Android Studio IDE: INTRODUCTION

Source: https://developer.android.com/studio/intro/index.html#project_structure

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA .

Java Development Kit (JDK)

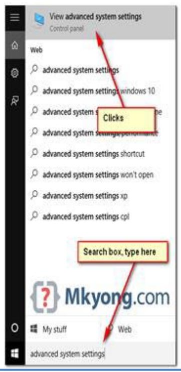
Google Search "Java JDK":

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

Select: Java SE (platform) → Windows x64

Installation process:

1. **Click "Next" mostly** → Until "*successfully installed JDK*" notice.
2. **Two folders created:**
   - Java/jdk1.8.0_91
   - Java/jre.8.0_91 (Java Runtime Environment)
3. **Set environment variables so that Android studio can recognize it:**
   - **Copy path:** C:\Program Files\Java\jdk1.8.0_91
   - **Open:** Computer → Property → Advanced System Setting → "Advanced" tab: **Environment Variables** → Check whether "**JAVA_HOME**" exists?
   - **If not exist**, create a **new System variable** (NOT USER_VARIABLE_FOR_ACER):
     - Variable name: JAVA_HOME
     - Variable value: C:\Program Files\Java\jdk1.8.0_91 (path to JDK);


View advanced system settings
Control panel
Web
advanced system settings
advanced system settings windows 10
advanced system s
Clicks
advanced system settings performance
advanced system settings shortcut
advanced system settings won't open
advanced system settings xp
advanced system settings cpl
Search box, type here
Mkyong.com
My stuff
Web
advanced system settings

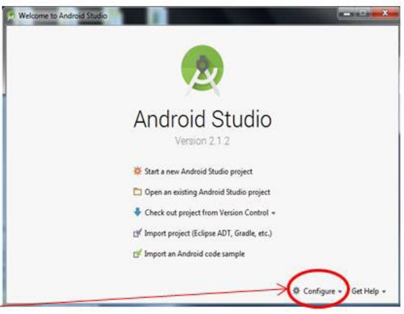
Android Studio IDE

Search "Android Studio"

→ Website: **https://developer.android.com/studio/index.html**

Installation process:

- ❑ **Click "Next" mostly** → Until "*successfully installed JDK*" notice.
- ❑ **Enable VT-x technology** in BIOS setting.

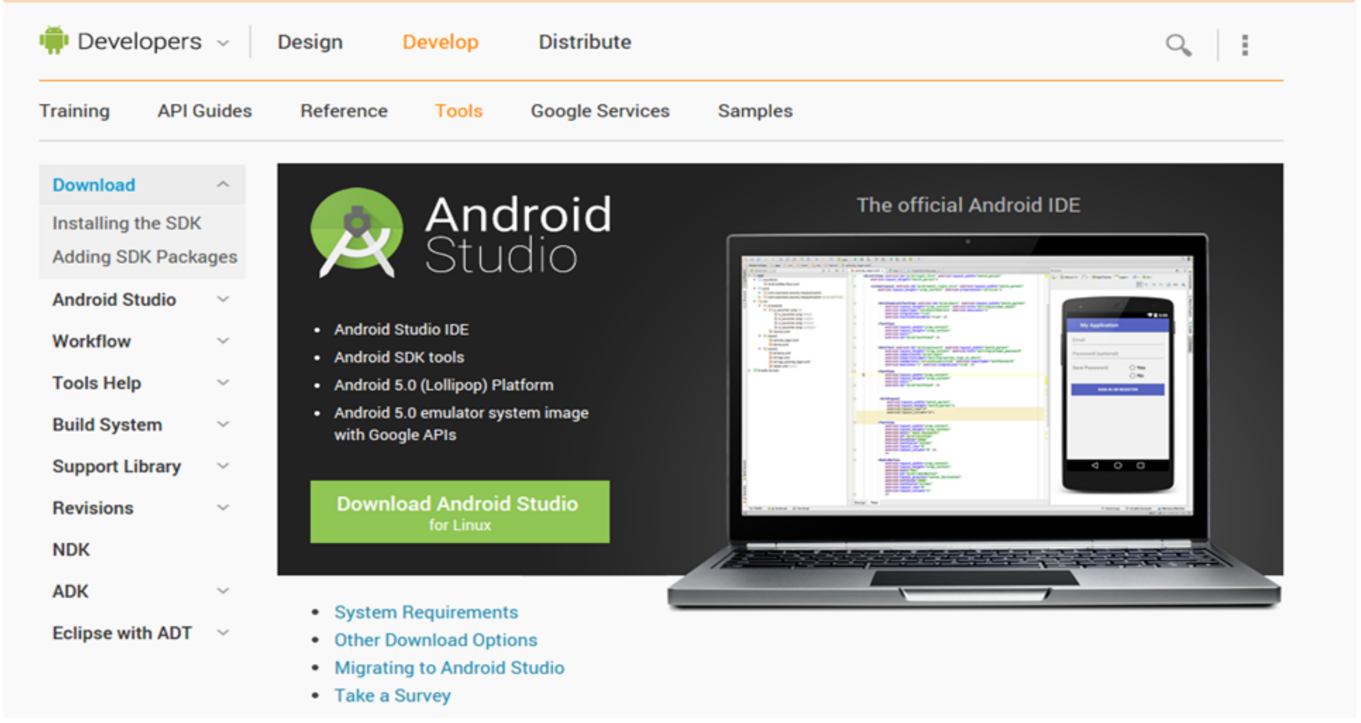

Welcome to Android Studio
Android Studio
Version 2.1.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

Before coding, configure:

- Configure → SDK Manager
- Make sure the options ticked: Intel x86 Emulator Accelerator, Android SDK Build Tools.

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

Android Studio IDE



Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

- Download
  - Installing the SDK
  - Adding SDK Packages
- Android Studio
- Workflow
- Tools Help
- Build System
- Support Library
- Revisions
- NDK
- ADK
- Eclipse with ADT

Android Studio

The official Android IDE

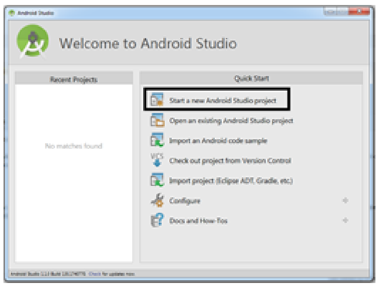
- Android Studio IDE
- Android SDK tools
- Android 5.0 (Lollipop) Platform
- Android 5.0 emulator system image with Google APIs

Download Android Studio for Linux

- System Requirements
- Other Download Options
- Migrating to Android Studio
- Take a Survey

Source: https://developer.android.com/studio/index.html

Create a new Android project

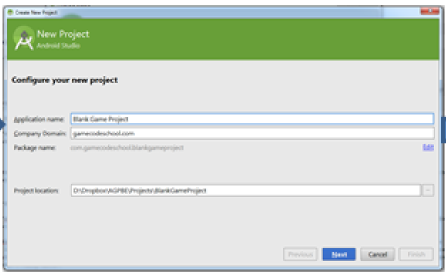


Welcome to Android Studio

Recent Projects: No matches found

Quick Start:
- Start a new Android Studio project
- Open an existing Android Studio project
- Import an Android code sample
- Check out project from Version Control
- Import project (Eclipse ADT, Gradle, etc.)
- Configure
- Docs and How-Tos

Start a new Android Studio Project



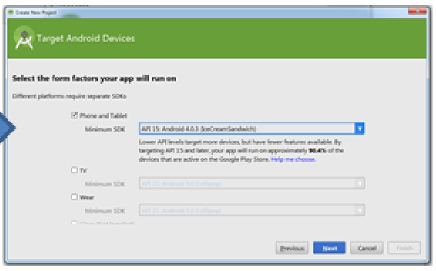
New Project
Android Studio

Configure your new project

Application name: Blank Game Project
Company Domain: gamecodeschool.com
Package name: com.gamecodeschool.blankgameproject
Project location: D:\Dropbox\AGPBE\Projects\BlankGameProject

Previous | Next | Cancel | Finish

Configure your new project dialog box



Target Android Devices

Select the form factors your app will run on

Different platforms require separate SDKs

☑ Phone and Tablet
Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available. By targeting API 15 and later, your app will run on approximately **90.4%** of the devices that are active on the Google Play Store. Help me choose.

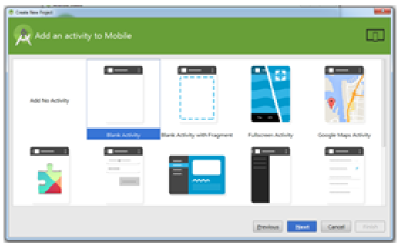
☐ TV
Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)

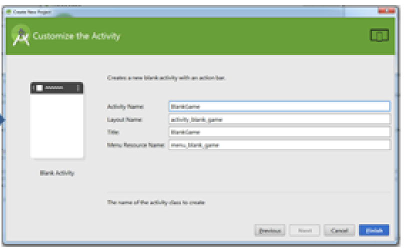
Previous | Next | Cancel | Finish

Target Android Devices screen

- Make sure the Phones and Tablets checkbox has a tick in it. It probably has by default.
- For the Minimum SDK to target option, a good rule of thumb is to pick a newer version of Android that does not exclude too many older devices. At the time of writing, **the API 19: Android 4.4 (KitKat)** is a good choice. But it doesn't matter too much while we are learning. If you are unsure just leave it at the default that Android Studio selects.



Add an Activity to mobile



Customize the Activity

Creates a new blank activity with an action bar.

Activity Name: BlankGame
Layout Name: activity_blank_game
Title: BlankGame
Menu Resource Name: menu_blank_game

Blank Activity

The name of the activity class to create

Previous | Next | Cancel | Finish

Customize the Activity

_Application name: **HelloWorld**
_Company Domain: **nz.ac.ames.project**

_Target Android devices: **Phone and Tablet**;
  + Minimum SDK: API 19: Android 4.4 (KitKat)

_Add an Activity to Mobile: **Empty Activity**;
  *+ Activity name: **MainActivity***
  *+ Layout name: **activity_main***

Android Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the **"Android" view**, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

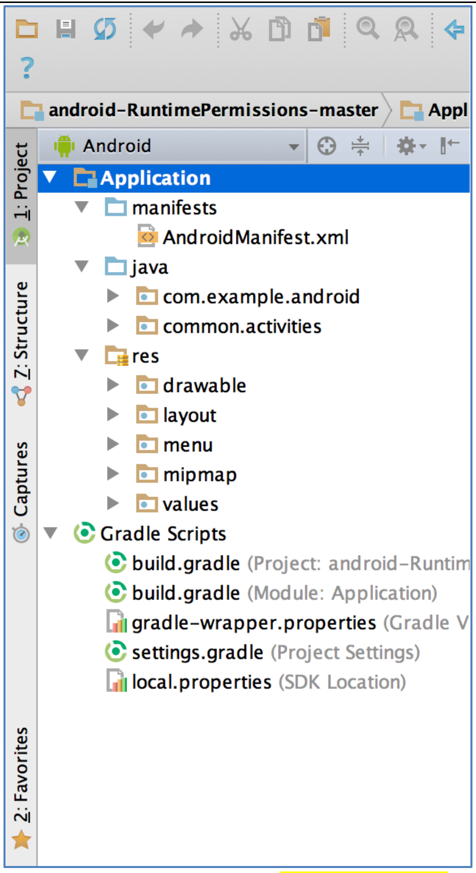
All the build files are visible at the top level under "**Gradle Scripts**".

"App" module contains the following folders:

- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Java source code files, including JUnit test code.
- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

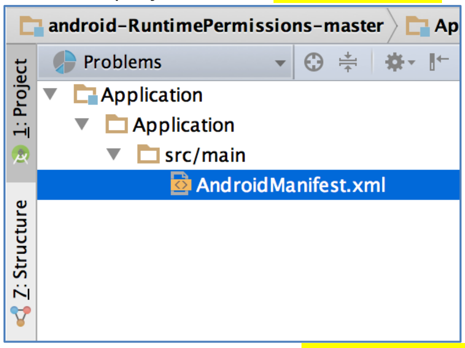
The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select "**Project**" from the **Project** dropdown.

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.



```
android-RuntimePermissions-master > Appl
Android
▼ Application
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▶ com.example.android
    ▶ common.activities
  ▼ res
    ▶ drawable
    ▶ layout
    ▶ menu
    ▶ mipmap
    ▶ values
▼ Gradle Scripts
    build.gradle (Project: android-Runtim
    build.gradle (Module: Application)
    gradle-wrapper.properties (Gradle V
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

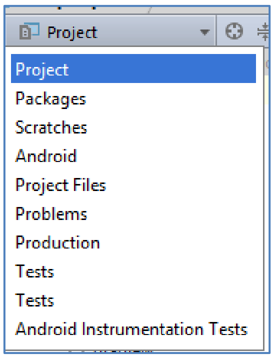
Figure 1. The project files in **"Android"** view.



```
android-RuntimePermissions-master > Ap
Problems
▼ Application
  ▼ Application
    ▼ src/main
        AndroidManifest.xml
```

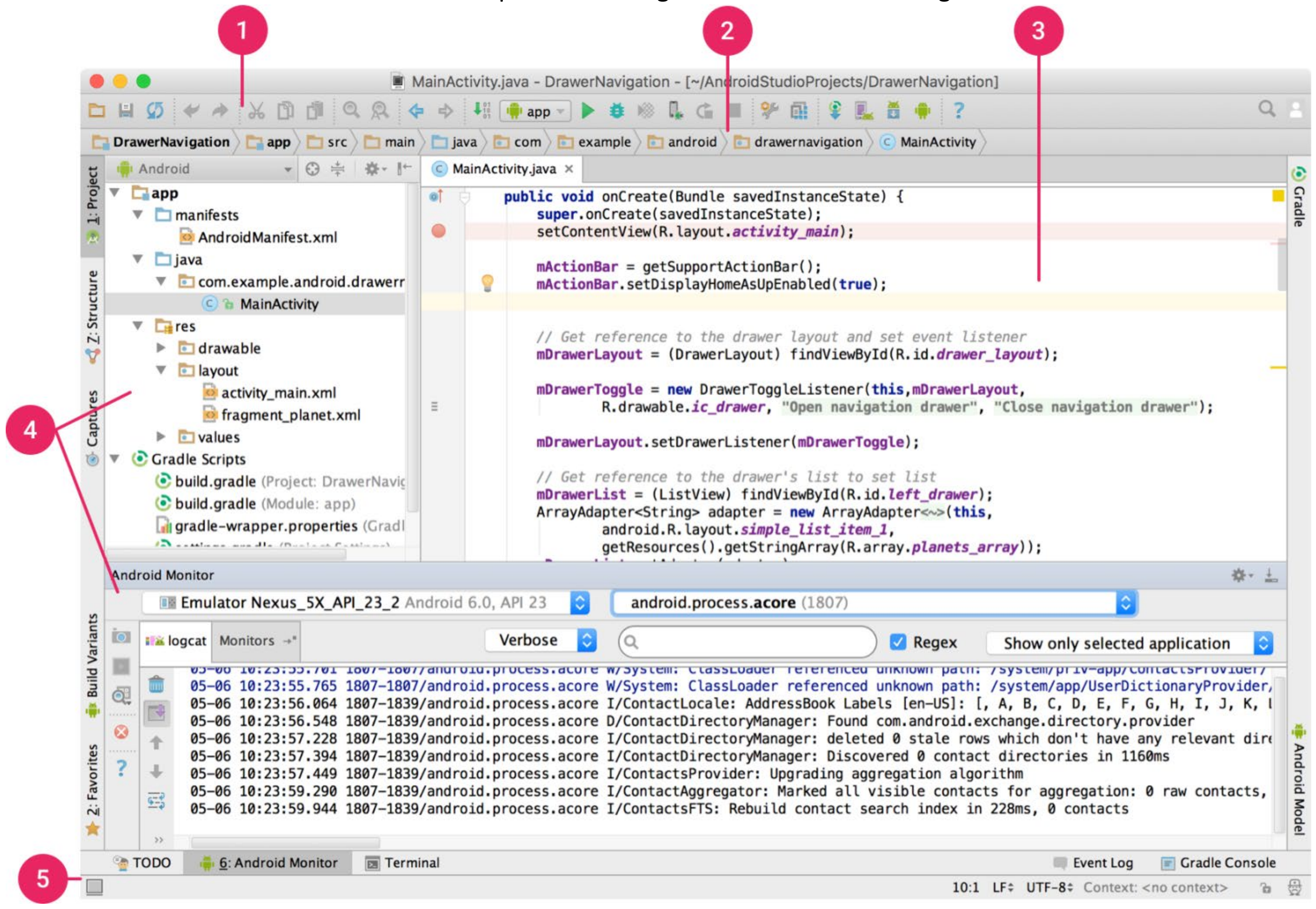
Figure 2. The project files in "**Problems**" view, showing a layout file with a problem.

Project Structure:
+ "Project" view
+ "Packages" view
+ "Scratches" view
+ "Android" view
+ "Project file" view
+ "Problems" view
+ "Production" view
+ Test view



The User Interface (UI)

The **Android Studio main window** is made up of several logical areas identified in figure 3.



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);


    // Get reference to the drawer layout and set event listener
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

    mDrawerToggle = new DrawerToggleListener(this,mDrawerLayout,
            R.drawable.ic_drawer, "Open navigation drawer", "Close navigation drawer");

    mDrawerLayout.setDrawerListener(mDrawerToggle);

    // Get reference to the drawer's list to set list
    mDrawerList = (ListView) findViewById(R.id.left_drawer);
    ArrayAdapter<String> adapter = new ArrayAdapter<~>(this,
            android.R.layout.simple_list_item_1,
            getResources().getStringArray(R.array.planets_array));
```

```
05-06 10:23:55.765 1807-1807/android.process.acore W/System: ClassLoader referenced unknown path: /system/app/UserDictionaryProvider,
05-06 10:23:56.064 1807-1839/android.process.acore I/ContactLocale: AddressBook Labels [en-US]: [, A, B, C, D, E, F, G, H, I, J, K, l
05-06 10:23:56.548 1807-1839/android.process.acore D/ContactDirectoryManager: Found com.android.exchange.directory.provider
05-06 10:23:57.228 1807-1839/android.process.acore I/ContactDirectoryManager: deleted 0 stale rows which don't have any relevant dire
05-06 10:23:57.394 1807-1839/android.process.acore I/ContactDirectoryManager: Discovered 0 contact directories in 1160ms
05-06 10:23:57.449 1807-1839/android.process.acore I/ContactsProvider: Upgrading aggregation algorithm
05-06 10:23:59.290 1807-1839/android.process.acore I/ContactAggregator: Marked all visible contacts for aggregation: 0 raw contacts,
05-06 10:23:59.944 1807-1839/android.process.acore I/ContactsFTS: Rebuild contact search index in 228ms, 0 contacts
```

Figure 3: The Android Studio main window

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project tool window.
3. The **editor window** is where you create and modify code. Depending on the current file type, this window can change. For example, when viewing a layout file, the editor window displays the layout editor and offers the option to view the corresponding XML file.
4. **Tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
5. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.
At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool Windows

You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows.

Table 1: Keyboard shortcuts for some useful tool windows

Tool Window	Windows and Linux	Mac
Project	**Alt+1**	**Command+1**
Version Control	**Alt+9**	**Command+9**
Run	**Shift+F10**	**Control+R**
Debug	**Shift+F9**	**Control+D**
Android Monitor	**Alt+6**	**Command+6**
Return to Editor	**Esc**	**Esc**
Hide All Tool Windows	**Control+Shift+F12**	**Command+Shift+F12**

Code Completion

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

Table 2: Keyboard shortcuts for code completion

Type	Description	Windows/Linux	Mac
Basic Completion	Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members.	**Control+Space**	**Control+Space**
Smart Completion	Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains.	**Control+Shift+Space**	**Control+Shift+Space**
Statement Completion	Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc.	**Control+Shift+Enter**	**Shift+Command+Enter**

Navigation

Here are some tips to help you move around Android Studio.

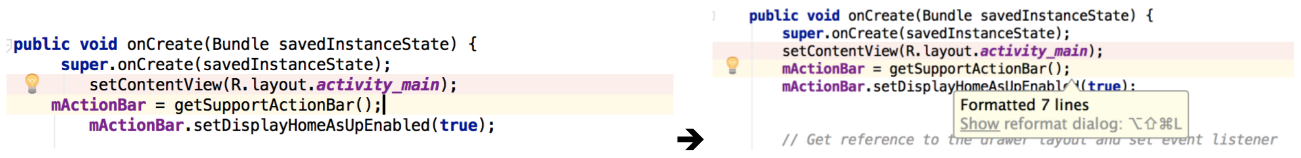
- Switch between your recently accessed files using the *Recent Files* action. Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
- View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
- Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N** (**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.
- Navigate to a file or folder using the *Navigate to File* action. Bring up the Navigate to File action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
- Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the Navigate to Symbol action by pressing **Control+Shift+Alt+N** (**Command+Shift+Alt+O** on a Mac).
- Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7**.

Style and Formatting

As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings. You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines.

To customize your code style settings, click: **File -> Settings -> Editor -> Code Style**
(**Android Studio > Preferences > Editor > Code Style** on a Mac.)

Although the IDE automatically applies formatting as you work, you can also explicitly call the ***Reformat Code*** action by pressing **Control+Alt+L**(**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I** (**Alt+Option+I** on a Mac).



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
  mActionBar = getSupportActionBar();
        mActionBar.setDisplayHomeAsUpEnabled(true);
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);


    // Get reference to the drawer layout and set event listener
```

Figure 4: Code before formatting and Code after formatting

Version Control Basics

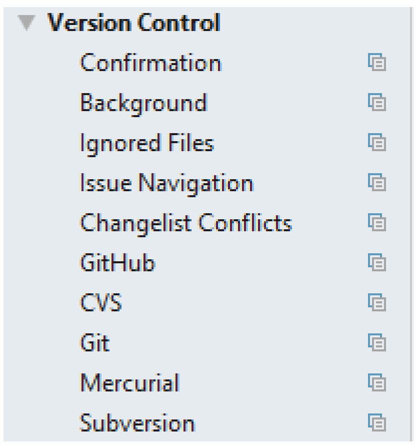
Android Studio **supports a variety of version control systems (VCS's),** including **Git**, **GitHub**, **CVS**, **Mercurial**, **Subversion**, and **Google Cloud Source Repositories**.

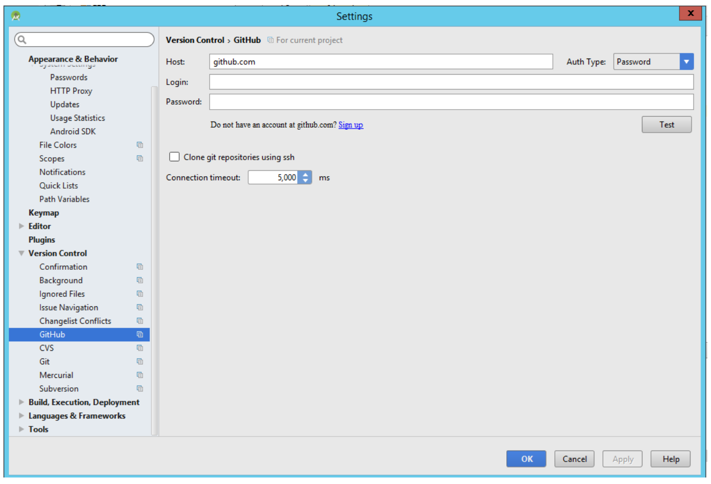
After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

1. From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
2. From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.

The VCS menu now displays a number of version control options based on the system you selected.

Note: You can also use the **File- > Settings -> Version Control** menu option to set up and modify the version control settings.





Version Control › GitHub For current project

Host: github.com
Auth Type: Password
Login:
Password:
Do not have an account at github.com? Sign up

☐ Clone git repositories using ssh
Connection timeout: 5,000 ms

The Build Process

The build process involves many tools and processes that convert your project into an **Android Application Package (APK)**. The build process is very flexible, so it's useful to understand some of what is happening under the hood.

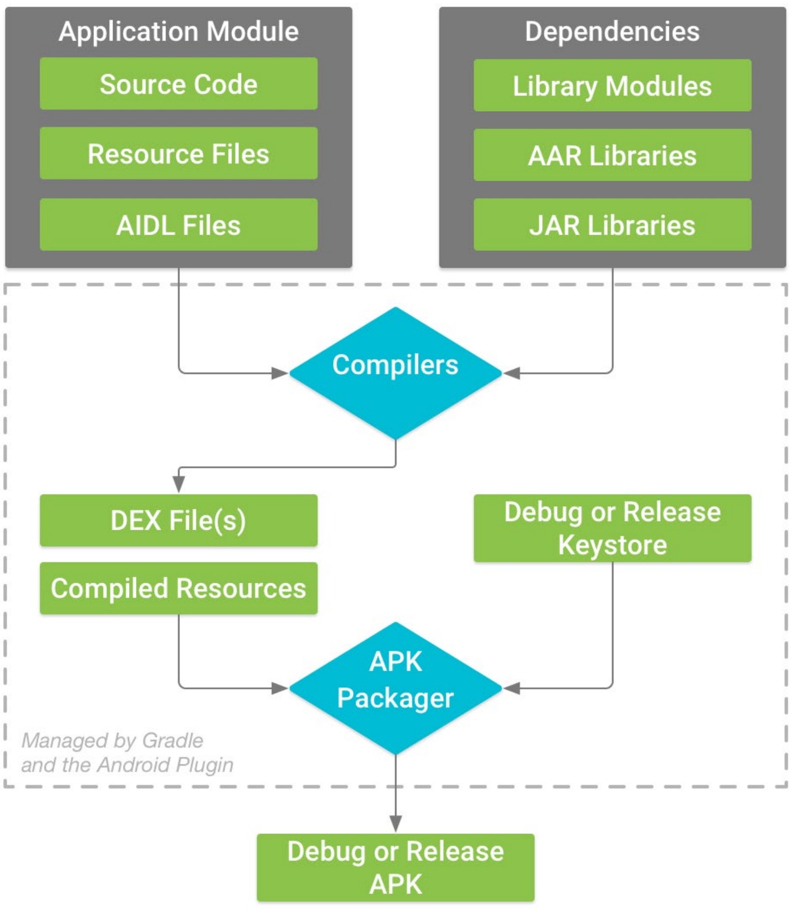


Figure 1: The build process of a typical Android app module

The build process for a typical Android app module, as shown in figure 1, follows these general steps:

1. **The compilers convert your source code into DEX (Dalvik Executable) files**, which include the bytecode that runs on Android devices, and everything else into compiled resources.
2. **The APK Packager combines the DEX files and compiled resources into a single APK**. Before your app can be installed and deployed onto an Android device, however, the APK must be signed.
3. **The APK Packager signs your APK** using either the debug or release keystore:
   a. If you are building a **debug version of your app**, that is, an app you intend only for testing and profiling, the packager signs your app with the debug keystore. Android Studio automatically configures new projects with a debug keystore.
   b. If you are building a **release version of your app** that you intend to release externally, the packager signs your app with the release keystore. To create a release keystore, read about signing your app in Android Studio.
4. Before generating your final APK, the packager uses the **zipalign** **tool to optimize your app** to use less memory when running on a device.

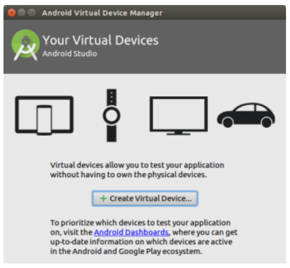
At the end of the build process, you have either a debug APK or release APK of your app that you can use to deploy, test, or release to external users.

Debug Tools

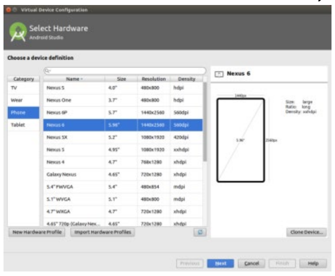
Create a virtual device (AVD)

+ Define a new Android Virtual Device (AVD) by opening the *AVD Manager*:

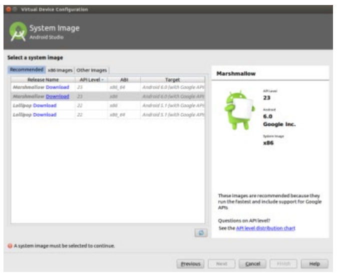
*Tools → Android → AVD Manager → Press the "**Create Virtual Device**" button;*


Android Virtual Device Manager
Your Virtual Devices
Android Studio
Virtual devices allow you to test your application without having to own the physical devices.
+ Create Virtual Device...
To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

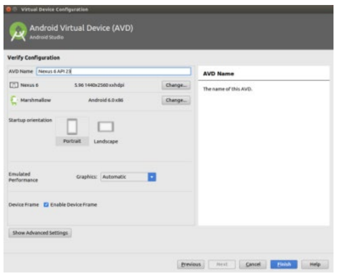
→



+ On the next screen select the latest API level for your AVD. You may need to select the option for additional images as highlighted in the following screenshot.



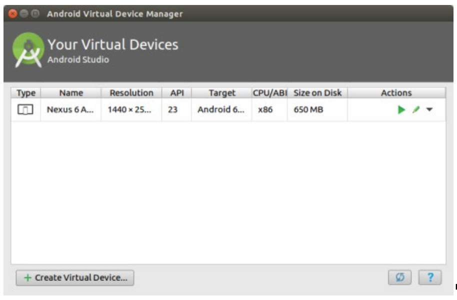
→



+ Afterwards press the *Finish* button. This will create the AVD configuration and display it under the list of available virtual devices.

Start your virtual device

Select ***"Run"* → *Run 'app'*** to start your application. This opens a dialog in which you can select your device to deploy your application to.



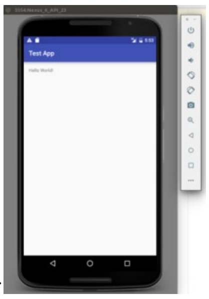
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 6 A...	1440 × 25...	23	Android 6...	x86	650 MB	

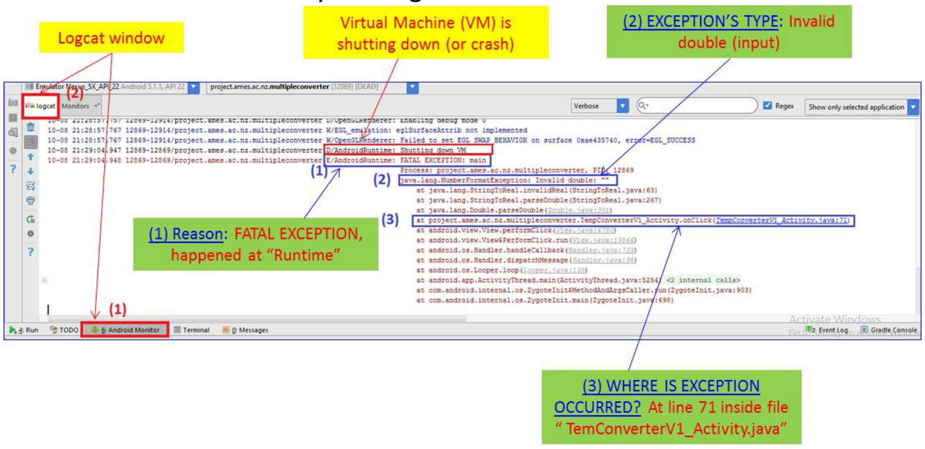
+ Create Virtual Device...

→



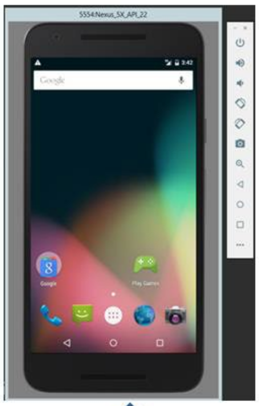
Log messages

When you build and run your app with Android Studio, you can view adb output and device log messages (logcat) by clicking **Android Monitor** at the bottom of the window. If you want to debug your app with the Android Device Monitor, you can launch the Device Monitor by clicking **Tools -> Android -> Android Device Monitor**.


Logcat window
Virtual Machine (VM) is shutting down (or crash)
(2) EXCEPTION'S TYPE: Invalid double (input)
Emulator Nexus_5X_API_22 Android 5.1.1, API 22
project.ames.ac.nz.multipleconverter (12869) [DEAD]
(2)
logcat
Monitors
Verbose
Regex
Show only selected application
(1)
(2)
(3)
(1) Reason: FATAL EXCEPTION, happened at "Runtime"
(1)
4: Run
TODO
6: Android Monitor
Terminal
0: Messages
Event Log
Gradle Console
(3) WHERE IS EXCEPTION OCCURRED? At line 71 inside file " TemConverterV1_Activity.java"

```
10-08 21:28:57.757 12869-12914/project.ames.ac.nz.multipleconverter D/OpenGLRenderer: Enabling debug mode 0
10-08 21:28:57.767 12869-12914/project.ames.ac.nz.multipleconverter W/EGL_emulation: eglSurfaceAttrib not implemented
10-08 21:28:57.767 12869-12914/project.ames.ac.nz.multipleconverter W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0xae435740, error=EGL_SUCCESS
10-08 21:29:04.947 12869-12869/project.ames.ac.nz.multipleconverter D/AndroidRuntime: Shutting down VM
10-08 21:29:04.948 12869-12869/project.ames.ac.nz.multipleconverter E/AndroidRuntime: FATAL EXCEPTION: main
                                                                        Process: project.ames.ac.nz.multipleconverter, PID: 12869
                                                                        java.lang.NumberFormatException: Invalid double: ""
                                                                           at java.lang.StringToReal.invalidReal(StringToReal.java:63)
                                                                           at java.lang.StringToReal.parseDouble(StringToReal.java:267)
                                                                           at java.lang.Double.parseDouble(Double.java:301)
                                                                           at project.ames.ac.nz.multipleconverter.TempConverterV1_Activity.onClick(TempConverterV1_Activity.java:71)
                                                                           at android.view.View.performClick(View.java:4780)
                                                                           at android.view.View$PerformClick.run(View.java:19866)
                                                                           at android.os.Handler.handleCallback(Handler.java:739)
                                                                           at android.os.Handler.dispatchMessage(Handler.java:95)
                                                                           at android.os.Looper.loop(Looper.java:135)
                                                                           at android.app.ActivityThread.main(ActivityThread.java:5254) <2 internal calls>
                                                                           at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:903)
                                                                           at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:698)
```

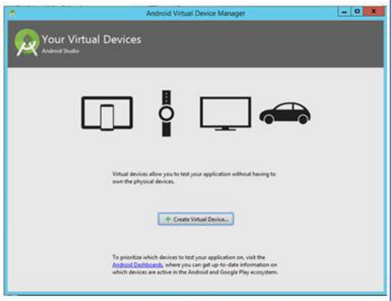
Emulator



Create a Android Virtual Device (AVD)

Click "AVD Manager" to create Android Virtual Device to test applications without the physical devices

- ✓ Click "**AVD Manager**" on toolbar;
- ✓ Select: Phone -> **Nexus 5**;
- ✓ Select system image: **Recommended**

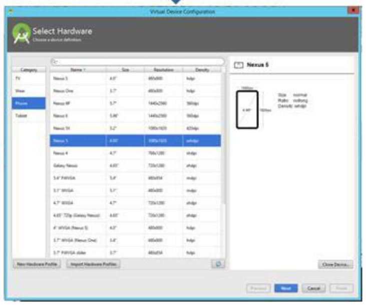

Android Virtual Device Manager

Your Virtual Devices
Android Studio

Virtual devices allow you to test your application without having to own the physical devices.

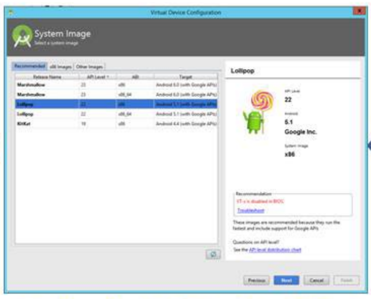
+ Create Virtual Device...

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.


Virtual Device Configuration

Select Hardware

Nexus 5

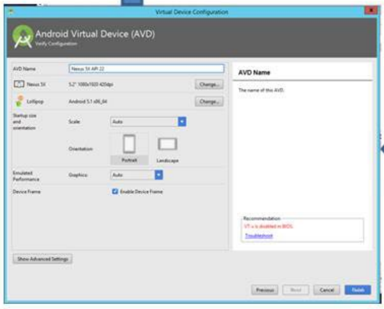

Virtual Device Configuration

System Image

Release Name	API Level	ABI	Target
Marshmallow	23	x86	Android 6.0 (with Google APIs)
Marshmallow	23	x86_64	Android 6.0 (with Google APIs)
Lollipop	22	x86	Android 5.1 (with Google APIs)
Lollipop	22	x86_64	Android 5.1 (with Google APIs)
KitKat	19	x86	Android 4.4 (with Google APIs)

Lollipop
API Level: 22
Android: 5.1
Google Inc.
System Image: x86

These images are recommended because they run the fastest and include support for Google APIs.


Virtual Device Configuration

Android Virtual Device (AVD)
Verify Configuration

AVD Name: Nexus 5X API 22
Nexus 5X: 5.2" 1080x1920 420dpi
Lollipop: Android 5.1 x86_64
Startup size and orientation: Scale: Auto; Orientation: Portrait
Emulated Performance: Graphics: Auto
Device Frame: ✓ Enable Device Frame

AVD Name
The name of this AVD.

Show Advanced Settings

API level: 22 & Android: 5.1

- **Investigate "AVD – verify configuration":**
  - **AVD Name**: ***Nexus 5X API 22***
  - **Configuration**:
    - 5.2"; 1080x1920; 420dpi,
    - Lollipop Android 5.1
  - **Start-up size and orientation**:
    - Start-up size (1px on screen, 2px on screen, 3px on screen) → **Select "AUTO"**
    - Orientation: portrait or landscape
  - **Emulated Performance**:
    - Different between software, hardware, and auto → **Select "auto"**
  - **Device frame**: tick "Enable Device Frame"

- **Investigate "Advanced settings":**
  - **Camera**:
    - Front camera: None, Emulated, Webcam0 → Select "none" for this exercise;
    - Back camera: Non, Emulated, Webcam 0 → Select "none" for this exercise;
  - **Network**:
    - Speed: FULL, GSM, HSCSD, GPRS, EDGE, UMTS, HSDPA → **Select 'FULL'**
    - Latency: None, UMTS, EDGE, GPRS → **Select 'None'**