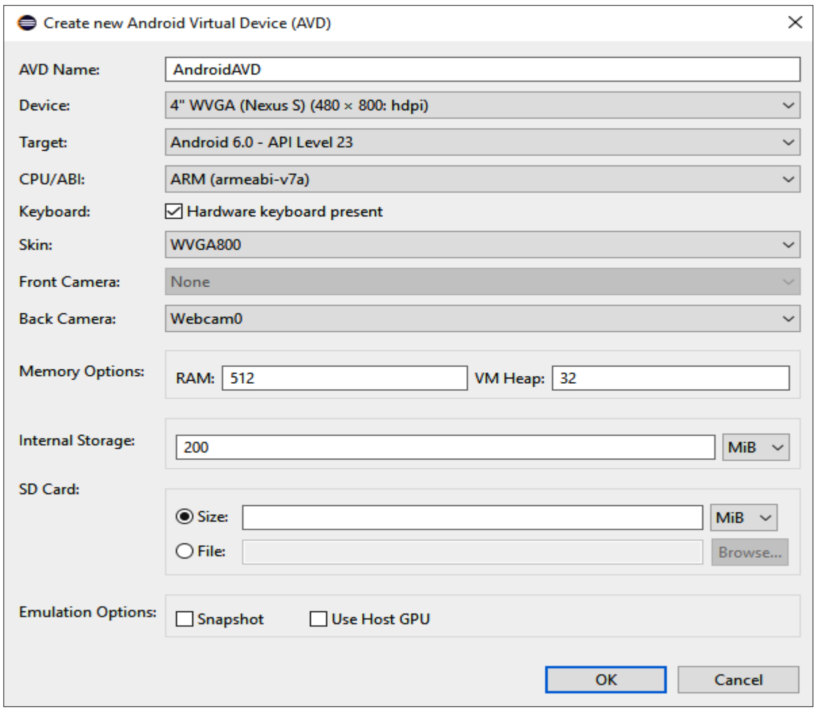
3.3 Creating Android Virtual Device

In this section, you will learn to create a new Android virtual Device in both IDEs (Eclipse and Android Studio).

3.3.1 Creating an AVD in Eclipse

You can create as many Android Virtual Devices as you want with different configuration. Attempt the following steps to create an AVD in Eclipse IDE:

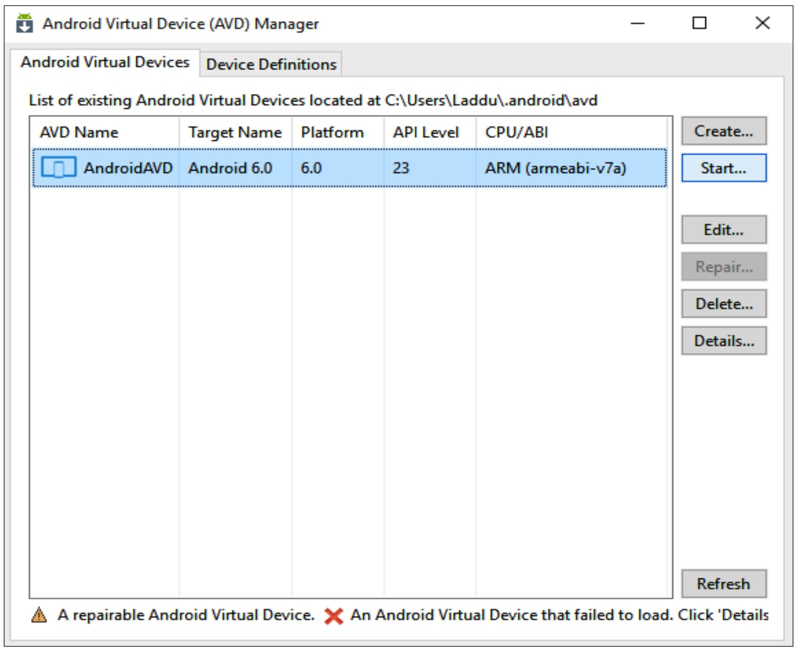
1. Launch the Eclipse and select "Android Virtual Device Manager" from "Window" menu. The "Android Virtual Device Manager" window will appear. If you have already created some AVD, that will be showed in the list. For creating a new AVD, there are two tabs in this window: "Android Virtual Devices" and "Device Definitions". In "Device Definitions" tab, a list of some known device definitions is available. You can select any one of these definitions to create an AVD. Otherwise, you can create a new customized AVD by clicking on the "Create..." button in "Android Virtual Devices" tab.
2. When you click on the "Create..." button, "Create New Android VirtualDevice" window will appear on the screen (as shown in Figure 2.10). In this window you will need to enter the details such as AVD name, device type, target API, etc.


Create new Android Virtual Device (AVD)

AVD Name:	AndroidAVD
Device:	4" WVGA (Nexus S) (480 × 800: hdpi)
Target:	Android 6.0 - API Level 23
CPU/ABI:	ARM (armeabi-v7a)
Keyboard:	☑ Hardware keyboard present
Skin:	WVGA800
Front Camera:	None
Back Camera:	Webcam0
Memory Options:	RAM: 512 VM Heap: 32
Internal Storage:	200 MiB
SD Card:	● Size: MiB ○ File: Browse...
Emulation Options:	☐ Snapshot ☐ Use Host GPU

Figure 2.10: Creating a New AVD

3. After filling all the details of AVD, click on the "OK" button to create a new AVD.Now your AVD will appear on the list of existing AVDs in the "Android Virtual Device Manager" window (as shown in Figure 2.11).


Android Virtual Device (AVD) Manager

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

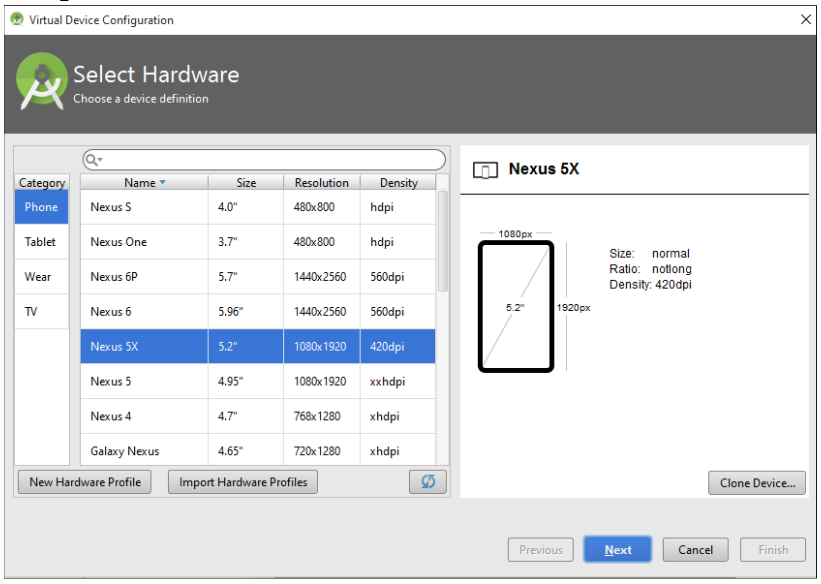
AVD Name	Target Name	Platform	API Level	CPU/ABI
AndroidAVD	Android 6.0	6.0	23	ARM (armeabi-v7a)

A repairable Android Virtual Device. ✗ An Android Virtual Device that failed to load. Click 'Details

Figure 2.11: List of Existing AVDs in AVD Manager Window

3.3.2 Creating an AVD in Android Studio

1. To create an AVD in Android Studio, Select the "AVD Manager" option in the "Android" submenu of "Tools" menu. An AVD Manager window will pop up. If you have already created some AVD, then it will be show in the list of AVDs. To create a new AVD click on the "Create New AVD" button. After clicking on the button, the "Virtual Device Configuration" window will pop up on the screen as show in Figure 2.12.



Category
Phone
Tablet
Wear
TV

Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6P	5.7"	1440x2560	560dpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5X	5.2"	1080x1920	420dpi
Nexus 5	4.95"	1080x1920	xxhdpi
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi

Nexus 5X

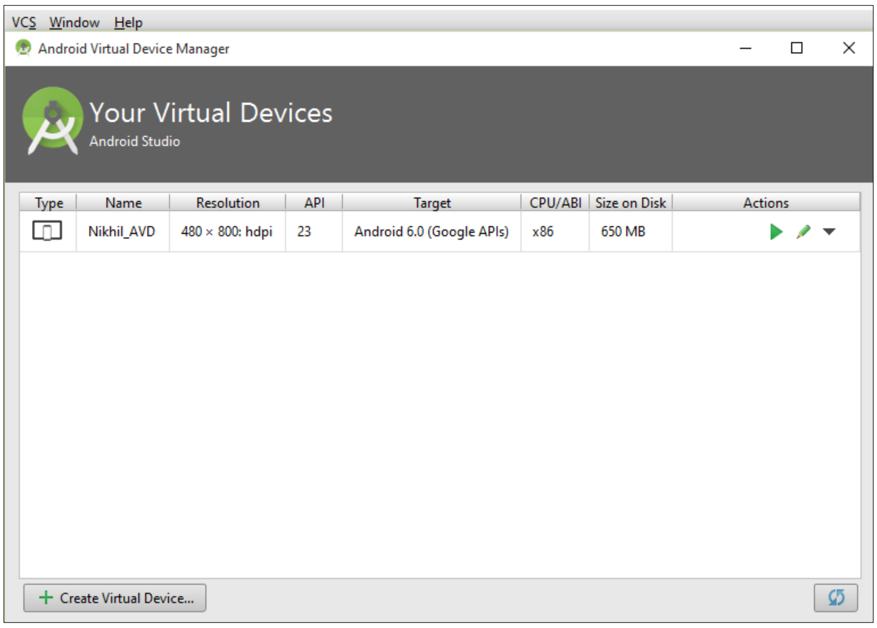
1080px
5.2"
1920px

Size: normal
Ratio: notlong
Density: 420dpi

Figure 2.12: Choosing device definition

2. Few definitions of some standard Android devices are listed here, as shown in the Figure 2.12. Either select any standard definition or create your own definition by clicking on the "New Hardware Profile" button. If you are creating your own hardware profile, then "Configure Hardware Profile" window will pop up where you can fill all necessary details of your device definition and provide a name to the definition. After setting the configuration click on the "Finish" button and then "Next" button in the next window ("Select a system image" window). This leads to a new window "Verify Configuration" where you should provide the name to the created AVD (for example NIK_AVD). Here you can change some attributes of AVD such that Android Version, resolution and orientation of the screen, etc. and complete the job by selecting the "Finish" button.

3. Now your created definition will be available in the list of AVDs in AVD manager window as shown in Figure 2.13.



Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nikhil_AVD	480 × 800: hdpi	23	Android 6.0 (Google APIs)	x86	650 MB	

Figure 2.13: AVD Manager is showing the created AVD

AVD has been created. The upcoming sections will discuss the process to run the app in the created AVD.

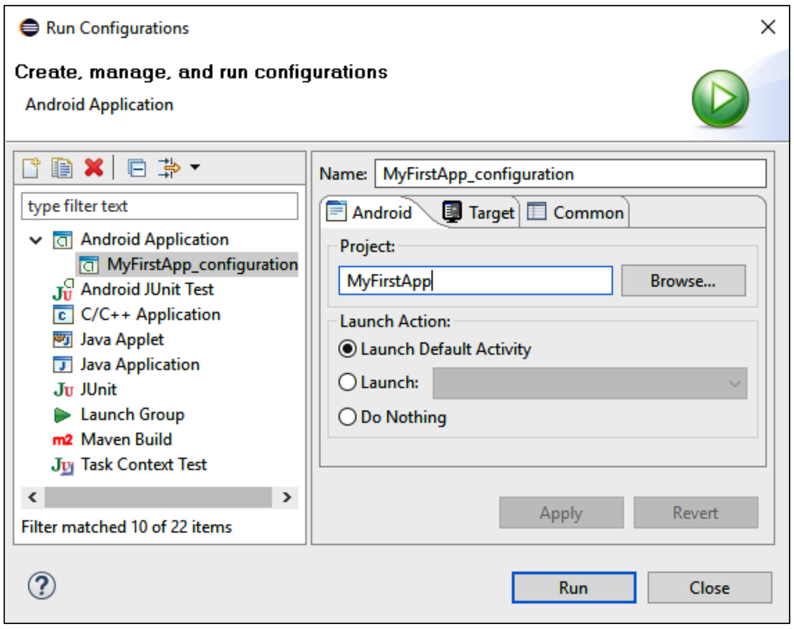
3.4 Creating and saving launch configuration

Whenever an app is created, you need to run and test the app against an AVD repeatedly by selecting the AVD on each launch. This extra overhead of selecting the AVD at each launch can be eliminated by tying up a specific AVD to a project. For this purpose, you need to save the launch configuration for the project (or application).

3.4.1 Creating and saving launch configuration in Eclipse

For creating and saving the launch configuration for a project in Eclipse, pursue the following steps:

1. In Eclipse, select "Run Configuration..." from the "Run" menu. After this, a "Run Configurations" window will open.
2. Now right click on the "Android Application" option and select the new option(or click on the new button). Type the name of the running configuration and, browse and select the project as shown in Figure 2.14.



Name: MyFirstApp_configuration

Project: MyFirstApp

Launch Action:

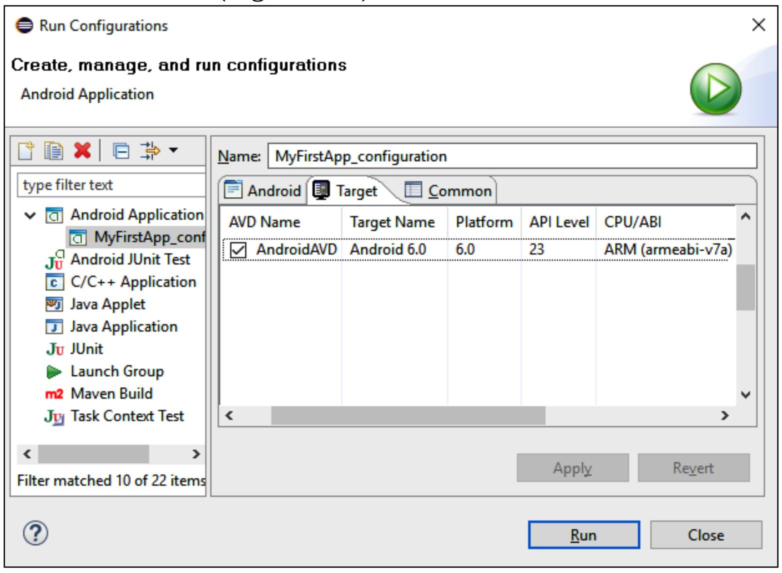
[✓ Launch Default Activity]

Launch:

Do Nothing

Figure 2.14 Manage Run Configuration in Eclipse

3. Now click on the target tab and select the appropriate AVD from available AVDs. (Figure 2.15)



Name: MyFirstApp_configuration

AVD Name	Target Name	Platform	API Level	CPU/ABI
☑ AndroidAVD	Android 6.0	6.0	23	ARM (armeabi-v7a)

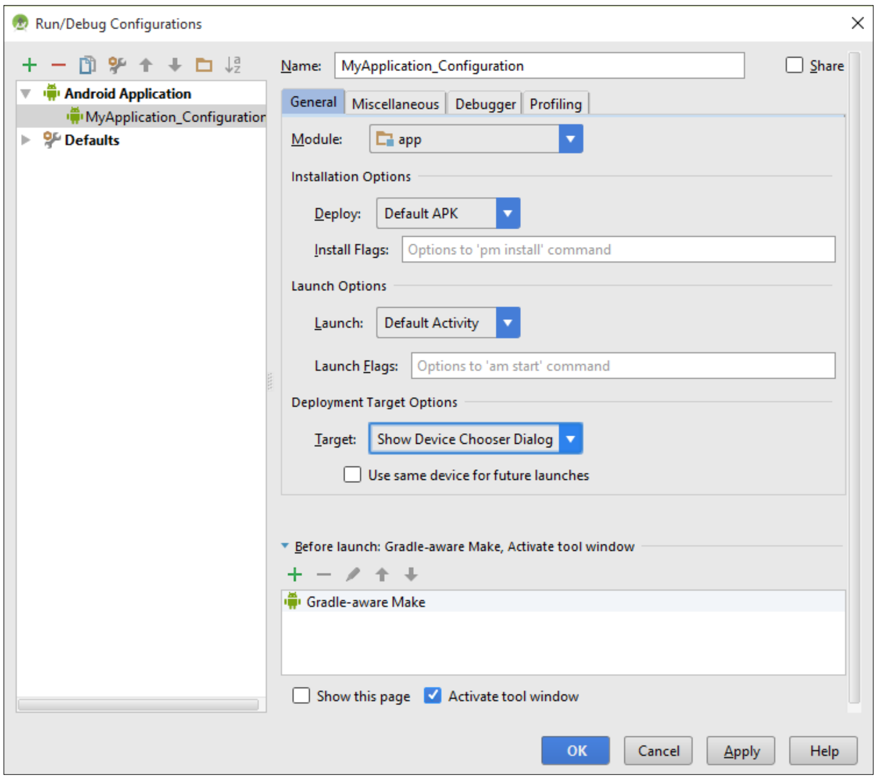
Figure 2.15 Selecting Compatible Device in Eclipse

4. On the click of the“Apply” button, the launch configuration willbe created. You can run the emulator by clicking on the “Run” button.

3.4.2 Creating and saving the launch configuration in Android Studio

For creating and saving the launch configuration for a project in Android Studio, pursue the following steps:

1. In Android studio, when you select the “Edit Configuration…” option from the “Run” menu, a “Run/Debug Configuration” window will pop up on the screen as shown in Figure 2.16.


Run/Debug Configurations

Name: MyApplication_Configuration
Share

Android Application
MyApplication_Configuration
Defaults

General | Miscellaneous | Debugger | Profiling

Module: app

Installation Options

Deploy: Default APK

Install Flags: Options to 'pm install' command

Launch Options

Launch: Default Activity

Launch Flags: Options to 'am start' command

Deployment Target Options

Target: Show Device Chooser Dialog

Use same device for future launches

Before launch: Gradle-aware Make, Activate tool window

Gradle-aware Make

Show this page ✓ Activate tool window

Figure 2.16: Setting the Run and Debug configuration in Android Studio

2. Click on the plus sign (+) button (window shown in Figure 2.16), to add new run configuration. Now assign new name to the run configuration, select the module (Application Name) to attach with the configuration, select the default launch Activity, select the default device to run and test the application, etc.
3. Now click on the “Apply” and “OK” button respectively. A new default launch configuration for the project in Android Studio will be created.

3.5 Running and debugging the app

After launch configuration has been created, in order to run the project, AVD should be launched either from Run configuration window or from AVD manager. It can take some time to launch the AVD as per the speed and configuration of your PC. You can also run the current project by selecting the “Run” option present in both IDEs whether it is Eclipse or Android Studio. In Eclipse, the running emulator will looks as shown in the Figure 2.17.

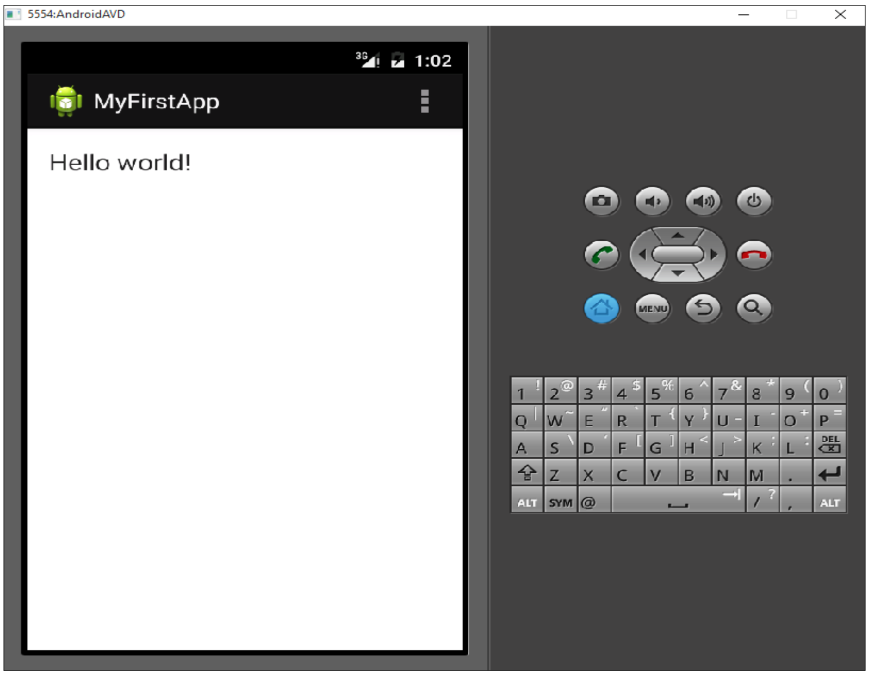


Figure 2.17 Running Android Emulator in Eclipse

While in Android Studio, the running emulator will look as show in Figure 2.18.

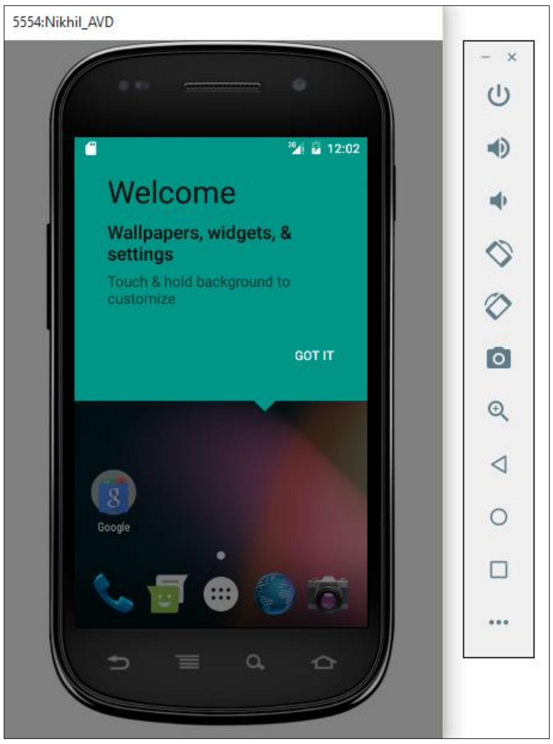

5554:Nikhil_AVD
12:02
Welcome
Wallpapers, widgets, & settings
Touch & hold background to customize
GOT IT
Google

Figure 2.18: Running Android Emulator in Android Studio

Video

This video will show you how to create an Android Virtual Device (AVD), save the run configuration and run the app in the created AVD. What is the role of an Android emulator?

https://www.youtube.com/watch?v=UNV1PvrcgmQ&index=5&list=PLXN-JCVb8z8BiyRhLO6HRNzfn5-TaTI_L

Unit summary

Summary

In this unit you learned about creating a new Android project in Eclipse and Android Studio IDEs. You have also been trained about creating the Android Virtual Devices to run, test and debug your application. Configuring and saving the launch configuration and associating an AVD with your project is also going to make your debugging and testing task easier. You also learnt about different project files that are created during the development process of an Android application. You also got an idea about different development tools, Android application components and adding permissions to your application.

Assignment

Assignment

Q 1: What is the difference between “Minimum Required SDK”, “Target SDK” and “Compile With”.

Answer:

*Minimum Required SDK:*The new versions of Android often provide best APIs for your application but you must continue to support previous versions of Android until older version devices get updated. So, during development process select the lowest version from the drop down list. As lower the version you select, more the devices it supports.

Target SDK: Select the highest API level that the application is going to work with. Since the application does not have forward compatibility issue for target API level. It is recommended to set highest level of API for Target SDK. Your application can still work with older versions up to Minimum SDK level that is set by you.

Compile With: You should select the most recent version of installed SDK to compile the project.

Q 2: Explain the role of R.java file.

Answer:

Android recommends that the logic part should be written in Java and design part should be written in XML, as by using these different technologies it gets easier to manage layouts, views, and other resources separately. The graphical components coded in XML are going to be used in Java code. But, how can it be done? The answer is – by using R.java. R.java is a Java file that is created automatically by Android development environment during the programming. It works as a bridge between logic part and design part. It interprets the references of UI components of XML file to the Java file and to other XML files. It is recommended not make any change in the file because you can alter the reference of any graphical component by mistake.

Q 3: Explain the role of AndroiManifest.xml file in an Android application.

Answer: AndroidManifest.xml file plays a very important role in Android development. This is the heart of any android application. It has all details about the application including package of the app, version of the app, minimum and maximum Android SDK versions supported by the app, themes, permissions, activities, intents, broadcast receivers, content providers and much more.

Q 4: What are different basic components of an Android application?

Answer: Building blocks or application components of an Android app defines the behaviour of the apps. Each Android application has four types of components; each component plays a specific role and has its own lifecycle. Each component is described below.

An Activity is a user interface provided as a screen with which a user interacts. For example, if you open a media player app, then the screen that allows you to play, pause, next or previous the song is an Activity. An Activity can either be fit into the whole screen or can float over other screens. An Android application can have multiple Activities linked to each other. The very first Activity that renders on the screen after opening the app is called main Activity (or default Activity).

A service is an application component that runs in the background and does not need any user interface. For example, if you are using a web browser and decide to download a file, then file would be downloaded in the background while you continue your interaction with the web browser. This is being done due to an application component called service. An Activity can start a service. You can also bind a service to an Activity to interact with it.

A content provider is a component that manages the application data stored in SQLite database, file system, on the web or any other storage location accessed by your application. Using the content provider other apps can access or modify the data. For example, Android system can provide the contact information to the messaging app through the content provider. It is also helpful in managing and querying the private data of the app.

A broadcast receiver is used to broadcast the announcements to whole system whether it is notification to status bar to alert the user or it is a message to another application or Activity. For example, if you receive a notification on status bar about completion of download receiving a text message or about the new update of the app, it happens because of the broadcast receivers. Even, a broadcast receiver can start a service on occurrence of a predefined event.

Q 5: What is the role of an Android emulator?

Answer: The Android Emulator is a virtual mobile device that runs in your computer. The emulator lets you run, test and debug your application. It provides you a mobile platform which does not need any real mobile hardware. You can connect to the internet; can make a call and SMS with the help of Android Emulator. It is a DVM implementation that needs an AVD instance created by the AVD Manager to show all above mentioned features.

Assessment

Assessment

Problem 1: Why do you manage the logic part and the design part separately? Which type of files contains the logic part and design part?

Problem 2: What is an .APK file? What are the ingredients of it? How is it created? Explain.

Problem 3: Explore and explain the contents of “res” folder.

Problem 4: What are the permissions?How can you add permissions to a Manifest file?

Problem 5: Why do you need to save the launch configuration for an Android project?

Online Test

Study Skills

Use the following URL to attempt the online quiz to test your skills after completing the unit 3:

http://tinyurl.com/colandroidtest2