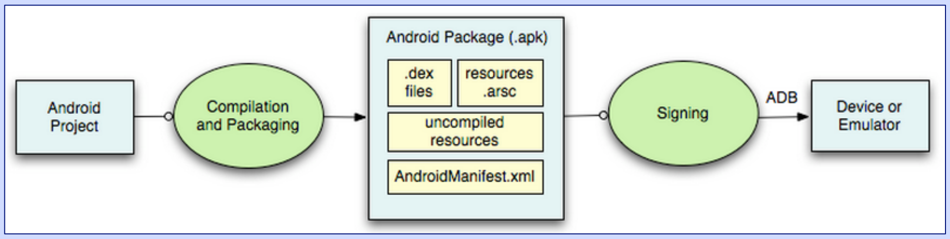
CS 478: Software Development for Mobile Platforms

Set 2: Installing and Using the Android Studio IDE

Ugo Buy
Department of Computer Science
University of Illinois at Chicago

January 17, 2019

Recall flow of app building



- Sources:

  http://developer.android.com/tools/building/index.html

  http://developer.android.com/guide/topics/resources/index.html

Flow of app building

1. Write app classes and define app resources
2. Compile and package your app
   - ➢ This will produce an .apk (application package) file containing both your .dex executables, packaged resources, and AndroidManifest file
3. Sign the app (debug vs. release mode)
4. Install and run the app (either on actual device or emulator)

Android Studio

- IDE for developing Android apps
  - — Based on JetBrains' IntelliJ IDEA environment
  - — Provide easy access to features in Android SDK
- Android SDK: Tool set for developing Android apps
  - — Compilation
  - — Debugging (Android Debugging Bridge)
  - — Handset emulator (QEMU-based) + system image running on it
  - — Software Libraries, Android platform
  - — Native Development Kit (NDK)
    - ➢ Support C/C++ library development

Android Studio (cont'd)

- Android Studio replaces old Eclipse-based IDE, called Android Development Tools (ADT)
  - — Caveat: Some learning materials still based on ADT,
  - — Refer to http://developer.android.com/index.html for latest materials
- Old IDE already included (1) graphical Layout Editor, (2) App signing, (3) NDK integration (as an Eclipse plugin)
- Main improvements with respect to ADT:
  - Logical formatting of app files
  - Automatic code completion
  - Android-specific refactoring
  - Gradle -- New app build process (replaces Apache Ant)

System requirements for Android Studio

- Java 8
  - — Main programming language
  - — Slow migration to Kotlin
  - — Can use C/C++ using Java Native Interface
- OS requirements
  - — MS Windows 7/8/10, or
  - — MAC OSX 10.10 → 10.13, or
  - — Linux GNOME or KDE desktop
- Hardware requirements
  - — 8GB RAM recommended
  - — 4GB Available disk/SSD space (recommended)
  - — 1280 x 800 screen resolution

Execution platforms

- Android's VMs:
  - — Dalvik virtual machine (until Kitkat)
  - — Android Run-Time (ART, since Lollipop)
- Both Dalvik and ART execute .dex files (optimized bytecode files)
- ART supports all Java 7 language features
- Most Java 8 language extensions supported as well
  - — Lambda expressions, type annotations, default and static interface methods are supported
  - — See https://developer.android.com/studio/write/java8-support

Downloading and configuring Android Studio

0. Prequel: If you don't have the *Java JDK*, first install it
   - — Go to: http://www.oracle.com/technetwork/java/index.html
   - — In *Top Downloads* list, select *Java SE* → *JDK* and get latest version for your OS (OS X vs. Windows)

   - — Mac OS X: Download .dmg file and move to *Applications* folder
   - — Go through installation process for your OS
   - — On Mac you are done; on Windows you must set up env. variables (under System Properties—Set up *JAVA_HOME* + Update *Path*)

Downloading and configuring Android Studio (cont'd)

1. Download *Android Studio* installer from http://developer.android.com/sdk/index.html
   — Mac OS X: Drag installer to *Applications* folder and run installer
   — Windows: Start installer, check all 3 components (SDK, AVD and HAXM), run installer
   — Excellent videos on youtube
   See, e.g., https://www.youtube.com/watch?v=gdeIol1m6A0
2. Now run Android Studio

Caveats

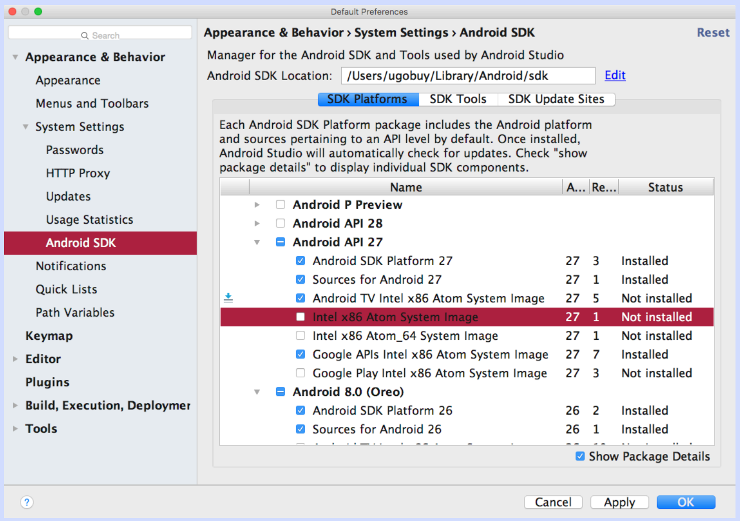
- On Mac OS X 10.10 and earlier, you'll need the *Java Run-time Environment* 1.6 (JRE 6), in addition to JDK 1.7 or 1.8
  — If missing, Studio installer will complain when you try launching it
  — Or modify JVM version in *AndroidStudio.app/Contents/info.plist* file to elude requirement (Change *JVMVersion* to *1.6*,1.7+*)
- Make sure you have Intel Hardware Accelerated Manager (HAXM) for faster device emulation
  — Sometimes asked during installation
  — Make sure to have at least SDK platform and Google API x86 system image

Caveats (cont'd)

- Basic installation just gives enough components to develop and run apps
- But you can add and remove components to suit your needs
  - — Under *SDK Tools,* include at least (1) *Android SDK Tools*, (2) *Android SDK Platform Tools*, (3) *Android SDK Build-Tools,* (4) *Android Emulator*, (5) *Google Play services,* (6) *Google Play APK Expansion Library* and (7) *Intel x86 Emulator Accelerator*
  - — Under SDK Platforms include at least (1) *Android SDK Platform xx*, (2) *Sources for Android xx*, (3) *Google APIs Intel x86 Atom System Image*, and (4) *Google APIs Intel x86 Atom_64 System Image*
  - — See next …

The SDK Manager window

- *Tools → Android → SDK Manager* or shortcut button



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/ugobuy/Library/Android/sdk Edit

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	A...	Re...	Status
☐ **Android P Preview**			
☐ **Android API 28**			
Android API 27			
✓ Android SDK Platform 27	27	3	Installed
✓ Sources for Android 27	27	1	Installed
✓ Android TV Intel x86 Atom System Image	27	5	Not installed
☐ Intel x86 Atom System Image	27	1	Not installed
☐ Intel x86 Atom_64 System Image	27	1	Not installed
✓ Google APIs Intel x86 Atom System Image	27	7	Installed
☐ Google Play Intel x86 Atom System Image	27	3	Not installed
Android 8.0 (Oreo)			
✓ Android SDK Platform 26	26	2	Installed
✓ Sources for Android 26	26	1	Installed

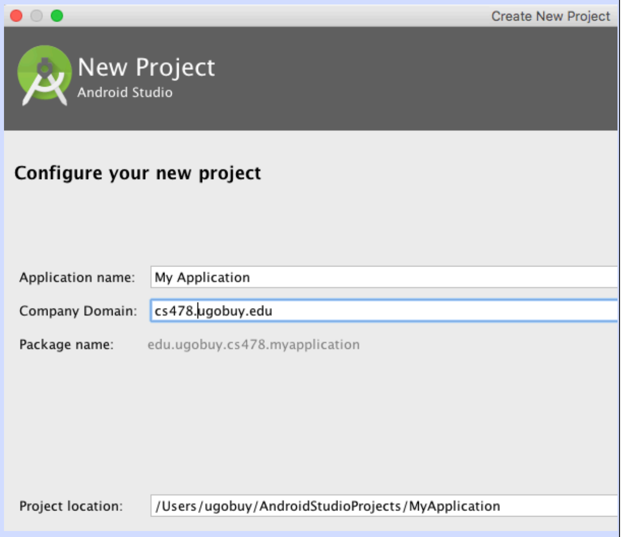
✓ Show Package Details

Caveats (cont'd)

- On Windows you will also need the Google USB Driver from *Extras*

Hello World app

1. Create new Android Studio project
   - Define app name (e.g., Hello World)
   - Define app domain (cs478.ugobuy.com)
   - Studio generates the app's package name (unique id)

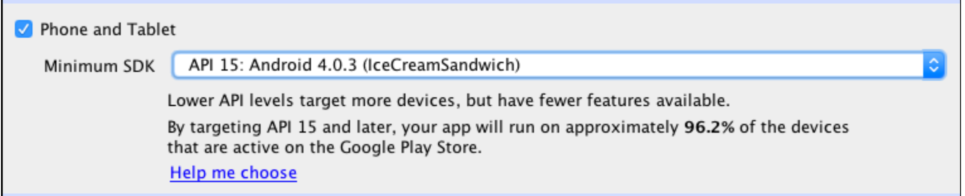


Configure your new project

Application name: My Application
Company Domain: cs478.ugobuy.edu
Package name: edu.ugobuy.cs478.myapplication
Project location: /Users/ugobuy/AndroidStudioProjects/MyApplication

Hello World app (cont'd)

2. Select target device and minimum SDK
   - Select Phone and Tablet, deselect the rest
   - Jellybean 4.1 (API version 16) is OK
   - *Help me Choose* shows cumulative distribution of Android version



☑ 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 **96.2%** of the devices that are active on the Google Play Store.

Help me choose

Hello World app (cont'd)

3. Next, select *Blank Activity*
4. Assign names to key elements of the app
   - Give it an activity name, title, layout name, menu resource name
5. Click finish
   - Code for running a "blank screen" generated automatically for you
   - Now Gradle "builds" your project for the first time
6. Now click on run button and run app on Android Virtual Device (AVD = emulated device)
   - If no AVD present, create one from *Tools* menu
   - *Tools* → *Android* → *AVD Manager* or use shortcut button in toolbar

Running configurations

- Test your app either on:
  1. An Android phone/tablet (if you have one)
     - Connect your device’s power cable to computer’s USB port
     - Click on ▶
     - Select *Choose a running device* and your phone/tablet
     - Shown on display using *DroidAtScreen* (Mac version)
  2. An emulated device (i.e., an AVD) on your computer
     - Select *Launch device* and one of the AVDs that you created with the *AVD Manager*

Android Virtual Devices (AVDs)

- Allow you to “see” your app running on potentially different physical devices, including Android TV and watches
- Configure, edit and start from *AVD Manager*
  - For efficiency, shut down any running AVDs before launching *AVD Manager*
  - Can choose amount of Ram (at least 1024 MB recommended, but beware of physical RAM on your computer)
- Rotate an AVD (landscape vs. portrait mode) using control panel tied to emulator
- Caveat: Cannot really test apps using certain physical hardware features (e.g., accelerometer, gyroscope, GPS receiver) on an AVD

AVDs (cont'd)

Advantages

- Cheap: Your cost == $0
- See how app will run on many different devices
  - — Support for some phones and tablets from Google and Samsung
  - — See how app will run on different OS versions; can choose any API level
- Can configure a lot of things
  - — Cameras
  - — Location
  - — Battery level
  - — Network latency/bandwidth

AVDs (cont'd)

Main Disadvantages

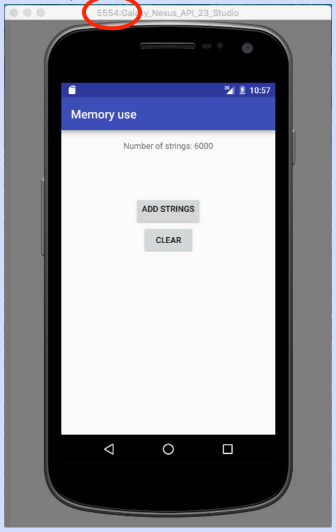
- Missing certain physical hardware features (e.g., accelerometer, gyroscope, GPS receiver, bluetooth/Wifi/NFC connectivity, USB connectivity) on an AVD
- Not the true user experience

Configuration capabilities

- — Network speed
- — Battery level, status
- — Location
- — Sending/receiving phone calls/SMS messages

The terminal tab

Find port number here.



- Use it, e.g., to telnet to phone
  - Select *Terminal* tab at display bottom
  - Connect with command:
    `telnet localhost <port-number>`
  - Find number by expanding emulator menu or zooming in emulator
  - E.g., `telnet localhost 5554`
- Use `help` command to learn features, e.g.,
  - `sms send 3125551212 “Ciao Ugo”`
  - `power capacity 10`
  - `power ac off`
  - `network speed lte`

Physical devices

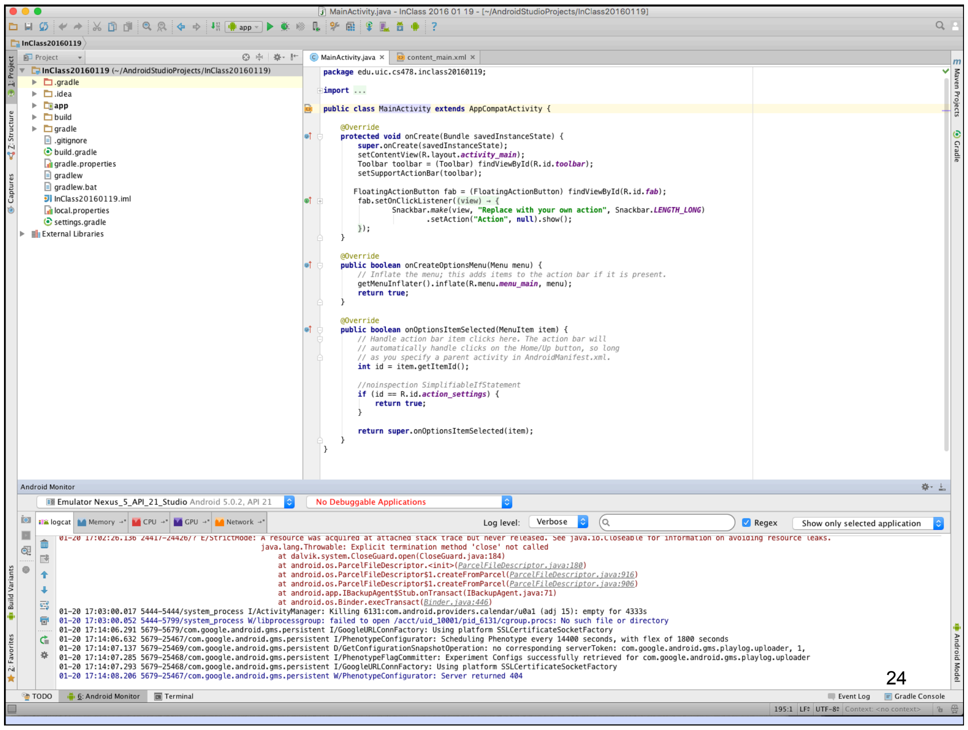
- Must configure phone/tablet to be in *developer mode*
  - On device select *Settings* → *About phone* → *Build number*
  - Touch *Build Number* item 7 (seven) times
  - Now main *Setting* screen will have *Developer options* item
  - Select item and turn on *USB debugging*

Android Studio: The navigation pane

- Android view: Logical arrangement of files (default)
  - ➢ Typically two packages shown (app and test)
- Project view: How files are stored in file system
- Stick to Android view, use project view only to see certain files (e.g., R.java)
- Android view: App package has three folders: (1) Manifests, (2) Java, and (3) Resources
  - ➢ Manifest file declares app components are files declaring app components and other useful information
  - ➢ Java files contain the source code of the application
  - ➢ Resource files define resources used by the application (See next)

Resources

- Various kinds of files that application needs, e.g.,
  - ➢ Strings
  - ➢ Layouts
  - ➢ Images
  - ➢ Views
  - ➢ Menus
  - ➢ etc.
- Access various resources through integer constants, defined for you in automatically-generated R.java file
  - ➢ We'll see how



MainActivity.java - InClass 2016 01 19 - [~/AndroidStudioProjects/InClass20160119]

```java
package edu.uic.cs478.inclass20160119;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

Emulator Nexus_5_API_21_Studio Android 5.0.2, API 21 — No Debuggable Applications — Log level: Verbose — ☑ Regex — Show only selected application

```
01-20 17:02:26.136 24417-24426/? E/StrictMode: A resource was acquired at attached stack trace but never released. See java.io.Closeable for information on avoiding resource leaks.
                                               java.lang.Throwable: Explicit termination method 'close' not called
                                                   at dalvik.system.CloseGuard.open(CloseGuard.java:184)
                                                   at android.os.ParcelFileDescriptor.<init>(ParcelFileDescriptor.java:180)
                                                   at android.os.ParcelFileDescriptor$1.createFromParcel(ParcelFileDescriptor.java:916)
                                                   at android.os.ParcelFileDescriptor$1.createFromParcel(ParcelFileDescriptor.java:906)
                                                   at android.app.IBackupAgent$Stub.onTransact(IBackupAgent.java:71)
                                                   at android.os.Binder.execTransact(Binder.java:446)
01-20 17:03:00.017 5444-5444/system_process I/ActivityManager: Killing 6131:com.android.providers.calendar/u0a1 (adj 15): empty for 4333s
01-20 17:03:00.052 5444-5799/system_process W/libprocessgroup: failed to open /acct/uid_10001/pid_6131/cgroup.procs: No such file or directory
01-20 17:14:06.291 5679-5679/com.google.android.gms.persistent I/GoogleURLConnFactory: Using platform SSLCertificateSocketFactory
01-20 17:14:06.632 5679-25467/com.google.android.gms.persistent I/PhenotypeConfigurator: Scheduling Phenotype every 14400 seconds, with flex of 1800 seconds
01-20 17:14:07.137 5679-25469/com.google.android.gms.persistent D/GetConfigurationSnapshotOperation: no corresponding serverToken: com.google.android.gms.playlog.uploader, 1,
01-20 17:14:07.285 5679-25468/com.google.android.gms.persistent I/PhenotypeFlagCommitter: Experiment Configs successfully retrieved for com.google.android.gms.playlog.uploader
01-20 17:14:07.293 5679-25468/com.google.android.gms.persistent I/GoogleURLConnFactory: Using platform SSLCertificateSocketFactory
01-20 17:14:08.206 5679-25467/com.google.android.gms.persistent W/PhenotypeConfigurator: Server returned 404
```

App build process

- Apps are compiled and linked automatically using *Gradle*
- Not an easy process because of many linked libraries going into apk archive
- Customize compilation/linking of your app using Gradle scripts
  - — File *local.properties* has Android SDK location that you are using (can be changed if you have multiple SDKs)

Language sensitive apps

- Strings are resources that can be customized, along with other resources
  - ➢ Found in *res->values->strings.xml* file for default display
- Use *values-it* directory for resources in Italian language, etc.
- Android will choose directory where to find resources depending on device configuration (as determined by user)
- XML file will assign a name to each string, e.g.,

```
<string name="upString">Up</string>
```

  - ➢ May add styling and formatting information
- Other res files refer to this string as `@string/upString`
- Java source code refers to string as `R.string.upString`

Creating apps

- Open Studio, select *new project* option from welcome window
- Use default, but choose unique *Company domain*
  - ➢ Used to create unique package name for your app
  - ➢ Should be different from other students, if want to publish your app
- Keep navigation pane in *Android* view, which arranges project's file storage structure into logical structure
- Observe project structure
  - ➢ *app* vs. *Gradle Scripts* folders
  - ➢ *app* folder has *manifests*, *java* and *res* subfolders
  - ➢ Gradle = compilation system for Android Studio projects

Editing apps

- Click on tab *MainActivity.java*; this file was generated automatically
  - ➢ Notice automatically-generated package name; this is the unique identification of your app
- Creating new classes and packages
- Refactoring code (^T on Mac, ^-alt-shift-t on Windows) and variable names (shift-F6)
- Code completion (^J on Windows, CMD-J on Mac)
- Java source code refers to resources using automatically generated class *R,* as in `R.string.upString`

R.java file

- Generated automatically
- Define class *R*, allowing access to resources in *res* folder programmatically
- Capture resources in Java language
- Use nested, static classes for the various kinds of resources
- E.g., nested class *R.string* declares static fields whose names match string names in */res/strings.xml* file
  - ➢ That way Java code can get a string as *R.string.helloThere*
- Nested class *R.layout* declares static fields whose names match layout files in *res/layout*, etc.
- In Studio, R.java file is found in folder `app/build/generated/source/r/debug/com.android.<project_name>/R.java`

Debugging Android apps

Two basic strategies

1. Debugging
   - Call *"Run→Debug as…"* instead of *"Run→Run as…"*
   - Support for standard debugging actions (break points, stepping through code, etc.)
   - Add breakpoints by clicking in the gutter; may specify conditions associated with breakpoint
   - Watch list of variables
2. Logcat output
   - Public predefined class *Log* allows printing to a log of messages produced by various applications
   - Produced also with normal run (w/o debugging)
   - https://developer.android.com/studio/profile/ddms.html

Logcat API and priorities

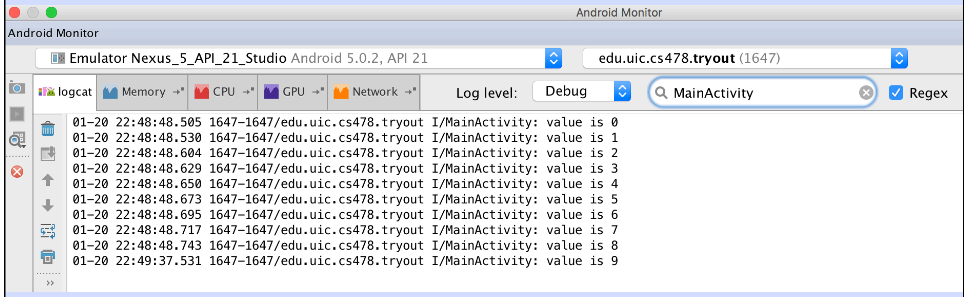
- *Log* class supports output of strings to the logcat
- Strings are given a priority level
- User can select priority level to be displayed
- Different static method for each priority level (2 *String* args per method)
- Static Methods with priorities from low to high
  - — `v()`: VERBOSE
  - — `d()`: DEBUG
  - — `i()` : INFO
  - — `w()`: WARN
  - — `e()`: ERROR
  - — `f()`: FATAL
  - — `s()`: SILENT

Example of logcat use

- Insert logcat statement in code where output is desired, e.g.,

```
Log.i("Main Activity:", "Counter is " + counter) ;
```

- Select logcat priority level INFO or lower; can filter by tag



Android Monitor

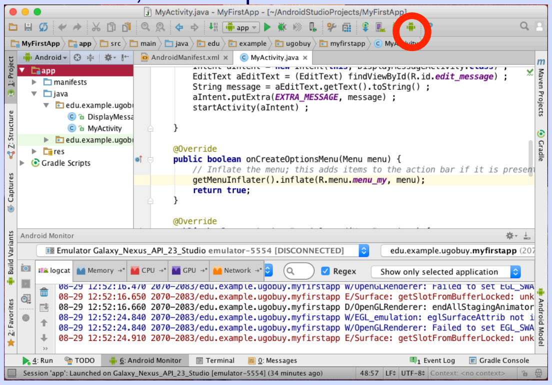
Emulator Nexus_5_API_21_Studio Android 5.0.2, API 21 — edu.uic.cs478.tryout (1647)

Log level: Debug — MainActivity — ✓ Regex

```
01-20 22:48:48.505 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 0
01-20 22:48:48.530 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 1
01-20 22:48:48.604 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 2
01-20 22:48:48.629 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 3
01-20 22:48:48.650 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 4
01-20 22:48:48.673 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 5
01-20 22:48:48.695 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 6
01-20 22:48:48.717 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 7
01-20 22:48:48.743 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 8
01-20 22:49:37.531 1647-1647/edu.uic.cs478.tryout I/MainActivity: value is 9
```

~~Android device monitor~~

- ~~Convenient tool for monitoring the status of the device (emulated or real)~~
  - ~~Part of Dalvik Debugging and Monitoring Service (DDMS)~~
- ~~Call from *Tools* menu or by selecting shortcut button)~~
  - No longer available, use the profiler instead



MyActivity.java - MyFirstApp - [~/AndroidStudioProjects/MyFirstApp]

```
        EditText aEditText = (EditText) findViewById(R.id.edit_message) ;
        String message = aEditText.getText().toString() ;
        aIntent.putExtra(EXTRA_MESSAGE, message) ;
        startActivity(aIntent) ;

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present
        getMenuInflater().inflate(R.menu.menu_my, menu);
        return true;
    }

    @Override
```

Android Monitor

Emulator Galaxy_Nexus_API_23_Studio emulator-5554 [DISCONNECTED] — edu.example.ugobuy.myfirstapp

✓ Regex — Show only selected application

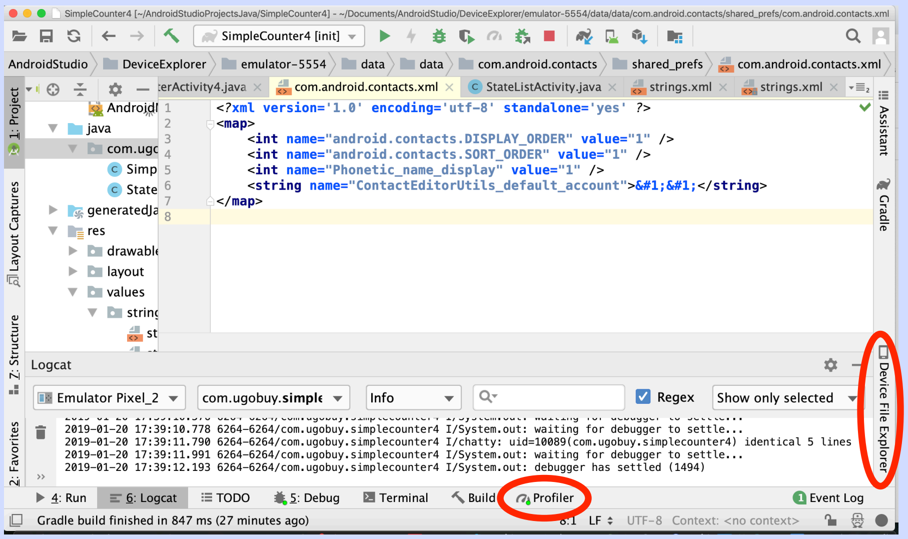
```
08-29 12:52:16.470 2070-2083/edu.example.ugobuy.myfirstapp W/OpenGLRenderer: Failed to set EGL_SWA
08-29 12:52:16.650 2070-2083/edu.example.ugobuy.myfirstapp E/Surface: getSlotFromBufferLocked: unk
08-29 12:52:16.660 2070-2083/edu.example.ugobuy.myfirstapp D/OpenGLRenderer: endAllStagingAnimator
08-29 12:52:24.840 2070-2083/edu.example.ugobuy.myfirstapp W/EGL_emulation: eglSurfaceAttrib not i
08-29 12:52:24.840 2070-2083/edu.example.ugobuy.myfirstapp W/OpenGLRenderer: Failed to set EGL_SWA
08-29 12:52:24.910 2070-2083/edu.example.ugobuy.myfirstapp E/Surface: getSlotFromBufferLocked: unk
```

Session 'app': Launched on Galaxy_Nexus_API_23_Studio [emulator-5554] (34 minutes ago)

More tools

- Two useful tools)
  - File explorer
  - Profiler
- These tools replace the old DDMS and Android Device Monitor tools
- File explorer: See file system actual or emulated device (aka AVD)
  - /data/data directory has files for each app installed on device, by package name
- Profiler: Examine CPU, network and memory utilization of device
  - See *MemoryUse* app

More tools – How to open



SimpleCounter4 [~/AndroidStudioProjectsJava/SimpleCounter4] - ~/Documents/AndroidStudio/DeviceExplorer/emulator-5554/data/data/com.android.contacts/shared_prefs/com.android.contacts.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name="android.contacts.DISPLAY_ORDER" value="1" />
    <int name="android.contacts.SORT_ORDER" value="1" />
    <int name="Phonetic_name_display" value="1" />
    <string name="ContactEditorUtils_default_account">&#1;&#1;</string>
</map>
```

Logcat: Emulator Pixel_2 · com.ugobuy.simple · Info · Regex ✓ · Show only selected

```
2019-01-20 17:39:10.778 6264-6264/com.ugobuy.simplecounter4 I/System.out: waiting for debugger to settle...
2019-01-20 17:39:11.790 6264-6264/com.ugobuy.simplecounter4 I/chatty: uid=10089(com.ugobuy.simplecounter4) identical 5 lines
2019-01-20 17:39:11.991 6264-6264/com.ugobuy.simplecounter4 I/System.out: waiting for debugger to settle...
2019-01-20 17:39:12.193 6264-6264/com.ugobuy.simplecounter4 I/System.out: debugger has settled (1494)
```

Gradle build finished in 847 ms (27 minutes ago)

A word to the wise

- Use the IntelliJ shortcuts to make your life easier
- Already seen some; full list can be viewed under *Help → Default Keymap Reference*
- Most notable according to David Gassner:
  - ENTER after class name will automatically import the class for you
  - ctrl-/ (cmd-/ on Mac) to comment or uncomment code
  - ctrl-SPACE to view code completion options
  - ctrl-shift-SPACE (cmd-shift-SPACE on Mac) does *smart code completion* (may or may not be what you wanted)
  - ctrl-shift-ENTER (cmd-shift-ENTER on Mac) does command completion

Useful links

- David Gassner's Lynda tutorial
  - http://www.lynda.com/Android-Studio-tutorials/Android-Studio-Essential-Training/361465-2.html
  - Caveat: Some information obsolete (e.g., memory monitor is found in *Android Device Monitor → Memory tab*)
- Satya Komatineni, *Understanding R.java* *http://knowledgefolders.com/akc/display?url=DisplayNoteIMPURL&reportId=2883&ownerUserId=satya*
- Java JDK download site:
  - http://www.oracle.com/technetwork/java/index.html
- Android SDK download site:
  - http://developer.android.com/sdk/index.html