Intro. to Android Programming in Java

Overall gist of programming in Android Java

1. Not a lot of hard programming
2. Much of it is WYSIWYG
   1. (What you see is what you get)
3. Components are the same as JAVA SWING
   1. label and text areas are named a little different
4. Layouts are basically the same
   1. but are not graphically manipulated
   2. ***NOT*** created in Java, but XML
5. Component details are changed in “properties”
   1. no code to change/create
   2. code for Component details are in XML
6. The new IDE REALLY does a lot of coding for you!!

Eclipse Project Layout

* it’s really important to understand the overall project layout
* then you can design your own project with it in mind

|  |
| --- |
| Overall Eclipse Project setup |
|  |

* java code
  + file for EACH activity window you create
  + will contain any button click, splash screens, etc…
* XML code
  + how the activity windows look
  + can be manipulated in either XML
* res/raw
  + raw (manually created) will contain any audio clips
  + drawable will contain your images
* manifest
  + covered later

Setting up your overall layout

* Need a overall goal!!
  + Hello World
* start first with the “window”
  + name each window
  + how they link together (much like an HTML page(s) )
  + what will they have on the windows, images, buttons, etc…
    - pages are rectangular
      * they can be landscape or portrait
    - have the same type of layouts as Java SWING

|  |
| --- |
| Window & Eclipse overall Layout – Hello World |
|  |

The coding for an Activity Window Layout

* can find in res/layout in the Eclipse Project
* each window is an XML file
  + need to click on the lower ???.xml tab to see the XML code
* can either design using IDE graphical interface OR XML code
  + both end up in activity\_main.xml in our example

|  |  |
| --- | --- |
| Accessing the Layout | |
|  |  |

The Pre-Built Layouts

* much like Java’s SWING layouts!!
* use the graphical side of the IDE
  + not XML
* must place the layout FIRST before adding components
  + at least easier
* select Layout under “Palette” in GUI Designer

|  |  |
| --- | --- |
| Layout Options | |
|  | * GridLayout * LinearLayout (Vertical/Horizontal) * Relative   + where we get to place GUI components randomly throughout the window * Many others |

Storing global constants in strings.XML

* under res/values in Eclipse project
* where we have set are variables, and where variables are referenced from any other XML document
  + XML must have “@string/…”
  + think of it as a global constant, a variable that is used many times, and IF you need to change it, you only have one location to edit
* In the other XML you CAN eliminate the reference and just put in a value
  + android:text=*"Lupoli’s hello\_world"*
* the variables created here are linked to any of the XML files

|  |  |
| --- | --- |
| strings.XML and how it’s linked | |
| strings.xml |  |
| <?xml version="1.0" encoding="UTF-8" standalone="no"?>  <resources>  <string name="app\_name">Lupoli\'s Favorites</string>  <string name="**hello\_world**">Hello world!</string>  <string name="menu\_settings">Settings</string>  <string name="title\_activity\_main">MainActivity</string>  </resources>  Notice the \’ for any special characters |
| activity\_main.xml |
| <RelativeLayout xmlns:android=*"http://schemas.android.com/...*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  <TextView  android:id=*"@+id/textView1"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentTop=*"true"*  android:layout\_centerHorizontal=*"true"*  android:layout\_marginTop=*"28dp"*  android:text=*"@string/****hello\_world****"*  tools:context=*".MainActivity"* /> |

Messing with the Emulator

* Slow to start up
* but you can LEAVE it “on” and each time you run the project it will update
  + there really is no need to restart it
  + just keep it running while editing your code
* must unlock first!!
  + slide the LOCK to the right to unlock
* if you application is not running, there are two options
  + click the middle bottom icon (apps) and find your application among the other emulated apps
    - the application SHOULD show up as an app
      * this did not always happen
  + you can set the project to run in the Emulator as soon as you press run
    - BUT, this needs to be updated if you have

|  |
| --- |
| Setting up the Run Configuration |
|  |

* the dreaded “Unfortunately, … has stopped”
  + there is an error in your code (well, kinda)
  + had to find the error the HARD WAY by commenting out blocks of code

Layouts (XML side)

* remember that each activity window has a layout of some sort
* you can manipulate the XML
  + but the XML has some order to it

|  |  |
| --- | --- |
| Viewing the Different sides of Layouts | |
| <RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  <TextView  …  android:text=*"@string/hello\_world"*  tools:context=*".MainActivity"* />  <LinearLayout  …  <Button  android:id=*"@+id/btnFood"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"Food"* />  <Button  android:id=*"@+id/btnToy"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"Toy"* />  <Button  …  <Button  …  </LinearLayout>  <TextView  …  android:text=*"11:23 11-5-2012"*  …  <ImageView  …  <DigitalClock  …  </RelativeLayout> |  |

* Notice how the layouts have a separate closing tag, but the components do not (at the end of the tag /> )
  + even components COULD have a separate closing tag

XML vs. Graphical Layout Programming

* each give you the option to add/edit components
* up to you which one you would like to use
* it will come with experience that sometimes one is better for certain situations over another
* just use the tabs to switch views
* in XML, Eclipse will try to help you once you start some code

|  |
| --- |
| Eclipse, that little helper you!! |
|  |

Formatting your XML Programming

* <CTR><SHIFT><F> (Windows) or <CMD><SHIFT><F> (MAC)
  + formats the XML code
  + not always the nicest looking

Adding image/sound files to your Eclipse Project

* image filename MUST NOT contain spaces or dashes
* for images, Android supports
  + .png (preferred),
  + .jpg and
  + .gif (not great)
  + open Window’s Explorer and drag/drop ***on top*** of the directory required
    - res/drawable-hdpi directory for images
      * hdpi – high density screens
      * mdpi – medium density
      * …
* for sounds, ***CREATE*** the “raw” directory within res
  + raw will contain your sound clips

Shortcuts on Setting up a new Java File

* make sure you have added “extends Activity” to the Java file BEFORE you go to Source
* Can use Source 🡪 Override/Implement Methods
  + then select what you need
    - onCreate
    - onPause

|  |
| --- |
| Developing your Java file quickly |
|  |

The Android Manifest(o) File

* Keeping it all together
* named AndroidManifest.xml
  + at the bottom of the list of items in the Eclipse Project
* tracks
  + name of the application
  + each activity window and action
  + other stuff
* ***we have to maintain*** if we add other activity windows
* the XML file must contain an entry for EACH activity window we use
  + so four in total in this exercise

|  |
| --- |
| The manifest with multiple activity window |
| ...  <application  android:icon=*"@drawable/ic\_launcher"*  android:label=*"@string/app\_name"*  android:theme=*"@style/AppTheme"* >  <activity  android:name=*".Main"*  android:label=*"@string/title\_activity\_main"* >  <intent-filter>  <action android:name=*"android.intent.action.MAIN"* />  <category android:name=*"android.intent.category.DEFAULT"* />  </intent-filter>  </activity>  <activity  android:name=*".Food"*  android:label=*"@string/title\_activity\_food"* >  <intent-filter>  <action android:name=*"android.intent.action.FOOD"* />  <category android:name=*"android.intent.category.LAUNCHER"* />  </intent-filter>  </activity>  </application>  ... |

* main is DEFAULT and rest are LAUCHER
* android <intent-filter> name matches the java file name but in CAPS
* android <activity> name matches the java file name (case too)
  + but NO .java
  + and has a . (dot) in front

Setting up a Splash screen

* splash screen is a window that will appear only for a few seconds
* used for an intro, advertisement, etc
* YOU ARE REQUIRED TO HAVE A SPLASH SCREEN FOR EACH APP
  + your name
  + class
  + Current Date
  + name of project

|  |  |
| --- | --- |
| Lupoli’s Splash Screen (in Planning) | |
|  |  |

* procedure
  + place any audio or images in respective Eclipse Package res/directories
  + create a new activity named splash.java and activity\_splash.xml respectfully
  + add pic XML file
  + in main.java we will create a **thread** that will open the splash screen, then close it

|  |
| --- |
| Splash Screen setup within Main.java |
| Thread splashTimer = **new** Thread(){    **public** **void** run() {  // **TODO** Auto-generated method stub  **try** {  *sleep*(5000);  startActivity(**new** Intent(Main.**this**, Splash.**class**));  } **catch** (InterruptedException e) {  e.printStackTrace();  }  **finally**{ } // notice empty  }    };  splashTimer.start(); |

|  |
| --- |
| Splash Screen setup within Splash.java |
| Thread splashTimer = **new** Thread(){    **public** **void** run() {  // **TODO** Auto-generated method stub  **try** {  *sleep*(5000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  **finally**{ finish(); }  }    };  splashTimer.start(); |

Adding Sounds to your Activity Windows

* uses MediaPlayer to set up for long sounds
  + **import** android.media.MediaPlayer;
* uses SoundPool for shorter sounds
  + but more complicated
* make sure your sound file is
  + placed in the created raw directory in your project
  + filename is in all lower case letters
* file types that work
  + .m4a
  + .ogg
  + .wav (hopefully)
  + .mp3
* place in raw folder in the project
  + clean after each addition
* sounds can be attached to a
  + GUI Component
    - button click
    - button will need to be imported (adroid.widget)
  + Overall Activity Window
* Few personal notes
  + I was unable to get it to work WITHIN a try
    - required the MediaPlayer to be final
  + always crashed

|  |
| --- |
| Sound Setup within the Splash Activity |
| **public** **class** Splash **extends** Activity {  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_splash*);    MediaPlayer buttonClickSound = MediaPlayer.*create*(Splash.**this**, R.raw.*wfreakndo1*);  buttonClickSound.start();    Thread splashTimer = **new** Thread(){    **public** **void** run() {  // **TODO** Auto-generated method stub  **try** {  *sleep*(5000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  **finally**{ finish(); }  }    };  splashTimer.start();  }  } |

Helpful Websites

* Android Stuff
  + <http://developer.android.com/guide/components/index.html>
    - Official website for Android programming
  + <http://www.mybringback.com/tutorial-series/362/android-getting-started-tutorial/>
    - Great videos, but wastes a lot of time (doesn’t edit any mistakes)
* Sounds
  + <http://www.fozzy42.com/SoundClips.html>
  + [www.soundjay.com](http://www.soundjay.com)
  + [http://soundbible.com/](http://soundbible.com/" \t "_blank)

Sources:

<http://www.fozzy42.com/SoundClips.html> (for free music clip)

<http://www.youtube.com/playlist?annotation_id=annotation_44305&feature=iv&list=PLB03EA9545DD188C3&src_vid=CxPh1tgiK2g>