Lab 2.1 – Create a Simple Android App

Overview:

In this lab you will build a simple Android app which will retrieve data from an XML file and display it.

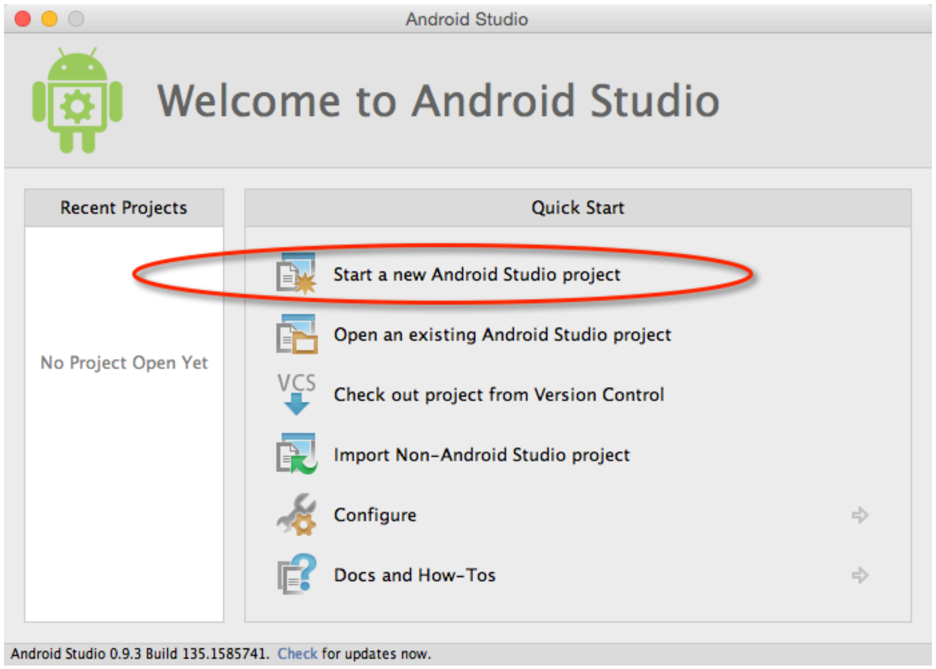
Setup:

Android Studio should already be installed. The SDK should be updated with the latest version of Android. Start Android Studio. On Windows machines, it may be necessary to run Android Studio in Administrator mode (right click and select "Run as Administrator")

Steps

1) Create a new Android Project in Android Studio

1.a) From the "Wecome to Android Studio" screen select "Start a New Android Studio Project"


Android Studio
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import Non-Android Studio project
Configure
Docs and How-Tos
Android Studio 0.9.3 Build 135.1585741. Check for updates now.

Enter the following values in the "Create New Project" wizard:

Application Name	Garage Sale App
Company Domain	com
Project Location	<place this in a directory you have permission to modify>

Select `Next`

1.b) On the "Select the form factors your app will run on" page of the wizard enter the following:

Phone and Tablet	check
Minimum SDK	API 19: Android 4.4

Select `Next`

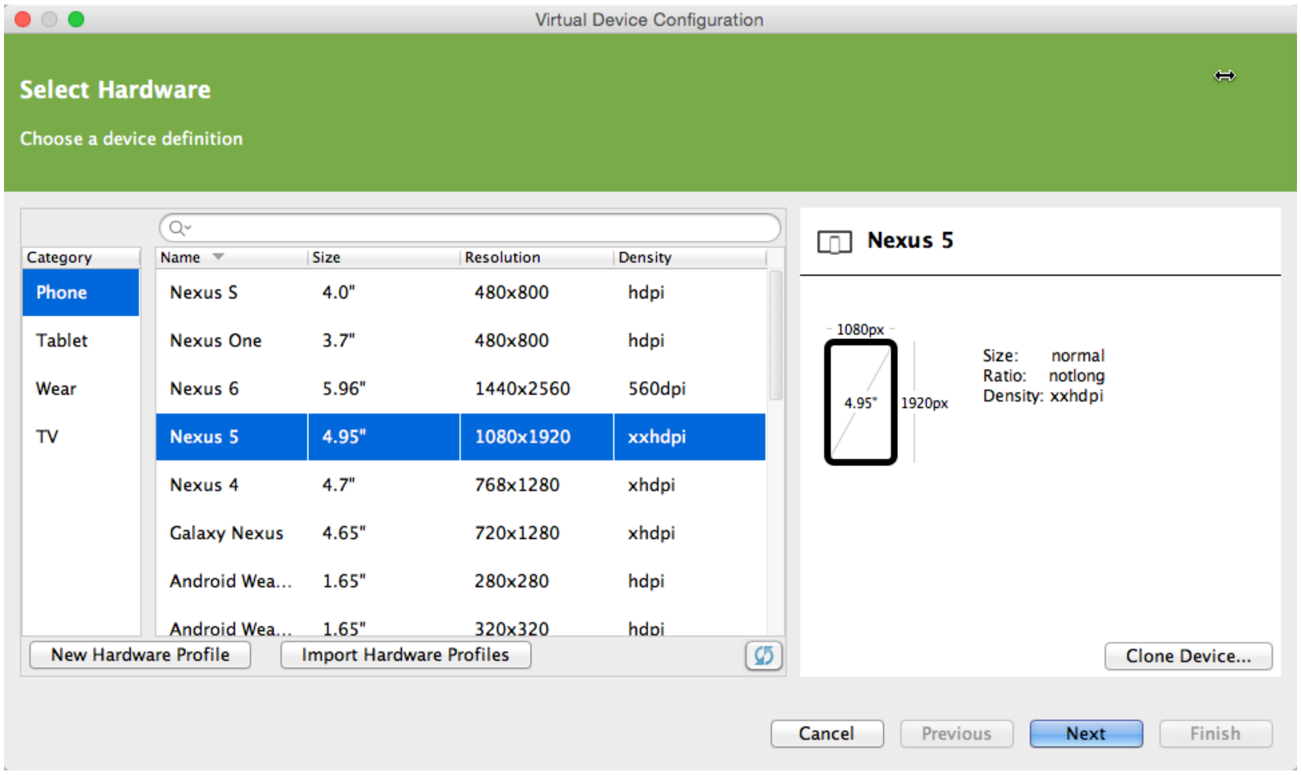
1.c) On the "Add an activity to Mobile" select "Blank Activity"

1.d) On the "Chose options for your new file" enter the following:

Activity Name	MainActivity

Layout Name	activity_main
Title	Garage Sales
Menu Resource Name	menu_main

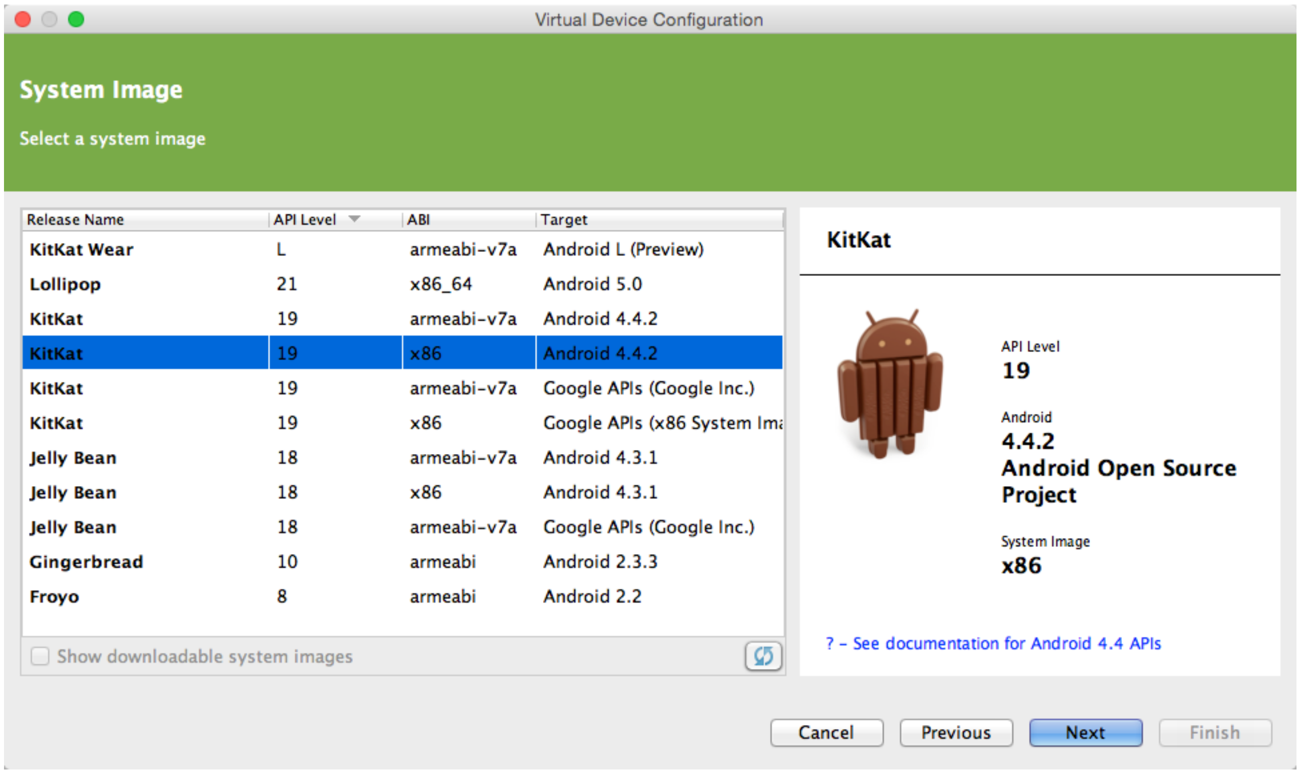
Click `Finish`
You should now have created a new project in your workspace.
2.) Configure an emulator and start it
2.a) From the menu, select: Tools → Android → AVD Manager
Click “Create Virtual Device”
Select `Phone` and `Nexus 5` and click `Next`



Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5	4.95"	1080x1920	xxhdpi
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi
Android Wea...	1.65"	280x280	hdpi
Android Wea...	1.65"	320x320	hdpi

Size: normal
Ratio: notlong
Density: xxhdpi

Select the "KitKat x86" System Image and click `Next`


Virtual Device Configuration

System Image

Select a system image

Release Name	API Level	ABI	Target
KitKat Wear	L	armeabi-v7a	Android L (Preview)
Lollipop	21	x86_64	Android 5.0
KitKat	19	armeabi-v7a	Android 4.4.2
KitKat	19	x86	Android 4.4.2
KitKat	19	armeabi-v7a	Google APIs (Google Inc.)
KitKat	19	x86	Google APIs (x86 System Ima
Jelly Bean	18	armeabi-v7a	Android 4.3.1
Jelly Bean	18	x86	Android 4.3.1
Jelly Bean	18	armeabi-v7a	Google APIs (Google Inc.)
Gingerbread	10	armeabi	Android 2.3.3
Froyo	8	armeabi	Android 2.2

Show downloadable system images

KitKat

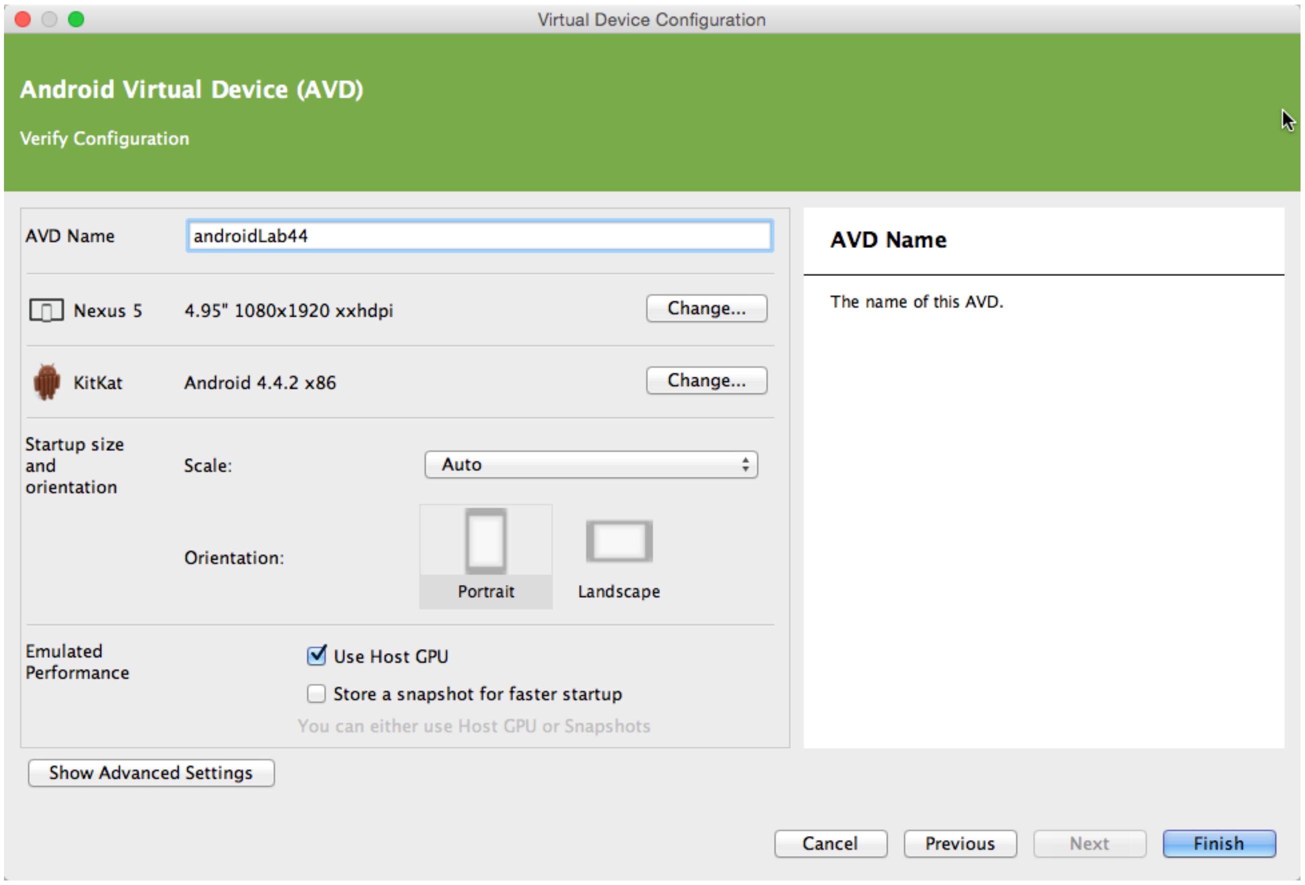
API Level
19

Android
4.4.2
Android Open Source Project

System Image
x86

? – See documentation for Android 4.4 APIs

Change the "AVD Name" to "androidLab44" and click `Finish`


Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD Name: androidLab44

Nexus 5 4.95" 1080x1920 xxhdpi

KitKat Android 4.4.2 x86

Startup size and orientation — Scale: Auto

Orientation: [✓ Portrait] Landscape

Emulated Performance — ✓ Use Host GPU; ☐ Store a snapshot for faster startup

You can either use Host GPU or Snapshots

AVD Name

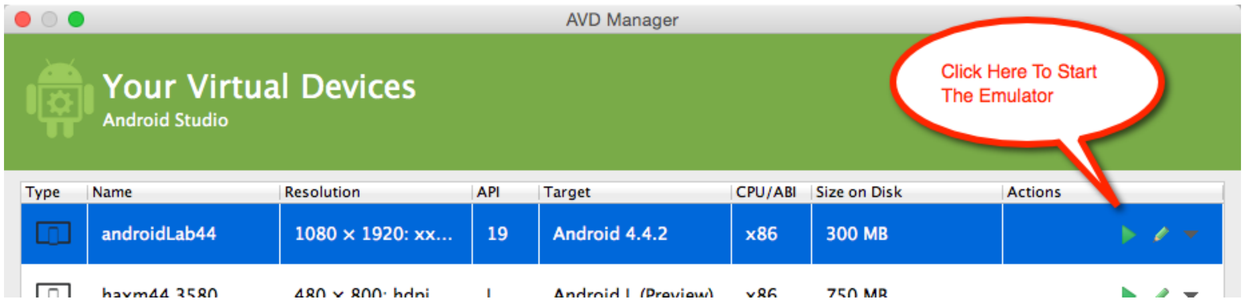
The name of this AVD.

The emulator is now configured but not yet running.

2.b) Start the emulator.

From the "Android Virtual Device (AVD)" screen select "android44".

Start the emulator by clicking the green play button.

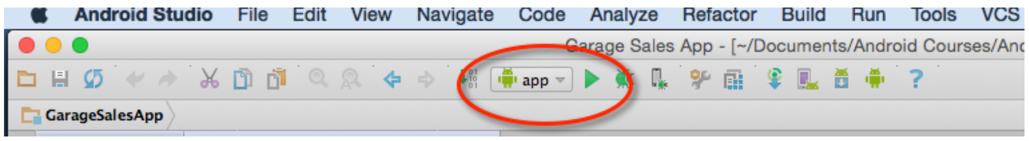


Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	androidLab44	1080 × 1920: xx...	19	Android 4.4.2	x86	300 MB	
	haxm44 3580	480 × 800: hdpi	L	Android L (Preview)	x86	750 MB	

Note: The emulator may take as long as 3 to 5 minutes to fully start. Wait for the home page on the emulator to confirm that it has started (not as the "ANDROID" splash screen).

2.c) Run the application on the emulator

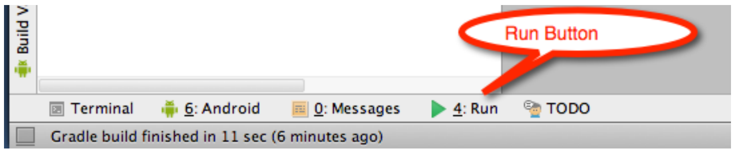
Click the green play button to the right of the run configuration list.



Review the console output. You should see output similar to the following:

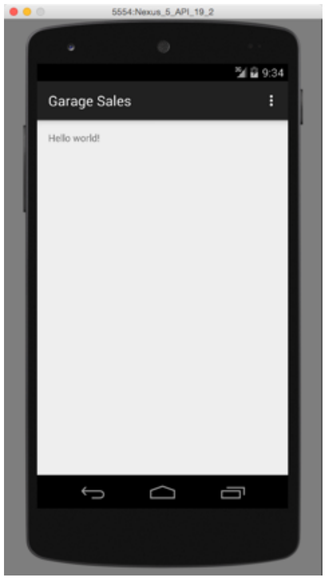
```
Waiting for device.
Target device: Nexus_5_API_19_2 [emulator-5554]
Uploading file
        local path: /Users/jamesharmon/Documents/Android Courses/Android Intro with Garagze/GarageSalesApp/ap
        remote path: /data/local/tmp/com.garagesalesapp
Installing com.garagesalesapp
DEVICE SHELL COMMAND: pm install -r "/data/local/tmp/com.garagesalesapp"
WARNING: linker: libdvm.so has text relocations. This is wasting memory and is a security risk. Please fix.
pkg: /data/local/tmp/com.garagesalesapp
Success


Launching application: com.garagesalesapp/com.garagesalesapp.MainActivity.
DEVICE SHELL COMMAND: am start -n "com.garagesalesapp/com.garagesalesapp.MainActivity" -a android.intent.acti
WARNING: linker: libdvm.so has text relocations. This is wasting memory and is a security risk. Please fix.
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.garagesalesa
```

Note: To view the console output, select the "Run" at the bottom left of the screen.



Gradle build finished in 11 sec (6 minutes ago)

The following should be visible in the emulator:



2.d) Review the following files with the instructor.

app/src/main/AndroidManifest.xml ... application manifest

app/src/main/res/layout/activity_main.xml ... layout for main activity

app/src/main/java/com.garagesalesapp/MainActivity.java ... implementation of main activity

build/generated/source/r/debug/com/garagesalesapp/R.java ... generated code

3) Import the library into the project

3.1) Select "Files → Import Module"

From the lab files directory select "garagze-library" as the "Source directory"

A new module will be created called "garagze-library"

The library contains the following code.

Package	Class File	Description
com.garagze.domain	Event.java	POJO object containing properties for garage sale events. The events location, description, identity and other properties are contained in it.
com.garagze.feed	EventXMLProcessor	Interface for XML parser
com.garagze.feed	EventXMLProcessorAndroidSAX	XML parser
com.garagze.service	EventService.java	Service layer for event data access.

The library also contains sample data. The directory "res/raw" contains a file called "naper_events.xml" which defines the garage sale events in XML format.

4) Call the library service method in your code.

4a) Change the "MainActivity" to look up events, calculate a count and display the count.

4b) Place an identifier on the TextView element in the layout

Edit the file "res/layout/activity_main.xml"

Add the following attribute to the "TextView" element

```
    android:id="@+id/textView"
```

This provides a unique identifier to the view element so that we can reference it from Java code. View "R.java" and notice the change to the "id" class. If you don't see "textField" then clean the project and a new "R.java" file should be generated. The "R.java" file will allow us to access elements by a logical name rather then the integer value (i.e. use "textField" instead of "0x7f090000").

4c) Modify the activity to display the number of events. Edit "MainActivity.java" and add the following code to the end of the "onCreate" method which was generated by the Android project wizard.

```
// Retrieve items from server and get count
int eventCount = getEventCount();

// Build display string
String displayText = "Number of events: " + eventCount;
```

```
// Reference view element and set display text
TextView textField = (TextView) findViewById(R.id.textView);
textField.setText(displayText);
```

Be sure to import the necessary Java classes. You will still get an error due to the reference to a method, "getEventCount", which hasn't been created yet. The "getEventCount" method will be created in the next step.

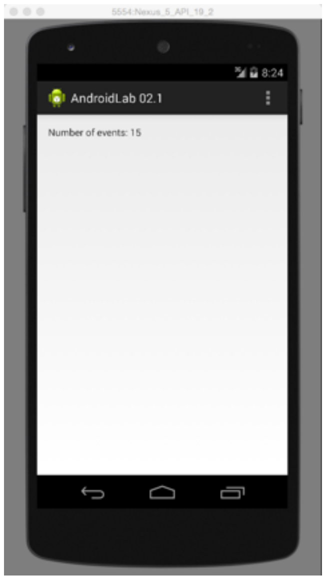
4d) Modify the activity to display the number of events.

Create a new method called "getEventCount" which will return the number of events. Start out by hard coding the number of events.

```
private int getEventCount() {
    return 100;
}
```

4e) Run the application and verify that it is working.

You should see the following results in your emulator:



Number of events: 15

Note: The number of events was hardcoded. In your code it should be 100.

4f) Now modify the "getEventCount" method to call the service layer to return lookup the events from the server.

```
private int getEventCount(this) {
    return EventService.getAllEvents(this).size();
}
```

Run the application again. The number of events should now be "15" which is how many events are in the "naper_events.xml" file.

Note: If the app is not working review the output in the LogCat view to see if a stack trace has been logged.

LAB 2.1

Last updated 2014-11-18 08:44:04 CST