Android Studio Cheat Sheet

Running and debugging code

Launch an application

If a file contains a class that has a `main(String[] args)` method, right-clicking in the editor will show a "*Run main()*" option. The same option will also appear if you right-click on that class in the "*Project*" view on the left.

Debug an application

The same menus containing "*Run main()*" also contain "*Debug main()*".

By default, Android Studio will run your code in debug mode and show the debug pane, without doing anything else.

To break when an exception is thrown, open the *Breakpoints* window via the "*View Breakpoints*" icon that appears in the debug pane, then check the "*Enabled*" box under "*Any exception*".

To break at some given point in your program, add a breakpoint there by clicking in the left margin, which will make a red circle appear representing a breakpoint. You can also do this using the application menu with *Run* > *Toggle line breakpoint*.

There are many kinds of breakpoints that can be useful in debugging, such as breaking whenever a specific field is changed. We strongly encourage you to read the documentation.

Run tests

If a file contains unit tests, you can right-click on the code of a test to show a "*Run*" option for that specific test, right-click on the code outside of a test to show a "*Run*" option for all tests in the file, or right-click on the file in the "*Project*" view on the left to also show a "*Run*" option for all tests in the file.

Debug a test

You can debug tests in the same way you can debug other kinds of code, with breakpoints either in tests or in the code under test.

Generating common code

Let's say you want a Point class with X and Y values. Due to how Java works, you'll need fields to store the values, a constructor to set the value, `equals` and `hashCode` methods to use the values in equality comparisons, and maybe getters and setters if you want a mutable class.

You don't need to write all of that yourself! You can let Android Studio generate it.

Write your class declaration including fields, then right-click the class name and click on "*Generate…*". You can then choose to generate a constructor, getters and setters, equals and hashCode, and more! When presented with a list of fields, use Ctrl+Click (or Command-Click on Mac) to select multiple fields.

Refactoring code

Rename

Right-click on a field, method, class, or anything else that has a name, and choose Refactor > Rename. You can then write the name you want, hit Enter, and Android Studio will do the

rename for you. This operation knows about language semantics, so it will only rename text that referred to the specific thing you renamed. For instance, if two classes have a field “name”, renaming one will not affect the other.

When renaming a lot, the Shift+F6 keyboard shortcut can come in handy instead of using the menu.

Quick fixes with Alt+Enter

Hit Alt+Enter (also known as Option-Enter on Mac) when your cursor is on some code to see what actions are available, such as quick fixes for warnings. This is a very useful shortcut!

Format code

Hit Ctrl+Alt+L (or Command-Option-L on Mac) to format a file. We encourage you to read the documentation to see all formatting options; but regardless of your preferences, even the default is better than no formatting.

Analyze code

We recommend you to run code analysis on your code. To do so, go to *Analyze* > *Inspect code …*

You will be asked on which part of the project the analysis needs to be performed. Once you validate, Android Studio will perform the analysis and open a new tab at the bottom of the window with all errors and warnings it found. You should go through them carefully as this list usually contains all sorts of code smells that lead to maintainability issues or, worse, bugs. You can tell Android Studio to hide a warning if you have A VERY GOOD reason to do so in the contextual menu accessible via a right-click. In this same menu, you can usually find the “light bulb” icon that proposes a solution to the warning/error it found.

Integration with other tools

Add a Gradle dependency

First of all, when opening/importing a gradle project in Android Studio, select the `build.gradle` file. In this way, Android Studio will correctly open it as a Gradle project.

If you need to add a new dependency, you can access the “project structure” menu either from *File* > *Project Structure…* or by hitting Ctrl+; (Command-; on Mac). You can also do it by modifying by hand the `build.gradle` file. To re-sync gradle, go to *File* > *Sync projects with Gradle files*. If you modify the `build.gradle` file, Android Studio will open a toolbar asking you to re-sync and offering to you a button to do so.

View Git history

You can access a git user interface from the *version control* tab at the bottom left of the window. From there, you can view the files that were modified since the last commit and, if you select one of them, you will see the actual lines that changed in the panel on the right. To commit, click on the green “tick” sign on the left or hit Ctrl+K (Command-K on Mac). A window appears where you can add the files you want to the commit, even line by line (at the bottom), and add a commit message before validating your commit.

At the bottom right of the main window, you can see on which git branch you are at the moment. You can open the branches menu by clicking on it. From there, you can push your current branch by clicking on it to open the submenu, or checkout to another branch (a new one or an existing one). You can also push from anywhere by hitting Ctrl+Shift+K (Command-Shift-K on Mac). Everything about Git is accessible via the *VCS* application menu.

If you need resources about Git, do not hesitate to search on Google, there are plenty available. Here you have a starting doc if needed: https://guides.github.com/introduction/git-handbook/

Other

Finding functionality

Hit Ctrl+Shift+A (Command-Shift-A on Mac) to open the *find action* menu. Iit is also accessible using the application menu: *Help* > *Find Action...*. You can then type the name of any action you need e.g., “comment a line”, “push”, ... and select it to perform it. From there, you can also learn the keyboard shortcut that corresponds to a given action.

Using the console

You can open a Terminal instance in the Android Studio window by pressing Alt+F12 (Option-F12 on Mac) or finding it via the *find an action* presented right above. This opens an instance of your system’s terminal in Android Studio. This is exactly the same terminal as you use in your system, not a special Android one, but in the Android Studio window to avoid extra window switching.

Advice from SwEng students

Please use this section to share with your classmates tips & tricks that work well for you or for others. Cool features of Android Studio, little-known settings, etc.
