- While it is tempting to plunge into running the example application created in the previous chapter, doing so involves using aspects of the Android Studio user interface which are best described in advance.

- Android Studio is a powerful and feature rich development environment that is, to a large extent, intuitive to use.

- That being said, taking the time now to gain familiarity with the layout and organization of the Android Studio user interface will considerably shorten the learning curve in later chapters.

- With this in mind, this chapter will provide an initial overview of the various areas and components that make up the Android Studio environment.

The Welcome Screen

- The welcome screen (Figure 5-1) is displayed any time that Android Studio is running with no projects currently open (open projects can be closed at any time by selecting the File > Close Project menu option).

- If Android Studio was previously exited while a project was still open, the tool will by-pass the welcome screen next time it is launched, automatically opening the previously active project.

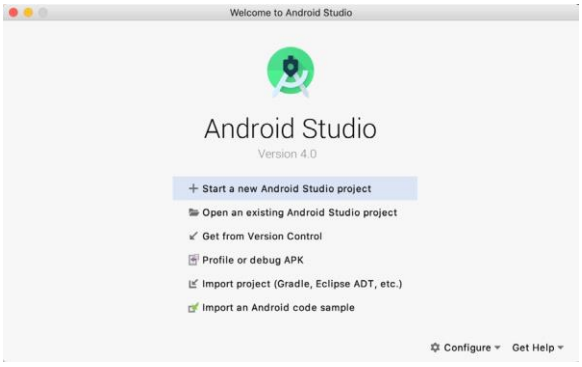


Figure 5-1

- In addition to a list of recent projects, the Quick Start menu provides a range of options for performing tasks such as opening, creating and importing projects along with access to projects currently under version control.

- In addition, the Configure menu at the bottom of the window provides access to the SDK Manager along with a vast array of settings and configuration options.

- A review of these options will quickly reveal that there is almost no aspect of Android Studio that cannot be configured and tailored to your specific needs.

- The Configure menu also includes an option to check if updates to Android Studio are available for download.

(continued)

The Main Window

- When a new project is created, or an existing one opened, the Android Studio main window will appear.
- When multiple projects are open simultaneously, each will be assigned its own main window.
- The precise configuration of the window will vary depending on which tools and panels were displayed the last time the project was open, but will typically resemble that of Figure 5-2.

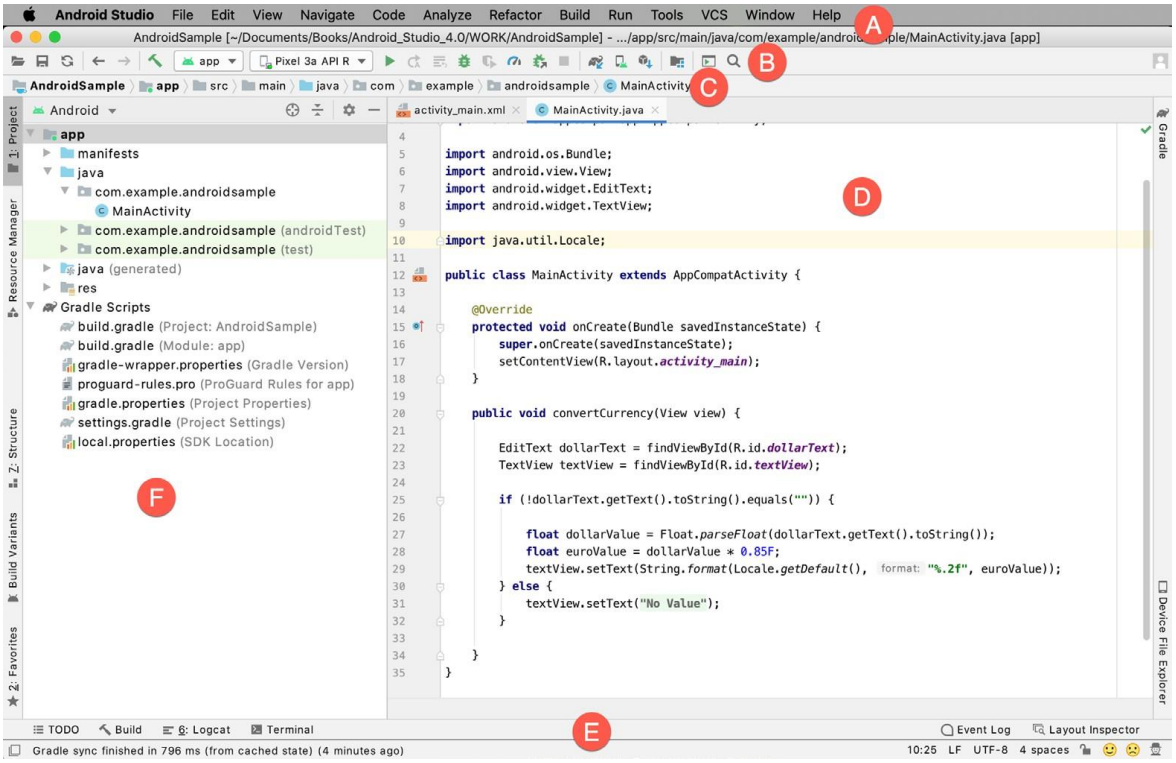


Figure 5-2

- The various elements of the main window can be summarized as follows:
  - A – Menu Bar – Contains a range of menus for performing tasks within the Android Studio environment.
  - B – Toolbar – A selection of shortcuts to frequently performed actions.

    The toolbar buttons provide quicker access to a select group of menu bar actions. The toolbar can be customized by right-clicking on the bar and selecting the Customize Menus and Toolbars… menu option.

    If the toolbar is not visible, it can be displayed using the View > Appearance > Toolbar menu option.
  - C – Navigation Bar – The navigation bar provides a convenient way to move around the files and folders that make up the project.

Clicking on an element in the navigation bar will drop down a menu listing the subfolders and files at that location ready for selection. This provides an alternative to the Project tool window.

Hide and display this bar using the View > Appearance > Navigation Bar menu option.

- D – Editor Window – The editor window displays the content of the file on which the developer is currently working.

  What gets displayed in this location, however, is subject to context.

  When editing code, for example, the code editor will appear.

  When working on a user interface layout file, on the other hand, the user interface Layout Editor tool will appear.

  When multiple files are open, each file is represented by a tab located along the top edge of the editor as shown in Figure 5-3.

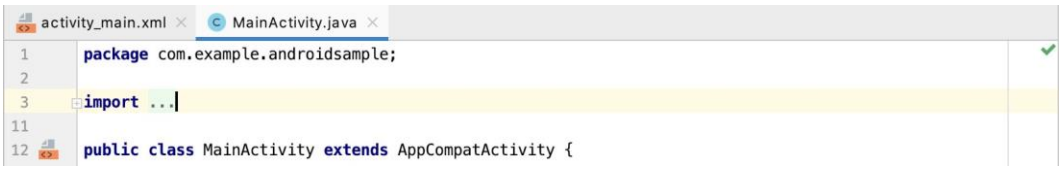


Figure 5-3

- E – Status Bar – The status bar displays informational messages about the project and the activities of Android Studio together with the tools menu button located in the far left corner.

  Hovering over items in the status bar will provide a description of that field.

  Many fields are interactive, allowing the user to click to perform tasks or obtain more detailed status information.

- F – Project Tool Window – The project tool window provides a hierarchical overview of the project file structure allowing navigation to specific files and folders to be performed.

  The toolbar can be used to display the project in a number of different ways.

  The default setting is the Android view which is the mode primarily used in the remainder of our studies.

  The project tool window is just one of a number of tool windows available within the Android Studio environment.

The Tool Windows

- In addition to the project view tool window, Android Studio also includes a number of other windows which, when enabled, are displayed along the bottom and sides of the main window.

- The tool window quick access menu can be accessed by hovering the mouse pointer over the button

located in the far left-hand corner of the status bar (Figure 5-4) without clicking the mouse button.

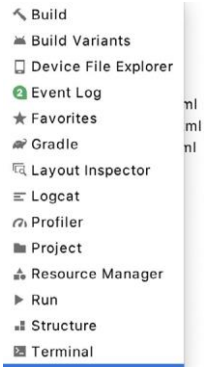


Figure 5-4

- Selecting an item from the quick access menu will cause the corresponding tool window to appear within the main window.
- Alternatively, a set of tool window bars can be displayed by clicking on the quick access menu icon in the status bar.
- These bars appear along the left, right and bottom edges of the main window (as indicated by the arrows in Figure 5-5) and contain buttons for showing and hiding each of the tool windows.
- When the tool window bars are displayed, a second click on the button in the status bar will hide them.
- Clicking on a button will display the corresponding tool window while a second click will hide the window.

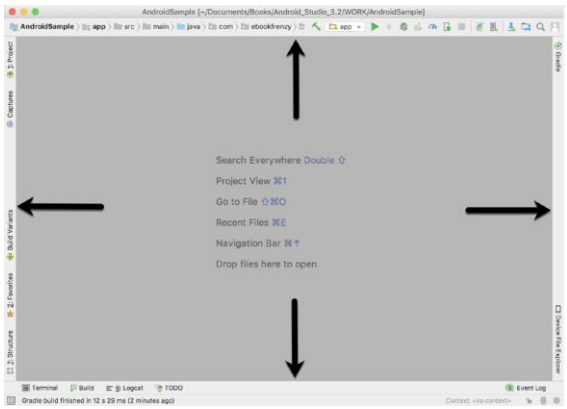


Figure 5-5

- Buttons prefixed with a number (for example 1: Project) indicate that the tool window may also be displayed by pressing the Alt key on the keyboard (or the Command key for macOS) together with the corresponding number.
- The location of a button in a tool window bar indicates the side of the window against which the window will appear when displayed.
- These positions can be changed by clicking and dragging the buttons to different locations in other

window tool bars.

- Each tool window has its own toolbar along the top edge.
- The buttons within these toolbars vary from one tool to the next, though all tool windows contain a settings option, represented by the cog icon, which allows various aspects of the window to be changed.
- Figure 5-6 shows the settings menu for the project view tool window.
- Options are available, for example, to undock a window and to allow it to float outside of the boundaries of the Android Studio main window and to move and resize the tool panel.

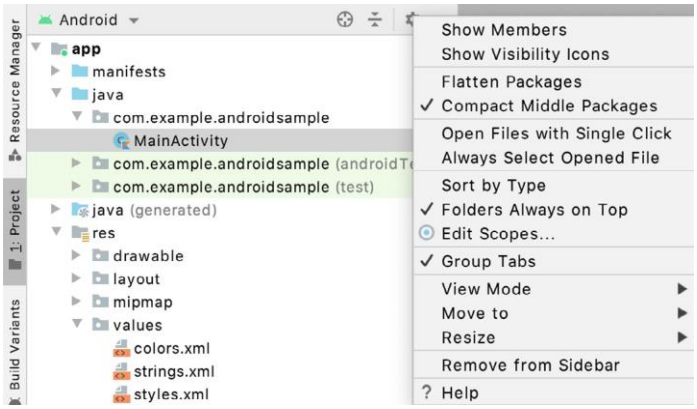


Figure 5-6

- All of the windows also include a far right button on the toolbar providing an additional way to hide the tool window from view.
- A search of the items within a tool window can be performed simply by giving that window focus by clicking in it and then typing the search term (for example the name of a file in the Project tool window).
- A search box will appear in the window's tool bar and items matching the search highlighted.
- Android Studio offers a wide range of tool windows, the most commonly used of which are as follows:
  - Project – The project view provides an overview of the file structure that makes up the project allowing for quick navigation between files.

    Generally, double-clicking on a file in the project view will cause that file to be loaded into the appropriate editing tool.
  - Structure – The structure tool provides a high level view of the structure of the source file currently displayed in the editor.

    This information includes a list of items such as classes, methods and variables in the file.

    Selecting an item from the structure list will take you to that location in the source file in the editor window.

- Favorites – A variety of project items can be added to the favorites list.

  Right-clicking on a file in the project view, for example, provides access to an Add to Favorites menu option.

  Similarly, a method in a source file can be added as a favorite by right-clicking on it in the Structure tool window.

  Anything added to a Favorites list can be accessed through this Favorites tool window.

- Build Variants – The build variants tool window provides a quick way to configure different build targets for the current application project (for example different builds for debugging and release versions of the application, or multiple builds to target different device categories).

- TODO – As the name suggests, this tool provides a place to review items that have yet to be completed on the project.

  Android Studio compiles this list by scanning the source files that make up the project to look for comments that match specified TODO patterns.

  These patterns can be reviewed and changed by selecting the File > Settings… menu option (Android Studio > Preferences… on macOS) and navigating to the TODO page listed under Editor.

- Logcat – The Logcat tool window provides access to the monitoring log output from a running application in addition to options for taking screenshots and videos of the application and stopping and restarting a process.

- Terminal – Provides access to a terminal window on the system on which Android Studio is running.

  On Windows systems this is the Command Prompt interface, while on Linux and macOS systems this takes the form of a Terminal prompt.

- Build - The build tool windows displays information about the build process while a project is being compiled and packaged and displays details of any errors encountered.

- Run – The run tool window becomes available when an application is currently running and provides a view of the results of the run together with options to stop or restart a running process.

  If an application is failing to install and run on a device or emulator, this window will typically provide diagnostic information relating to the problem.

- Event Log – The event log window displays messages relating to events and activities performed within Android Studio.

  The successful build of a project, for example, or the fact that an application is now running will be reported within this tool window.

- Gradle – The Gradle tool window provides a view onto the Gradle tasks that make up the project build configuration.

  The window lists the tasks that are involved in compiling the various elements of the project into an

executable application.

Right-click on a top level Gradle task and select the Open Gradle Config menu option to load the Gradle build file for the current project into the editor.

Gradle will be covered in greater detail later.

- Profiler – The Android Profiler tool window provides realtime monitoring and analysis tools for identifying performance issues within running apps, including CPU, memory and network usage.

  This option becomes available when an app is currently running.

- Device File Explorer – The Device File Explorer tool window provides direct access to the filesystem of the currently connected Android device or emulator allowing the filesystem to be browsed and files copied to the local filesystem.

- Resource Manager - A tool for adding and managing resources and assets such as images, colors and layout files contained with the project.

Android Studio Keyboard Shortcuts

- Android Studio includes an abundance of keyboard shortcuts designed to save time when performing common tasks.

- A full keyboard shortcut keymap listing can be viewed and printed from within the Android Studio project window by selecting the Help > Keymap Reference menu option.

Switcher and Recent Files Navigation

- Another useful mechanism for navigating within the Android Studio main window involves the use of the Switcher.

- Accessed via the Ctrl-Tab keyboard shortcut, the switcher appears as a panel listing both the tool windows and currently open files (Figure 5-7).

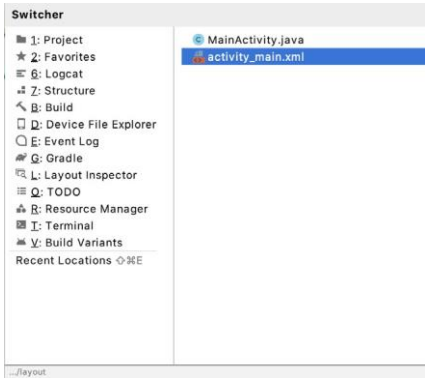


Figure 5-7

- Once displayed, the switcher will remain visible for as long as the Ctrl key remains depressed.

- Repeatedly tapping the Tab key while holding down the Ctrl key will cycle through the various selection

options, while releasing the Ctrl key causes the currently highlighted item to be selected and displayed within the main window.

- In addition to the switcher, navigation to recently opened files is provided by the Recent Files panel (Figure 5-8).

- This can be accessed using the Ctrl-E keyboard shortcut (Cmd-E on macOS).

- Once displayed, either the mouse pointer can be used to select an option or, alternatively, the keyboard arrow keys used to scroll through the file name and tool window options.

- Pressing the Enter key will select the currently highlighted item.

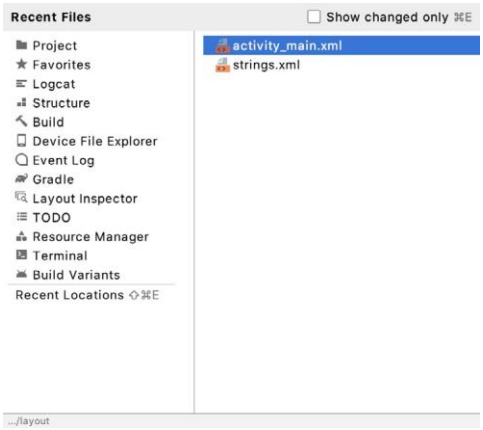


Figure 5-8

Changing the Android Studio Theme

- The overall theme of the Android Studio environment may be changed either from the welcome screen using the Configure > Settings option, or via the File > Settings… menu option (Android Studio > Preferences… on macOS) of the main window.

- Once the settings dialog is displayed, select the Appearance option in the left-hand panel and then change the setting of the Theme menu before clicking on the Apply button.

- The themes available will depend on the platform but usually include options such as Light, IntelliJ, Windows, High Contrast and Darcula, and Figure 5-9 shows an example of the Darcula main window:

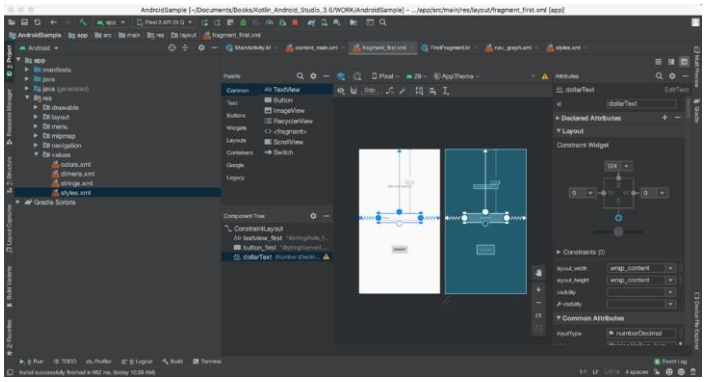


Figure 5-9

The Basics of the Android Studio Code Editor

- Developing applications for Android involves a considerable amount of programming work which, by definition, involves typing, reviewing and modifying lines of code.

- It should come as no surprise that the majority of a developer's time spent using Android Studio will typically involve editing code within the editor window.

- The modern code editor needs to go far beyond the original basics of typing, deleting, cutting and pasting.

- Today the usefulness of a code editor is generally gauged by factors such as the amount by which it reduces the typing required by the programmer, ease of navigation through large source code files and the editor's ability to detect and highlight programming errors in real-time as the code is being written.

- As will become evident in this chapter, these are just a few of the areas in which the Android Studio editor excels.

- While not an exhaustive overview of the features of the Android Studio editor, this chapter aims to provide a guide to the key features of the tool.

- Experienced programmers will find that some of these features are common to most code editors available today, while a number are unique to this particular editing environment.

- The Android Studio editor appears in the center of the main window when a Java, Kotlin, XML or other text based file is selected for editing.

- Figure 5-10, for example, shows a typical editor session with a Java source code file loaded:

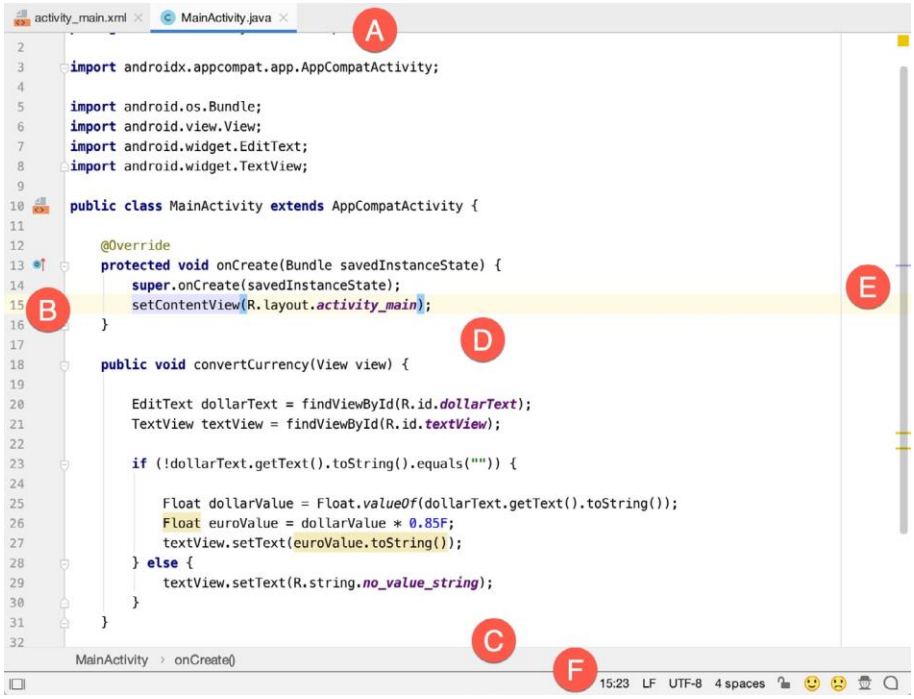


Figure 5-10

- The elements that comprise the editor window can be summarized as follows:

  - A – Document Tabs – Android Studio is capable of holding multiple files open for editing at any one time.

    As each file is opened, it is assigned a document tab displaying the file name in the tab bar located along the top edge of the editor window.

    A small dropdown menu will appear in the far right-hand corner of the tab bar when there is insufficient room to display all of the tabs.

    Clicking on this menu will drop down a list of additional open files.

    A wavy red line underneath a file name in a tab indicates that the code in the file contains one or more errors that need to be addressed before the project can be compiled and run.

    Switching between files is simply a matter of clicking on the corresponding tab or using the Alt-Left and Alt-Right keyboard shortcuts.

    Navigation between files may also be performed using the Switcher mechanism (accessible via the Ctrl-Tab keyboard shortcut).

    To detach an editor panel from the Android Studio main window so that it appears in a separate window, click on the tab and drag it to an area on the desktop outside of the main window.

    To return the editor to the main window, click on the file tab in the separated editor window and drag and drop it onto the original editor tab bar in the main window.

  - B – The Editor Gutter Area - The gutter area is used by the editor to display informational icons and controls.

    Some typical items, among others, which appear in this gutter area are debugging breakpoint markers, controls to fold and unfold blocks of code, bookmarks, change markers and line numbers.

    Line numbers are switched on by default but may be disabled by right-clicking in the gutter and selecting the Show Line Numbers menu option.

  - C – Code Structure Location - This bar at the bottom of the editor displays the current position of the cursor as it relates to the overall structure of the code.

    In the following figure, for example, the bar indicates that a setOnClickListener method call is currently being edited, and that this line of code is contained within the onCreate method of the MainActivity class.

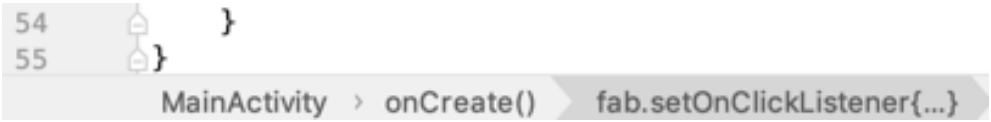


Figure 5-11

Selecting an element within the bar will move the cursor to the corresponding location within the code file.

For example, selecting the onCreate() entry will move the cursor to the top of the onCreate method within the source code.

- D – The Editor Area – This is the main area where the code is displayed, entered and edited by the user.

  Later sections of this chapter will cover the key features of the editing area in detail.

- E – The Validation and Marker Sidebar – Android Studio incorporates a feature referred to as "on-the-fly code analysis".

  What this essentially means is that as you are typing code, the editor is analyzing the code to check for warnings and syntax errors.

  The indicator at the top of the validation sidebar will change from a green check mark (no warnings or errors detected) to a yellow square (warnings detected) or red alert icon (errors have been detected).

  Clicking on this indicator will display a popup containing a summary of the issues found with the code in the editor as illustrated in Figure 5-12:

Figure 5-12

The sidebar also displays markers at the locations where issues have been detected using the same color coding.

Hovering the mouse pointer over a marker when the line of code is visible in the editor area will display a popup containing a description of the issue (Figure 5-13):

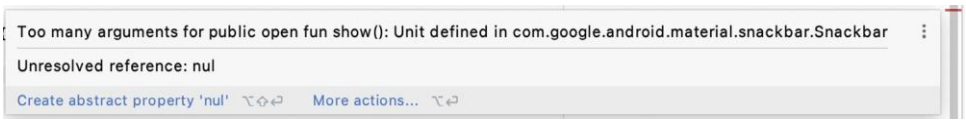


Figure 5-13

Hovering the mouse pointer over a marker for a line of code which is currently scrolled out of the viewing area of the editor will display a "lens" overlay containing the block of code where the problem is located (Figure 5-14) allowing it to be viewed without the necessity to scroll to that location in the editor:

(continued)

Figure 5-14

It is also worth noting that the lens overlay is not limited to warnings and errors in the sidebar.

Hovering over any part of the sidebar will result in a lens appearing containing the code present at that location within the source file.

- F – The Status Bar – Though the status bar is actually part of the main window, as opposed to the editor, it does contain some information about the currently active editing session.

  This information includes the current position of the cursor in terms of lines and characters and the encoding format of the file (UTF-8, ASCII etc.).

  Clicking on these values in the status bar allows the corresponding setting to be changed.

  Clicking on the line number, for example, displays the Go to Line dialog.

- Having provided an overview of the elements that comprise the Android Studio editor, the remainder of this chapter will explore the key features of the editing environment in more detail.

Splitting the Editor Window

- By default, the editor will display a single panel showing the content of the currently selected file.

- A particularly useful feature when working simultaneously with multiple source code files is the ability to split the editor into multiple panes.

- To split the editor, right-click on a file tab within the editor window and select either the Split Vertically or Split Horizontally menu option.

- Figure 5-15, for example, shows the splitter in action with the editor split into three panels:

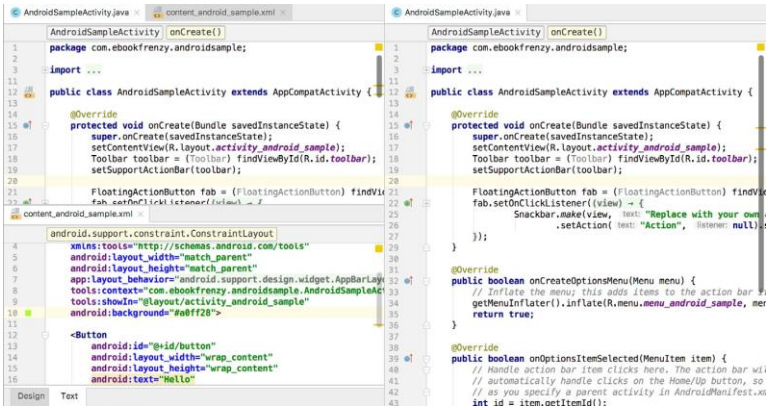


Figure 5-15

- The orientation of a split panel may be changed at any time by right-clicking on the corresponding tab and selecting the Change Splitter Orientation menu option.

- Repeat these steps to unsplit a single panel, this time selecting the Unsplit option from the menu.

- All of the split panels may be removed by right-clicking on any tab and selecting the Unsplit All menu option.

- Window splitting may be used to display different files, or to provide multiple windows onto the same file, allowing different areas of the same file to be viewed and edited concurrently.

Code Completion

- The Android Studio editor has a considerable amount of built-in knowledge of Java programming syntax and the classes and methods that make up the Android SDK, as well as knowledge of your own code base.

- As code is typed, the editor scans what is being typed and, where appropriate, makes suggestions with regard to what might be needed to complete a statement or reference.

- When a completion suggestion is detected by the editor, a panel will appear containing a list of suggestions. In Figure 5-16, for example, the editor is suggesting possibilities for the beginning of a String declaration:

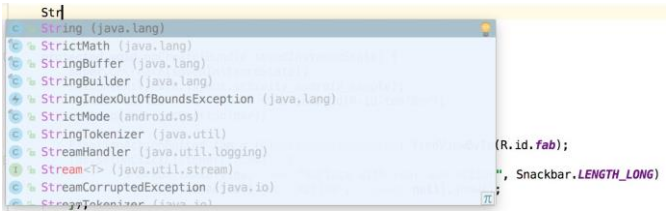


Figure 5-16

- If none of the auto completion suggestions are correct, simply keep typing and the editor will continue to refine the suggestions where appropriate.

- To accept the top most suggestion, simply press the Enter or Tab key on the keyboard.

- To select a different suggestion, use the arrow keys to move up and down the list, once again using the Enter or Tab key to select the highlighted item.

- Completion suggestions can be manually invoked using the Ctrl-Space keyboard sequence.

- This can be useful when changing a word or declaration in the editor.

- When the cursor is positioned over a word in the editor, that word will automatically highlight.

- Pressing Ctrl-Space will display a list of alternate suggestions.

- To replace the current word with the currently highlighted item in the suggestion list, simply press the Tab key.

- In addition to the real-time auto completion feature, the Android Studio editor also offers a system referred to as Smart Completion.

- Smart completion is invoked using the Shift-Ctrl-Space keyboard sequence and, when selected, will provide more detailed suggestions based on the current context of the code.

- Pressing the Shift-Ctrl- Space shortcut sequence a second time will provide more suggestions from a wider range of possibilities.

- Code completion can be a matter of personal preference for many programmers.

- In recognition of this fact, Android Studio provides a high level of control over the auto completion settings.

- These can be viewed and modified by selecting the File > Settings… menu option (or Android Studio > Preferences… on macOS) and choosing Editor > General > Code Completion from the settings panel as shown in Figure 5-17:

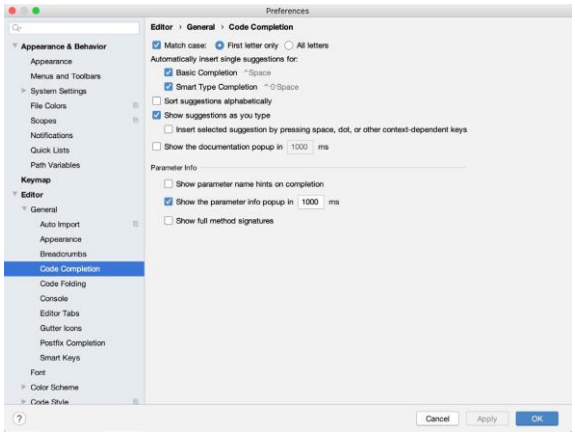


Figure 5-17

Statement Completion

- Another form of auto completion provided by the Android Studio editor is statement completion.

- This can be used to automatically fill out the parentheses and braces for items such as methods and loop statements.

- Statement completion is invoked using the Shift-Ctrl-Enter (Shift-Cmd-Enter on macOS) keyboard sequence. Consider for example the following code:

```
myMethod()
```

- Having typed this code into the editor, triggering statement completion will cause the editor to automatically add the braces to the method:

```
myMethod()
{
}
```

Parameter Information

- It is also possible to ask the editor to provide information about the argument parameters accepted by a method.

- With the cursor positioned between the brackets of a method call, the Ctrl-P (Cmd-P on macOS) keyboard sequence will display the parameters known to be accepted by that method, with the most likely suggestion highlighted in bold:

Figure 5-18

Parameter Name Hints

- The code editor may be configured to display parameter name hints within method calls.

- Figure 5-19, for example, highlights the parameter name hints within the calls to the `make()` and `setAction()` methods of the Snackbar class:

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener((view) → {
        Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction( text: "Action",  listener: null).show();
});
```

Figure 5-19

- The settings for this mode may be configured by selecting the File > Settings (Android Studio > Preferences on macOS) menu option followed by Editor > Appearance in the left-hand panel.

- On the Appearance screen, enable or disable the Show parameter name hints option.

- To adjust the hint settings, click on the Configure... button, select the programming language and make any necessary adjustments.

Code Generation

- In addition to completing code as it is typed the editor can, under certain conditions, also generate code for you.

- The list of available code generation options shown in Figure 5-20 can be accessed using the Alt-Insert (Cmd-N on macOS) keyboard shortcut when the cursor is at the location in the file where the code is to be generated.

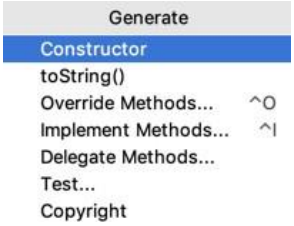


Figure 5-20

- For the purposes of an example, consider a situation where we want to be notified when an Activity in our project is about to be destroyed by the operating system.

- As will be outlined in later, this can be achieved by overriding the onStop() lifecycle method of the Activity superclass.

- To have Android Studio generate a stub method for this, simply select the Override Methods… option from the code generation list and select the onStop() method from the resulting list of available methods:

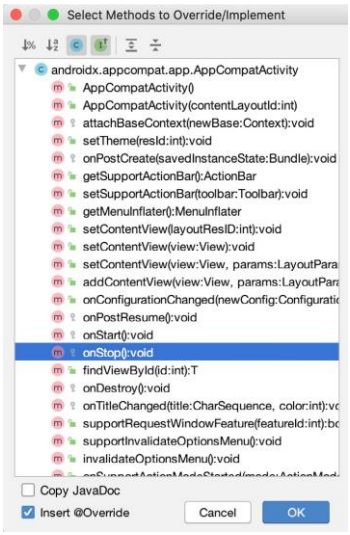


Figure 5-21

- Having selected the method to override, clicking on OK will generate the stub method at the current cursor location in the Java source file as follows:

```
@Override
protected void onStop()
{
  super.onStop();
}
```

Code Folding

- Once a source code file reaches a certain size, even the most carefully formatted and well organized code can become overwhelming and difficult to navigate.

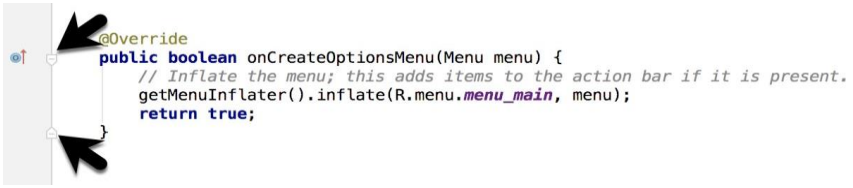
- Android Studio takes the view that it is not always necessary to have the content of every code block visible at all times.

- Code navigation can be made easier through the use of the code folding feature of the Android Studio editor.

- Code folding is controlled using markers appearing in the editor gutter at the beginning and end of each block of code in a source file. Figure 5-22, for example, highlights the start and end markers for a method

declaration which is not currently folded:

Figure 5-22



- Clicking on either of these markers will fold the statement such that only the signature line is visible as shown in Figure 5-23:

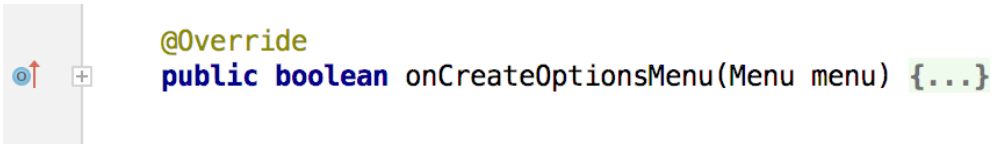


Figure 5-23

- To unfold a collapsed section of code, simply click on the '+' marker in the editor gutter.

- To see the hidden code without unfolding it, hover the mouse pointer over the "{…}" indicator as shown in Figure 5-24.

- The editor will then display the lens overlay containing the folded code block:

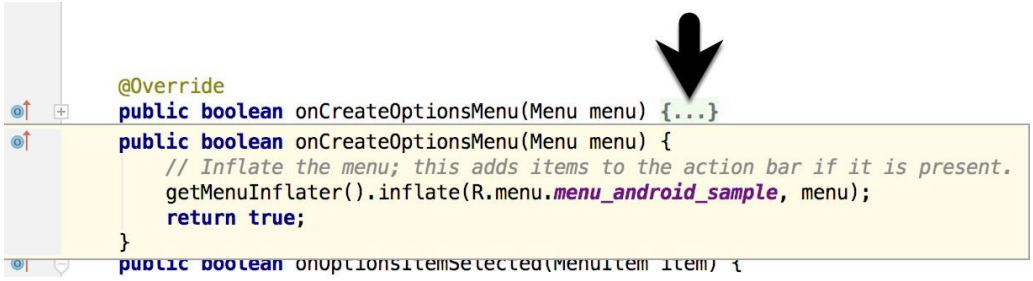


Figure 5-24

- All of the code blocks in a file may be folded or unfolded using the Ctrl-Shift-Plus and Ctrl-Shift-Minus keyboard sequences.

- By default, the Android Studio editor will automatically fold some code when a source file is opened.

- To configure the conditions under which this happens, select File > Settings… (Android Studio > Preferences… on macOS) and choose the Editor > General > Code Folding entry in the resulting settings panel (Figure 5-25):

(continued)

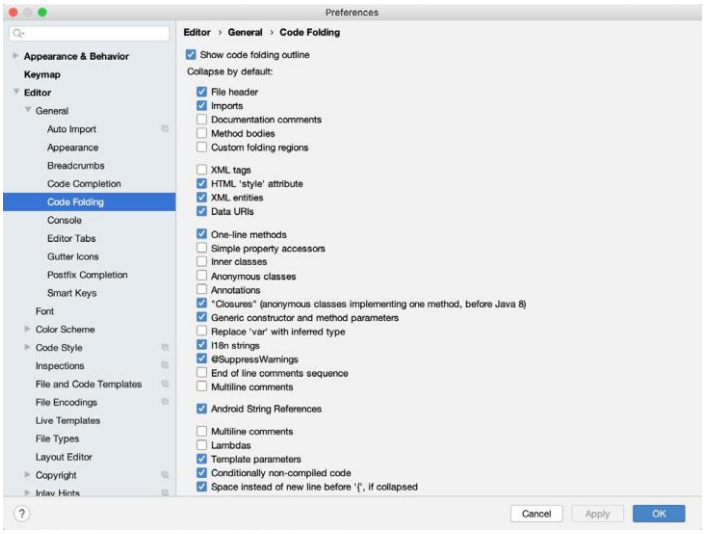


Figure 5-25

Quick Documentation Lookup

- Context sensitive Java and Android documentation can be accessed by placing the cursor over the declaration for which documentation is required and pressing the Ctrl-Q keyboard shortcut (Ctrl-J on macOS).

- This will display a popup containing the relevant reference documentation for the item.

- Figure 5-26, for example, showsthe documentation for the Android Snackbar class.

Figure 5-26

- Once displayed, the documentation popup can be moved around the screen as needed.

- Clicking on the push pin icon located in the right-hand corner of the popup title bar will ensure that the popup remains visible once focus moves back to the editor, leaving the documentation visible as a reference while typing code.

Code Reformatting

- In general, the Android Studio editor will automatically format code in terms of indenting, spacing and nesting of statements and code blocks as they are added.

- In situations where lines of code need to be reformatted (a common occurrence, for example, when cutting and pasting sample code from a web site), the editor provides a source code reformatting feature

which, when selected, will automatically reformat code to match the prevailing code style.

- To reformat source code, press the Ctrl-Alt-L (Cmd-Opt-L on macOS) keyboard shortcut sequence.
- To display the Reformat Code dialog (Figure 5-27) use the Ctrl-Alt-Shift-L (Cmd-Opt-Shift-L on macOS).
- This dialog provides the option to reformat only the currently selected code, the entire source file currently active in the editor or only code that has changed as the result of a source code control update.

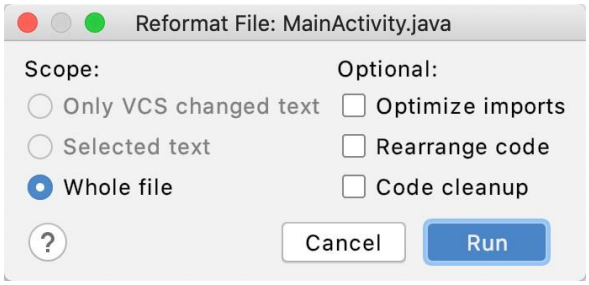


Figure 5-27

- The full range of code style preferences can be changed from within the project settings dialog.
- Select the File > Settings menu option (Android Studio > Preferences… on macOS) and choose Code Style in the left-hand panel to access a list of supported programming and markup languages.
- Selecting a language will provide access to a vast array of formatting style options, all of which may be modified from the Android Studio default to match your preferred code style.
- To configure the settings for the Rearrange code option in the above dialog, for example, unfold the Code Style section, select Java and, from the Java settings, select the Arrangement tab.

Finding Sample Code

- The Android Studio editor provides a way to access sample code relating to the currently highlighted entry within the code listing.
- This feature can be useful for learning how a particular Android class or method is used.
- To find sample code, highlight a method or class name in the editor, right-click on it and select the Find Sample Code menu option.
- The Find Sample Code panel (Figure 5-28) will appear beneath the editor with a list of matching samples.
- Selecting a sample from the list will load the corresponding code into the right-hand panel:

(continued)

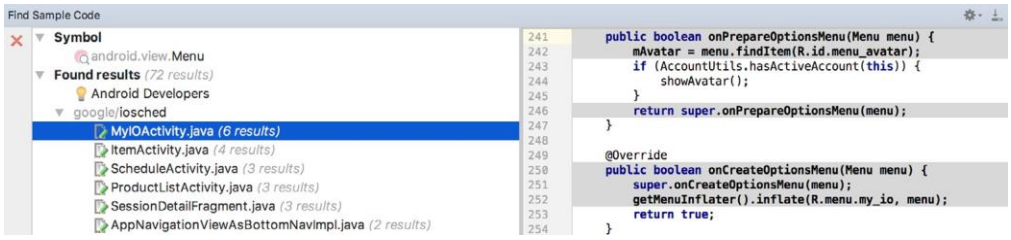


Figure 5-28

Live Templates

- As you write Android code you will find that there are common constructs that are used frequently.
- For example, a common requirement is to implement a for loop.
- Live templates are a collection of common code constructs that can be entered into the editor by typing the initial characters followed by a special key (set to the Tab key by default) to insert template code.
- To experience this in action, type fori in the code editor followed by the Tab key and Android Studio will insert the following code at the cursor position ready for editing:

```
for (int i = 0; i < ; i++)
{

}
```

- To list and edit existing templates, change the special key, or add your own templates, open the Preferences dialog and select Live Templates from the Editor section of the left-hand navigation panel:

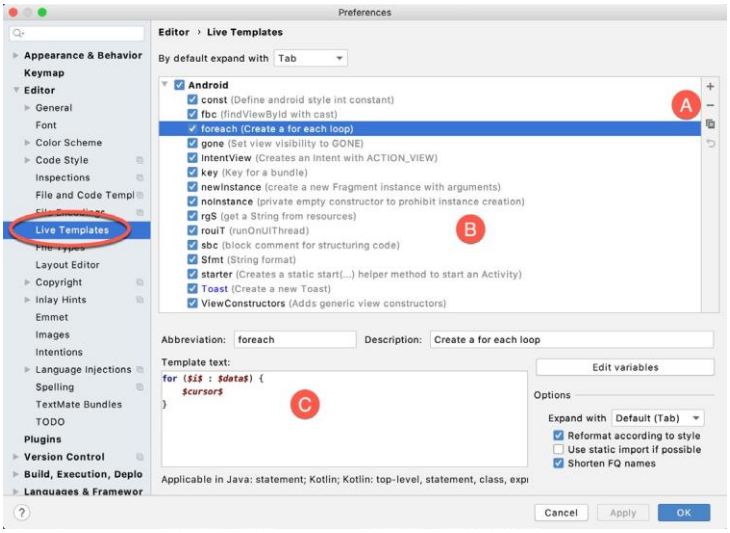


Figure 5-29

- Add, remove, duplicate or reset templates using the buttons marked A in Figure 5-29 above.
- To modify a template, select it from the list (B) and change the settings in the panel marked C.

Summary

- The primary elements of the Android Studio environment consist of the welcome screen and main window.
- Each open project is assigned its own main window which, in turn, consists of a menu bar, toolbar, editing and design area, status bar and a collection of tool windows.
- Tool windows appear on the sides and bottom edges of the main window and can be accessed either using the quick access menu located in the status bar, or via the optional tool window bars.
- There are very few actions within Android Studio which cannot be triggered via a keyboard shortcut.
- A keymap of default keyboard shortcuts can be accessed at any time from within the Android Studio main window.
- The Android Studio editor goes to great length to reduce the amount of typing needed to write code and to make that code easier to read and navigate.
- In this chapter we have covered a number of the key editor features including code completion, code generation, editor window splitting, code folding, reformatting, documentation lookup and live templates.