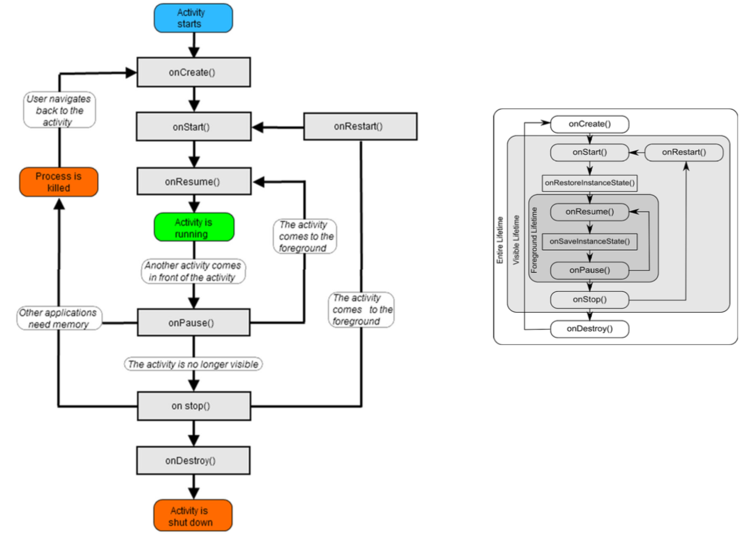
Let 2.

* Basic building blocks: Activities, layouts, Views, Events, 2 ways – Early vs. Late Binding
* No need to instantiate a new object, using the reference of 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.
  + Start typing the event handler and let android studio fill in the mundane details.
  + btnClickMe.setOnClickListener (new View.OnClickListener() {…}
* Native Executable
  + Program that runs, directly on a computer.
  + What are some IDEs/languages that can compile Native EXE’s?

?? C++, C

* + 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?
* JVM (java virtual machine) will becomes BYTE code after compiled.
* BYTE code is standard and can run on JVM’s anywhere,
  + Pros/Cons???
  + Pros: cross platform(平台)
  + Cons: slow
* Android program run
  + Natively, inside a VM or in a Managed Environment.
* DVM (Dalvik Virtual machine) is very similar to the JVM, but is lightweight and can spawn itself quickly.
  + Every android App runs in its own DVM.
* DVM Apps are based on Java, but there is an extra compilation and a different Virtual Machine.
* Gradle
  + The build Manager for Android.
  + Manages Dependencies
  + SDK(Software Developer Kit) Versions
  + Order of compilation
  + Third party Library integration (JARs).
  + Let’s take a peek, **build.gradle** (app module).
* Activity Life Cycle
  + Activities have multiple transitional states.
  + When transitioning between states, events will trigger, you can write handlers for these events. \* Eg, onCreate, onPause, onResume, etc.
  + 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.
* Software Components
  + Android has unusual names for some of its components.
  + Here is what is on tap for today.
    - Activates, Layouts, Views, Toasts, Snackbars
  + Activities:
    - 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.
  + 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 grate 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, test boxes, buttons, radio buttons, check buttons, image wrappers, listviews, calendars, clocks and much more.
  + Toasts
    - A Toast is used to generate transient messages.
      * Eg, connected to a WIFI hotspot.
      * fThey should popup and then go away
    - Syntax: Toase.makeText(getBaseContext(), “I would like to propose a Toast.”, Toast.LENGTH\_LONG).show();
* Events 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 points 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 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 method at run time, handy if you want to swap out methods, or disable long running methods when not needed.
  + 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.setOnClickListencer(new On…), as soon as you start typing the parameters, you will be prompted to autogenerate the rapper. Hit Return, not Tab.
  + 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
      * NumRows,
    - 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();
* Bundles and Activity Lifecycle
  + Activity state: eg, text in a textview, or the color of a button, or the position of the seekbar.
  + When might an activity be destroyed and recreated: screen rotation, changing Locale, app going in background.
  + Users expect their activity state to persist through these destructions/recreations.
  + How is this done: using bundle of information.
  + A bundle is just a hashmap of info.
* Data Bundles in Android
  + Bundle objects are the preferred way of passing data in Android.
  + Sometimes the Bundle can hold Application State information
    - Eg, when an activity is destroyed and recreated.
  + Sometimes the Bundle can be used to pass information.
    - Eg, when one activity initiates the creation of another activity.
* APK
  + Is android package, basically in zip format with the apk extension. The apk contains several things, but primarily it contains a manifest, compiled code, and resources.
  + Some of these resources are uncompiled.
* Resources
  + Build process results in an apk file which is eventually signed and uploaded to an Android Device.
  + Resources are stored separately. They aren’t “baked in”.
  + Android takes the added step of assigning consistent ID’s for resources with the same name. More on this later.
  + This is a great model. It means we can easily change things without adding code to handle device configurations.
* Things that will trigger resources difference and even dynamic changes.
  + Android Framework will automatically choose resources to render, eg, strings, images, layouts, etc. depending on the device, it’s pixel density, it’s locale, it’s orientation, etc.
* Things that will trigger this switch:
  + Changing Locale
  + Changing Language of phone, how are these different?
  + Different device sizes
  + Different device orientations
  + Different device DPI’s (resolutions)
* A Locale object represents a specific geographical, political, or cultural region. An operation that requires a Local to perform its task is called locale-sensitive and uses the Local to tailor information for the user.
* What is inside an .apk file?
  + Apk file is generated as a result of compiling/recompiling your java code into dex code.
  + The .apk file also includes:
    - A manifest, listing all of the stuff inside
    - A set of resources which can include Images & Strings.
  + Why not just compile the Images and Strings into the byte-code?
    - App would certainly be faster if resources were baked in.
  + Every resource gets its own unique ID. R.java
  + These IDs are typically referenced in the xml files, but can also be referenced from other java source code.
  + Separation of resources from compiled code simplifies automatic configuration process when the app runs on different devices and in different locales.
* String resource
  + By making Strings a resource, it becomes easier to handle locale changes.
  + Eg, if you change your phone’s Locale to France, and want your App to be in French, create a new resource file. Just right click on String.xml and choose new Values resource File.
* Handle Multiple languages.
  + Adding a string resource.
    - Choose Locale as a qualifier, note it will disappear after you select it.
    - Choose a Language then choose then locale specific dialect.
  + Next change the local of your device in settings.
* Layout Resources
  + A layout is just a resource. Eg, activity\_main.xml is a resource with its own dynamically generated id in the R.java file.
  + Layout typically get: inflated” in the onCreate event of your Java App.
    - Eg, setContentView( R.layout.activity\_main);
  + You as a developer can generate multiple layouts for the same activity.
    - Why
* Image Resources
  + Not support for adding localized images, must do it manually.
  + Can be automatically maintained by Android.
  + Android Studio will automatically change the pixel density of images for you, so your images look good on different devices.
  + App icon is stored in the various mapmap-xxx folers.
  + We keep images at /app/src/main/res/drawable
  + Image files must always be lower case.
  + The icon is referred to in the AndroidMainifest.xml. Change the file ic\_launcher and you change the default icon. /res/new/image Asset
* Grid Layout
  + D
* Introduction to Intents
  + Intents are a key feature of android
  + An intent is a message
    - Message can target a single Activity
    - Or it can be broadcast, and multiple apps can respond.
  + For example, if you are in an app that has street addresses (eg, in your Contact, from Yelp, whatever) and click “navigate” an intent is broadcast, and app apps hat can perform navigation wake up. You pick the one you want.
  + These other apps have intent-filters and android notifies them that you want navigation.
  + Today we will focus on spawning multiple activities with intents.
* Intents and Activity Management
* Passing Data in a bundle, from One Activity to the Next
  + Bundles are generally used to pass data between activities.
  + It’s a mapping of string values to different parcelable types.
  + You can pass the data with an intent:
    - Intent intent = new intent(getApplicationContext(), SecondActivity.class);
    - Intent.putExtra(“key”, value);
  + You can receive the extras in a bundle
    - Bundle bundle = intent.getExtras();
    - String data = bundle.getString(“key”);



Pinch point – Debugging Wirelessly

* How to debug your physical device wirelessly
* No rooting, no additional sdks.

Lecture 4

Handy props:

* Wrap\_content, Match\_parent, Weights and WeightSum, Can be a good generic alternative to generating multiple layouts
* Handy Notes: working with images,
  + Be sure image name is all lowercase -to be safe, used to be required.
  + Someimtes andoird can’t render pngs, just resave as a different file type.
  + App icon is automatically a mipmap, “pre-calculated sequences of images to look good on multiple resolutions”.
* Adding icons is done a bit differently then adding other images.

Ref Last week’s material (in weChat posted)

* Activity Life Cycle Additional Callbacks:
  + onSaveInstanceState(..) : allow you to add key/value pairs to the outstate of the app
  + onRestoreInstanceState: retrieve value back to the variable from which it was originally collected...
* Intents
  + Are a key feature of android.
  + An intent is a message.
    - Message can target a single Activity
    - Or it can be broadcast, and multiple apps can respond.
    - Eg, click “navigate” an intent is broadcast, and all app perform navigation wake up. You pick one you want.
  + These other apps have intent-filters and android notifies them that you want navigation.
* Intents and Activity Management
  + Eg, App with two activities.
  + Activity 1 spawn Activity2:
    - In the AnforidManifest.xml we list all of the Activities that are part of the project. We also specify unique properties of these Activities, a signature...
    - Activity 1 will broadcast an intent to open Activity2 by using this signature.



Passing Data in a bundle, from One Activity to the Next

* Bundles are generally used to pass data between activities.
* It’s a mapping of string values to different parcelable types.

You can pass the data with an intent:

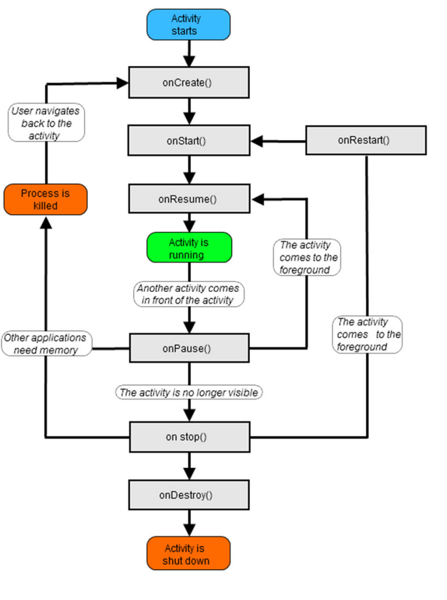
Intent intent = new Intent(getApplicatonContext(), SecondActivity.class);

Intent.putExtra(“key”, value);

You can receive the extras in a bundle

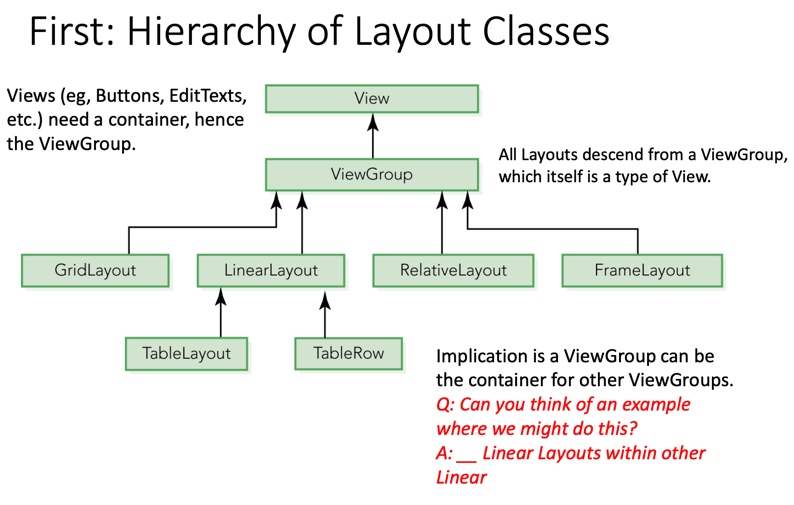
Bundle bundle = intent.getExtras();

String data = bundle.getString(“key”);



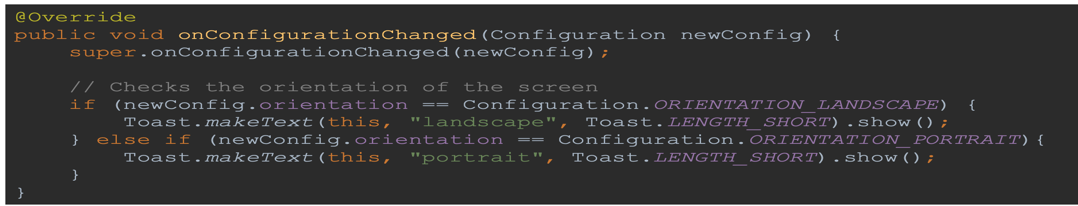
Dynamically Creating Views

* Views can be created at runtime too.
* This is not the preferred mode of development.
  + It removes the Android Studio Layout Designer, which can be quite helpful.
  + It also makes it more difficult for large teams manage code.
  + There can be some advantages however.
* When would you want to Dynamically Create Views.
* Good reasons
  + You want to be able to easily reference your components in an array.
  + You are unsure what the design would look like in advance.
    - Eg. You are designing based on some sort of database query.
* Bad reasons
  + Old C++ developers etc.
* First: Hierarchy of Layout Classes
  + Views ( eg, buttons, EditTexts, etc) need a container, hence the ViewGroup.



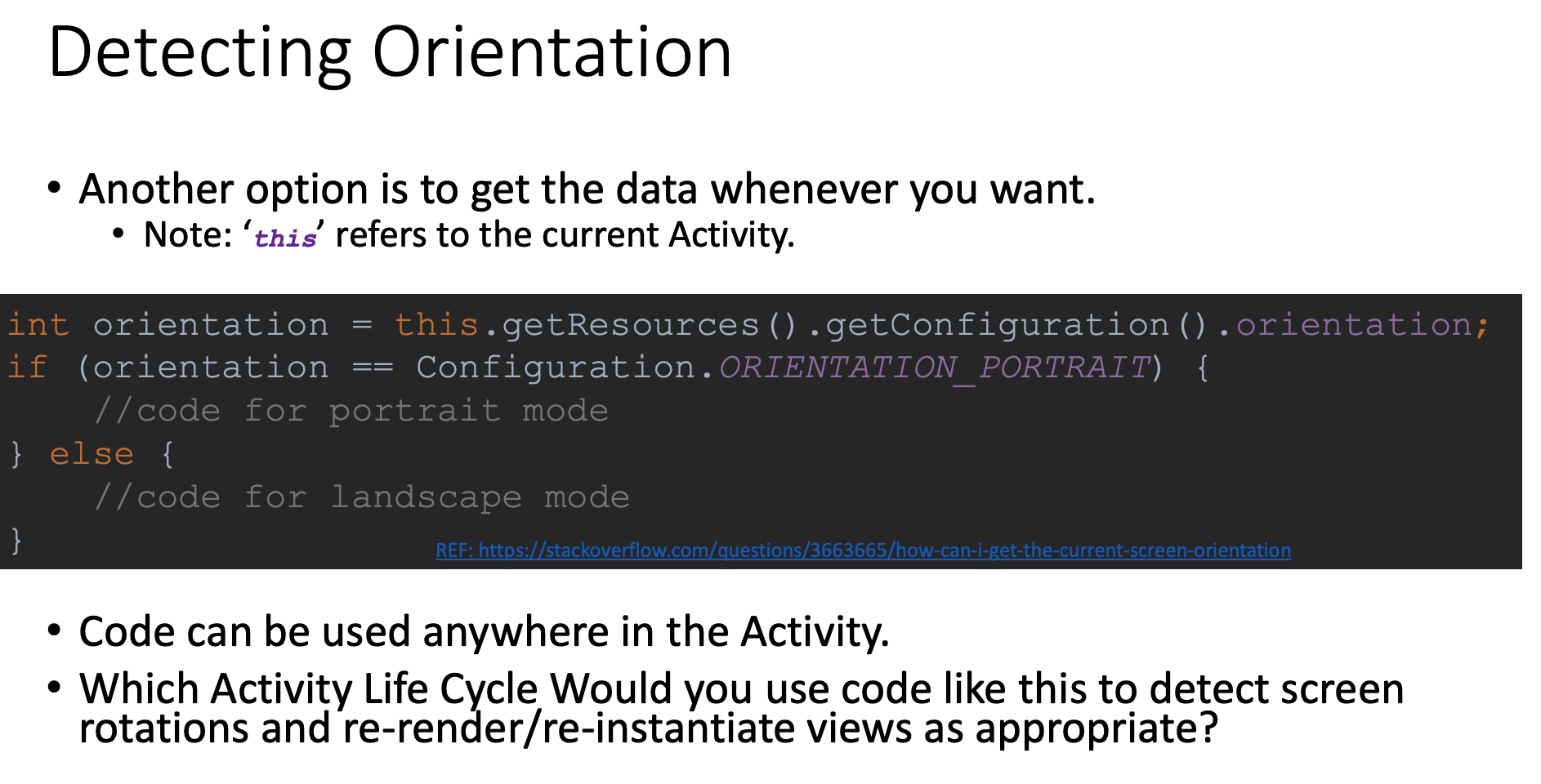
Detecting Device Orientation

* One way to identify when orientation changes is to check the onConfigurationChanged(..) : for any configuration changes(you handle there), you will receive a call to current activities method instead of restart. If you not handle, will restart and not call onConfigurationChagned()



Detecting Orientation

* Another option is to get the data whenever you want.



* + (same as rotation)OnPause()-> onSaveIntranceState() -> onStop() -> onStart() -> onRestoreInstanceState() -> OnResume()
* The Accelerometer detects motion on your device on all these axes
  + Step-by-step
  + Accelerometer is a Sensor, we therefore use the Android SensorManager object to retrieve sensor data.

1, instantiate A sensorManager Object

2, Bind a onSensorChanged Event to this object

3, Start Listening for Movement

4, Stop Listening whenever you leave an activity, why?

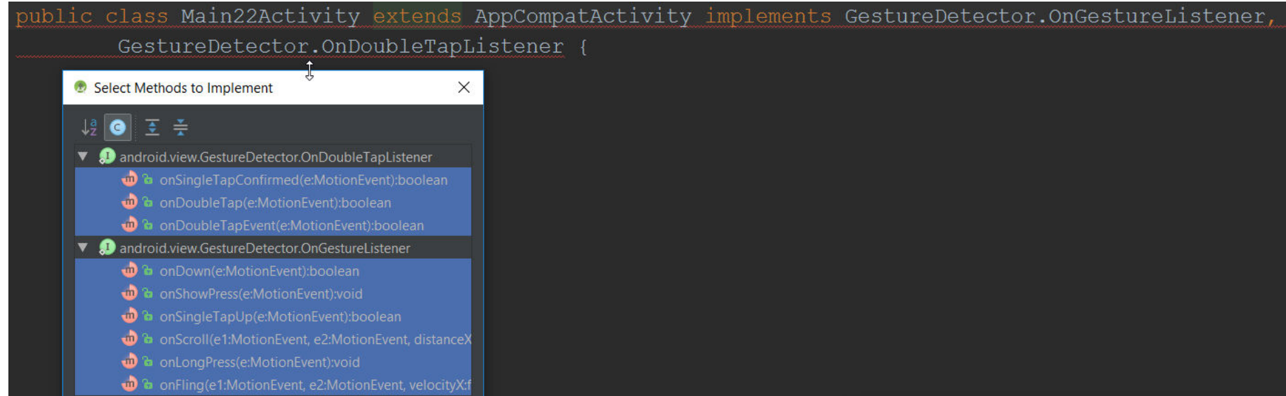
* + - * Save energy.

5, Best way to explain this is to walk through the code.

* Ref: EI-Sheikh – accelerometertesting.apk

Implementing Gesture Interfaces

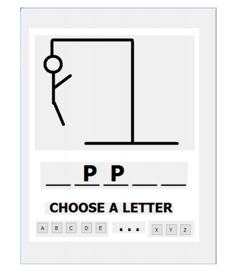
* Android Studio makes it simple to implentent Gesture Event Handlers
* Implement **GestureDetector.OnDoubleTapListener**, then hit {Alt-Ins}



Consuming Gestures before they get passed the Activity

* There is already a default handler for screen touches, onTourchEvent which is implemented in the base Activity Class.
* A touch is different than a Gesture,
* We must override this method locally in order to be able to consume touches locally.
* Don’t forget to call the super class so that our event chain is not broken.

Gestures and Touch Events

* Implementing Touch Events in Android is very convenient.
* The classes needed to handle gestures and Touch are implemented for you.
  + You just need to know what events to track and how they are triggered.
  + Very convenient
* Gestures and Touch are a multifaceted approach.
  + Instantiate a GestureDetector Object.
  + Implement the gestureDetector Interfaces that you want.
  + Different of scrolling vs Fling:
    - Scrolling: general process of moving the viewport. When scrolling is x and y called panning.
    - Dragging: type of the scrolling that user drags finger across the touch screen. Simple dragging is often implemented by overriding onScroll() in GestureDetector.OnGestureListener
  + Fling: is the type of scrolling that user drags and lifts her finger quickly. When lifts her finger, you generally want to keep ccrolling(moving the viewport). By implemented by overriding onFling() in GestureDetector.OnGestureListener.
* Pinchpoint: Error opening existing project, Gradle Snyc Failed: due to their source had a different directory structure and may have built their code using a different sdk than you have. (project opening doesnot have the same build tools as you. confusing as sometimes show you too much the hood). You need ot change is in ..\app\build\build.gradle.
* Pinchpoint: what is android instant Run and Do I need it?
  + It makes the complie, build, run process faster, by shipping changes over to an existing complied sdk on your device.
* Enable “instant run” and clean project and rebuilt it. 

Lecture 5

* Listviews can be used to present a set of choices, sort of like a dropdown(spinner)。
* Frame Layout
  + Simple, optimized for rendering objects on top of each other(layers).
  + Often used by Fragments.
  + Dynamically resized itself to the largest view it contains.
  + Gets its name because it can be used to automatically frame images.
* Animation
  + Objects in Android Applications can be used to provide subtle feedback to a user.
    - Leads to a more professional App, but don’t over do it.
  + There are several ways to animate views within your App.
  + Key Animations:
    - Transparency
    - Stretching
    - Translation
    - Rotation
  + There are many others, but the above are the key ones we will use.
* Basic Animation
  + All views have a view property animator.
    - View.animate() returns a reference to the view animator object
  + Some typical methods that are called with the animator object
    - setDuration (int) – the amount of time the animation will occur
    - alpha(float) – sets the alpha transparency. 0 = invisible, 1 = completely visible
    - alpha(float).withlayer() – helps improve performance when rendering complex views (prevents re-rendering entire view each time).
    - Eg., the following code would change the transparency from whatever it was, to 25%, taking 5 seconds to complete. 
* More complex animations with xml
  + Another way to create animations, this second approach best for complex animations.
    - Animations can be resource based, just like layouts, images, menus, etc.
    - Create a folder canned res/anim
    - Add your animation xml to this folder.
  + Room and rotate an object.
* Animation interpolators
  + Interpolators are animation resources used to modify speed and type of movement of animations. Many come built into Android.
* How to use?
  + 1, carete an anim resource file. Eg.fadeup.xml
  + 2, Bind the xml to an Animation Object. Call startAnimation.
* Asynch animations.
  + Animations are Asyn
  + All the animations in an anim.xml file happen simultaneously.
* Asyn and Sync animation resources.
* Make sync: use the startOffset property, which specifies a delay. It still asyc, but acts sunc
* What would the view bound to this animation do?

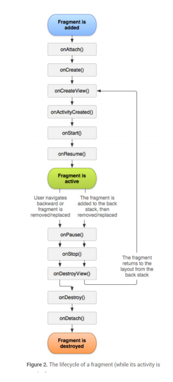
Auto completion Telnetting into Virtual Device

* Getting Tired of declaring components -> auto hotkey

Fragments & Inter-fragment communication

* Fragments:
  + Reusable GUI components grouped together in a Layout.
  + Think of it as a way to group views and logic into a reusable component.
  + An activity can have multiple fragments
  + Activity host the fragments
    - Fragments cannot run without a host.
    - Fragments should not talk to each other directly.
      * If they did they would be tightly coupled.
      * Tight coupling is bad practice.
    - Communicate through an interface to the Activity instead.
* Why use Fragments?
  + Reuse of code (what else is new)
  + Excellent way to handle multiple layouts and devices, Eg, Bid vs. Small screen
  + Change with device type and resolution
  + Provide a reusable framework with which to work.
  + Activities responsible for determining how to layout fragments as well as providing the communication factility between fragments.
* Fragments
  + Can be thought of as individual building blocks that can encapsulate multiple Veiws and Routines.
  + These building blocks are then used by Activities to presents a consistent look and feel, even may different sizes and resolutions.
* Creating a new Fragment
  + Uncheck options to create factory methods and interface callbacks when create a new object.

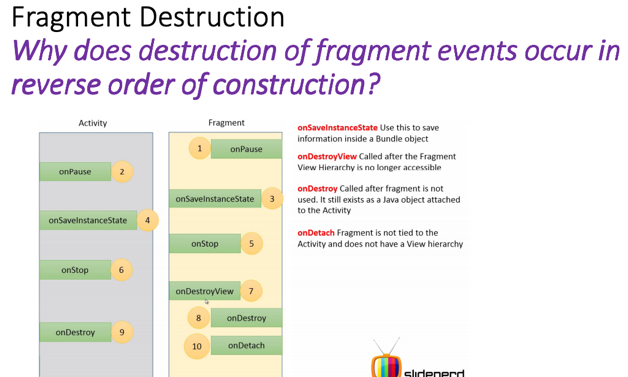
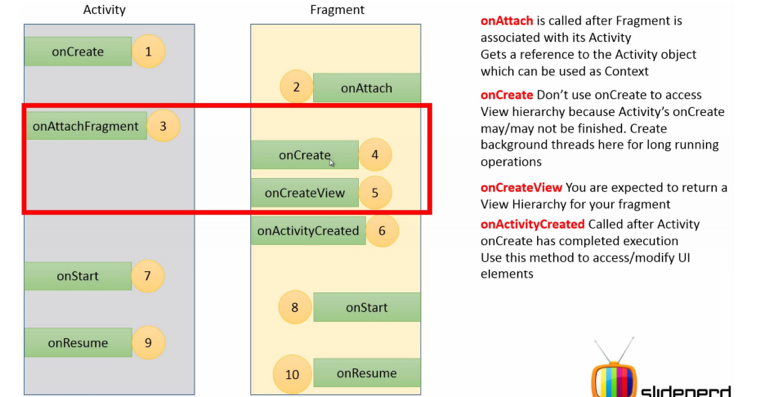
Fragment Lifecycle

* Hosted by an Activity, Fragment life cycle is tightly coupled to its host.
* Fragment construction events occur just AFTER that of the Host. Destructed before host.
  + Eg, onCreate, onCreateView, etc. – get called before the corresponding Fragment events’.
  + onDestory, - gets called after the corresponding Fragment events’.
  + Q? why const vs. destr in opposite order.
    - Activity hold fragments, and fragment is a case of activity.
* Fragment life Cycle tied directly to the host Activity.
  + Eg, when an activity stops, Pauses, Resumes, etc, so too will the underlying fragment(s).
* Fragment provides a reusable ui
* Can be combined to satisfy different screen resolution
* Can be dynamically “inflated”, allowing for custom UI’s

Fragment has own life cycle>

* onCretate – called after Activities onCreate – DON’T perform GUI Updates here. Why?
  + - noCreateView – Pass View structure to Activity.
* onActivityCreate – it is now safe to modify UI Elements.
* onPause – called when user leaves fragment. Good time to backup changes/settings.

Fragment instantiation, what happen when?



Inflating Fragments

* sometimes it makes sense to not inflate all available fragments
  + eg, you may want to hide information from a user

What is a Layoutinflator

* This comes up whenever want to instantiate Views from an XML Layout File.
  + Eg, in a fragment we might call. Make XML to object

Fragment Manager

* Responsible for maintaining references to all fragments within an Activity.
* Use findFragemntByid, to locate and interact with fragments.
* Provides a transactional mechanism for interacting with fragments.

Saving Fragment State

* There are times when it is important to save fragment state information
* Eg, when fragment is destroyed
* And then recreated.
* The App looks more professional and the user experience will improve when an App can remember its former state.
* How does Android Studio do it?

Saving Fragment State: onSaveInstanceState

* Uses a Java HashMap to store and then retrieve key/value pairs.
* An specialized Fragment event is called just before a fragment is destroyed
  + onSaveInstanceState(bundle outstate)
* similar to storing extra info when navigating through intents/activities.

In Class Activity

* Add logging to Fragment class in the usual way. Use Log.e( not Log.i)
* Override the following methods.
* Fragment:
* --(initating events) onAttach, onCreate, onCreateView, onActivityCreated, onstart, onResume
* --(terminating events) onPause, onSaveInstanceState, onStop, onDestroyView, onDestroyView, onDetach

Bind the fragment to the MianActivity

* Try it, and track the program flow??

Fragment Communication

* Fragments do not communicate directly to one another. Why not??
  + Have to bundle together, use one have to involved another.
* Use the main activity the “host” to pass messages.

Tight VS. Loose Coupling

* Two software modules, wish to communicate with each other (tight coupling).
  + Import them into your App and directly instantiate their objects.
  + Make them each implement a known interface, then pass references to their objects as that interface type (loos coupling).
* Both approaches above will work. What are some pros&cons of each approach??
  + Invoke both fragments may or may not together need.
* Ever wonder why interfaces are called contracts??
  + All the abstract functions have to be implemented in any class test.

Setting yourself apart: Use interfaces.

* What if software Module A, only sometimes needs module B?
* Would you be able to compile Software module A, without including/shipping Software module B??
  + Interfaces are a little more work to be sure.
  + You ability to create these software contrasts will set you apart from other engineers.
* Try it, choose image and send a message
  + Use a ListView to select from multiple items.

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 Components Tree.??
* Grid Settings
  + numRow,
* Span Multiple rows, columns

Pinch Points: Changing Minimum SDK

* Step1, open build.gradle\*, change targetSdkVersion & minSdkVersion values to what you desire.
* Step2, in android studio menu, find “build” and choose “clean project”

Change linear layout in XML.

Let 7.

Tight vs. Loose Coupling

* If you have two software modules communicate with each other,
  + Import them into your App and directly instantiate their objects.
  + Or make each implement a known interface, then pass reference to their objecs as that interface type.
* Both approaches above will work. What are some pros & cons of each approach?
* Ever wonder why interfaces are called contracts??

Android v6.0+ permission model

* Grant permissions as needed instead of older version required permission in advance.

AppCompat

* Sometimes activates and fragments that we create descend from the Activity and Fragmetn Class.
* Other items, they descend from Class prefixes with the Appcompat. The import statemnts are a bit different in this case too.
* Q: When would you want to descend from an AppCompatActivity?
* A: when you want your App to work on older phones with older versions of Android.
* R is class that grading you as you program.

Shared Preferences

* SimpleSharedPref App.
* Demo

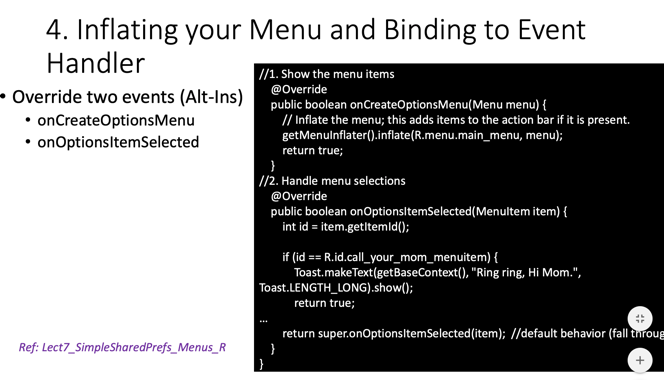
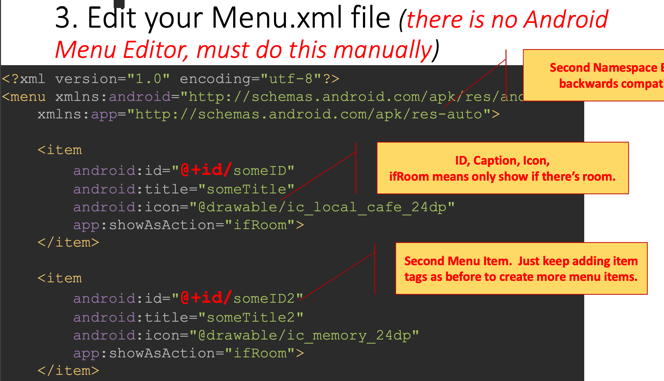
1, menus

* + Adding a Menu Item.
  + First create a folder under res called “menu”
    - Note you may already have this depending on the type of project you initially created.
  + Right Click on Menu and select New Menu Resource File
  + Provide a name for your menu
  + If you want to be selective you can apply a filter just like you did for dimensions, Locale, etc.

2, Vector Asset Studio – Adding Menu Icons

* + Its best to load the menu icons in advance of creating the menus
    - This will ensure you can establish proper references to the icons.

3, Edit your Menu.xml file (these is no Android Menu Editor, must do this manually)



Fragment Transactions

* Fragment Manager
  + Every Activity has a FragmentManager associated with it.
  + Grab a reference to it using
    - getFragmentmanager()
    - getSupportFragmentManager() // why are there two?
    - Then can use it to find Fragments associated with your Activity.
    - Very similar to findViewByID(), eg, from last week…
  + Eg, BottomFragment receivingFragment = (BottomFragment) getSupportfragmentManger().findFragmentById(R.id.fragment);
  + What happened to findViewByID?

Fragment Transactions

* There I another fragment management class.
* Antoehr way to interact with fragments, but at a transactional level.
* Frist:
  + What is a transaction?
  + What does commit do in a database?
  + What does rollback do in a database?
  + Why do we need them in a database?
* We can use commit and rollback with fragemnt transactions, for the same reasons we do in a database.

Fragment Transactions

* Fragments are bound to the Activity when they are “attached”, ref: onAttach() of the Fragment.
* To get a reference to the fragment in code, use the FragmetnManager.
  + Every activity has one, just get the reference.
* To interact with the fragments,
* Instantiate them
* Add them to a layout
* Find the reference to the FragmentTransaction Class.
  + Again, every Activity has one.

Fragment Transactions

* Common Calls are:
  + Show(), hide(), replace(), attach(), detach()
* Calling hide and replace actually destroys the fragment.
  + Implication is you have to save state
  + Nice thing is that it uses a bundle in the same way Activities do.
* Calling detach does NOT destroy the fragment.
* When would you want to do this instead of calling hide()?

Q&A??

* How do a fragment work?
* Can fragments run on their own? -> Not
* How many transactions can you run in a FragmentTransaction class? ->
* True/False: FindViewByID() will find fragments? -> False (getFragmentMangager, findFragmentID())
* True/False Fragments exist as a Java Object. -> True
* True/False: it’s fine for one fragment to call another. -> False
* What is best practice for fragment communication. -> Activity implements each interface of fragments.
* What tight coupling? -> two modules communicate each other directly.
* What hosts a fragment. -> Each activity has a FragmentManager.
* True/False: it is safe to interact with fragment views during the onCreate event of the fragment. -> False. Have to wait onCreate finished, then call fragment, and view.
* True/False: Fragments get inflated during the onCreateView() method.

Custom Listviews

-A touch of elegance-

Listviews

* ListViews can be used to present a set of choices, sort of like a dropdown (spinner). Recall this simple ListView code from earlier in the semester.

Protected void onCreate (Bundle savedInstanceState) {

Super.onCreate(saveInstantState);

setContentView(R.layout.activity\_main);

String[] menu = { “Burger”, “Pizza”, “Chicken”};

ListAdapter LA = new ArrayAdapter<String> (this, Android.R.layout.simple\_list\_item\_1, menu);

ListView lstMyList = (ListView) findViewById(R.id.lstMyList);

lstMyList.setAdapter(LA);

lstMyList.setOnItemClickListener( new AdapterView.onItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String menuItem = String.valueOf(Parent.getItemAtPosition(position)) ;

Toast.makeText( getBaseContext(), menuItem, Toast.LENGTH\_LONG);

}

});

Custom ListViews – step and step

* Think atomically, that is in terms of the items in each of the rows of your custom list view.
* Steps:
  + 1, Create the layout of a row (no java source)
    - 1, Edit the layout as if you were developing a fragment
  + 2, Setup Data Resources
    - 1, Eg, Simplest is a parallel array of websites, drawables, text, etc.
    - Each index of these arrays will map to an item in the listview.
  + 3, Create and Bind a Custom Adapter
    - 1, Binds the Data Resources to the ListView’s Layout
  + 4, Override core Adapter methods:
    - 1, so that the list render the way you want.
    - 2, So that onClick events are handled properly.

Write your own Custom Adapter: Override the BaseAdapter Class

* BaseAdapter Provides Mapping Between:
  + Elements (indices) of the datasource
  + Individual Views within each Row of the listview.
* Write your Custom Adapter that extends BaseAdapter class.
* Override/Implement the 4 coure methods of BaseAdapter.

Public int getCount()

Public Object getItem(int position)

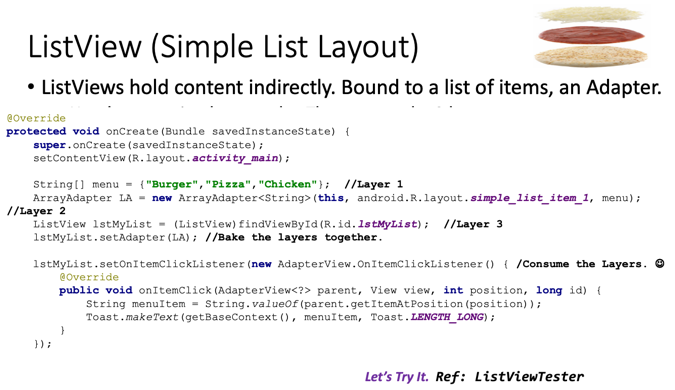
Public long getItem(int position)

Public view getView(int position, View convertView, ViewGroup parent)

* Do what do the 4 method do?

ListView and Adapters

* ListViews can be used to present a set of choices. d
* ListViews indirectly hold content (text or other views)
* Adapters actually contain the content
* ListView Bind to the Adapters.
* How do the Adatpters get content?
  + Simplest way is from an Array, eg, an Array of Strings.
* ListView(Simple List Layout)
  + ListViews hold content indirectly. Bound to a list of items, an Adapter.



Custom ListViews More complex

* The text list view’s adapter was an array of Strings.
* But you could put an array of almost anything into your Adapter.
* A custom listview is just a more complex version of a simple text listview.
* Used to incorporate multiple views into a list item.
* Makes an App Look and Feel much more professional
* Nice way to allow scrolling an App.
* Very similar to RecyclerView

Start Building

* Think atomically, that is in terms of the items in each of the rows of your custom list view
* Steps. P18.
* For Overriden BaseAdapter Methods.
  + getCount(), return total number of rows
  + getItem(), get the object that was clicked.
  + getItemID() return the ID(a numeric) of the clicked object
  + getView(): gets the view that was selected. This is where the secret sauce goes.
* baseAdapter mapping datasource and row of listvView.

What is a RecyclerView

* It’s a modernized Custom ListView.
* Much more efficient.
* Enforecs usage of ViewHolder Design Pattern.
* Enforces app compatibility all the way back to Androidv7.
  + Implication: your entire App should also be AppCompat to version7.
* RecyclerView does;t come pre-installed.

Intents : Location :

* Location
* More on intents
* Instant can be implicit or Explicit
* Explicit intent:
  + We use an explicit intent when we know what we want to do and how we want to do it.
* IMPLICIT INTENT
  + We use an implicit intent when we know what we want to do, but we trust our system to have an App that can do it for us.
  + Implicit intents take a Uniform Resource Identifier as a parameter, a string that identifies a task to be achieved. For example, make a phone call, open a map, send an SMS.
  + You might have several applciatiosn on your device that can handle these intents.
  + Android framework let’s the user choose the app to complete the task.
  + For this reason only install reliable Apps.
* Intents Example: location
* Let start simple by broadcasting an intent to utilize location services. Don’t worry about the syntax.
* Did we forget to update the Permissions in the Manifest?
* We did not specify Google maps Anywhere? How will it know to open Google Maps.

Broadcasting the Intent

* In this case we are broadcasting an intent
* It’s like sending up the bat signal. You hope Batman will answer, but you might get someone else, since many people are listening.

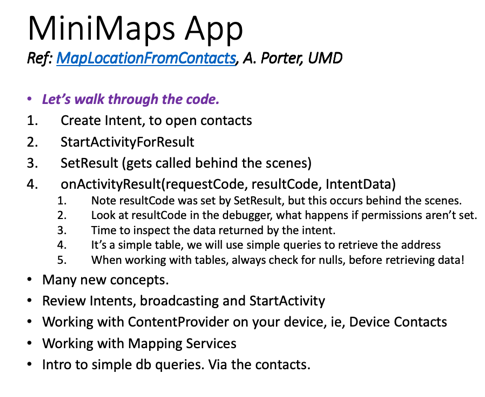
StartActivityForResultContentProvidersContacts, SMS, Phone, Maps

StartActivityForResult vs. StartActivity

* Sometimes the calling App wants a result back from its broadcasted intent. Eg, was the call successful?
* Two way to start an Activity.
  + StartActivity(intent)
    - Simply pass it an intent, and a new activity will get spawned.
  + StartactivityForResult(Intent, RequestCode)
    - Very similar to above
    - Also takes a request code to be returned later (by reference).
    - Returns a value to the Calling Activity.

StartActivityForResult vs. startActivity

* Once the new activity is spawned, it is responsible for setting its own result.
  + Not care the result of user does
* Activity.setResult()
  + Typical Result Codes:
    - RESULT\_CANCELED – eg, user hit back button.
    - RESULT\_OK – eg, activity completed normally
* setResult works hand in hand with the onActivityResult event.
* Void onActivityResult ( int requestCode, int resultCode, Intent data)
  + requestCode – the code we initially passed in StartActivityForResult
  + resultCode – result of the Activity we just called. Hopefully RESULT\_OK.
  + Intent – Reference to the original Intent, but only if we passed one into SetResult initially (SetResult is overloaded and the Intent is optinal)
* MiniMaps App & Revisiting Intents

Lecutre 7

Sometimes the calling App wants a result back from its broadcasted intent. Eg, Was the call successful?

* Two ways to start an Activity.
* StartActivity(intent)
  + Simply pass it an intent, and a new Activity will get spawned.
* StartActivityForResult(Intent, RequestCode)
  + Very similar to above
  + Also takes a request code to be returned later(by reference).
  + Returns a value to the Calling Activity.

Specific Intents - Localized speech to text Photos Video

* Preview: Starting Activites
  + Remember an activity can be started in two ways.
  + startActivtiy (Intent i);
    - the Activity + Intent over the fence and hope…
  + Or StartActivityForResult, start the activity and get a result back, eg, data+success/failure info.
  + StartActivityForResult(Intent I, int requestCode); // requestCoee is like dye, we will track this later.
  + Result gets passed back through an event, not a return value.
  + Protected void onActivityResult(int requestCode, int resultCode, Intent data)

What is a RecyclerView

* It’s modernized Custom ListView
* Much more efficient.
* Enforces usage of ViewHolder Design Pattern
* Enforces app compatibility all the way back to Android v7.
  + Implication: Your entire App should AppCompat to version 7.

Recycler View: Where does the name com from?

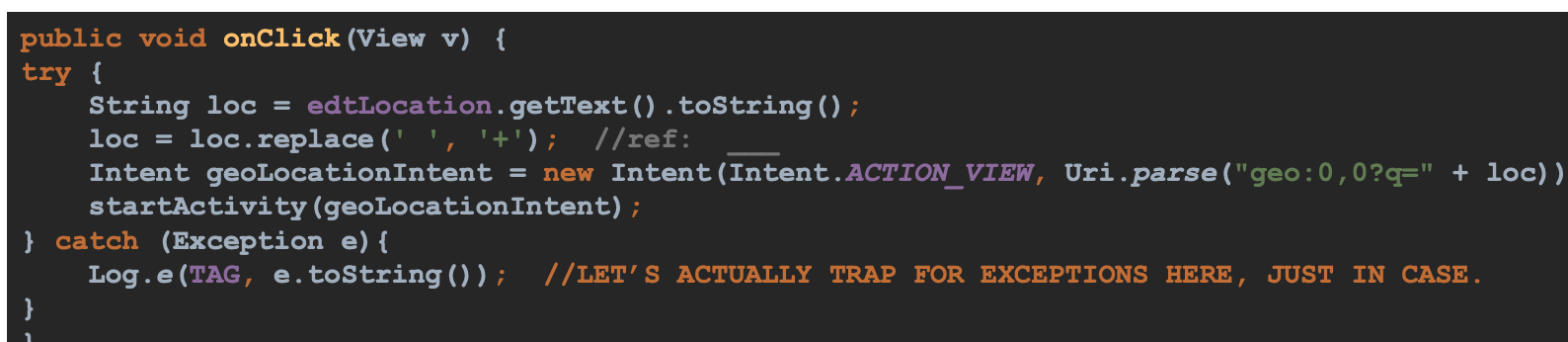
* RecyclerView – similar to a Customized ListView, but better.
* Declare -> Class; Instantiate -> instance; recycle -> Refer back to the ViewHolder Reference, instead of repeatedly calling findViewByID(..)
* Recycler View only comes in an AppCompat Flavor, meaning it will work with versions 7 and higher.
* RecyclerView doesn’t come pre-installed.
  + In the Pallete, select Common, then click on the download arrow next to “RecyclerView”
* New Android Permission Model, Android 6/7. Users grant permissions as needed. but 5 and before, it was necessary to publish your permission in advance.

Lecture 8.

More on Intents

* Intents can be implicit or Explicit.
* Explicit Intent
  + We use an explicit intent when we know what we want to do and how we want to do it.
* Implicit intent:
  + We use an implicit intent when we know what we want to do, but we trust our system to have an App that can do it for us.
  + Implicit intents take a Uniform Resource Identifier as a pareameter, a string that identifies a task to be achieved. For example, make a phone call, open a map, send an SMS.
  + You might have several applications on your device on your device that can handle these intents.
  + Android Framework let’s the user choose the app to complete the task.
  + For this reason only install reliable Apps.
* Intents Example: location, location.
  + Some of the more interesting Apps utilize location services, Eg. Yelp Waze, etc.
  + You can use it too, Free.
  + You can either perform your own geocoding.
  + Or you can leverage someone else’s work, eg, Google Maps via intents.
* It’s all about:

Let’s start simple by broadcasting an intent to utilize location services. Don’t worry about the syntax (yet).



Broadcasting the Intent

* In this case we are broadcasting an intent.
* It’s like sending up the bat signal. You hope Batman will answer, but you might get someone else, since many people are listening.

Pulling it all together

Localized Speech to text photos video

* Review: Starting Activites
* Remember an Activity can be started in two ways.
  + StartActivity(Intent i); // just throw
    - The Activity + Intent over the fence and hope…
  + Or StartActivitForResult, start the activity and get a result back, eg, data + success/failure info.
* StartActivityForResult( Intent I, int requestCode); // requestCode is like dye, we will track this later.
* Result gets passed back through an event, not a return value.
* Protected void onActivityResult(int requestCode, int resultCode, Intent data)

Chapter 3,

* 1, **We must import View.OnClickListener** to use code like View.OnClickListener
* 2, **onClick** is the abstract method of View.OnClick Listener
* 3, **addView** method to add a child View to a parent View.
* 4, add **this** inside the private class Y, which to access the current object of the Y class.
* 5, access the current object of the **X.this** class inside the private
* 6, **getWindowManager().getDeraultDisplay.getSize(size);** to retrieve the size of the screen.
* 7, the data type of this is **Context** in the code of GridLayout gridLayout = new GridLayout(this)
* 8, **setRowCount** to set the number of rows of the grid
* 9, **setCountentView** method to set the view for an activity
* 10, **Button b = new Button(Activity);** to instantiate a button.
* 11, **LayoutParams** class to be used by views to tell their parents how they wan to laid out.
* 12, **setGravity** method to specify the alignment of the text within a TextView.

Chapter 7  
1 A.setOnTouchListener

2.B getAction  
3. C bringToFront  
4. B GestureDetector  
5. C Public static inner interfaces of GestureDetector 6. D Constants that the action can be compared to 7. B onTouchEvent

18  
protected void onCreate( Bundle savedInstanceState ){

super.onCreate( savedInstanceState ); // Your code goes here

GestureAndTapHandler gth = new GestureAndTapHandler( );

detector = new GestureDetector( this, gth ); detector.setOnDoubleTapListener( gth );

detector.onTouchEvent( event ); }

19  
public boolean onTouchEvent( MotionEvent event ){

// Your code goes here detector.onTouchEvent(event)

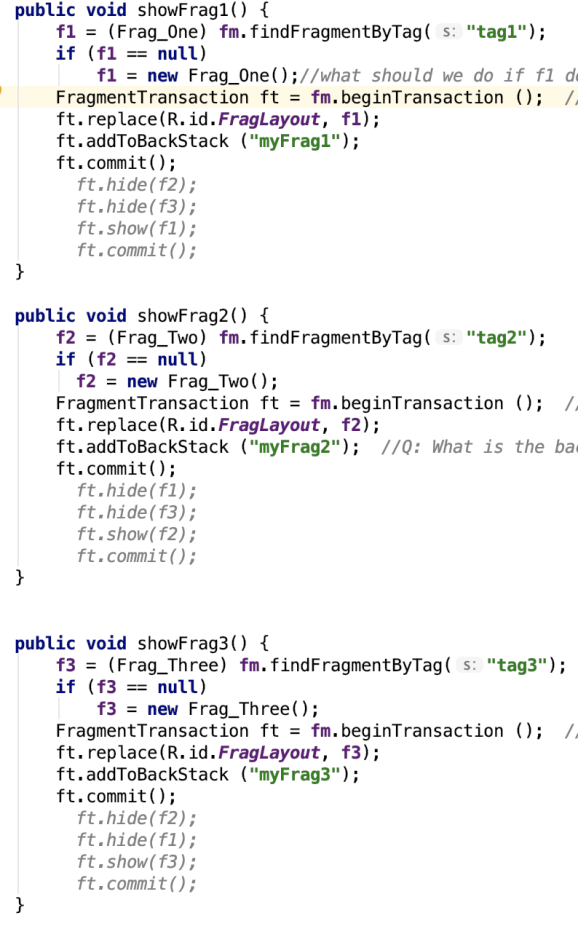
}

**W6\_P1**

Code I modified to make the project run properly:

⚫The first way to modify:  
Use replace rather than hide and show, attach and detach

⚫In methods showFlag2 and showFlag3, they lack **f2 = (Frag\_Two)**



**fm.findFragmentByTag("tag2");** and **f3 = (Frag\_Three) fm.findFragmentByTag("tag3");**

⚫In case f1 and f2 might be null, method showFlag1 and showFlag2 should add **if ( f1 == null)**

**f1 = = new Frag\_One(); if ( f3 == null) f3 = = new Frag\_Three();**⚫In showFlag1 method, replace the code **ft.hide(f2); ft.hide(f3); ft.show(f1);** with

**ft.replace(R.id.FragLayout,f1);ft.addToBackStack ("myFrag1");** In showFlag3 method , replace the code **ft.detach(f1); ft.detach(f2); ft.attach(f3);** with **ft.replace(R.id.FragLayout,f3);ft.addToBackStack ("myFrag3");**

**If use replace, the Activity Lifecycle would be:  
The last flagment: onPause onStop onDestroyView  
Flagment that should show: onCreateView onActivityCreated, onStart, onResume**

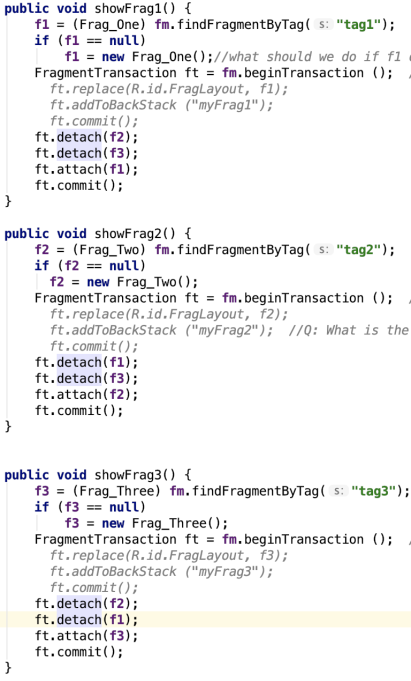
⚫The second way to modify:  
Use hide and show rather than replace, which won’t cause change in Activity Lifecycle, But remember to also add fragment2 and fragment3 first



⚫The third way to modify:  
Use attach, detach, also remember to add fragment2 and fragment3 first

The lifecycle would be

**The last flagment: onPause onStop onDestroyView  
Flagment that should show: onCreateView onActivityCreated, onStart, onResume**



HW 7. Bouns points

**1. Why is this component called a Recycler View? What is getting recycled?**

Because instead of creating views every time user scrolls, views in RecyclerView creates views once and reuse them if needed.

The views are getting recycled, they will be filled with different data when user scrolls up/down.

**2. What are some advantages that the Recycler View has over a CustomListView?**

1. RecyclerView brings more powerful animation function while ListView lacks support of good animations
2. The concept of ViewHolder: it’s optional for ListView but mandatory for RecyclerView, it helps solve many problems faced in ListView. There is no need to call findViewById().
3. RecyclerView supports both grids and lists.

**3. Which of the two components are easier to work with and why?**

ListView is easy to use because you don’t have to customize your ViewHolder, which is required to do for RecyclerView.

**4. Why did Google move the CustomListView component to the Legacy Tab in the Tool Pallete? Is this the same thing as deprecating this functionality?**

CLV is moved to Legacy Tab mainly because the new solution(RecyclerView) has better performance.

I think it’s same. The purpose for Google that moves CLV to Legacy Tab is, if you just get start with this area, you should choose RecyclerView, but if you have already used CLV for a long time and still want to use it, you can still find it at Legacy Tab.

**5. What is required, in terms of gradle settings/changes to utilize the Recycler View?**

implementation 'com.android.support:recyclerview-v7:28.0.0'



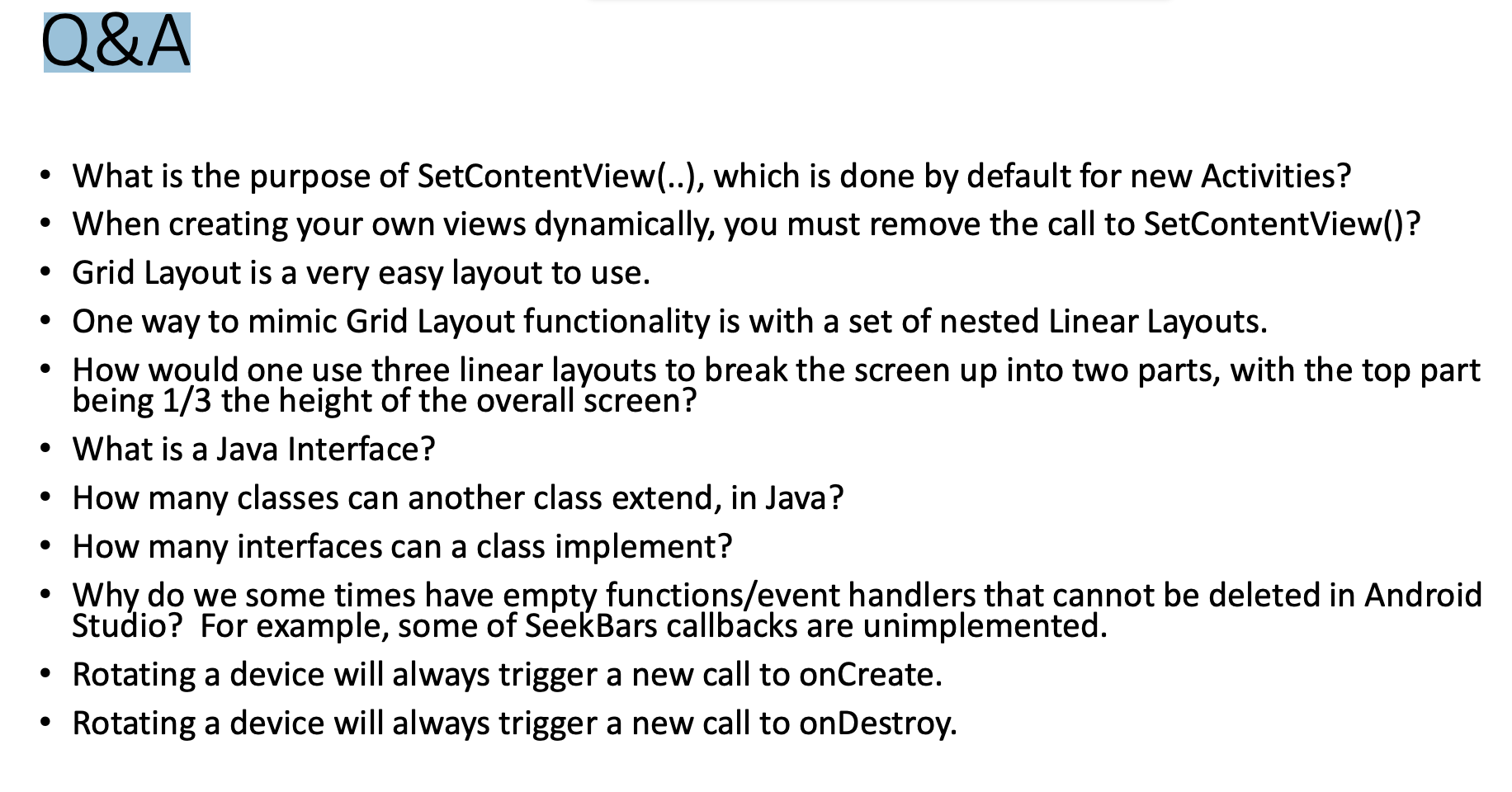
1, when activity destroy, save middle state. onRestoreInstance. Bundle 传activity, eg, use Instant.

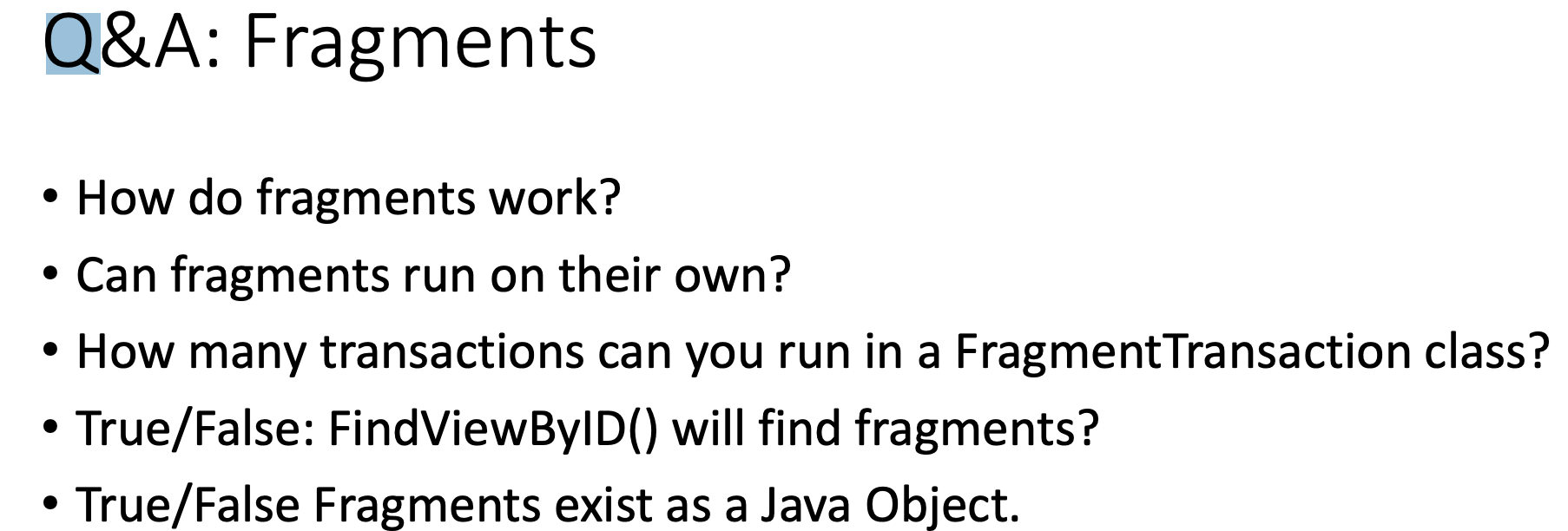
2, False

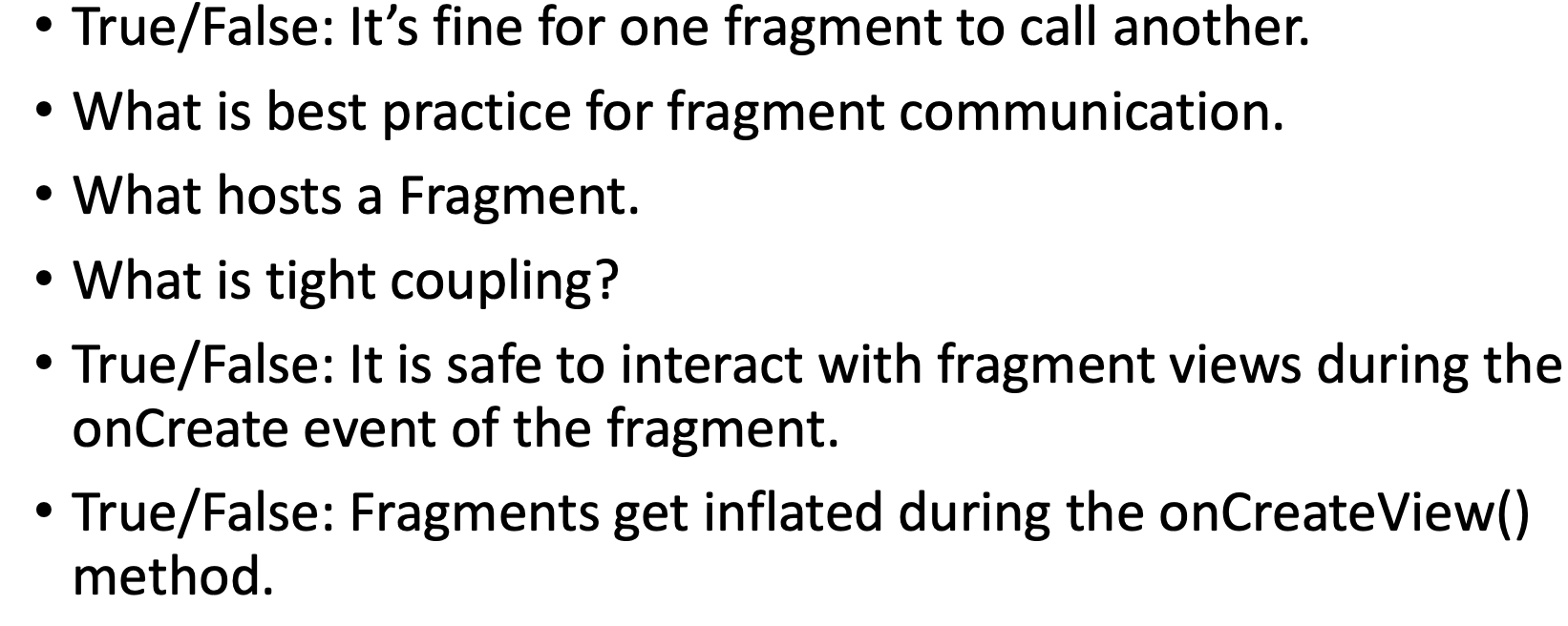
3, onSaveInstanceState, treanform data by key, values pairs.

3, False. Byte code, java code complie to Byte, run on DVM.

4, save and transform data.







Chapter 3:

·       14, creates a GridLayout within the current context and sets its number of rows to four and its number of columns to two.

GridLayout gd =  new GridLayout(this);

gd.setColumnCount(2);

gd.setRowCount(4);

·       15, This code create a button within the current context.

Button button = new Button(getContext());

·       16, This code creates a 5 \* 2 two-dimensional array of buttons within the current context.

Button [][]  buttons = new Button[5][2];

for( int i = 0; i < 5;  i++) {

         for(int  j=0; j < 2 ; j ++) {

                     buttons[i][j] = new Button(getContext());

         }

}

·       17, this code adds a Button object named b, specifying its width and heights as 200 pixels each, to an already created GridLayout object named gl.

Button b = new Button(getContext);

b.setLayoutParams(new LinearLayout.LayoutParams(200, 200));

gl.addView(b);

·       21, This code checks if the button that was clicked is a button name b. If it is, it outputs to Logcat YES, otherwise, it outputs to Logcat NO.

Private class ButtonHandler implements View.OnClickListener {

         Private static final String TAG = “LOG”;

         Public void onClick(View v) {

                     If(v == b) {

                                 Log.i(TAG , “YES”);

                     }else{

                                 Log.i(TAG  G, “NO”);

         }

}

Chapter 4:

·       1, The TableLayout class can be used to organize various GUI conpoents: **As a table of rows and columns.**

·       2, The direct superclass fo LinearLayout and RelativeLayout is **ViewGroup**

·       3, TableLayout and TalbeRow are direct subclasses of **LinearLayout**

·       4, The RelativeLayout class is a good choice to organize various GUI components **So that we position components relative to other components**

·       5, in **android.content** package is the intent class

·       6, After you have created an intent for a new activity, **startActivity**method of the Activity class you call with that intent parameter in order to start a new activity?

·       7, **onRestart** method of the activity class is automatically called when an activity is about to restart?

·       8, What methods of the Activity class (and in what order) are automatically called when an activity is first created?

**On create, on start and onResume**

·       9, **onStop** method of the activity class is automatically called when an activity becomes invisible to the user?

·       10, Two activities can share the same data?

o  **Yes, for example by each accessing a public static instance variable from another class.**

Chapter 7

1 A.setOnTouchListener

2.B getAction

3. C bringToFront

4. B GestureDetector

5. C Public static inner interfaces of GestureDetector

6. D Constants that the action can be compared to

7. B onTouchEvent

18

protected void onCreate( Bundle savedInstanceState ){

   super.onCreate( savedInstanceState );

   // Your code goes here

   GestureAndTapHandler gth = new GestureAndTapHandler( );

   detector = new GestureDetector( this, gth );

   detector.setOnDoubleTapListener( gth );

    detector.onTouchEvent( event );

}

19

public boolean onTouchEvent( MotionEvent event ){

  // Your code goes here

   detector.onTouchEvent(event)

}

Chapter 3:

· 14, creates a GridLayout within the current context and sets its number of rows to four and its number of columns to two.

GridLayout gd = new GridLayout(this);

gd.setColumnCount(2);

gd.setRowCount(4);

15, This code create a button within the current context.

Button button = new Button(getContext());

16, This code creates a 5 \* 2 two-dimensional array of buttons within the current context.

Button [][] buttons = new Button[5][2];

for( int i = 0; i < 5; i++) {

for(int j=0; j < 2 ; j ++) {

buttons[i][j] = new Button(getContext());

}

}

17, this code adds a Button object named b, specifying its width and heights as 200 pixels each, to an already created GridLayout object named gl.

Button b = new Button(getContext);

b.setLayoutParams(new LinearLayout.LayoutParams(200, 200));

gl.addView(b);

· 21, This code checks if the button that was clicked is a button name b. If it is, it outputs to Logcat YES, otherwise, it outputs to Logcat NO.

Private class ButtonHandler implements View.OnClickListener {

Private static final String TAG = “LOG”;

Public void onClick(View v) {

If(v == b) { Log.i(TAG , “YES”);

}else{Log.i(TAG G, “NO”);

}

}

Chapter 4:

· 1, The TableLayout class can be used to organize various GUI conpoents: As a table of rows and columns.

· 2, The direct superclass fo LinearLayout and RelativeLayout is ViewGroup

· 3, TableLayout and TalbeRow are direct subclasses of LinearLayout

· 4, The RelativeLayout class is a good choice to organize various GUI components So that we position components relative to other components

· 5, in android.content package is the intent class

· 6, After you have created an intent for a new activity, startActivitymethod of the Activity class you call with that intent parameter in order to start a new activity?

· 7, onRestart method of the activity class is automatically called when an activity is about to restart?

· 8, What methods of the Activity class (and in what order) are automatically called when an activity is first created?

On create, on start and onResume

· 9, onStop method of the activity class is automatically called when an activity becomes invisible to the user?

· 10, Two activities can share the same data?

o Yes, for example by each accessing a public static instance variable from another class.

Chapter 7

1 A.setOnTouchListener

2.B getAction

3. C bringToFront

4. B GestureDetector

5. C Public static inner interfaces of GestureDetector

6. D Constants that the action can be compared to

7. B onTouchEvent

18

protected void onCreate( Bundle savedInstanceState ){

super.onCreate( savedInstanceState );

// Your code goes here

GestureAndTapHandler gth = new GestureAndTapHandler( );

detector = new GestureDetector( this, gth );

detector.setOnDoubleTapListener( gth );

detector.onTouchEvent( event );

}

19

public boolean onTouchEvent( MotionEvent event ){

// Your code goes here

detector.onTouchEvent(event)

}