Learning Unit 1

1.2 Android Studio

MOAATH ALRAJAB

Learning Unit 1

- 1.1 Introduction
  - Android History
  - Android Architecture
  - Getting Started
- 1.2 Android Studio
  - IDE parts
  - Creating a project
  - Installing required plugins
- 1.3 Android Application
  - Concept and components
  - Manifest file

Meet Android Studio

- Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA .
- On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:
- A flexible Gradle-based build system
- A fast and feature-rich emulator

Credit: Kirk Scott

- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Project Structure

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

Types of modules include:

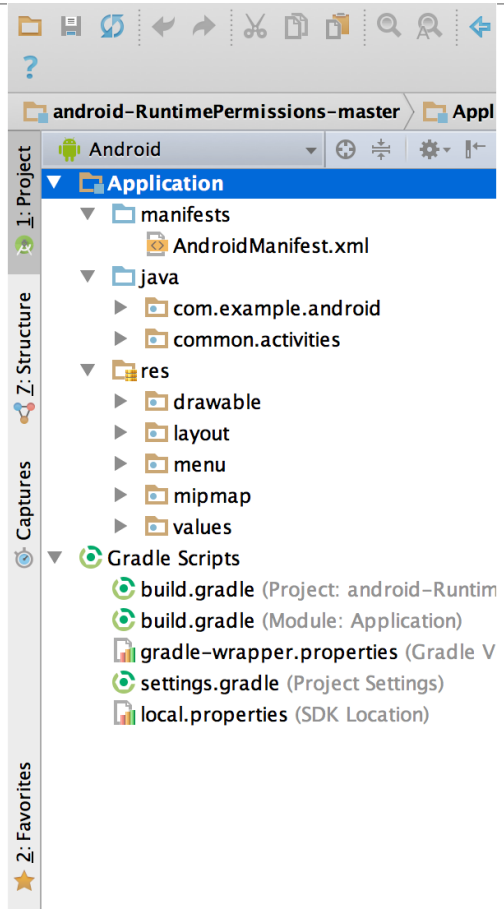
Android app modules

Library modules

Google App Engine modules

[See the following overhead]

By default, Android Studio displays your project files in the Android project 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** and each app module contains the following folders:

manifests: Contains the AndroidManifest.xml file.

java: Contains the Java source code files, including JUnit test code.

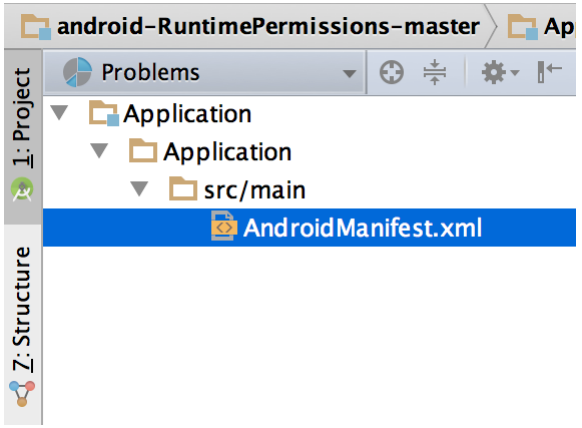
res: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from the IDE flattened representation.

To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

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.



The User Interface

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

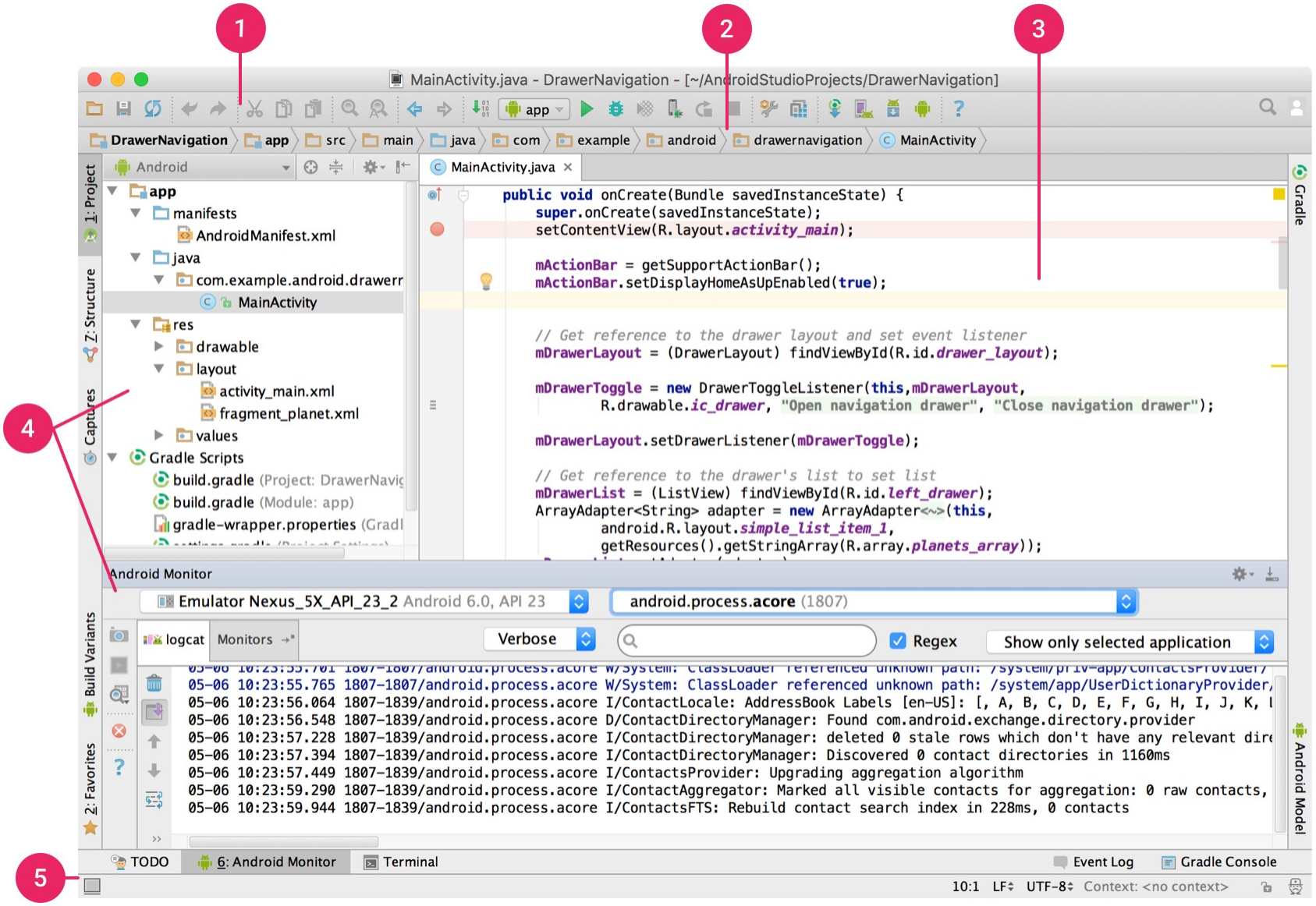
[1. The **toolbar**

2. The **navigation bar**

3. The **editor window**

4. **Tool windows**

5. The **status bar**


1
2
3
4
5
MainActivity.java - DrawerNavigation - [~/AndroidStudioProjects/DrawerNavigation]
DrawerNavigation
app
src
main
java
com
example
android
drawernavigation
MainActivity
Android
app
manifests
AndroidManifest.xml
java
com.example.android.drawerr
MainActivity
res
drawable
layout
activity_main.xml
fragment_planet.xml
values
Gradle Scripts
build.gradle (Project: DrawerNavig
build.gradle (Module: app)
gradle-wrapper.properties (Gradl
MainActivity.java
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));
Android Monitor
Emulator Nexus_5X_API_23_2 Android 6.0, API 23
android.process.acore (1807)
logcat
Monitors
Verbose
Regex
Show only selected application
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,
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
TODO
6: Android Monitor
Terminal
Event Log
Gradle Console
10:1
LF
UTF-8
Context: <no context>
1: Project
Z: Structure
Captures
Build Variants
2: Favorites
Gradle
Android Model

The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.

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.

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.

Tool windows give you access to specific tasks like project management, search, version control, and more.

You can expand them and collapse them.

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.

Searching the IDE

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

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work.

By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

To expand or collapse a tool window, click the tool's name in the tool window bar.

You can also drag, pin, unpin, attach, and detach tool windows.

Definitely Useful: Restoring the Default Layout

To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.

[Note: Whether you commit this to memory or not, it's important to be able to muddle through the menu in order to bring back windows that you've accidentally lost.]

To show or hide the entire tool window bar, click the window icon in the bottom left-hand corner of the Android Studio window.

To locate a specific tool window, hover over the window icon and select the tool window from the menu.

For Power-User Wannabes

It seems unlikely that most people would master the keyboard shortcuts and other items shown on the following overheads unless they went down the Android Studio rabbit hole

However, the list may have value as a semi-comprehensive introduction to some of the environment's capabilities

A few of the items listed are definitely useful

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

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**.

This enables *Distraction Free Mode*.

To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use *Speed Search* to search and filter within most tool windows in Android Studio.

To use Speed Search, select the tool window and then type your search query.

Code Completion

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

Code completion is definitely useful, but as an eternally beginning programmer, quite often I find it to be a distraction

Type	Description	Windows and 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.

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**.

2.2 “Update the IDE and Tools”

Things may have gotten better than in the past, but this is the voice of experience speaking:

It can be a mistake to accept development environment updates during the course of the semester (ask before update)

It is possible that system updates will cause something which was previously working not to work…

In general, you should just be able to install the latest stable release, and there is no particular need to update after that

It is certainly not advisable to use anything but a stable release

Especially since the hardware devices we're using are not the most recent, the latest updates to Android should be of no consequence to us

Having given the preliminary warning, it’s worth knowing how updates are accomplished

It is always possible that you will want or need to update at some point

You may also want to know more so that you can protect yourself from unwanted updates

The online documentation follows

Once you install Android Studio, it's easy to keep the Android Studio IDE and various SDK tools up to date with automatic updates for Android Studio and the Android SDK Manager.

Android Studio Update Channels

Android Studio notifies you with a small bubble dialog when an update is available for the IDE, but you can manually check for updates by clicking **Help** > **Check for Update** (on Mac, **Android Studio** > **Check for Updates**).

Updates for Android Studio are available from the following release channels:

Canary Channel: These are bleeding-edge releases, updated roughly weekly.

Although these builds are subject to more bugs, they do get tested and they want to offer early access so you can try new features and provide feedback.

This channel is **not recommended for production development**.

Dev Channel: These are hand-picked canary builds that survived a full round of internal testing.

Beta Channel: These are release candidates based on stable canary builds, released to get feedback before going into the stable channel.

Stable Channel: The official stable release that is available for download at developer.android.com/studio.

By default, Android Studio offers updates from the Stable Channel.

If you'd like to try one of the other versions of Android Studio—known collectively as the Preview Channels—you can choose to receive updates from one of those instead.

If your system crashes midway through the semester and you have to re-install:

Most likely you will just take the latest stable version, even if it's later than the one you installed initially

If you remember the version you originally installed, it may be possible to re-install it

As you recognize, either way, you may have follow-on problems as a result

So, there's no real reason to mess with this…

To change your update channel, proceed as follows:

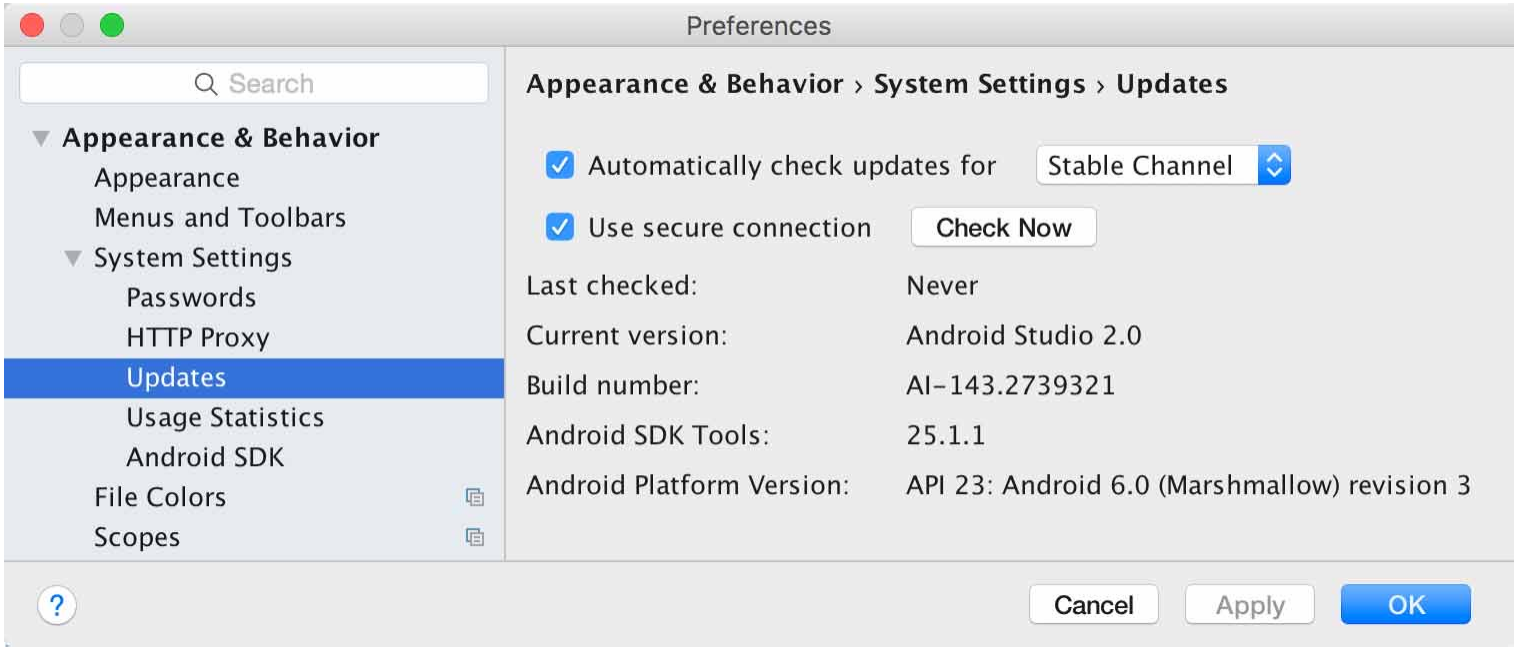
Open the **Preferences** window by clicking **File > Settings** (on Mac, **Android Studio > Preferences**).

In the left panel, click **Appearance & Behavior > System Settings > Updates**.

Be sure that **Automatically check for updates** is checked, then select a channel from the drop-down list (see figure 1).

Click **Apply** or **OK**.

[See the following overhead]


Preferences
Search
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Updates
Usage Statistics
Android SDK
File Colors
Scopes
Appearance & Behavior › System Settings › Updates
Automatically check updates for
Stable Channel
Use secure connection
Check Now
Last checked:
Never
Current version:
Android Studio 2.0
Build number:
AI-143.2739321
Android SDK Tools:
25.1.1
Android Platform Version:
API 23: Android 6.0 (Marshmallow) revision 3
Cancel
Apply
OK

Android SDK Tool Updates

The Android SDK Manager provides the SDK tools, platforms, and other components you need to develop your apps.

To open the SDK Manager, click **Tools > Android > SDK Manager** or click **SDK Manager** in the toolbar.

When an update is available for a package you already have, a dash appears in the check box next to the package.

To update an item or install a new one, click the check box so it shows a checkmark.

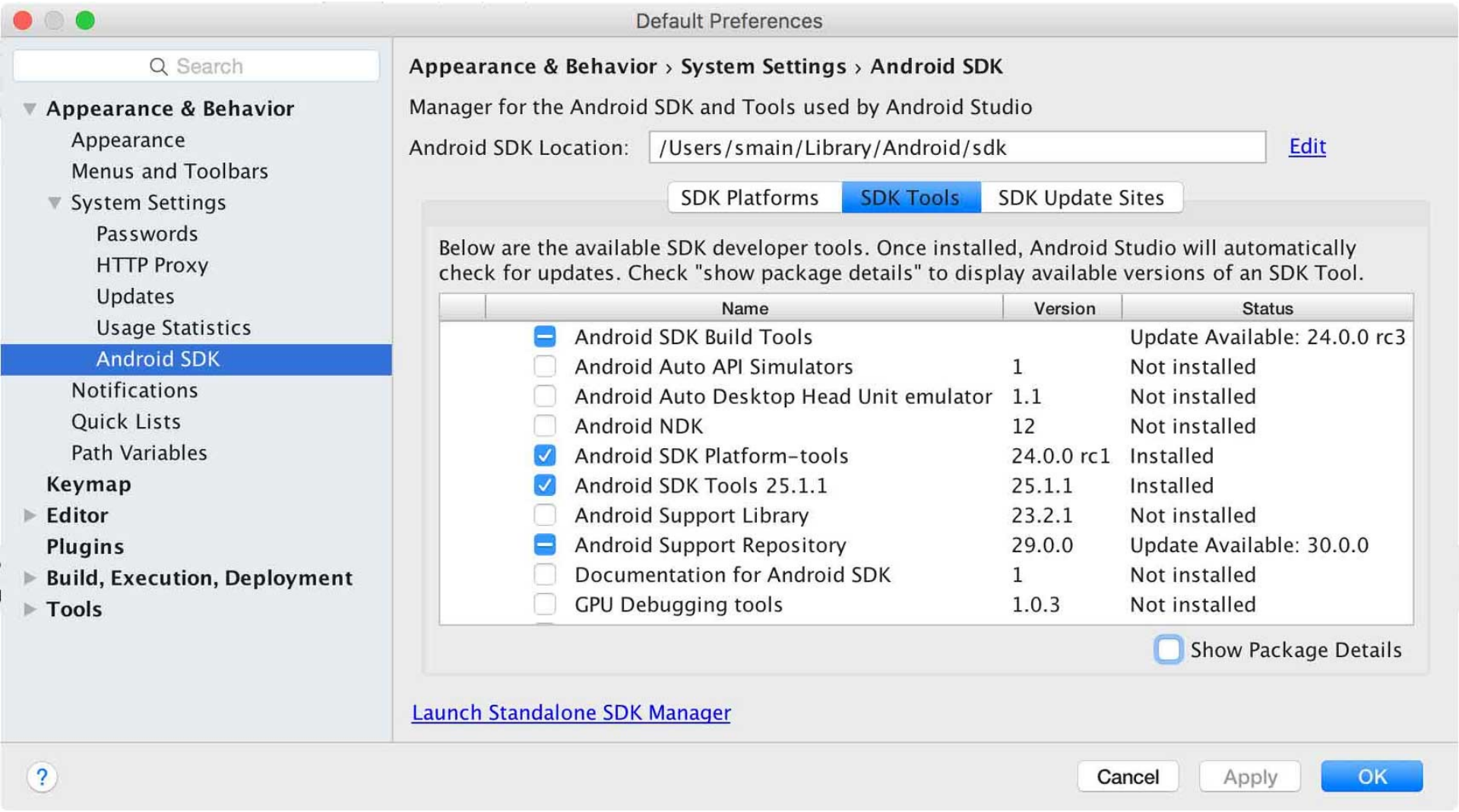
To uninstall a package, click to clear the check box.

Pending updates are indicated in the left column with a download icon.

Pending removals are indicated with a red cross.

To update the selected packages, click **Apply** or **OK**, then agree to any license agreements.

[See the following overhead]


Default Preferences
Search
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Updates
Usage Statistics
Android SDK
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: /Users/smain/Library/Android/sdk
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build Tools
Update Available: 24.0.0 rc3
Android Auto API Simulators
1
Not installed
Android Auto Desktop Head Unit emulator
1.1
Not installed
Android NDK
12
Not installed
Android SDK Platform-tools
24.0.0 rc1
Installed
Android SDK Tools 25.1.1
25.1.1
Installed
Android Support Library
23.2.1
Not installed
Android Support Repository
29.0.0
Update Available: 30.0.0
Documentation for Android SDK
1
Not installed
GPU Debugging tools
1.0.3
Not installed
Show Package Details
Launch Standalone SDK Manager
Cancel
Apply
OK

Note on Android Studio

In the past, making sure you had the right drivers for a device attached to your development machine could be an issue

It was the one critical thing that would require going through the update interface

It appears to go much more smoothly now

However, problems are always possible

2.3 “Workflow Basics”

This topic is very general and should be “non-threatening”

The idea is that there is a logical order of activities, a path to be followed, in order to go from an idea to an implementation

There’s no harm in a little high level visualization of what that path consists of

Developer Workflow Basics

The workflow to develop an app for Android is conceptually the same as other app platforms.

However, to efficiently build a well-designed app for Android, you need some specialized tools.

The following list provides an overview of the process to build an Android app and includes links to some Android Studio tools you should use during each phase of development.

1. Set up your workspace

This is the phase you probably already finished: Install Android Studio and create a project.

This is your first homework as of this Learning Unit 1.

For a walkthrough with Android Studio that teaches some Android development fundamentals, also check out the guide to Building Your First App.

2. Write your app

Now you can get to work.

Android Studio includes a variety of tools and intelligence to help you work faster, write quality code, design a UI, and create resources for different device types.

For more information about the tools and features available, see Write Your App.

3. Build and run

During this phase, you build your project into a debuggable APK package that you can install and run on the emulator or an Android-powered device.

For more information about how to run your code, see Build and Run Your App.

You can also begin customizing your build.

For example, you can create build variants that produce different types of APKs from the same project, and shrink your code and resources to make your APK file smaller.

For an introduction to customizing your build, see Configure Your Build.

4. Debug, profile, and test

This is the iterative phase in which you continue writing your app but with a focus on eliminating bugs and optimizing app performance.

Of course, creating tests will help you in those endeavors.

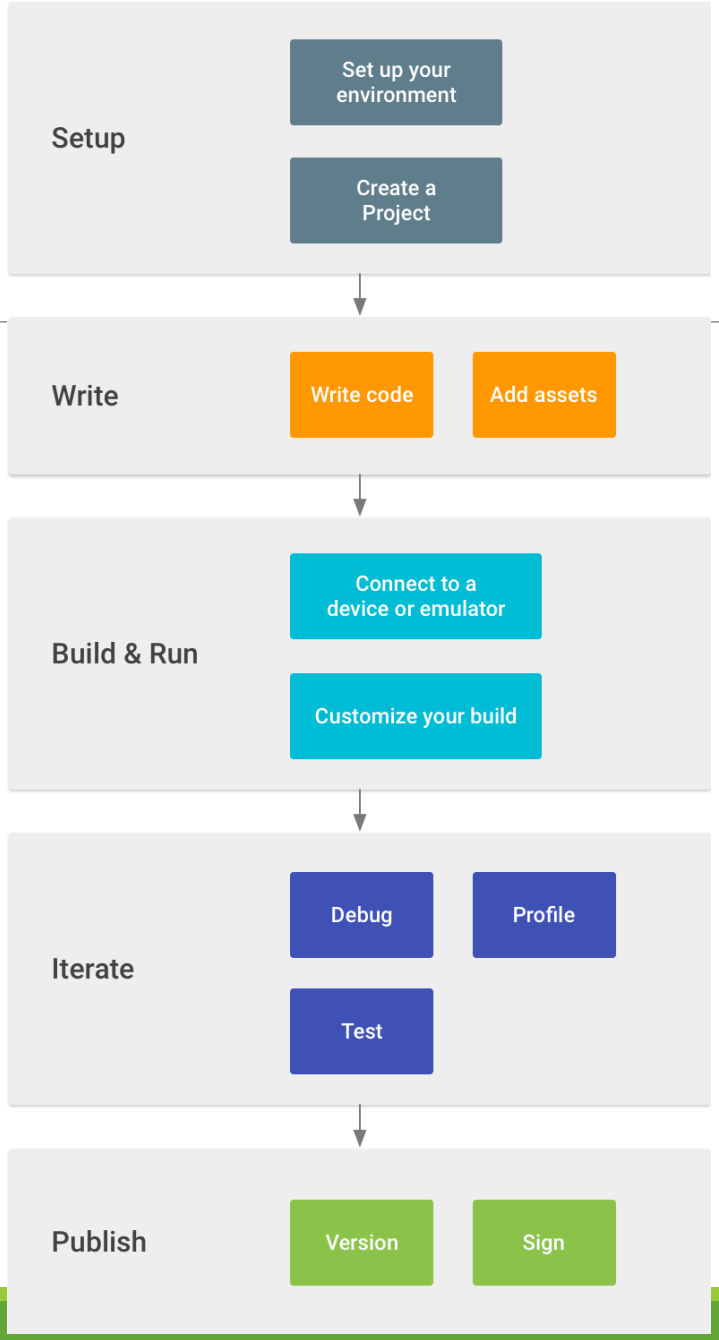
For information about basic debugging tasks, read Debug Your App and Write and View Logs.

To view and analyze various performance metrics such as memory usage, network traffic, CPU impact, and more, use Android Monitor.

5. Publish

When you're ready to release your app to users, there are just a few more things to consider, such as versioning your app and signing it with a key.

For more information, see the Publishing Overview.


Setup
Set up your environment
Create a Project
Write
Write code
Add assets
Build & Run
Connect to a device or emulator
Customize your build
Iterate
Debug
Profile
Test
Publish
Version
Sign

Concluding Comments

You are probably already familiar with the repetitive, cyclical nature of code development

You are constantly having to go back and debug, fix old features, and add new ones

Code, run, debug repeat

A colorful box diagram makes it look like fun rather than frustration ☺

The final step, publishing, is an option

The typical student project is not something that is going to acquire a mass, paying audience, but the principle is important and valid

Once you learn how to use Android, the potential is there to create something other people might want

GooglePlay provides a means for making it available to anyone in the world

In the vast infrastructure of Android, hardware, telecommunications, development software, etc., this is where you might fit in:

App developer

Java Editor	Performs auto-completion, real-time syntax checking, and shows API documentation. The editor's productivity enhancements can decrease the time required to program an Android app.
Layout Editor	Allows the UI to be created by dragging and dropping buttons, checkboxes, and other UI components onto the UI screen. The editor also shows what the UI will look like on different screen sizes and orientations.
Theme Editor	For creating and modifying an app's font properties and color schemes. A theme gives the app a unified and consistent look and feel.
Debugger	For setting breakpoints in the code and watching code execute line-by-line. Apps can be debugged while running on an actual Android device.
Android SDK Manager	Allows developers to install new and updated SDK tools, platforms, and other components necessary to develop Android apps. The build tools and at least one Android platform must be installed before apps can be created.
Android AVD Manager	Allows developers to create Android Virtual Devices (AVDs). An AVD defines the hardware characteristics of an Android device that an app is to be tested on. AVDs can be created for a wide range of phones and tablets.
Android emulator	Simulates an Android device on a development computer using an AVD configuration. Android Studio can run and debug an app on the emulator or on an actual Android device.

Next

1.3 Android Application