CS 213 – Software Methodology
Spring 2019

Sesh Venugopal

Lecture 20 – Apr 19
Android Programming

ListView/Multiple Activities/Raw resource
Example: Rutgers Bus Routes

Project

Make an Android application project called RU NB Bus Routes (or whatever).

Choose the Empty Activity option when asked to select a template.

(You should have a
`public class MainActivity extends AppCompatActivity`)

Refactor>Rename MainActivity to Routes, and activity_main.xml to routes_list.xml

Part 1:
Showing a List of Route Names

routes_list.xml layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     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”
     android:orientation="vertical"
     tools:context="com.example.sesh.runbbusroutes.Routes">

     <ListView
         android:id="@+id/routes_list"
         android:layout_width="match_parent”
         android:layout_height="match_parent”/>

</LinearLayout>
```

placeholder ListView, the actual list will be populated in the Java code

See https://developer.android.com/guide/topics/ui/binding

Routes in `strings.xml` file

```
<resources>
      ...
       <string-array name="routes_array">
              <item>A: College Ave/Busch</item>
              <item>B: Livingston/Busch</item>
              <item>C: Busch Commuter Shuttle</item>
              <item>EE: College Ave/Douglass-Cook</item>
              <item>F: College Ave/Douglass-Cook</item>
              <item>H: College Ave/Busch</item>
              <item>LX: College Ave/Livingston</item>
              <item>REXL: Douglass/Livingston</item>
              <item>REXB: Douglass/Busch</item>
       </string-array>
</resources>
```

We will read these names into our `ListView` when the app executes.

See https://developer.android.com/guide/topics/resources/string-resource.html

Layout for each ListView item

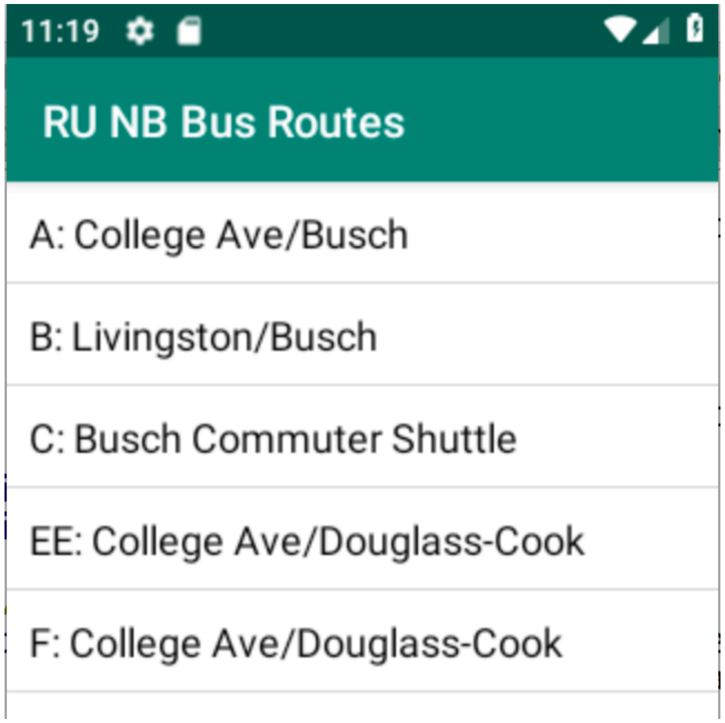
Make a file called route.xml in the layout folder, with a top level TextView tag, with values for text size, text color (e.g. black), background color (e.g. white), and padding.

route.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff"
    android:textColor="#000000"
    android:textSize="18sp"
    android:padding="10dp" />
```

#ffffff (hex) = White (RGB = 255,255,255)

#000000 (hex) = Black (RGB = 0,0,0)



RU NB Bus Routes

A: College Ave/Busch

B: Livingston/Busch

C: Busch Commuter Shuttle

EE: College Ave/Douglass-Cook

F: College Ave/Douglass-Cook

Coding the main (launch) activity

When the app is launched, the `onCreate` method in the `Routes` activity will be called. See the Manifest:

```
...
<activity android:name=".Routes">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

This activity will need to get the list of routes from `strings.xml` and populate the `ListView`

Coding the main (launch) activity

Load the list of routes from `strings.xml` and populate the `ListView`

`Routes.java`

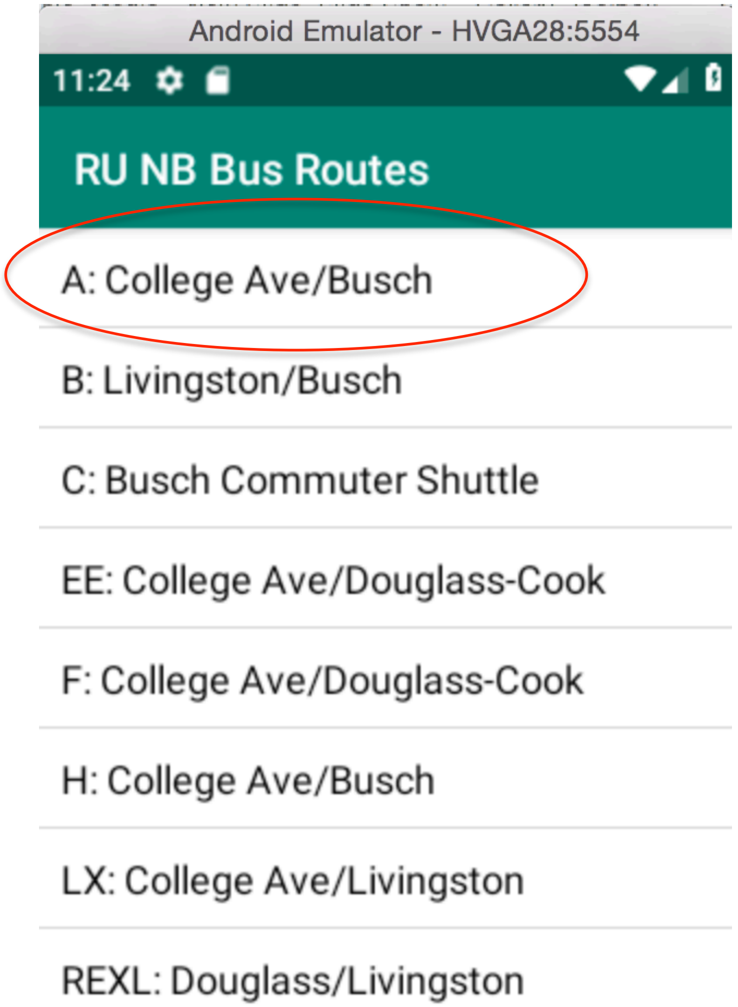
```
private ListView listView;
private String[] routeNames;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.routes_list);

   listView = findViewById(R.id.routes_list);
   routeNames = getResources().getStringArray(R.array.routes_array);
   ArrayAdapter<String> adapter =
      new ArrayAdapter<>(this, R.layout.route, routeNames);
   listView.setAdapter(adapter);
}
```

RUN THE APP!

Part 2:
Showing Details (Sequence of Stops) for Routes

List Item Selection – Showing Route Stops

When any of the list items (routes) is clicked, we want to show another screen with the details of all the stops on that route:



Android Emulator - HVGA28:5554

11:24

RU NB Bus Routes

A: College Ave/Busch

B: Livingston/Busch

C: Busch Commuter Shuttle

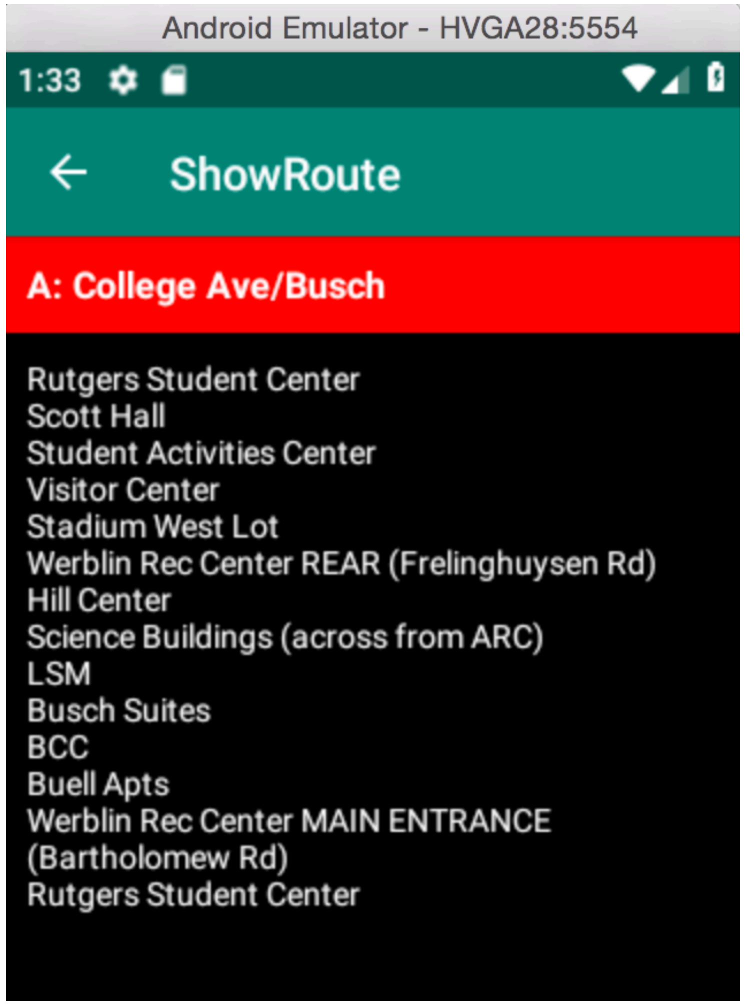
EE: College Ave/Douglass-Cook

F: College Ave/Douglass-Cook

H: College Ave/Busch

LX: College Ave/Livingston

REXL: Douglass/Livingston



Android Emulator - HVGA28:5554

1:33

ShowRoute

A: College Ave/Busch

Rutgers Student Center
Scott Hall
Student Activities Center
Visitor Center
Stadium West Lot
Werblin Rec Center REAR (Frelinghuysen Rd)
Hill Center
Science Buildings (across from ARC)
LSM
Busch Suites
BCC
Buell Apts
Werblin Rec Center MAIN ENTRANCE (Bartholomew Rd)
Rutgers Student Center

List Item Selection – Showing Route Stops

This requires us to do the following:

- Get the route details from somewhere (an input file)
- Set up a click listener for the list items
- Have the event handling code launch a new *detail activity* that has its own layout, and shows the details of the stops on the selected route
- Have the main activity pass to the new activity info on which item/route was clicked

Setting up a read-only input file

If a file is read-only, it can be placed in the resource space of the app:

- Set up a folder called `raw` within the res folder
- Place the file (`routes.txt`) in the `raw` folder

See https://developer.android.com/guide/topics/resources/providing-resources.html

`res/raw/routes.txt`

A

```
Rutgers Student Center
Scott Hall
Student Activities Center
Visitor Center
Stadium West Lot
Werblin Rec Center REAR (Frelinghuysen Rd)
Hill Center
Science Buildings (across from ARC)
LSM
Busch Suites
BCC
Buell Apts
Werblin Rec Center MAIN ENTRANCE (Bartholomew Rd)
Rutgers Student Center
*****************
Livingston Student Center
Quads
...
```

Each block of lines up to
****
is the sequence of stops on a route. The blocks are in the order of the routes listed in the array resource itemized in `strings.xml`

Reading from read-only input file

The raw resource file can be loaded into the program via `R.raw.routes`

`Routes.java`

```
...
private String[] routeDetails;   ← field in Routes class
...
InputStream is = getResources().openRawResource(R.raw.routes);
Scanner sc = new Scanner(new InputStreamReader(is));
routeDetails = new String[routeNames.length];

for (int i = 0; i < routeNames.length; i++) {
    StringBuilder sb = new StringBuilder();
    String line = sc.nextLine();
    while (!line.startsWith("*")) {
       sb.append(line);
       sb.append("\n");
       line = sc.nextLine();
    }
    routeDetails[i] = sb.toString();
}
...
```

Setting up a click listener/event handling code

Routes.java

```
...
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    public void onItemClick(AdapterView<?> parent, View view,
                            int position, long id) {
        showRoute(position);
    }
});
...
```

Or, you can use a lambda since `AdapterView.OnItemClickListener` is a functional interface:

```
...
listView.setOnItemClickListener(
    (p,v,pos,id) -> showRoute(pos));
...
```

The default Java compiler version setting in Android Studio is 1.7. Since lambdas are only supported in versions 1.8 and up, you need to change version to 1.8 by going to File -> Project Structure -> app and setting Source Compatibility and Target Compatibility to 1.8

Setting up a click listener/event handling code

Routes.java

```
public static final String ROUTE_NAME = "route_name";
public static final String ROUTE_DETAIL = "route_detail";


private void showRoute(int pos) {
   Bundle bundle = new Bundle();
   bundle.putString(ROUTE_NAME,routeNames[pos]);
   bundle.putString(ROUTE_DETAIL,routeDetails[pos]);
   Intent intent = new Intent(this, ShowRoute.class);
   intent.putExtras(bundle);
   startActivity(intent);
}
```

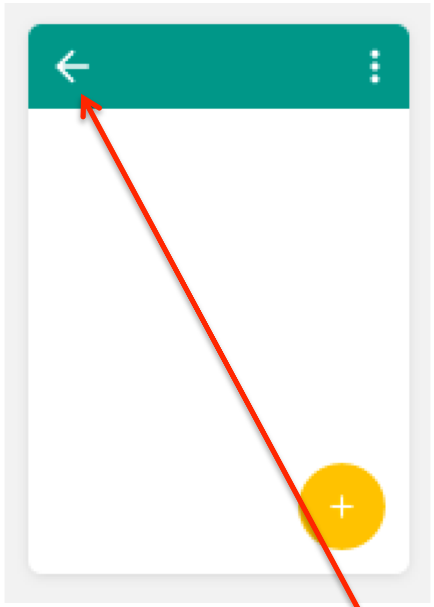
Activity to launch for showing route details.
MUST BE LISTED IN MANIFEST FILE

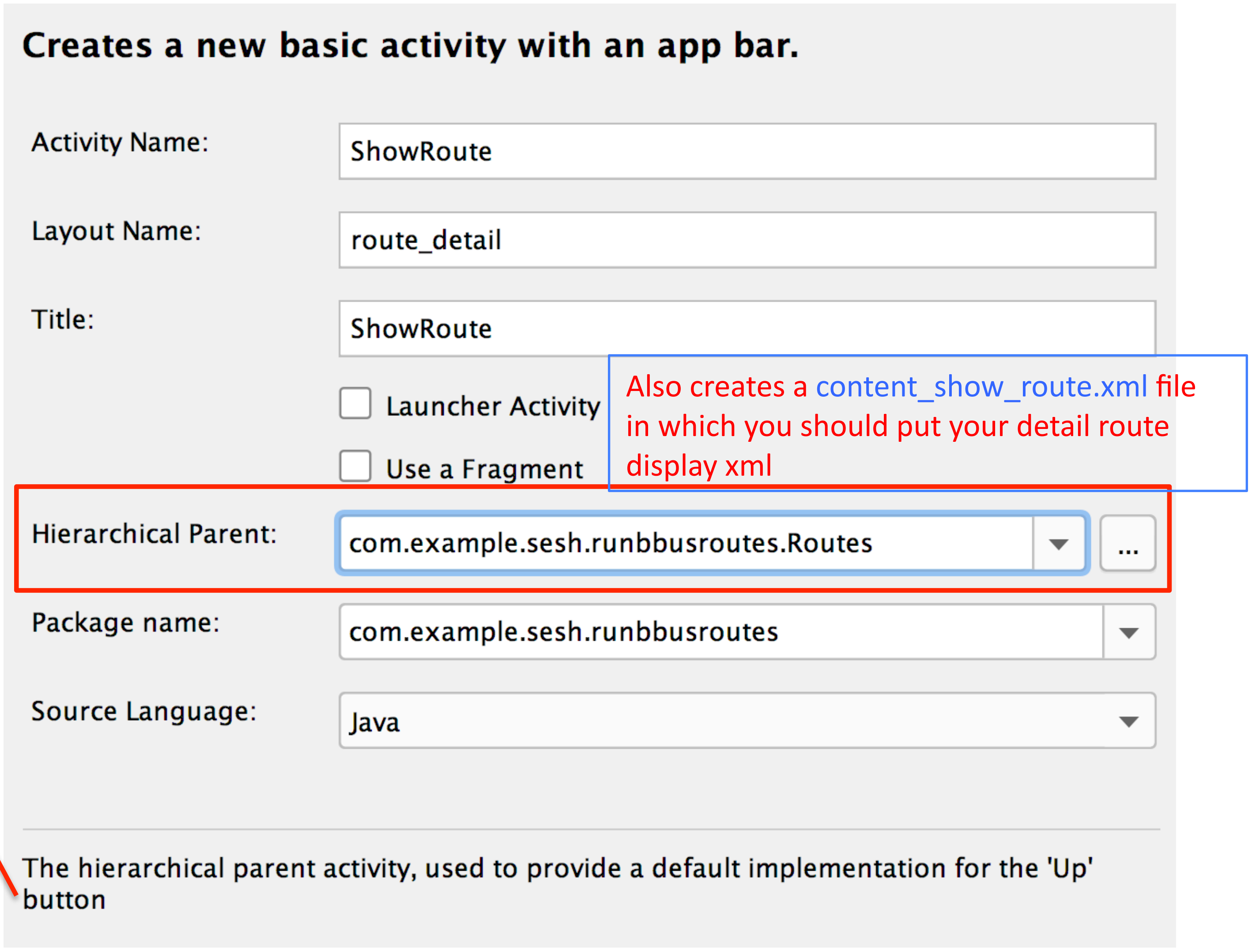
A `Bundle` can be populated with key,value mappings, and can be passed to another activity as info

See https://developer.android.com/guide/components/activities/parcelables-and-bundles.html

Implementing activity ShowRoute

File -> New -> Activity -> Basic Activity





Creates a new basic activity with an app bar.

Activity Name:	ShowRoute
Layout Name:	route_detail
Title:	ShowRoute
	☐ Launcher Activity
	☐ Use a Fragment
Hierarchical Parent:	com.example.sesh.runbbusroutes.Routes
Package name:	com.example.sesh.runbbusroutes
Source Language:	Java

Also creates a content_show_route.xml file in which you should put your detail route display xml

The hierarchical parent activity, used to provide a default implementation for the 'Up' button

Listing of `ShowRoute` activity in Manifest file

```
<activity

    android:name=".ShowRoute"
    android:label="@string/title_activity_show_route"
    android:parentActivityName=".Routes"
    android:theme="@style/AppTheme.NoActionBar">
    <meta-data
       android:name="android.suppory.PARENT_ACTIVITY"
        android:value="com.example.sesh.runbbusroutes.Routes" />

</activity>
```

See https://developer.android.com/guide/topics/manifest/manifest-intro.html

Designing a layout for ShowRoute

content_show_route.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent”
    android:layout_marginTop="?attr/actionBarSize">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/route_name"
        android:background="#ff0000"
        android:textColor="#ffffff"
        android:padding="10dp" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/route_detail"
        android:padding="10dp"
        android:textColor="#ffffff"
        android:background="#000000" />

</LinearLayout>
```

Red background

Need the name of the route And the detail content to be ID'd so they can be assigned values in the Java code

Black background

Coding activity ShowRoute

```
public class ShowRoute extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.route_detail);

        ...  // other auto-generated stuff

        // get the name and detail from bundle
        Bundle bundle = getIntent().getExtras();
        String routeName = bundle.getString(Routes.ROUTE_NAME);
        String routeDetail = bundle.getString(Routes.ROUTE_DETAIL);

        // get the name and detail view objects
        TextView routeNameView = findViewById(R.id.route_name);
        TextView routeDetailView = findViewById(R.id.route_detail);

        // set name and detail on the views
        routeNameView.setText(routeName);
        routeDetailView.setText(routeDetail);
    }
}
```

TAKE IT FOR A SPIN!!