Unit 2- Installation and configuration of Android

Course Outcome:

Configure Android environment and development tools.

Unit Outcomes:

2a. Describe function of the given component to operate the specified IDE.
2b. Explain the given term related to virtual machine.
2c. Explain the given basic term related to Android development tools.
2d. Describe the features of given android emulator.
2e. Describe the steps to configure the given android development environment

---

Contents:

2.1 Operating System, Java JDK, Android SDK

Operating System

- A mobile OS is an operating system for smartphones, tablets, PDAs, or other mobile devices.
- Mobile OSs combine features of a personal computer OS with other features useful for mobile or handheld use; usually including, and most of the following considered essential in modern mobile systems;

Touchscreen, cellular, Bluetooth, Wi-Fi, GPS mobile navigation, camera, video camera, speech recognition, voice recorder, music player, etc

Java JDK

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent). JVM is a part of Java Run Environment (JRE).

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JRE

The Java Runtime Environment (JRE) is a set of software tools for development of Java applications. It combines the Java Virtual Machine (JVM), platform core classes and supporting libraries.

Android SDK

Android development starts with the Android SDK (Software Development Kit). It is a software development kit that enables developers to create applications for the Android platform.

The Android SDK (software development kit) is a set of development tools used to develop applications for Android platform. The Android SDK includes the following:

- Required libraries
- Debugger
- An emulator
- Relevant documentation for the Android application program interfaces (APIs)
- Sample source code
- Tutorials for the Android OS

2.2 Android Development Tools (ADT)

1. Android Studio

Developed by Google, Android Studio is an all-rounder integrated development environment that allows the Android developers to get what they desire without an Integrated Development Environment or IDE.

Android has Gradle-base support that has features like visual layout editor, intelligent code editor, real-time profilers and APK analyzer. It acts just like any other Java IDE in terms of error investigating and file hierarchy.

2. Visual Studio – Xamarin

Xamarin was launched in 2011 which is the best free IDE for delivering an enterprise-quality, cross-platform approach. **Xamarin** supplies add-ins to Microsoft **Visual Studio** that allows developers to build **Android**, iOS, and Windows apps within the IDE

3. IntelliJ IDEA

The framework based assistance, productivity boosters, unobtrusive intelligence, duplicates, and inspections are provided with the IDE. Using this IDE, you can do in-depth coding, quick navigation, and error analysis. It supports mobile app development with the help of Java, Scala, Kotlin, Groovy.

4. Eclipse IDE

It is one of the most popular IDES of Android apps. The open-source software is free to use. Released under the Eclipse Public License, it holds a large community having plenty of plugins and configurations. Highly customizable offers full support for Java programming language and XML.

Android Development IDEs	Languages	Target OS	Runs On	Audience	License	Price
Android Studio	Java C C++ Kotlin	Android	Windows MacOS Linux	Experienced	Freeware	Free
Eclipse	Java C C++ C# JavaScript Python more	Android iOS Linux MacOS Windows	Any OS supporting Java	Professional Developers	Eclipse Public License	Free
Visual Studio (with Xamarin)	C++ C C# Visual Basic PHP JavaScript more	Cross-Platform Windows Android iOS more	Windows MacOS Linux	Experienced	Proprietary, Visual Studio Code is Open Source MIT	Free to $2,999+
IntelligJ	Java	Any OS	Windows	Professional	Proprieta	Free to

IDEA	Scala Groovy Kotlin JavaScript TypeScript SQL	supporting Java	MacOS Linux	Java Developers	ry, Community Edition is Apache 2.0 License	$499/year
NetBeans	Java C C++ HTML PHP JavaScript others	Cross-platform	Windows MacOS Linux Solaris	Professional Developers	CDDL 1.0 and GPL2	Free
Komodo	Java JavaScript Python PHP HTML Ruby others	Cross-platform	Windows MacOS Linux	Professional Web and mobile developers	Proprietary, Komodo Edit is Mozilla Public License	Free to $394+
Cordova	HTML CSS JavaScript	Cross-platform Android Windows iOS MacOS Ubuntu	Windows MacOS Linux	Experienced Web developers	Apache 2.0 License	Free
PhoneGap	HTML CSS JavaScript	Cross-platform Android iOS	Windows MacOS Linux Android Windows Phone	Web developers	Apache 2.0 License	Free
App Inventor	Kawa	Android	Windows MacOS Linux	Students and amateurs	MIT License	Free
AIDE	Java C C++ XML HTML CSS JavaScript	Android Web	Android	Amateurs or mobile professionals	Proprietary	Free with in-app purchases

2.3 Android Virtual Devices (AVDs)

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to

simulate in the Android Emulator. The AVD Manager is an interface you can launch from Android Studio that helps you create and manage AVDs.

To open the AVD Manager, do one of the following:

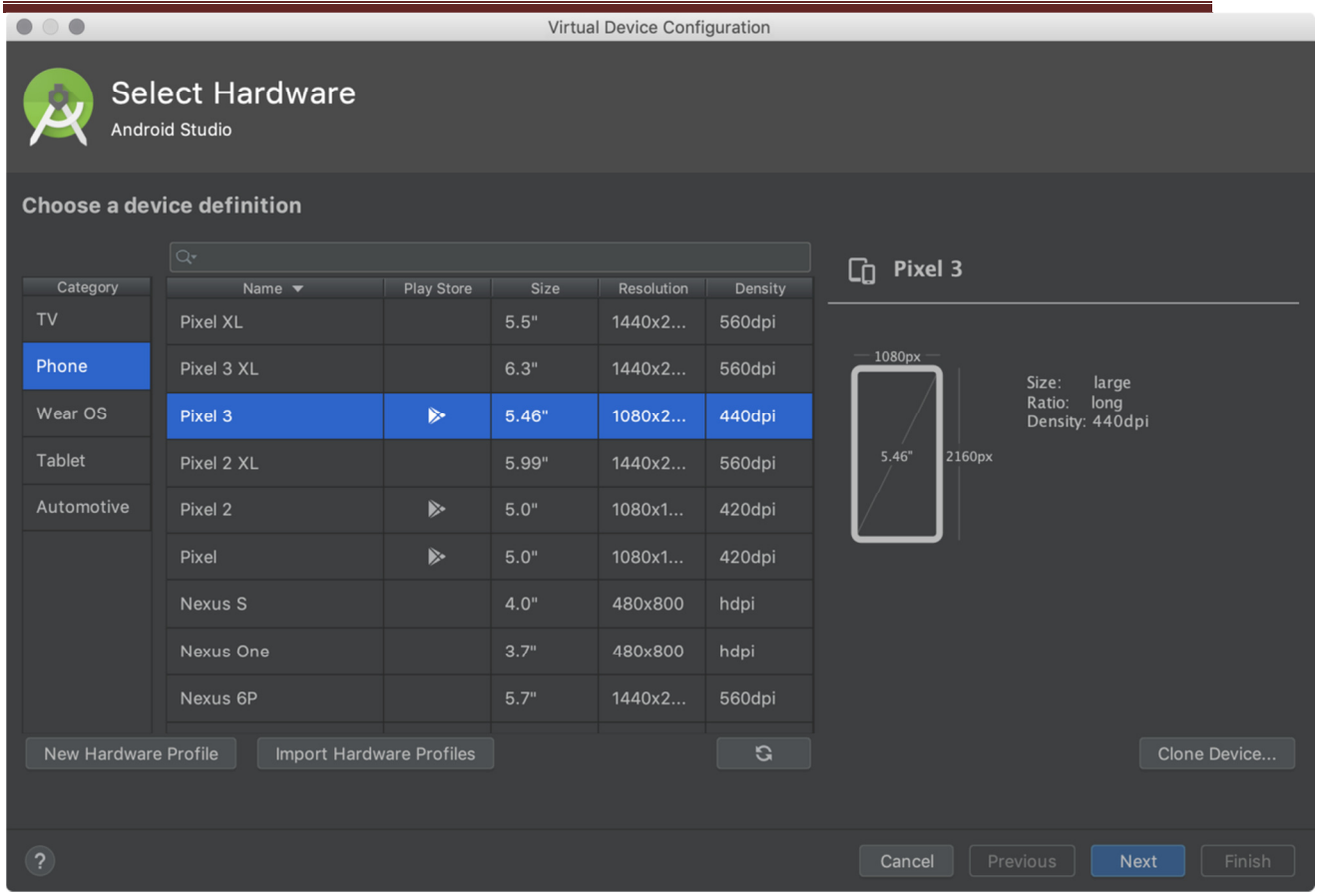
- Select **Tools > AVD Manager**.
- Click **AVD Manager**  in the toolbar.

To create a new AVD:

1. Open the AVD Manager by clicking **Tools > AVD Manager**.

2. Click **Create Virtual Device**, at the bottom of the AVD Manager dialog.

   The **Select Hardware** page appears.

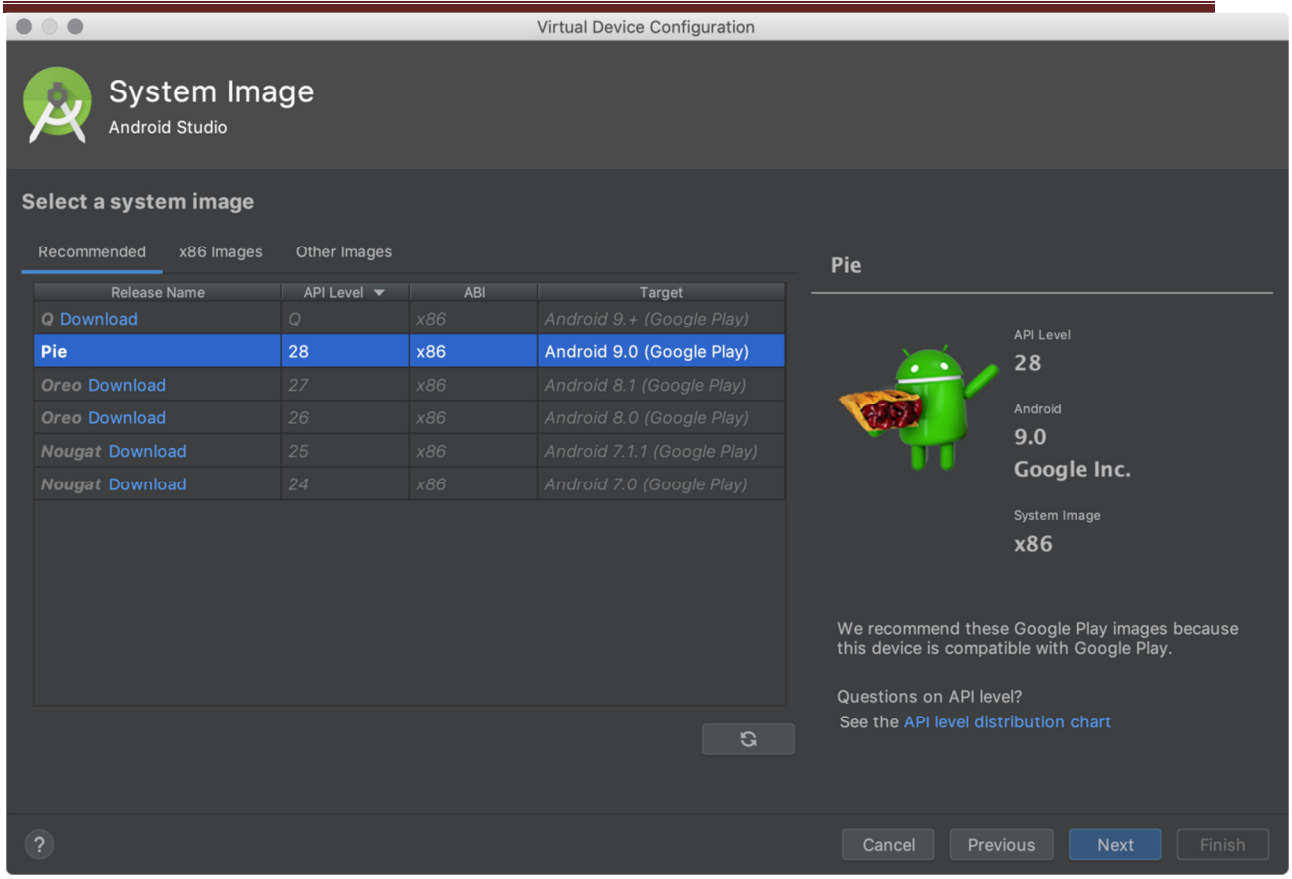


Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully CTS compliant and may use system images that include the Play Store app.

3. Select a hardware profile, and then click **Next**.

   If you don't see the hardware profile you want, you can create or import a hardware profile.

   The **System Image** page appears.



4. Select the system image for a particular API level, and then click **Next**.

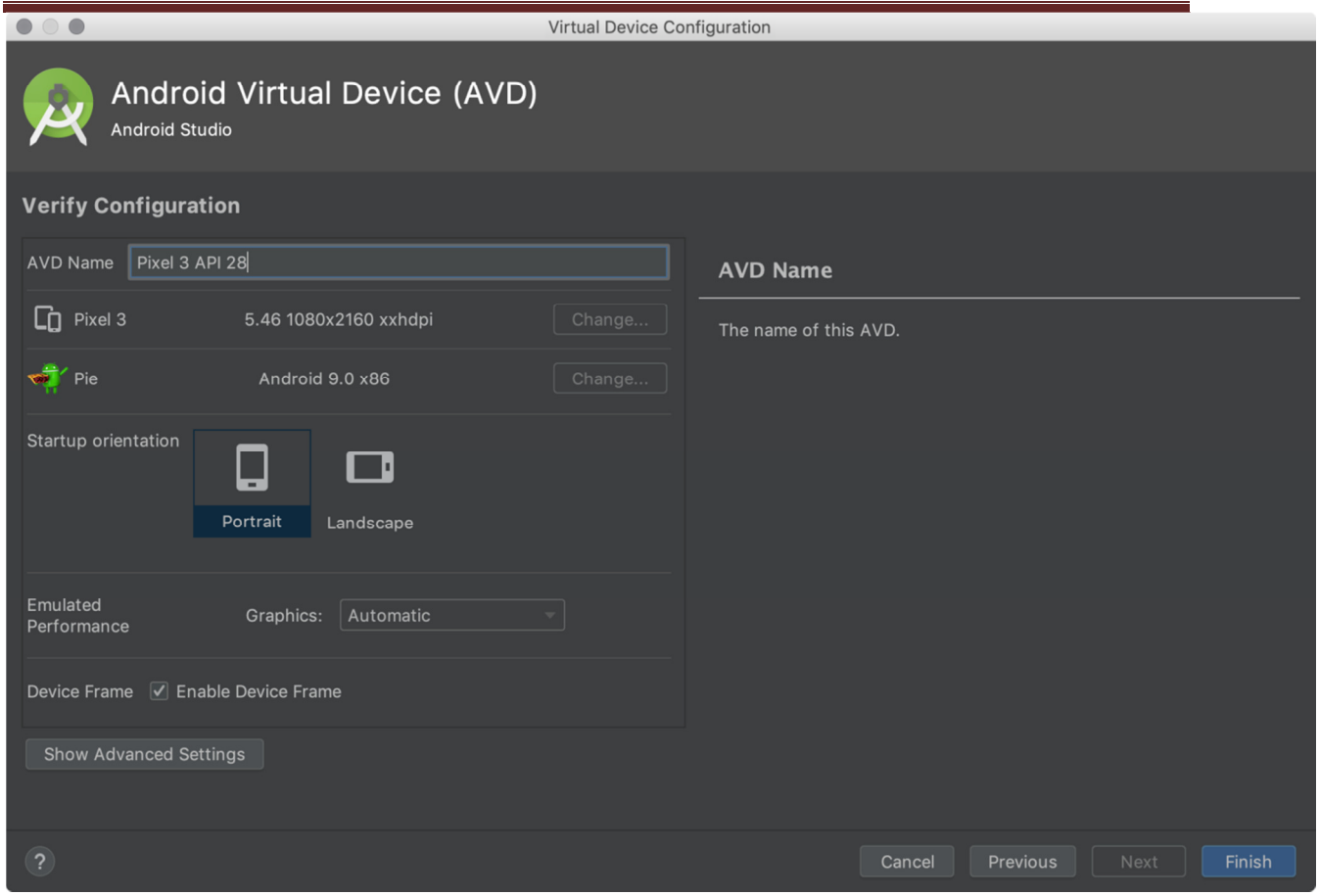
   The **Recommended** tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator.

   If you see **Download** next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

   The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the `minSdkVersion` attribute of the app manifest file. For more information about the relationship between system API level and `minSdkVersion`, see Versioning Your Apps.

   If your app declares a `<uses-library>` element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

   The **Verify Configuration** page appears.



Change AVD properties as needed, and then click **Finish**.

Now you get a new AVD ready for launching your apps on it.

2.4 Emulators

The Android Emulator simulates Android devices on your computer so that you can test your application on a variety of devices and Android API levels without needing to have each physical device.

The emulator provides almost all of the capabilities of a real Android device. You can simulate incoming phone calls and text messages, specify the location of the device, simulate different network speeds, simulate rotation and other hardware sensors, access the Google Play Store, and much more.

Testing your app on the emulator is in some ways faster and easier than doing so on a physical device. For example, you can transfer data faster to the emulator than to a device connected over USB.

The emulator comes with predefined configurations for various Android phone, tablet, Wear OS, and Android TV devices.

In short, An **Android emulator** is an **Android** Virtual Device (AVD) that represents a specific **Android** device. You can use an **Android emulator** as a target platform to run and test your **Android** applications on your PC. Using **Android emulators** is optional.

Launch the Android Emulator without first running an app

To start the emulator:

1. Open the AVD Manager.
2. Double-click an AVD, or click **Run**

The Android Emulator loads.

While the emulator is running, you can run Android Studio projects and choose the emulator as the target device. You can also drag one or more APKs onto the emulator to install them, and then run them.

2.5 Dalvik Virtual Machine (DVM)

As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for *memory*, *battery life* and *performance*.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

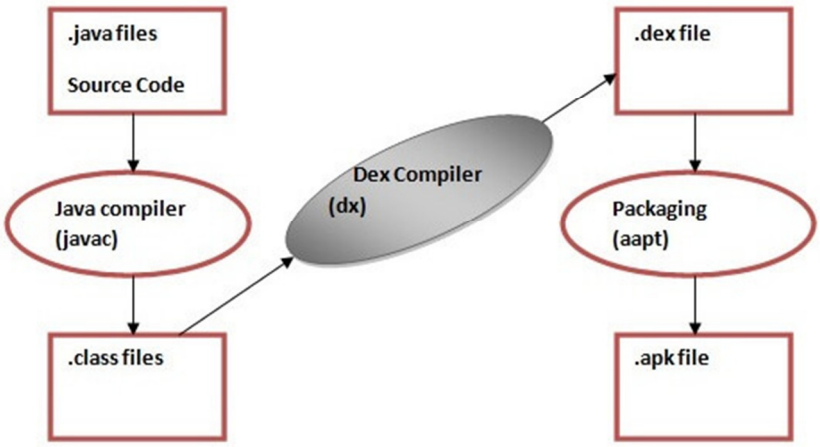


Fig.: The compiling and packaging process from the source file

The **javac tool** compiles the java source file into the class file.

The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

The **Android Assets Packaging Tool (aapt)** handles the packaging process.

Difference between JVM and DVM

DVM (Dalvik Virtual Machine)	JVM (Java Virtual Machine)
It is Register based which is designed to run on low memory.	It is Stack based.
DVM uses its own byte code and runs ".Dex" file. From Android 2.2 SDK Dalvik has got a Just in Time compiler	JVM uses java byte code and runs ".class" file having JIT (Just In Time).
DVM has been designed so that a device can run multiple instances of the VM efficiently. Applications are given their own instance.	Single instance of JVM is shared with multiple applications.
DVM supports Android operating system only.	JVM supports multiple operating systems.
For DVM very few Re-tools are available.	For JVM many Re-tools are available.
There is constant pool for every application.	It has constant pool for every class.
Here the executable is APK.	Here the executable is JAR.

2.6 Steps to install and configure Android Studio and SDK

Installation
Follow steps below for complete installation and configuration of Android Studio.

Step 1) Download Android Studio
You can download Android Studio from this link or go to developer.android.com homepage and search for downloads. Choose appropriate platform either for windows, mac or linux. Following are the pre requirements for windows operating system.

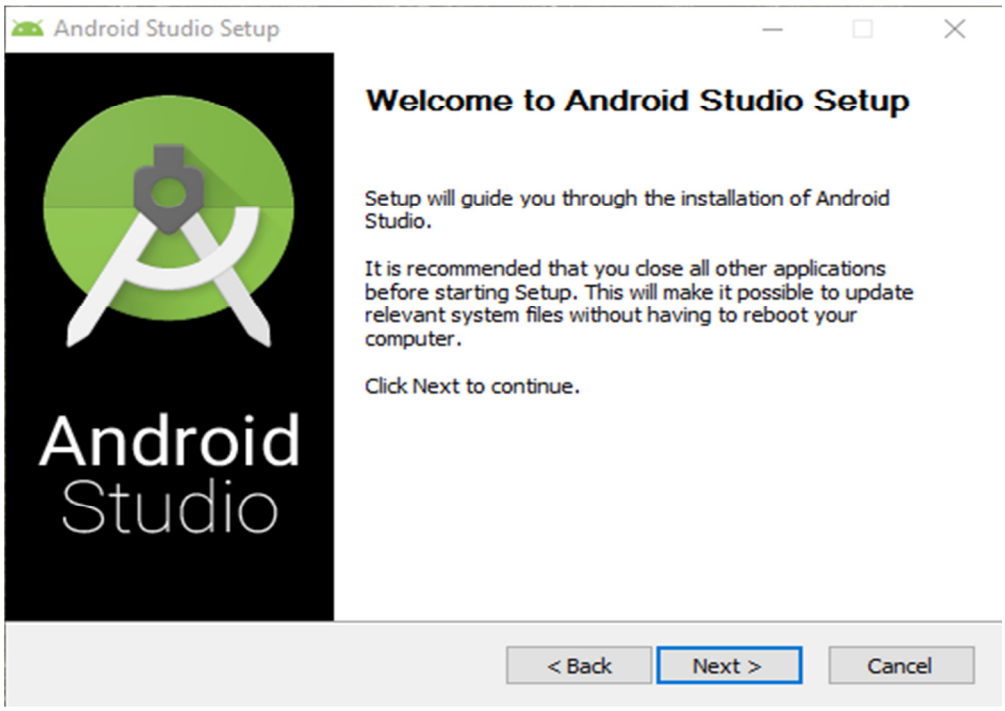
Pre-requirements

- Microsoft windows 7/8/10 (32 or 64 bits)
- Minimum 3GB RAM (recommended 8GB)
- 2GB disk space
- 1280 x 800 minimum screen resolution size
- Intel processor for accelerated emulator
- Android SDK

Note: If you don't have Android SDK, you can download with Android studio. Go to the end of download's page and find android-studio-bundle-162.4069837-windows.exe it includes SDK also.

Step 2) Run .exe file

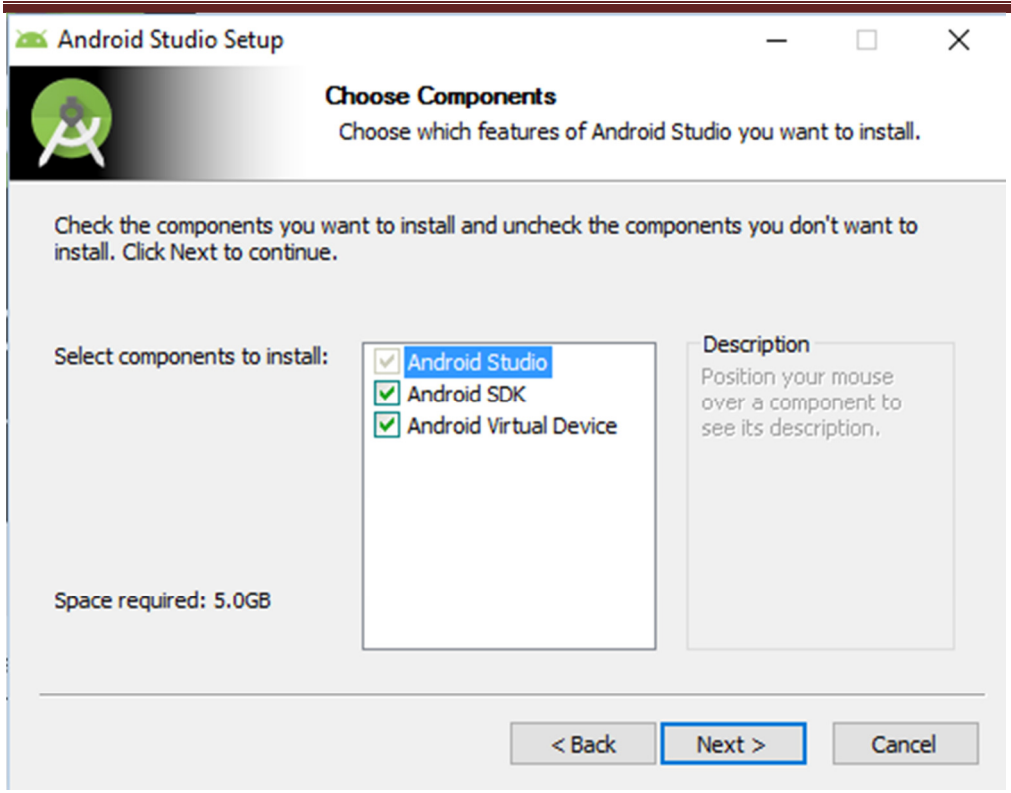
Now the next step is to launch .exe file you just download. Following screen will appear



Step 1: Run .exe file

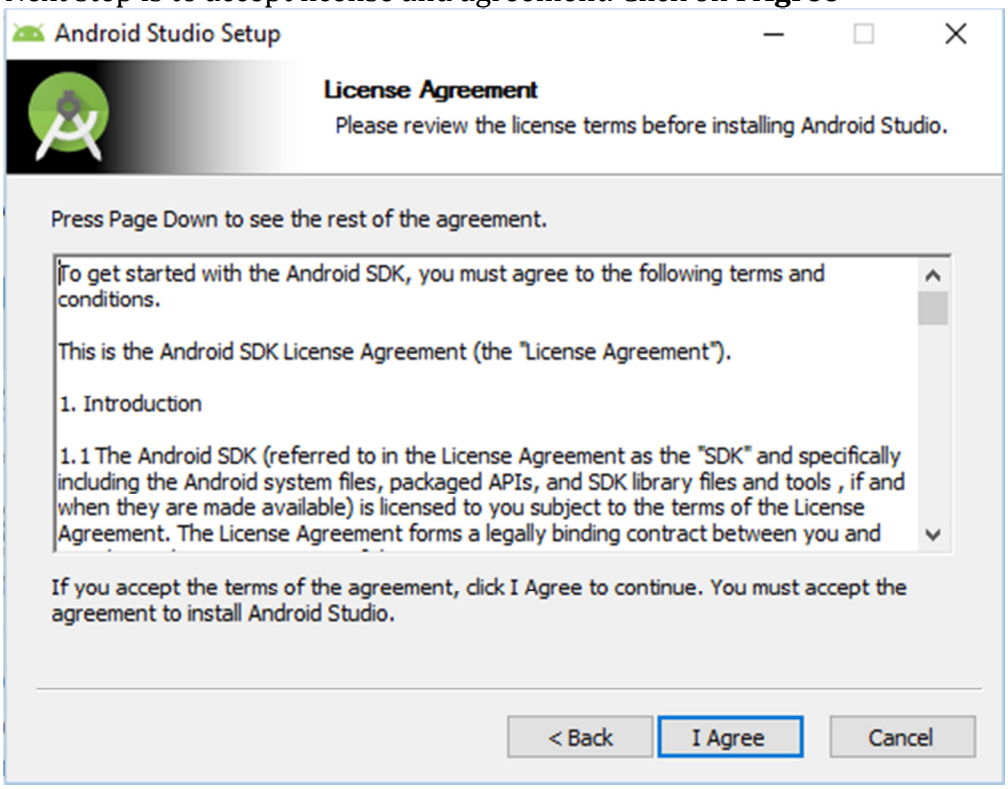
Click next and select Android SDK checked if you don't have it already. Better is to leave the default settings.

Make sure Android virtual device is also checked.



Step 2: Choose components

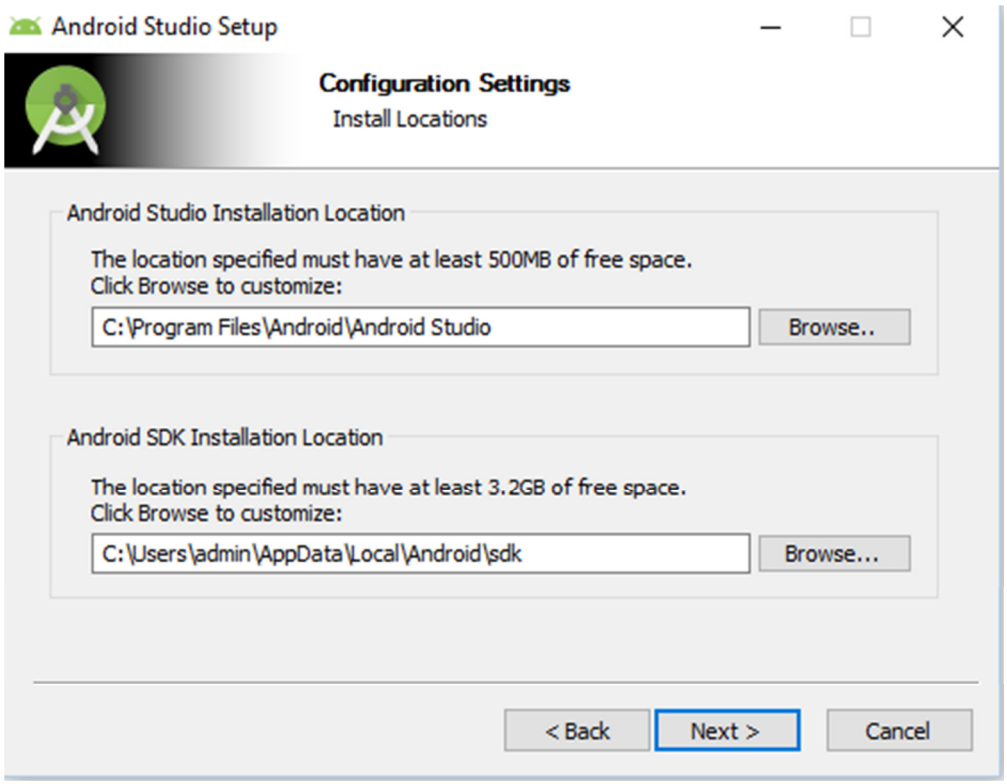
Next step is to accept license and agreement. Click on **I Agree**



Step 3: Accept license

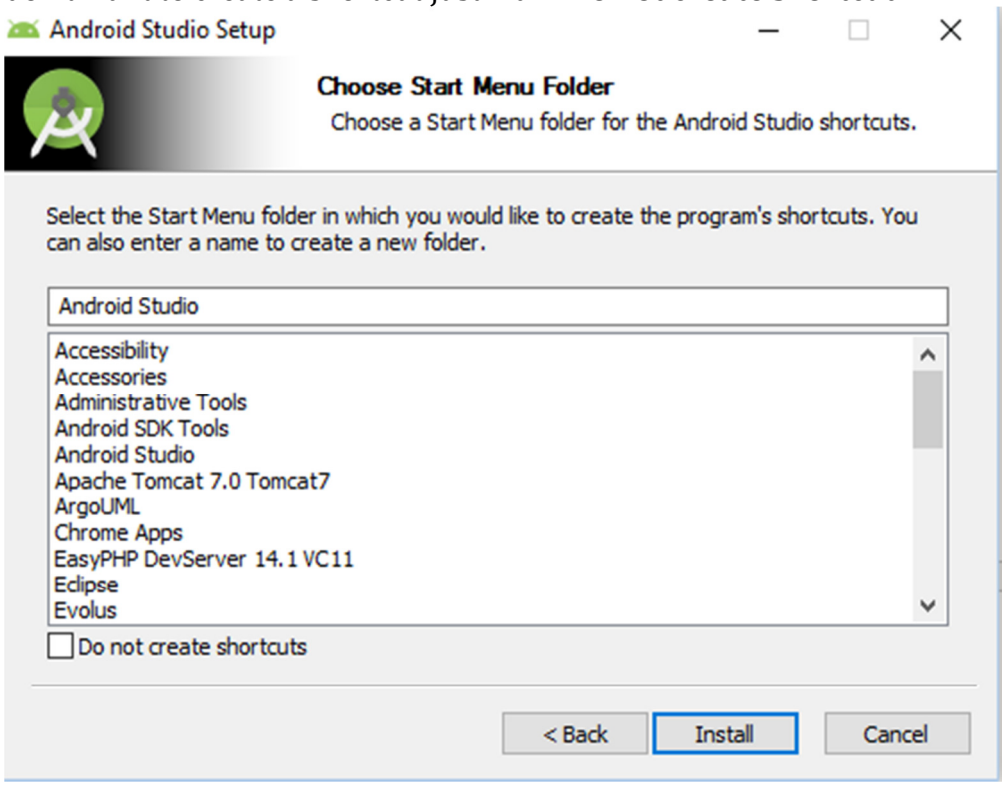
Next step is to set location of installation. Please make sure your disk has minimum required space before clicking on Next. For Android Studio installation location must

have at least 500MB free space. For Android SDK installation, selected location must have at least 3.25GB free space.



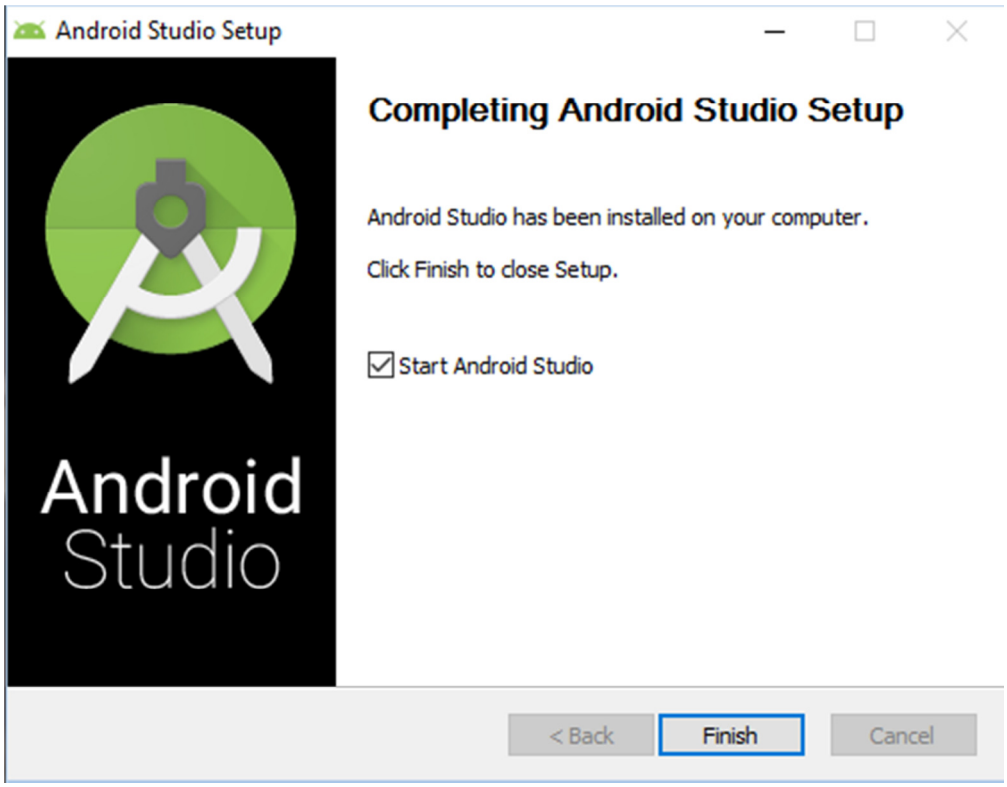
Step 4: Install location

Next step is to choose the start menu folder, where you want to create shortcut. If you don't want to create a shortcut just mark **Do not create shortcut**.



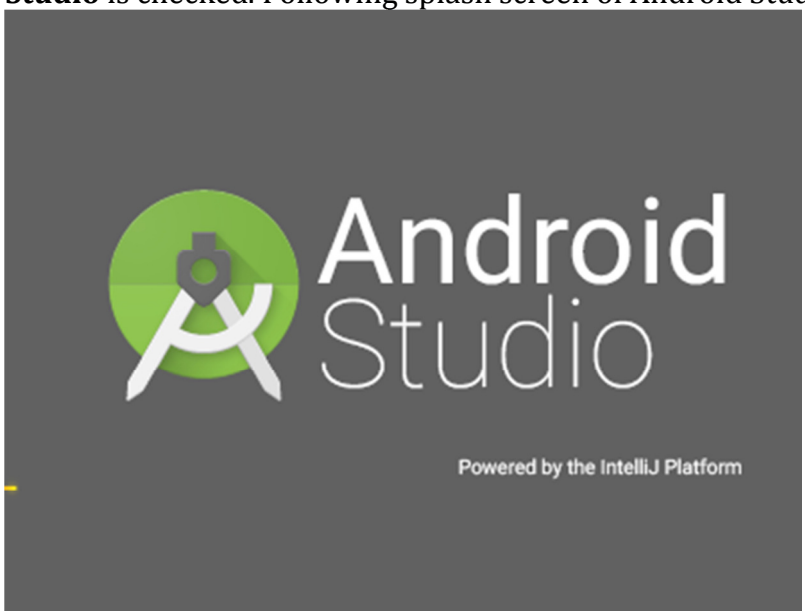
Step 5: Choose start menu folder

And hit **Install** button.
It will start installation. Once it's done following window will appear.



Step 6: Finish

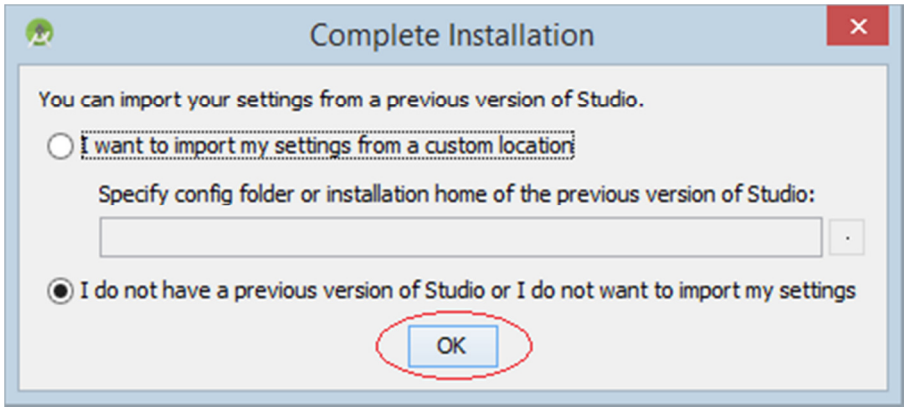
This informs you installation has completed. Click **Finish**. Make sure **Start Android Studio** is checked. Following splash screen of Android Studio will appear.



Step 7: Android Studio Splash Screen

Step 3) Configure Android Studio

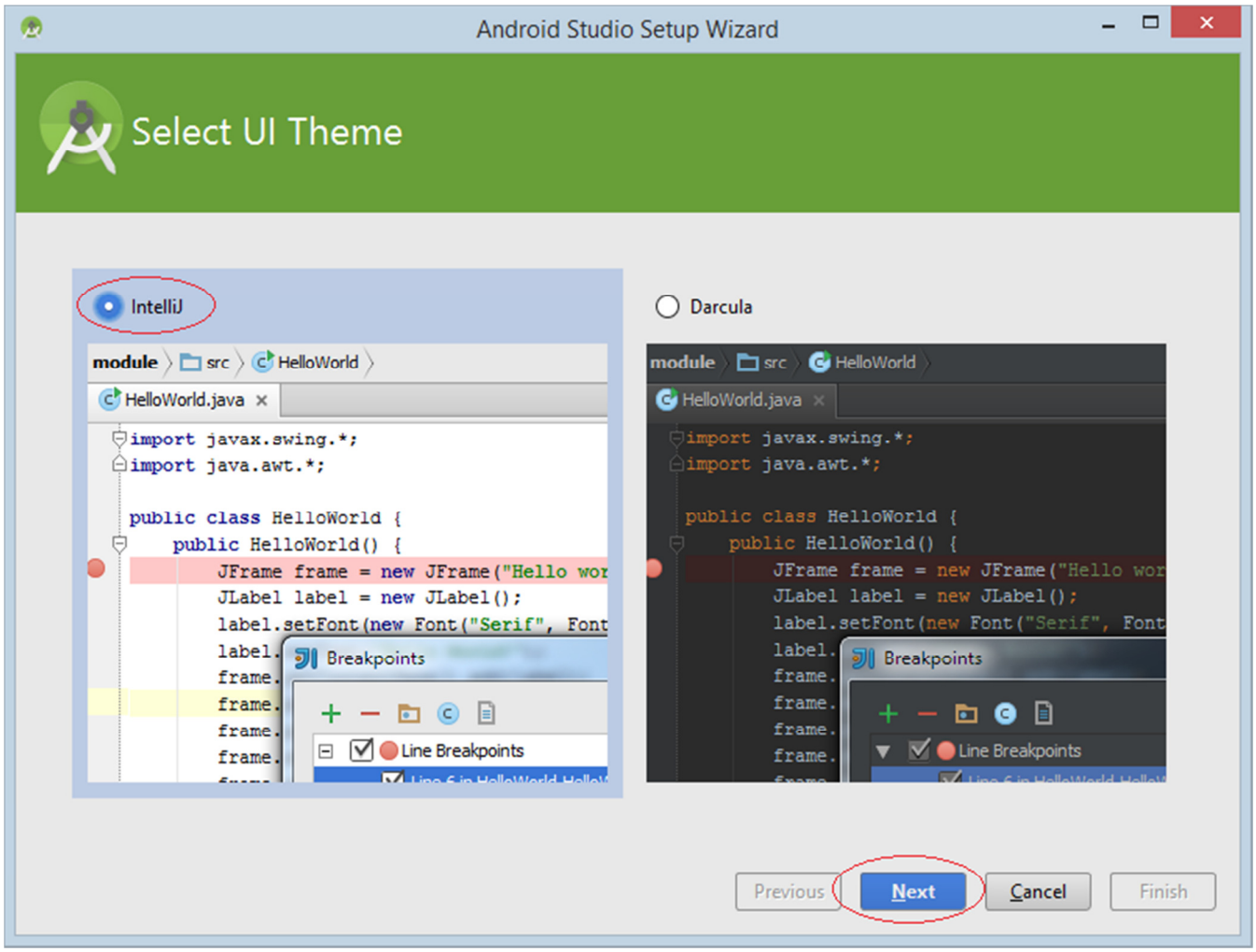
When you run it for the first time it will ask for Android Studio settings.



Step 8: Import settings

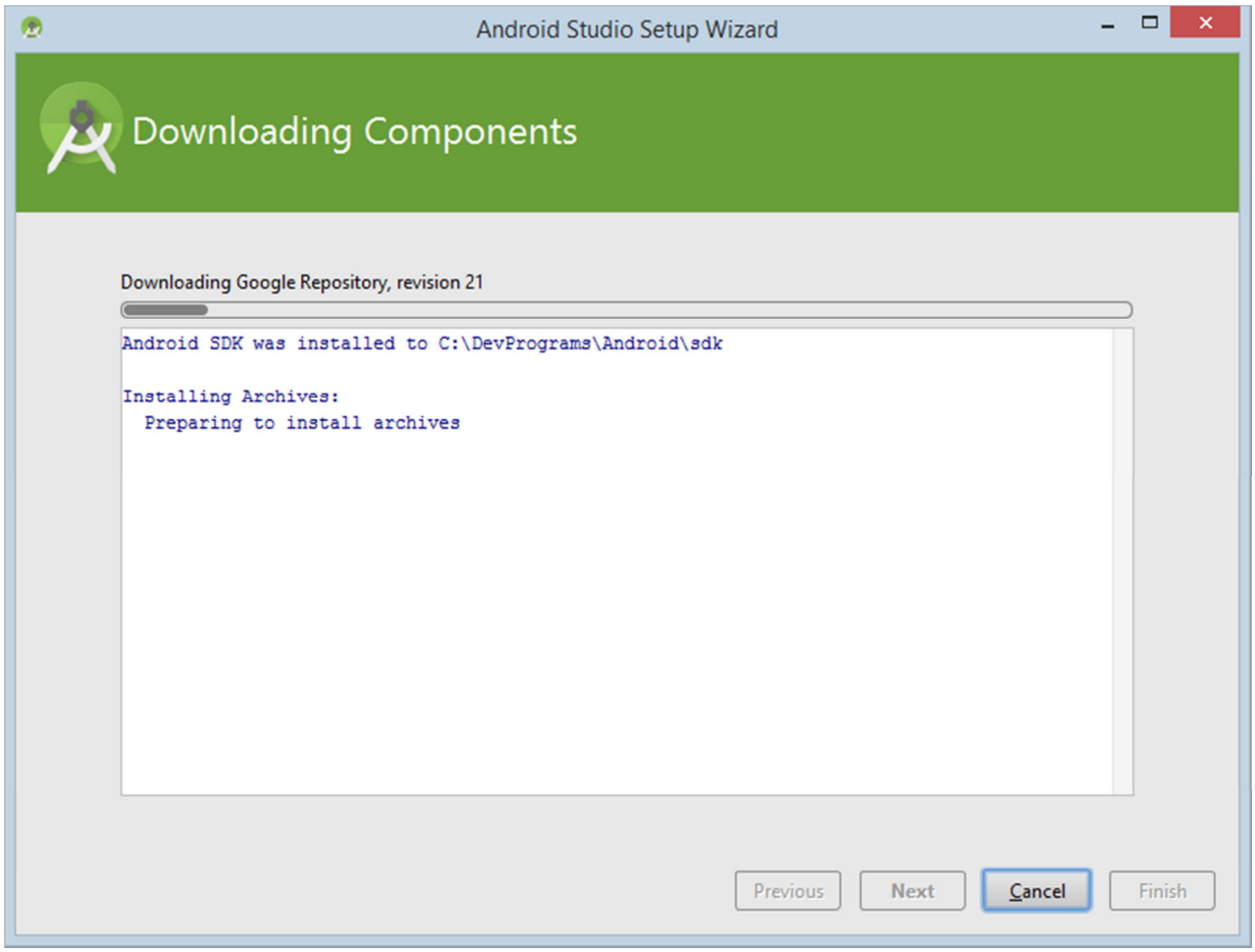
If you don't have any previous settings click on the second option (I don't have a previous version of Studio or I don't want to import my settings).

Select a theme and click next.



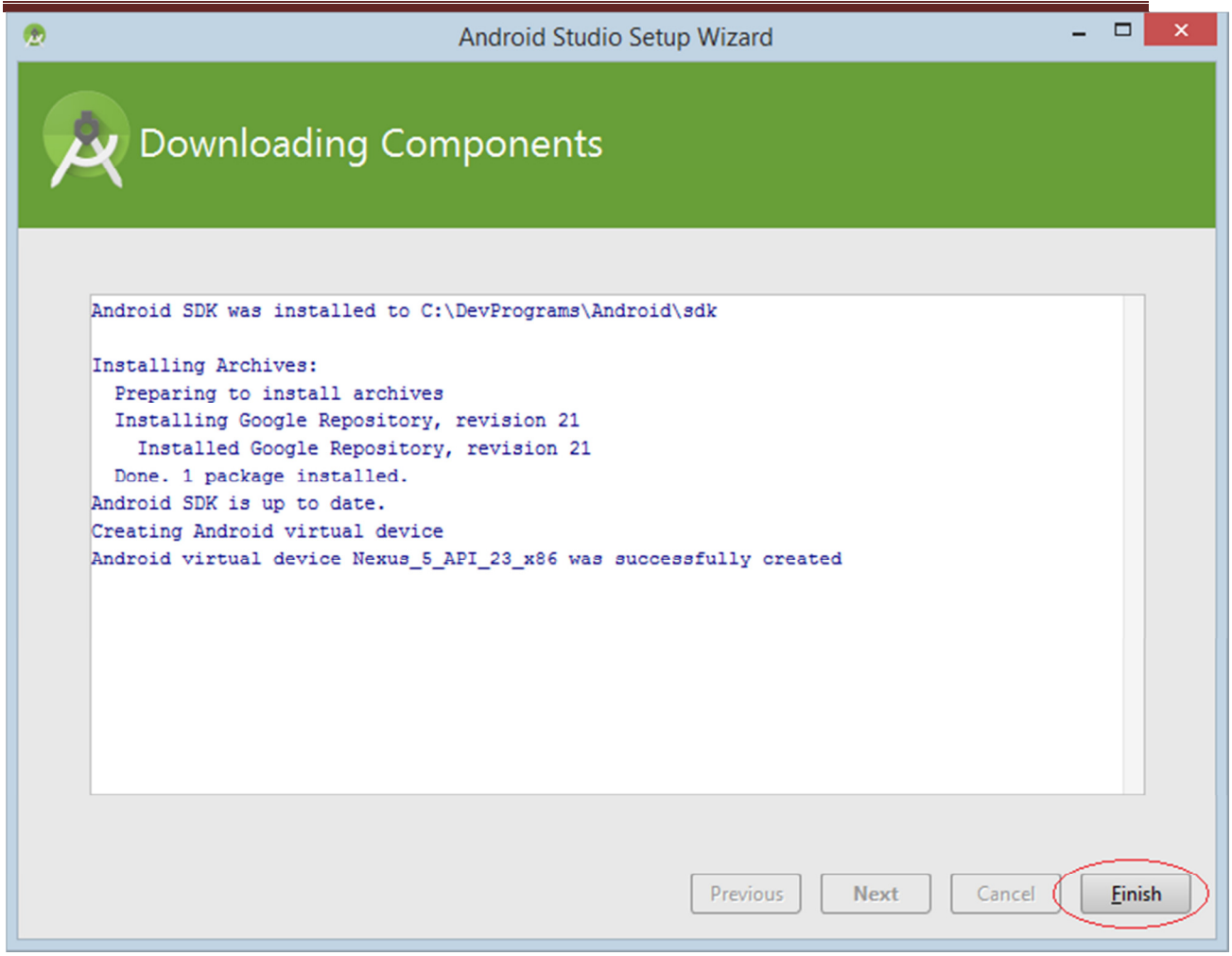
Step 9: Select theme

At the very first run it needs to download some necessary components, wait till it completes.



Step 10: Download components

And it's all done.

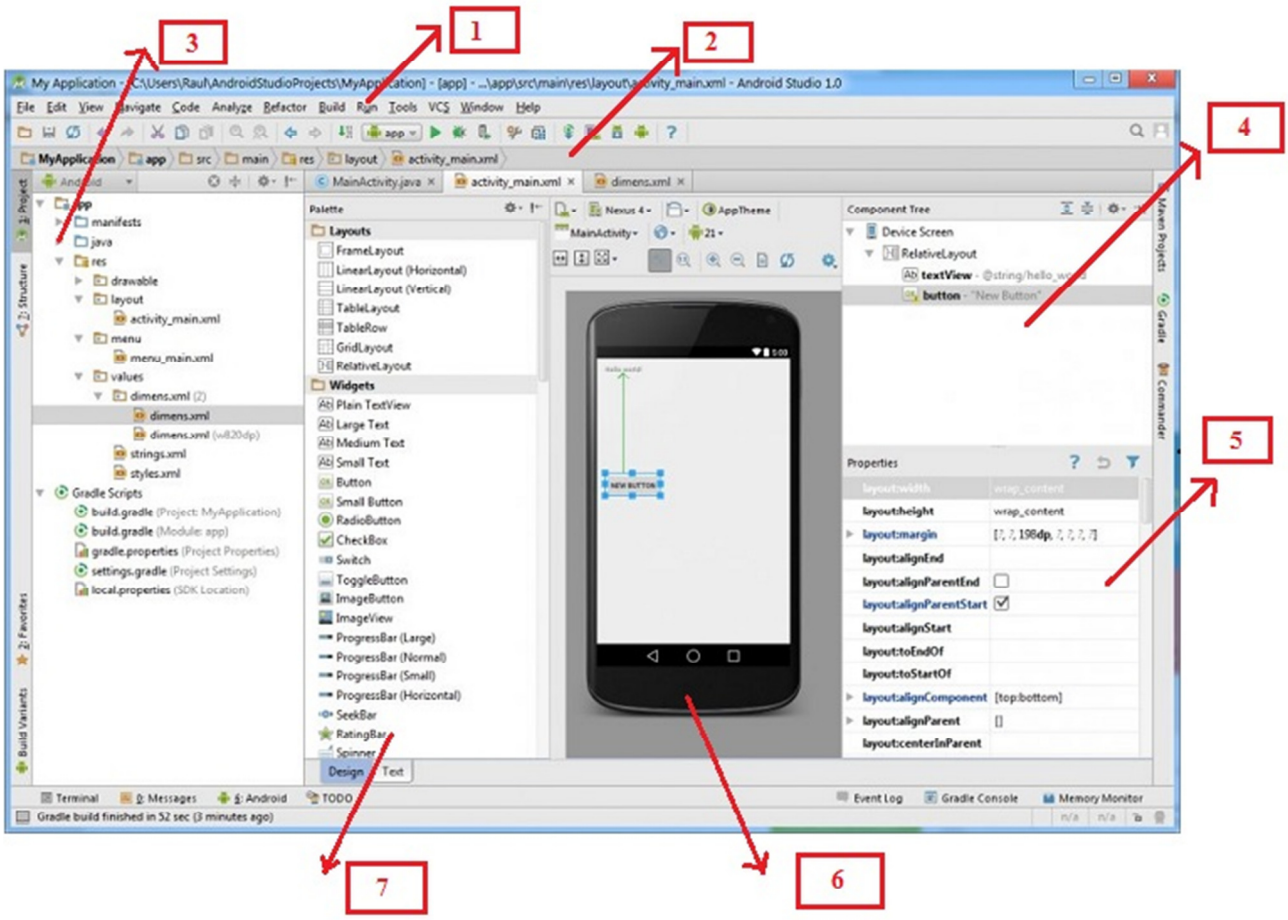


Step 11: Done

Click on **Finish** and start building your Android apps.

Introduction to Android Studio User Interface

Android Studio is an Integrated Development Environment (IDE). You have seen download and installation in this tutorial. Let's learn some basics of Android Studio. Here is the a screenshot of a running Android studio.



Android Studio screen

Red mark shows

1: Tool bar- It is collection of many tools like cut, copy, paste, run debug and others.
2: Navigation bar- It helps you to navigate through the recent open files of your project.
3: Project hierarchy- It is the hierarchy of your project's folders.
4: Component Tree- It shows component used in an activity in the form of a tree structure.
5: Properties window- It shows properties of selected item on the screen.
6: Layout editor- It shows graphical layout, how your app will look like.
7: Palette window- Palette window shows component, layouts, and widgets available in Android Studio.

Create an Android project

This lesson shows you how to create a new Android project with Android Studio, and it describes some of the files in the project.

To create your new Android project, follow these steps:

1. Install the latest version of Android Studio.

2. In the **Welcome to Android Studio** window, click **Start a new Android Studio project**.

Figure 1. Android Studio welcome screen

If you have a project already opened, select **File > New > New Project**.

3. In the **Choose your project** window, select **Empty Activity** and click **Next**.
4. In the **Configure your project** window, complete the following:

- Enter "My First App" in the **Name** field.
- Enter "com.example.myfirstapp" in the **Package name** field.
- If you'd like to place the project in a different folder, change its **Save** location.
- Select either **Java** or **Kotlin** from the **Language** drop-down menu.
- Select the checkbox next to **Use androidx.* artifacts**.
- Leave the other options as they are.

5. Click **Finish**.

After some processing time, the Android Studio main window appears.

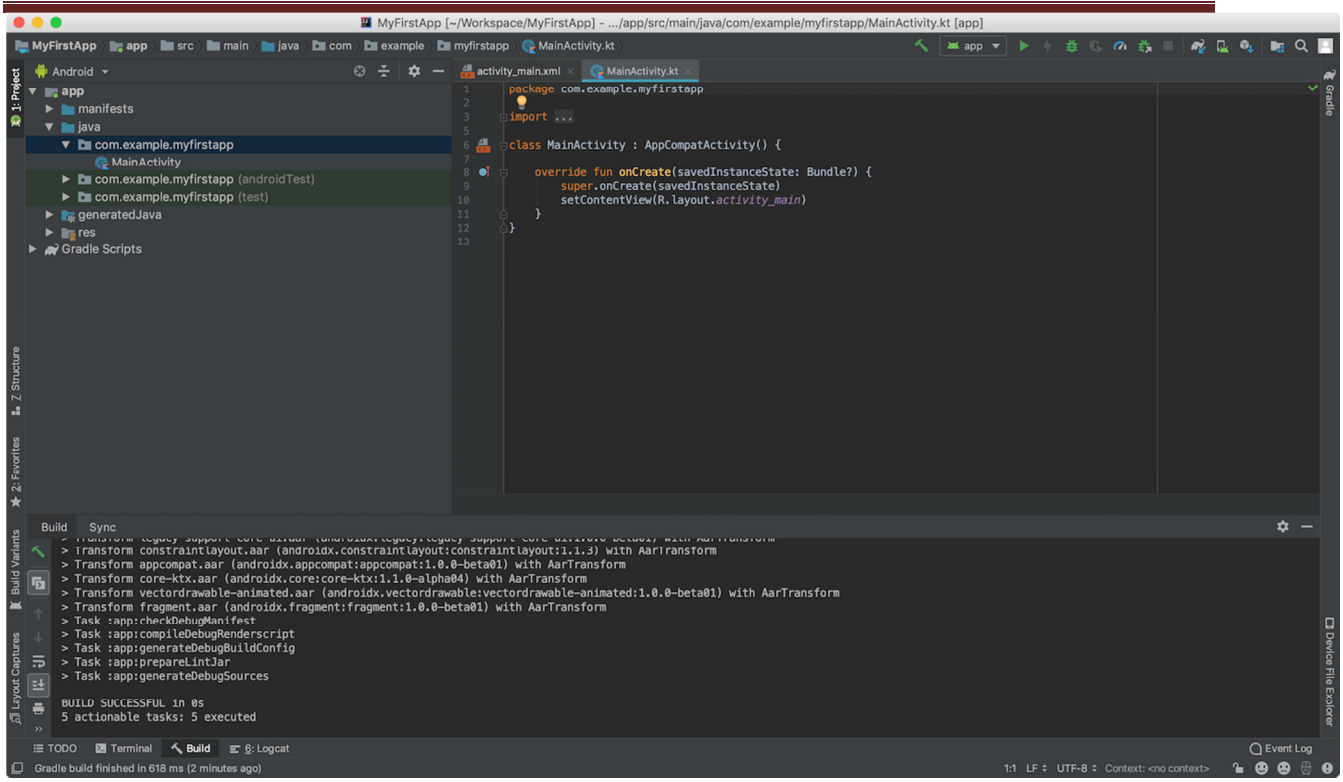


Figure 2. Android Studio main window

Now take a moment to review the most important files.

First, be sure the **Project** window is open (select **View > Tool Windows > Project**) and the Android view is selected from the drop-down list at the top of that window. You can then see the following files:

app > java > com.example.myfirstapp > MainActivity

This is the main activity. It's the entry point for your app. When you build and run your app, the system launches an instance of this Activity and loads its layout.

app > res > layout > activity_main.xml

This XML file defines the layout for the activity's user interface (UI). It contains a TextView element with the text "Hello, World!"

app > manifests > AndroidManifest.xml

The manifest file describes the fundamental characteristics of the app and defines each of its components.

Gradle Scripts > build.gradle

There are two files with this name: one for the project, "Project: My First App," and one for the app module, "Module: app." Each module has its own build.gradle file, but this project currently has just one module. Use each

module's build.file to control how the Gradle plugin builds your app. For more information about this file, see Configure your build.

Run your app

In the previous section, you created an Android app that displays "Hello, World!" You can now run the app on a real device or an emulator.

Run on a real device

Set up your device as follows:

1. Connect your device to your development machine with a USB cable. If you developed on Windows, you might need to install the appropriate USB driver for your device.
2. Perform the following steps to enable **USB debugging** in the **Developer options** window:

a. Open the **Settings** app.

b. If your device uses Android v8.0 or higher, select **System**. Otherwise, proceed to the next step.

c. Scroll to the bottom and select **About phone**.

d. Scroll to the bottom and tap **Build number** seven times.

e. Return to the previous screen, scroll to the bottom, and tap **Developer options**.

f. In the **Developer options** window, scroll down to find and enable **USB debugging**.

Run the app on your device as follows:

1. In Android Studio, select your app from the run/debug configurations drop-down menu in the toolbar.
2. In the toolbar, select the device that you want to run your app on from the target device drop-down menu.

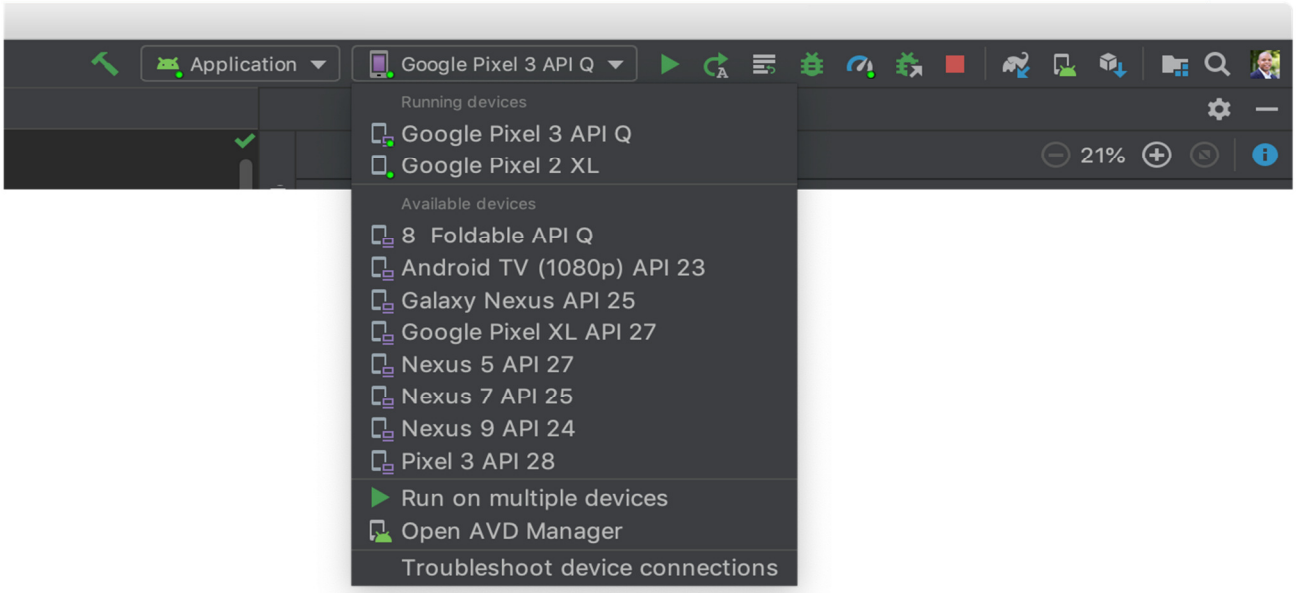


Figure 1. Target device drop-down menu

1. Click **Run**  .

   Android Studio installs your app on your connected device and starts it. You now see "Hello, World!" displayed in the app on your device.

   To begin to develop your app, continue to the next lesson.

Run on an emulator

Run the app on an emulator as follows:

1. In Android Studio, create an Android Virtual Device (AVD) that the emulator can use to install and run your app.
2. In the toolbar, select your app from the run/debug configurations drop-down menu.
3. From the target device drop-down menu, select the AVD that you want to run your app on.

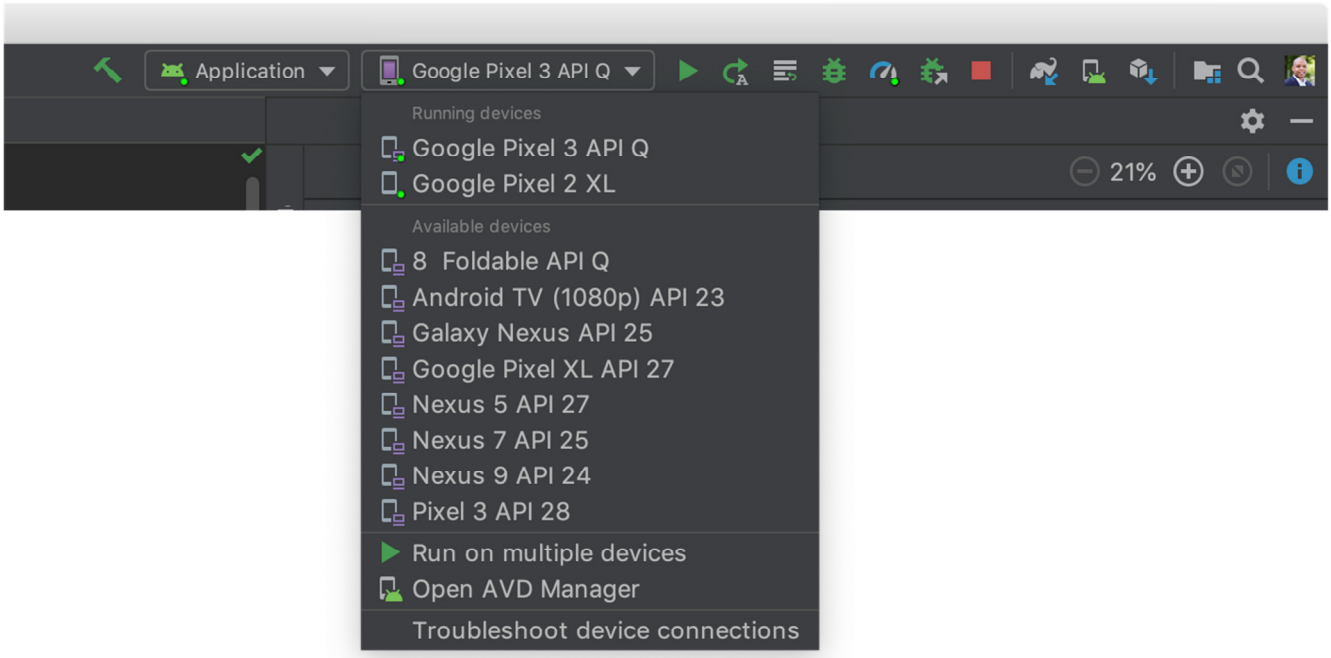


Figure 2. Target device drop-down menu

4. Click **Run**  .

   Android Studio installs the app on the AVD and starts the emulator. You now see "Hello, World!" displayed in the app.