**121`**

ITMD 455/555 *Intelligent Device Applications* Lab 3

#### Famous Quotes App- 50 points

**Introduction**. Alright enough of the cold weather apps! This lab concentrates on the fundamentals of using Intents and multiple activities. You will create a ListView full of quotes, which will serve as the master view then when the user clicks on a particular quote they will be taken to a detail view of the full quote!

Also included in this package is a Splash screen for grad credit that will be added!

**STEP 1 Creating a New Android Project**

Create a new project with the Application Name called **Quotes**. Use API 23: Marshmallow, as your minimum SQK when going through the **Target Android Devices** wizard. Create an Empty Acitivity for your new project. Name your activity **QuoteReaderActivity** and include **activity\_quote\_reader** as your Layout Name. Keep defaults that appear as checked.

**STEP 2 Crafting your activity for your project. Your first activity screen will contain a list with quotes and thumbnails. Lets start by creating the UI for this.**

Open up your activity\_quote\_reader.xml file and include the following code:

<ListView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/quotes\_list"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

</ListView>

This will create a list view with a width and height equal to the size of the screen.

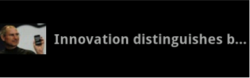
Delete your defaulted *TextView* tag included in the file.

**STEP 3 Add in rows to your list.**

In this step we will create rows and add them to our list. We know that a row needs to

have a thumbnail and text.

Here is how the first row is going to look:



I know it’s Steve Jobs, but who knows without him maybe there would be no Android!?

Lets start by creating the layout for the row. Create a new xml file in the res/layout directory. To add a XML, right click on your layout folder and choose New>XML>Layput XML File. Name your layout “list\_item\_layout”. Use a **RelativeLayout** as your Layout “Root Tag:” option. Next, open it in the raw XML view and replace the contents with the following objects:

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent" android:layout\_height="100dip"**>  
 <**ImageView  
 android:id="@+id/thumb"  
 android:layout\_width="60dip"  
 android:layout\_height="60dip"  
 android:layout\_centerVertical="true"  
 android:layout\_alignParentLeft="true"**/>  
 <**TextView  
 android:id="@+id/text"  
 android:layout\_toRightOf="@+id/thumb"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textSize="18dip"  
 android:layout\_marginLeft="10dip"  
 android:layout\_centerVertical="true"  
 android:singleLine="true"  
 android:ellipsize="end"  
 android:textStyle="bold"**/>  
</**RelativeLayout**>

Notice this creates an ImageView for the thumbnail and a TextView for the quote. Notice too that the parent view used in this layout is a RelativeLayout. This layout allows you to position its children relative to each other and itself. The ImageView needs to be placed to the left of the layout and it needs to be centered vertically hence those properties are set to true. The TextView also needs to contain text in a single line and if the length increases it should ellipsize, therefore the single line and ellipsize properties are set.

Okay now that we have the UI in place, lets add some code so that we can use it for each list item.

**STEP 4 Adding in a quotes and a data source for your adapter.**

For this step you will add in a class to serve as your adapter to help paint your

layout on your screen.

Open up your res/values/strings.xml file and add in the following quotes to your syntax:

<string name=*"quote\_1"*>Innovation distinguished between a leader and a follower</string>

<string name=*"quote\_2"*>I want to put a ding in the universe!</string>

<string name=*"quote\_3"*>People don\'t know what they want until you show it to them.</string>

<string name=*"quote\_4"*>The only way to be truly satisfied is to do what you believe is great work.</string>

<string name=*"quote\_5"*>That\'s been one of my mantras -- focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple. But it\'s worth it in the end because once you get there, you can move mountains.</string>

<string name=*"quote\_6"*>Great things in business are never done by one person, they\'re done by a team of people.</string>

<string name=*"quote\_7"*>You can\'t connect the dots looking forward; you can only connect them looking backward.</string>

<string name=*"quote\_8"*>Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work.</string>

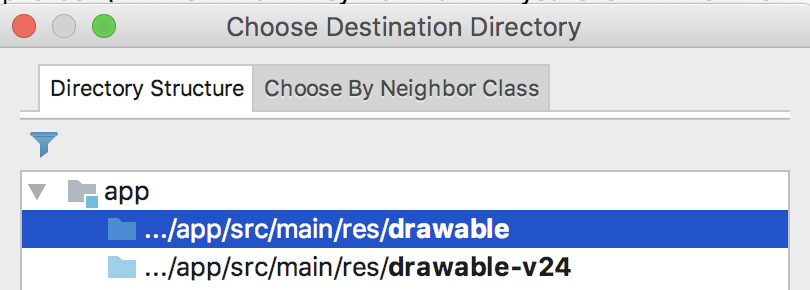
<string name=*"quote\_9"*>I\'m the only person I know that\'s lost a quarter of a billion dollars in one year. ... It\'s very character-building.</string>

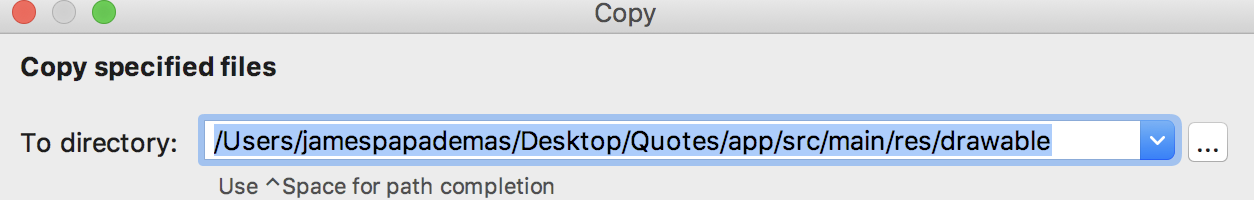
<string name=*"quote\_10"*>It\'s more fun to be a pirate than join the Navy.</string>

Notice you have to precede any single quotes with a backslash. You’ll be replacing some

quotes soon with your own, so not to worry Android lovers!

Next download the zip file I have provided in Blackboard under your Lab 3 link entitled ‘steve.zip’ that will serve as some pics to now add (copy in) into your drawable folder. Note the pics just serve as an example for now, as you can delete them later and add in your own! Sansphots below show pop ups to guide you thru your paste result.





Now that we have all our data, lets create our data source class. Right click on your package named **com.example.mypackage.quotes** locateddirectly under your **java folder** and choose **New** then select **Java Class**. Name it **DataSource**, and press OK when complete. Make sure the Package for the file is set to whatever package QuoteReaderActivity.java is part of (so no import is needed later). Add the following lines of code within the class:

**private** ArrayList<Integer> mPhotoPool;

**private** ArrayList<Integer> mQuotePool;

**private** ArrayList<Integer> mPhotoHdPool;

**public** ArrayList<Integer> getmPhotoHdPool() {

**return** mPhotoHdPool;

}

**public** ArrayList<Integer> getmPhotoPool() {

**return** mPhotoPool;

}

**public** ArrayList<Integer> getmQuotePool() {

**return** mQuotePool;

}

**public** DataSource() {

mPhotoPool = **new** ArrayList();

mQuotePool = **new** ArrayList();

mPhotoHdPool = **new** ArrayList();

}

All that we have done here is create 3 variables, generated their getter methods, and initialized the variables in the constructor.

You’ll notice you get warnings that it can’t find ArrayList with the supplied code so to recitfy the situation simply by adding in the necessary import namely

**import** java.util.ArrayList;

towards the top of the file, under your package name!

Next we need to populate the array lists with data which are the pictures plus the quotes. Notice that we have defined our array list to contain integers (ex. *steve\_1,* *quote\_1,…)*. The reason for this is when you drop in any resource into the resource directory the build system creates a *unique* identifier for it which is represented by an integer.

Okay now lets create 3 separate functions to populate our ArrayLists. Add the following functions above the constructor definition within your DataSource class.

**private** **void** setupPhotoPool() {

mPhotoPool.add(R.drawable.*steve\_1*);

mPhotoPool.add(R.drawable.*steve\_2*);

mPhotoPool.add(R.drawable.*steve\_3*);

mPhotoPool.add(R.drawable.*steve\_4*);

mPhotoPool.add(R.drawable.*steve\_5*);

mPhotoPool.add(R.drawable.*steve\_6*);

mPhotoPool.add(R.drawable.*steve\_7*);

mPhotoPool.add(R.drawable.*steve\_8*);

mPhotoPool.add(R.drawable.*steve\_9*);

mPhotoPool.add(R.drawable.*steve\_10*);

}

**private** **void** setupQuotePool() {

mQuotePool.add(R.string.*quote\_1*);

mQuotePool.add(R.string.*quote\_2*);

mQuotePool.add(R.string.*quote\_3*);

mQuotePool.add(R.string.*quote\_4*);

mQuotePool.add(R.string.*quote\_5*);

mQuotePool.add(R.string.*quote\_6*);

mQuotePool.add(R.string.*quote\_7*);

mQuotePool.add(R.string.*quote\_8*);

mQuotePool.add(R.string.*quote\_9*);

mQuotePool.add(R.string.*quote\_10*);

}

**private** **void** setupPhotoHDPool() {

mPhotoHdPool.add(R.drawable.*steve\_hd\_1*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_2*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_3*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_4*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_5*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_6*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_7*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_8*);

mPhotoHdPool.add(R.drawable.*steve\_hd\_9*);

mPhotoHdPool.add(R.drawable.*apple\_hd*);

}

Each of these functions is filling in data into our ArrayLists. Call these functions in your constructor like so:

setupPhotoPool();

setupQuotePool();

setupPhotoHDPool();

We will also add another function for the size of the data source. In our case the size of the data source is 10 i.e. we have 10 images and 10 quotes hence we can simply return the size of any one of our ArrayLists.

Add the following function above your constructor:

**public** **int** getDataSourceLength() {

**return** mPhotoPool.size();

}

That is all we need. Do a Save All at this point. Next we need to use this data source and populate our list.

**STEP 4 Creating your adapter.**

Open QuoteReaderActivity.java from the java folder and add in the following class named QuoteAdapter (code shown in **bold** below) directly within your QuoteReaderActivity class. This class should inherit from the BaseAdapter class and it should therefore implement all the abstract methods with minimal implementations.

Here’s what the class should look like now (no need for any other extraneous code):

**public** **class** QuoteReaderActivity **extends** AppCompatActivity {

**public class QuoteAdapter extends BaseAdapter {**

**@Override**

**public int getCount() {**

**return 0;**

**}**

**@Override**

**public Object getItem(int arg0) {**

**return null;**

**}**

**@Override**

**public long getItemId(int arg0) {**

**return 0;**

**}**

**@Override**

**public View getView(int arg0, View arg1, ViewGroup arg2) {**

**return null;**

**}**

**}**

/\*\* Called when the activity is first created. \*/

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.activity\_quote\_reader);

}

}

**Clean up your imports to resemble the following at this point:**

**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.BaseAdapter;

Lets try and understand what each method does.

* getCount(): Returns the number of items present in the data set.
* getItem(int position): Gets the data item associated with the specified position in the data set.
* getItemId(int position): Gets the row id associated with the specified position.
* getView( int position, View convertView, ViewGroup parent): This is used to get a view that displays the data at the specified position in the data set.

Now that we have a basic understanding of all the functions, add a constructor to our adapter and some private variables. Add the following lines of code to the QuoteAdapter class:

**private** Context mContext;

**private** LayoutInflater mInflator;

**private** DataSource mDataSource;

**public** QuoteAdapter(Context c) {

mContext = c;

mInflator = (LayoutInflater)

mContext.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);

mDataSource = **new** DataSource();

}

Add any imports where you see highlighted text in red as a flag. Go to the end of the flagged line and just do Alt+ Enter to allow for the automatic addition of a needed import statement.

The new mInflator variable will be used here to instantiate the list\_item\_layout XML file into corresponding View objects as will be shown below in the code.

Next change all the overridden functions as follows:

@Override

**public** **int** getCount() {

**return** mDataSource.getDataSourceLength();

}

@Override

**public** Object getItem(**int** position) {

**return** position;

}

@Override

**public** **long** getItemId(**int** position) { **return** position; }

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

ImageView thumbnail;

TextView quote;

**if**(convertView == **null**) {

convertView = mInflator.inflate(R.layout.*list\_item\_layout*, parent,

**false**);

}

thumbnail = convertView.findViewById(R.id.*thumb*);

thumbnail.setImageResource(mDataSource.getmPhotoPool().get(position));

quote = convertView.findViewById(R.id.*text*);

quote.setText(mDataSource.getmQuotePool().get(position));

**return** convertView;

}

Again add in import statements where your red flagged. The main function here is the getView function, since it is called for every item in the list view. It is responsible for binding the data source with the list view item. In it we first check if the **convertView** passed into the function is null, the reason we do this is to check if the convertView returned is not a recycled view. The adapter does not create a new view for each and every item. Instead it creates a set number of views and reuses them as the users scrolls through the list.

Once this is done we get a reference to our image view and text view and fill it with data from our data source. The position variable returned is used as an index in your data source.

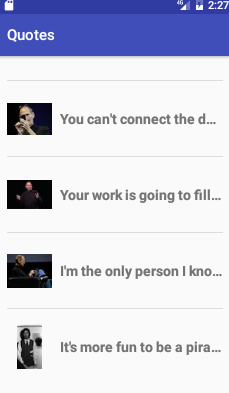
**STEP 5 Finalizing your Master View.**

We now have our adapter and data source set so we just need to tell our list view to start using our adapter. Add the following ListView declarations in your onCreate method within your QuoteReaderActivity class (an import will be needed)

ListView mListView = findViewById(R.id.*quotes\_list*);

mListView.setAdapter(**new** QuoteAdapter(**this**));

Now run the application with an acceptable emulator (I use the Nexus 5 API 28 x86 emulator) , and you should see a screen like this:

****

**Sweeet! Make sure to scroll up/down to see all quotes!**

**STEP 6 Setting up your detail screen.**

Next we want to *extend* our app features so the user can tap a quote to see it in full detail. It will show a large size picture and the quote written at the bottom.

Creating a new UI. Create a layout for your detail view screen by creating a new xml file in your res/layout directory and name it “quote\_detail.xml”. Use ScrollView as your “Root Tag:” Layout option. Add the following lines to the file:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<RelativeLayout

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

<ImageView

android:id=*"@+id/image"*

android:layout\_alignParentTop=*"true"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"350dip"*

android:layout\_marginLeft=*"5dip"*

android:src=*"@drawable/steve\_1"* />

<TextView

android:id=*"@+id/quote"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@+id/image"*

android:textSize=*"24sp"* />

</RelativeLayout>

</ScrollView>

Here notice that the root view is a scroll view and it has only one child, which is a relative layout. This relative layout contains our image view for the picture and text view for the quote. The reason why a scroll view is used is since the quote could be long there is a possibility that it might go off the screen. A scroll view is smart enough to understand when its child view is bigger than the screen size and will *automatically* give you a scroll bar to scroll down!

**STEP 7 Add in the QuoteDetail activity.**

Lets add the thrid and last activity that is responsible for presenting the

quote\_detail.xml layout details to the user.

Create a new **class** and name it QuoteDetail, make *sure* that it

**extends** **AppCompatActivity** along with the appropriate imports. Add in the following

code and the needed imports to your file:

**public** **class** QuoteDetail **extends** AppCompatActivity{

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.quote\_detail);

}

}

**STEP 8 Connecting your two activities (like a Master/Detail view).**

An activity in Android is instantiated with an Intent. An Intent is used to start an activity.

In QuoteReaderActivity.java add the following at the end of your onCreate function. Make sure to included needed imports!

mListView.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {

@Override

**public** **void** onItemClick(AdapterView arg0, View arg1, **int** position,

**long** arg3) {

Intent i = **new** Intent(QuoteReaderActivity.**this**, QuoteDetail.**class**);

i.putExtra("position", position);

startActivity(i);

}

});

What we are doing here is setting an item click listener and every time an item is clicked we create an Intent!

Notice that we pass the position variable to the intent, this is used to represent the position in the data source for the item clicked.

Next add the following lines (appearing in boldface) in QuoteDetail.java:

**private ImageView mImageView;**

**private TextView mQuote;**

**private int mPosition;**

**private DataSource mDataSource;**

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*quote\_detail*);

**Intent i = getIntent();**

**mPosition = i.getIntExtra("position", 0);**

**mDataSource = new DataSource();**

**mImageView = findViewById(R.id.*image*);**

**mQuote = findViewById(R.id.*quote*);**

**mImageView.setImageResource(mDataSource.getmPhotoHdPool().**

**get(mPosition));**

**mQuote.setText(getResources().getString(mDataSource.getmQuotePool()**

**.get(mPosition)));**

}

Include needed imports and then save all your files at this point to update your package. As you can see we retrieve the position sent by the QuoteReaderActivity from the intent and use that to fill in the image view and the text view.

A last thing we need to do before you can run the application is to add an entry for this activity explicitly in your manifest as a rule of thumb rule whenever we add a new activity

to the manifest.

So add these actvity tag lines to the manifest (AndroidManifest.xml):

<activity android:name=*".QuoteDetail"* />

Now run your application and test it. You should have a complete working app with both a master and detail view! Double Sweeet!!!

**STEP 9 Adjustments to your view. Adding in your own quotes and images.**

Go back your strings.xml file and adjust the last 5 quotes to some *other* famous

quotes.

Hunt down some appropriate pictures and replace the steve\_5 thru steve\_10 png’s with appropriate pics that match your quotes of some famous person. Replace also the steve\_hd\_5 thru steve\_hd\_10 png’s, which serve as somewhat larger versions of your non-hd files. Of course as you can notably see, the files when you ran your app show that the master view serves up a thumbnail file while the hd version serves as a larger file rendered in the detail view with the appropriate detailed quote. Also notice that the smaller files do not necessarily have to match the larger files but should somehow represent the person you’re quoting!

**STEP 10 Attention all Grads! Adding in a splash screen.**

As good practice for your final project, add in a splash screen to learn the process.

Your boss has complemented you on this great app! Now he/she wants a dynamic

splash screen added to the mix! As you know a splash screen is the first screen scene

at start up.

1. To add in a splash screen, grab an appropriate image and add it into your drawable folder.
2. Create a java file called Splash. Time the splash screen to appear at least for 5 seconds. When the timer is complete, allow an intent to load in your QuoteReaderActivity class file.
3. Make sure your Splash activity file is logged in your **manifest** file with an intent to launch. Finally tweak your manifest file to load the appropriate screen first at start up...Your file’s activity’s section should end up looking something like this:

<activity

android:name=*"com.example.quotes.Splash"*

android:label=*"@string/title\_activity\_splash"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*"com.example.quotes.QuoteReaderActivity"* />

<activity android:name=*".QuoteDetail"*/>



My Splash screen snapshot!

R.I.P. Stever!!!

**STEP 11 Submitting your assignment:**

For **full** credit turn in the following **pdf** files filled with this:

1. All your java files- should be 3, 4 if you’re doing the Splash screen thingy.
2. Your 5 XML files (including strings.xml and your AndroidManifest.xml files)- 6 if you’re doing the Splash screen.
3. Snapshots of your Master view of some quote other than the supplied Steve Jobs quotes and your Detail view also of this corresponding quote you picked from your Master view.
4. Submit your Splash screen snapshot as well if you are doing that for credit.