Topics in Computer Science: Mobile Application Development

Lecture 2

Prof. El-Sheikh

CS-591 – Mobile Application Development

Today's Agenda

- Quick Review from last week
  - Will always do this.
- DVM vs. ART
- What's inside an Android App.
  - Manifest, Activities, Views,etc.
- Gradle Build Manager
- Working with the Android Studio IDE, some tips and a brief tour.
  - Activities, Layouts, Views
  - Using (and Reusing) Android Resources.
  - Intro to Debugger.
- Android Activity Lifecycle
  - Logging State Changes

Today's Agenda

- Software Components Basic Building Blocks of an App.
  - Activities
  - Layouts
  - Views
  - Events, 2 ways - Early vs. Late Binding
- More on Consuming Events, OnClick vs. OnLongClick
- Generating a Toast
- Team Building Activity: Elevator Pitches
- Will perform hands on exercises, building our own simple apps.
- Worksheet 2

Administriva & Reminders

- Reminder: Textbook is available from Publisher with a 30% discount.
  - How many people have the textbook?
- Bring your Laptops,
  - Run the emulator (if you need it) as soon as you arrive. ☺
  - **Please target Android Lollipop, 5.1**
- Reminder: Attendance, Collaboration and Participation are mandatory.
  - Use Piazza to help each other.
    - **Small amounts of extra credit for those who post often, esp. those who provide "good" answers.**
  - Will drop lowest homework.
  - Don't miss Midterm or other important classes.
  - Course is highly collaborative. **No Makeups.**
  - **If you are overloaded or have limited time please drop and retake when you have more availability.**
- Feel free to work in groups on all assignments
  - You must specify clearly who you worked with.

Administriva & Reminders

- No Makeups of any kind, unless you just added the class.
- Will drop lowest homework, with the exception of a couple required.
- **Please don't use electronics while I am lecturing. It's rude and will affect your Participation Grade.**
- It's a seminar and you will be expected to pursue quite a bit of research on your own, but I will ground the research for you.
- You will also be expected to work well with your peers. - **(1) Be nice! (2) Don't be an excuse maker.**

Homework Philosophy

Homework Philosophy

- Big Believer in Collaboration
  - Homework is just another way to collaborate.
- So, why limit students?
- My approach is to allow students great latitude when collaborating, going as far as to allow them to submit jointly, provided they document who they work with.
  - Won't students "cheat", it's not cheating if you call it collaborating.
- Won't some students just "skate"?
  - Nope, not if you closely tie homework to classwork/projects/and other assessments. Nope, not if you provide ongoing assessments and checkin frequently

~~• Javert!~~

Review from Last Week

(REF SLIDES FROM LAST WEEK)

Pivot

- This course will require you to pivot and utilize your understanding of Java to develop rich Mobile Applications for Android.
- This is an advanced level course.
- You will be required to do your own research during the second half of the course in support of your Final Project.
  - Eg, how to interact with eBay, AWS, Twitter, Facebook.
  - Most of these API's are well documented.

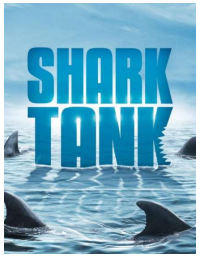
Some Expectations:
Things You are Responsible For

- Being a good problem solver
- Installing Android Studio
- Troubleshooting your own PC
- Being a good neighbor and collaborator
- Doing necessary research
- Pivoting
- Being Nice to your Team Mates. 

Brain Storming

&

Elevator Pitches



In Class Activity

- A good app is more than just a collection of technologies.
- Consider the following ideas (next slide) for possible Apps, choose two and delve more deeply, generating an elevator pitch, something you might share if you were on Shark Tank.

- If you don't like any of these ideas, feel free to come up with one of your own.
- Use the pitch for Parking Miracle as a guide.
- Each team will pitch these ideas for 5 minutes each.

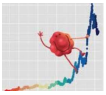
Brainstorming with Elevator Pitches

- Parking Miracle – a parking App that uses crowdsourcing to identify opening spots and the crowd to protect your vehicle.

- Fine – a Parking Ticket Fighting & Payment Service.
- 30 minute DVR – Watching TV can take feel like a waste of time.
  - Let's take control and speed it up.

- Whiner – a Drive Through Feedback App.
- Searchable Multimedia Clip Database
  - To be used as a pallette of media.

- Telemarketer Terminator – An App that blocks telemarketers



- Stock Ticker Alert. Get alerted when a stock matches your trend.
- Lobster Alert, App that uses GoPro's SDK to identify the big catch.

Parking Miracle

Shark Tank Pitch:

- Parking in a busy metro is fraught with difficulties.
  - Will I find a spot?
  - Will I get a ticket?
  - Is anyone looking out for my vehicle while I am away?
- Introducing Parking Miracle, the App that not only utilizes crowd sourcing to identify Parking Spots, but also uses the crowd to help you avoid fines and protect your vehicle.
- How does it work?
  - Smartphones are equipped with sensors that not only tell you where you are, but also whether or not you are driving or have parked. Why not leverage this capability to identify where others have parked, and then send out an alert when they drive off? Just like Waze does for traffic, both automatically and with users who want to help.
  - Let's take it a step further. Sometimes you might be running late and not be able to feed the meter. Instead of just sourcing the crowd for info, let's ask the crowd for help with an *"I NEED A MIRACLE"** Alert,
  - Let others with the App know you're in trouble. They could even be reimbursed automatically with your PayPal, Venmo, or PayU ID.
  - Every member would get one of these awesome stickers too!
  - Let people know to keep an eye out for these vehicles. Dropping a quarter in the meter, or letting users know when something is happening to their car, eg, getting hit by another parking car, or vandalism, etc.

* Inspired by all the Dead-Heads who have gone to concerts without a ticket.

Review: Labels, Buttons and Checkboxes, Oh my...

- Let's try working with a few common components in a few different languages.
- I'll Try it. (if time)
- You pick
  - Java Window Builder
  - C#
  - VB
  - Delphi
- Let's Try again with Android Studio.
- Pivoting: in other languages, eg, C#, VB, Delphi….
  - What is the equivalent of a View in other languages?
  - What is the equivalent of an Activity?
  - What is the equivalent of a group of Activities?

Key Concepts

- Even though you "instantiated" the Views by dragging them onto the Activity. Your Activity's Java Source doesn't have a reference.
- Don't instantiate a new object.
- Find the reference to the one that was created by Android Studio.
  - `Button btnClickMe;`
  - `btnClickMe = (Button)findViewById(R.id.btnClickMe);`
- Binding events, eg, onClick for a button is not automatic as it is with other languages. Here's the trick:
  - Start typing the event handler and let android studio fill in the mundane details.
  - `btnClickMe.setOnClickListener(new View.OnClickListener() {...}`

Native EXE's, Virtual Machines & Managed Code

What is a Native Executable

- Program that runs, directly on a computer.
- *What are some IDEs/languages that can compile Native EXE's?*
  - _____
  - _____
  - _____
  - _____

- Native Executables are (typically) unbounded, can go anywhere on the computer and do anything. The OS, offers some minimal protection.
- *What are some Pros/Cons of Native EXE's?*
- Very _____, but can also be _____.

Virtual Machines can also Run Programs

- An example of this the JVM, Java Virtual Machine.
- When Java code gets compiled, it doesn't become an executable that can run directly on your computer.
  - It becomes BYTE code.
- The BYTE code is then interpreted by another running process, the JVM.
  - A VM is just a software application that emulates a physical machine.
  - The JVM is like a machine inside a machine. The running program is much more secure. *At least that was the plan when Java was invented…* ☺
- BYTE code is standard and can run on JVM's anywhere, PC's, Macs, Cars, Toasters, whatever.
- Pros/Cons?

What is Managed Code?

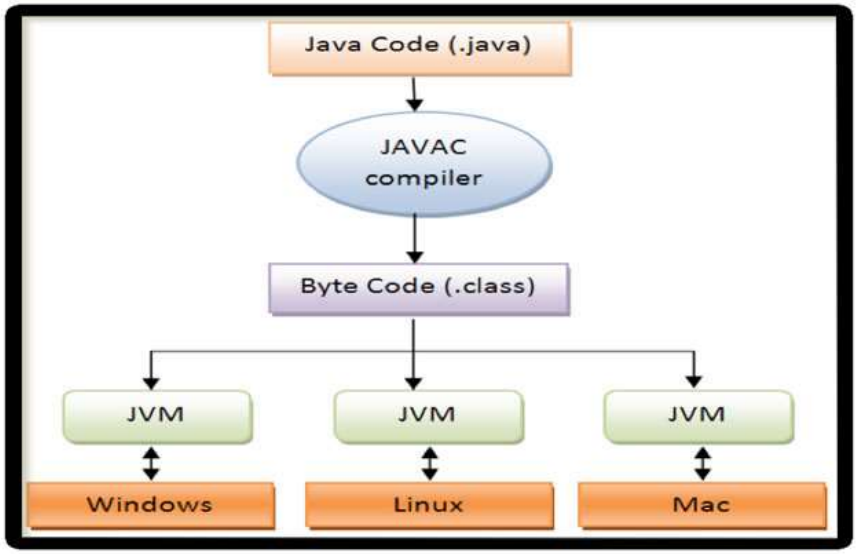
- Managed code is the next evolution of running code inside virtual machines. Came about when MS got sued by SUN (Maker of Java).
- Code is compiled into an executable. It will run directly on the CPU, but only with a Manager running with it.
- The first example of managed Code is the CLR, Common Library Runtime by Microsoft.
- C# gets compiled into a Managed EXE. The EXE will not run without the CLR.
- Not a VM. It runs “next” to the Exe, “Managing”. This includes Memory Management, Garbage Collection and Security. Eg, no Pointers in C#.

So How do Android Programs Run?

- Natively? Inside a VM? In a Managed Environment?
- All of the above are possible.
- But like Microsoft's .NET, Google's Android evolved to run in a Managed Environment, the ART, Android Runtime.
- Also like .NET, Google got sued by SUN@!!

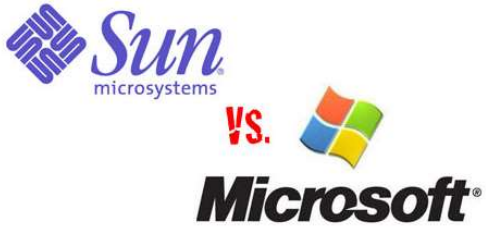
Mommy, “Where does managed code come from?”

- Java gained popularity in the late 1990’s when there were many different types of computers connecting to the Internet. HTML was X-platform, but insufficient.
- Vendors wanted a way to run their Applications on any computer with any architecture/OS. They also only wanted to write the code once. This could not be done natively, eg, with C++
  - Note: Macs were using several different chipsets (not including Intel)
- Customers wanted to be sure these Applications were secure.
- Enter Java and its JVM.



Ref: CartoonStock.com, www.slideshare.net/nextvision321/what-isja

The Battle for Managed Code



- In the late 1990's Java was getting increasing amounts of Market Share.
- Many companies had licensed Java, and integrated into their own development environments.
- Microsoft's implementation of Java would support code written on an MS platform, but wouldn't necessarily work across platforms.
- Sun sued MS for $35 Million. *(THAT'S IT ?!?)*
- They eventually settled for $20 Million, Microsoft was permanently prohibited from using "Java compatible" trademarks.

Guess Who the Big Bear is?

Ref: https://www.cnet.com/news/sun-microsoft-settle-java-suit/

The Battle for Managed Code

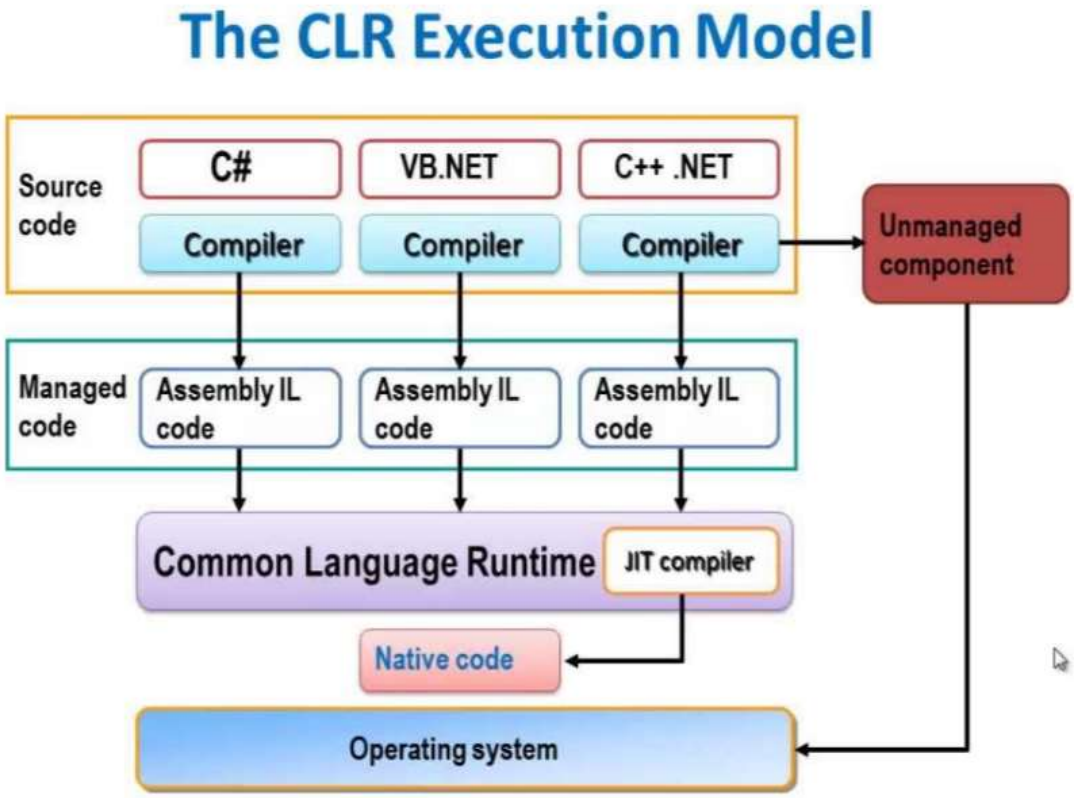
- So what did Microsoft do?
- A few years later they built their own managed code environment .NET. as well as a new programming language to go with it, C#.
- Unfortunately, Microsoft raided the best people from other companies to get this done. Eg, Borland.
- Borland lost 34 of their key people including their best Architects, including the guy who invented Turbo Pascal and Delphi, Anders Hejlsberg.
  - **Anders went on to co-invent C# at Microsoft!**

"It's like you're in the desert and Microsoft is stealing your water bottle." – Delbert Yocam (President, Borland. Sept., 1997)

- .NET is now very popular and many programming languages (eg, VB, C++, Delphi, Python, etc.) can compile down to MSIL (byte code).

How does Microsoft's Managed Code work?

- C# code compiles to an intermediate language (MSIL). Just like Java Byte Code.
- Like Java, .NET, has a virtual machine, *well sort of*, The CLR actually compiles Just in Time, then keeps the compiled code in memory for later reuse.
- You are not limited to compiling just Microsoft products. Many vendors have .NET languages.
- Notes:
  - More recent implementations of Java have a JIT.
  - The CLR is "smart". If it detects a segment of code being called frequently it will recompile with additional optimizations for better performance.



JVM vs. DVM vs. ART NET

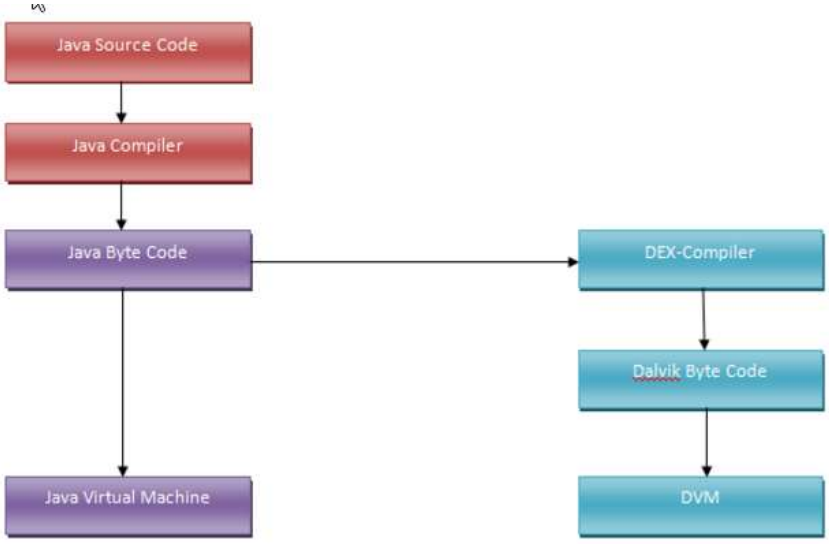
- What is the JVM again?
- DVM is very similar to the JVM, but is lightweight and can spawn itself quickly.
  - This is a must! Why?
- Every Android App runs in its own DVM.
  - It's possible to get around this, and let Apps share a DVM, but it's not typical.
  - *Well that's how it used to be anyway. Now Android uses the ART. More later…*
- DVM is very similar to JVM.
- Nice Refs:
  - http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf
  - http://stackoverflow.com/questions/13577733/how-an-android-application-is-executed-on-dalvik-virtual-machine

From Source to Executable

- Native EXE's
  - Eg, C++ Source → C++ COMPILER→ NATIVE EXE
  - NATIVE EXE'S ARE FAST, BUT INSECURE.
- JVM Apps
  - EG, Java Source → Java COMPILER→ Java Byte Code →JVM
  - Slower (interpreted), more secure, runs in sandbox.
- DVM Apps are based on Java, but there is an extra compilation and a different Virtual Machine.
  - EG, Java Source → Java COMPILER→ Java Byte Code →Dalvik COMPILER → DVM
  - Takes a while to compile. Things get compiled twice!
  - Why the hassle? JVM is not Free, DVM is Free. DVM is also optimized for mobile apps?
  - Why use a VM at all? Secure, also "guaranteed" to run across different devices. Just like Java.

Compiling Android Code: Slow as a Twice Baked Potato

- Multiple Compiles
- It's still a new development environment.
- Requires Patience.
- Even the IDE's are not as feature rich.
- Let's try creating a simple app with a button.
  - We'll use a few different IDE's to compare and contrast.



- Ref: http://stackoverflow.com/questions/13577733/how-an-android-app

From Dalvik to ART

- Dalvik: Cooking a baked potato twice is delicious, it is time consuming.
  - Why not bake the potato in advance and store in the freezer?
- **Who is ART?** *Kind of a Nerd, but with super powers.*

- Compiles code ahead of time (AOT) into runnable executable.
- Very similar to running managed code in .NET framework.
- Takes longer to install, because of the additional compilation, but runs and starts up quicker.
- Launched Experimentally in Android KitKat 4.4
- Now it's the default standard.

What is Gradle?

Gradle

- Gradle is the Build Manager for Android.
- Manages Dependencies
- SDK Versions
- Order of compilation
- Third party Library integration (JARs).
- *Stuff you don't want to have to worry about.*
- *Let's take a peek,* **`build.gradle`** *(app module).*
  - *You usually won't have to go in there,*
    - *Unless your compiling someone else's code, then you may have to tweak SDK versions (to match the ones you have).*
    - *Or adding a third party library to your project (dependencies section).*

Activity Life Cycle

Activity Lifecycle

- Activities have multiple transitional states.
  - When transitioning between states, events will trigger, you can write handlers for these events.
    - Eg, onCreate, onPause, onResume, etc.
  - Ref: http://developer.android.com/training/basics/activity-lifecycle/index.html

  - Some important lifecycle events:
  - onCreate, onPause, onStop, onResume, onSaveInstanceState, onRestoreInstanceState

Restoring Saved Activity State

- There are many (many) lifecycle events.
- The onCreate(..) event gets called first. It also receives a bundle that can be used to set state.
- Another (better) way to restore state is to use the onRestoreInstanceState().
  - only gets called by the android system, if there is a bundle to restore.
- Nice because you don't have to check if the bundle is null.-

Instead of restoring the state during `onCreate()` you may choose to implement `onRestoreInstanceState()`, which the system calls after the`onStart()` method. The system calls `onRestoreInstanceState()` only if there is a saved state to restore, so you do not need to check whether the `Bundle` is null.
Ref: https://developer.android.com/guide/components/activities/activity-lifecycle.html#saras

Software Components

- Android has unusual names for some of its components.
- Here is what is on tap for today.
  - Activities
  - Layouts
  - Views
  - Toasts
  - Snackbars (if time allows)

Activities

- An Activity is a component of your application that has a screen to accomplish something, eg, a Calculator, a Web Browser, a Hello World.
- In other languages an Activity might be thought of as a single Window, Form or Web Page.
- An App, can have multiple activities, just like a web site can have multiple pages, or a program can have multiple windows.
- Each application has only one Main Activity. This gets specified in the Android Manifest. ***(Let's See it.)***

Layouts

- Layouts are used to organize and arrange components (Views).
- They can also be used to automatically pad and fill proportional areas of the screen.
  - For example with large and small screens and when rotating between landscape and portrait mode.
- Let's look at some of these, relative, grid, linear.
- Tips and thoughts:
  - Grid layout is great for placing components with precision
  - Best to drag onto Component Tree (don't mix and match)
  - Relative layout is not easy to work with, let's try copying and pasting
  - Linear Layout can be used to hide and show entire rows/columns of widgets.

Views

- Views are the basic building blocks of your App. There are many, these include labels, text boxes, buttons, radio buttons, check buttons, image wrappers, listviews, calendars, clocks and much more.
- Let’s explore a few of these in Android and in one other IDE.

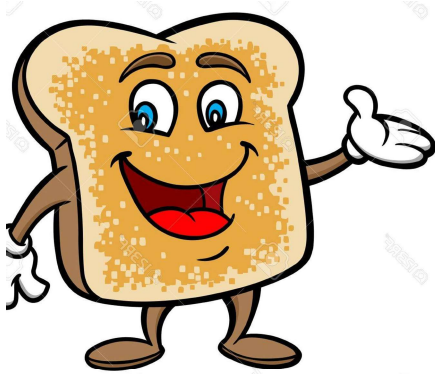
Celsius to Fahrenheit Converter

- Let’s Do it together
- 2 ways if we have time.

Activity: Fourth Grade Flash Card App *(if time)*

- Implement a Third Grade Flash Card App for Addition and Subtraction.
- The app will provide students with ten random Problems.
- Students will enter their response in a textbox. Use a Single Activity. Feel free to use the template to the right as a guide.
- Your program will utilize Java as the code-behind layer of your Activity and will generate 10 random problems, students will enter the answer into the textbox, then depress submit. After 10 submissions the student's score should popup in a Toast.
- We will improve upon the design later, don't worry about telling students which problems they got right or wrong, or giving them immediate or delayed feedback.

•Provide a descriptive Storyboard for your App. Much of the design and user interaction is up to you, but Feel free to use the above template as a guide.
•Demonstrate your application running.



Logging *And* *Toasts*

Toasts

- A Toast is used to generate transient messages.
  - Eg, connected to a WIFI hotspot.
  - They should popup and then go away.
- These messages pop up, then go away.

- Syntax:

```
Toast.makeText(getBaseContext(),"I would like to propose a Toast.",
Toast.LENGTH_LONG).show();
```

Event Based Programming

Events

- There is no main() program in a GUI.
- The entry point is the first activity, and once there,
- Everything that occurs from that point forward is event driven.
  - onClick (most common)
  - onLongClick
  - onHover
  - onScroll
  - onShow
  - onHide
  - Etc.
- This is a very good concept to demo in other IDE's, since the events are well organized. Pick one, and I will demo (quickly).

Events, 2 primary ways

- Early Binding
  - Identify the methods to be attached to specific events before the program runs.
- Late Binding
  - Identify the methods at run time, handy if you want to swap out methods, or disable long running methods when not needed.

- **Now let's try it Android Studio (two ways). How is it different?**

Let's Try It, me then you

- Write a simple program with a single button, that when clicked will make a Toast "I have been clicked."

- *Note: This is mostly to ensure your Android Studio is setup properly!*

Lab Exercise (I'll do it)

- Create an Android App to compute Celsius to Fahrenheit.
- Display the result in a Toast.
- Use a Relative Layout.
- Redo, but display the result in an EditText

Pinch Points – Buttons Always Caps

- Buttons are always Caps? Why?
- It has to do with the theme, check your styles.xml (in the res folder)
- You can simply change to another style, eg,
- Or fix at the individual button levels, by adding the line.
- **`android:textAllCaps="false"`**
- Note this property is NOT available in design view. ☹

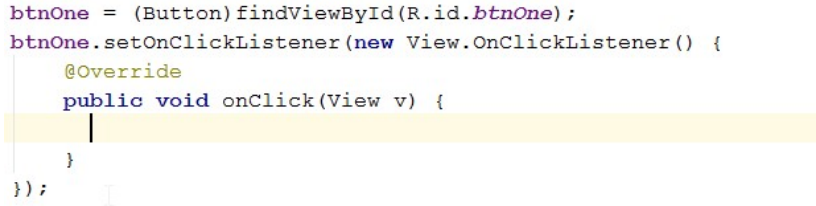
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button Two"
    android:textAllCaps="false"
    android:id="@+id/btnTwo"
    android:layout_below="@+id/btnOne"
    android:layout_alignStart="@+id/btnOne"
    android:layout_marginTop="52dp" />
```

Pinch Points: Auto Completing Event Handlers for Runtime Binding.

- Writing event handlers in Android requires too much code. How can I simplify the process?
  - Create a reference to a Button Object in Java
  - Bind the Button reference to the one instantiated in the view, by using FindViewByID(..)
  - Start typing
  - btnOne.setOnClickListener(new On…), as soon as you start typing the parameters, you will be prompted to autogenerate the wrapper. Hit Return, not Tab.
  -

Pinch Points: Auto Completing Event Handlers for Runtime Binding.

- Be sure to click Enter, when the drop down appears.



Pinch Points: How use GridLayout

- Best to lay things out in Component Tree.
  - This will add things in order.
  - Important: Don’t mix and match, either lay things out on the device or on the Component Tree.
- Grid Settings
  - NumRow,s
- Span Multiple rows, columns

Pinch Points: Syntax for generating a Toast

```
Toast.makeText(getBaseContext(),"I would like to propose a
Toast.", Toast.LENGTH_LONG).show();
```

Next week, we will look at SnackBar, which is similar, but Allows a callback function.

Pinch Points, technique of overriding methods

- In the Java source code window, type Alt-Insert, then choose Override Methods.
- Eg, some activity lifecycle methods you might override include.

```
protected void onCreate(Bundle savedInstanceState)
protected void onRestart()
protected void onResume()
protected void onStart()
protected void onStop()
protected void onPause()
protected void onDestroy()
```

Reference Materials (optional reading)

- Android Calculator App, Step by Step Tutorial

- http://www.i-programmer.info/programming/android/5937-android-adventures-building-the-ui.html