MainActivity.java -------------------------------------------------------------------------------------------------------------------

**package** com.quarry.thetour;

**import** android.os.Bundle;

**import** android.app.Activity;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.View;

**import** android.content.Intent;

**public** **class** MainActivity **extends** Activity {

@Override

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

//Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.*main*, menu);

**return** **true**;

}

// This is will begin the sncMap Activity

**public** **void** onBeginClick(View view){

startActivity(**new** Intent(**this**, SncMapActivity.**class**));

}

// Will provide the user with the ability to choose options in the

// action bar or action overflow

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item){

**switch** (item.getItemId()){

**case** R.id.*action\_instructions*:

//openInstructions();

**return** **true**;

**case** R.id.*action\_settings*:

//openSettings();

**return** **true**;

**default**:

**return** **super**.onOptionsItemSelected(item);

}

}

}

Activity\_main.xml ------------------------------------------------------------------------------------------------------------------

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

<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"*

android:background=*"#FFFFFF"*

android:orientation=*"vertical"*

tools:context=*".BuildingIndexActivity"* >

<!-- The first imagView is the image of the building,, because it was

one of the first graphics I worked on, it may not scale correctly

on different devices. But this can be fixed by switching it with an

image of the maximum image size supported by android -->

<ImageView

android:id=*"@+id/building\_background"*

android:contentDescription=*"@string/content\_backgroundimage"*

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

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

android:layout\_alignParentTop=*"true"*

android:scaleType=*"centerCrop"*

android:src=*"@drawable/background\_building"* />

<!-- The second ImageView in this layout contains the graphic created

in Adobe Illustrator. Android works best with .png files especially

if the user morphs them into 9-patch drawing,, I'll explain that in

later documentation -->

<ImageView

android:id=*"@+id/graphic\_background"*

android:contentDescription=*"@string/content\_backgroundgraphic"*

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

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

android:layout\_alignParentBottom=*"true"*

android:layout\_alignParentLeft=*"true"*

android:scaleType=*"fitXY"*

android:src=*"@drawable/background\_graphic"* />

<!-- This Textview contains the snc\_welcome String located in the values

folder, it allows Android to display text on the screen -->

<TextView

android:id=*"@+id/building\_title"*

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

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

android:layout\_gravity=*"center\_vertical"*

android:gravity=*"center"*

android:text=*"@string/snc\_welcome"*

android:textColor=*"#FFFFFF"*

android:textSize=*"30sp"* />

<!-- This is the "Begin Tour" button and when pressed, will send the user to

the sncMap Activity. The Android's default button is gray and Square, kind

of boring. The user may change the 'style' of a button by settings its background

to a selector,, more on that later -->

<Button

android:id=*"@+id/btn\_to\_info"*

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

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

android:layout\_alignParentBottom=*"true"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginBottom=*"20dp"*

android:background=*"@drawable/ibtn\_begintour\_selector"*

android:gravity=*"center"*

android:minHeight=*"24dip"*

android:minWidth=*"32dip"*

android:onClick=*"onBeginClick"*

android:paddingLeft=*"15dip"*

android:paddingRight=*"15dip"*

android:text=*"@string/btn\_begintour"* />

</RelativeLayout>

SncMapActivity.java ---------------------------------------------------------------------------------------------------

**package** com.quarry.thetour;

**import** android.os.Bundle;

**import** android.app.AlertDialog;

**import** com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListener;

**import** com.google.android.gms.maps.GoogleMap.OnMapClickListener;

**import** com.google.android.gms.maps.CameraUpdateFactory;

**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;

**import** com.google.android.gms.maps.GoogleMap;

**import** com.google.android.gms.maps.SupportMapFragment;

**import** com.google.android.gms.maps.model.LatLng;

**import** com.google.android.gms.maps.model.Marker;

**import** com.google.android.gms.maps.model.MarkerOptions;

**import** com.google.android.gms.common.ConnectionResult;

**import** com.google.android.gms.common.GooglePlayServicesClient;

**import** com.google.android.gms.location.LocationClient;

**import** android.location.Location;

**import** android.support.v4.app.FragmentActivity;

**import** android.support.v4.app.NavUtils;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.widget.Toast;

**import** android.app.Activity;

**import** android.app.DialogFragment;

**import** android.app.Dialog;

**import** android.content.Intent;

**import** android.content.IntentSender;

**import** android.content.res.AssetManager;

**import** java.io.\*;

**import** java.util.ArrayList;

//This is the activity brought about by the Button (btn\_begintour)

//it sends the user to view that map and it begins in the middle of

//St. Norbert Campus.

**public** **class** SncMapActivity **extends** FragmentActivity **implements**

GooglePlayServicesClient.OnConnectionFailedListener,

GooglePlayServicesClient.ConnectionCallbacks, OnMapClickListener {

// This is used to check if Google Play Services is available

// when the user attempts to find their own location

**private** **final** **static** **int** *CONNECTION\_FAILURE\_RESOLUTION\_REQUEST* = 9000;

**private** GoogleMap sncMap;

**boolean** touchCoords = **false**; // used to switch on/off displaying touched

// coordinates on the map

**private** **double** sncLat = 44.445423;

**private** **double** sncLong = -88.067085;

**private** LatLng centercampus = **new** LatLng(sncLat, sncLong);

**private** LocationClient mLocationClient;

ArrayList<String> indexBList = **new** ArrayList<String>();

// This will be used to pass all of the names

//of the buildings to the indexbuildingactivity

ArrayList<Marker> mArray = **new** ArrayList<Marker>();

Location currentLocation;

// This functions initializes the sncMap Activity, inflating the layout.

// of the activity. The layout is chosen in the SetContentView function

// from the layout resources.

@Override

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

**super**.onCreate(savedInstanceState);

mLocationClient = **new** LocationClient(**this**, **this**, **this**);

setContentView(R.layout.*map\_snc*);

getActionBar().setDisplayHomeAsUpEnabled(**true**);

}

// this method will initialize the connection to the location client

// when the activity begins.

@Override

**protected** **void** onStart() {

**super**.onStart();

// connect the client

mLocationClient.connect();

}

// if the user soft-exits the Activity,, like rotating the device ,,

// this is called to re-start the sncMap Activity.

// This function is used to catch Intents thrown from the IndexActivity or the

// Building Info Activity. It catches the name of the building thrown and then

// determines its location on the map and animates the map to that position.

**protected** **void** onResume() {

**super**.onResume();

setUpMapIfNeeded();

Intent catchbuilding = getIntent();

**if** (catchbuilding.hasExtra("buildingtitle") || catchbuilding.hasExtra("buildingname")) {

String name;

// String test = catchbuilding.getStringExtra("buildingtitle");

**if**(catchbuilding.hasExtra("buildingtitle")){

name = catchbuilding.getStringExtra("buildingtitle");

catchbuilding.removeExtra("buildingtitle");

}

**else**{

name = catchbuilding.getStringExtra("buildingname");

catchbuilding.removeExtra("buildingname");

}

// Toast.makeText(this, name, Toast.LENGTH\_SHORT).show();

**if**(!mArray.isEmpty()){

/\*

\* new AlertDialog.Builder(this) .setTitle("Compared To:")

\* .setMessage(name) .show();

\*/

**for** (**int** i = 0; i < mArray.size(); i++) {

**if**(mArray.get(i).getTitle().equals(name)){

Toast.*makeText*(**this**, "Here it is!", Toast.*LENGTH\_LONG*)

.show();

sncMap.animateCamera(CameraUpdateFactory.*newLatLng*(mArray.get(i).getPosition()));

mArray.get(i).showInfoWindow();

}

}

}

}

}

@Override

**protected** **void** onSaveInstanceState(Bundle bundle) {

**super**.onSaveInstanceState(bundle);

}

// when the user switches screens we want to make sure that we disconnect

// from the location services because it drains batteries.

@Override

**protected** **void** onStop() {

mLocationClient.disconnect();

**super**.onStop();

}

// This function will add the academic buildings to the map from

// academicbuildings.txt in the 'assets' folder.

//

**public** **void** PopulateAcademicBuildings() {

AssetManager mgr;

String line = **null**;

**try** { // issue a try/catch in case any errors occur with the .txt file

mgr = getAssets();

InputStream is = mgr.open("academicbuildings.txt");

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr); // this will read the

// data from the

// file

String[] array; // will hold the data read from the file after it

// has been split

String buildingName, latitude, longitude;

**while** ((line = br.readLine()) != **null**) { // read one line at a time

// until end of file has

// been reached

array = line.split(" ");

// must accommodate for building name lengths here

**if** (array.length == 3) { // one name

buildingName = array[0];

indexBList.add(buildingName); // indexBList will later be

// used in

// BuildingIndexActivity

latitude = array[1]; // to create a list of buildings

longitude = array[2];

} **else** **if** (array.length == 4) { // two names

buildingName = array[0] + " " + array[1];

indexBList.add(buildingName);

latitude = array[2];

longitude = array[3];

} **else** { // three names, array.length == 5

buildingName = array[0] + " " + array[1] + " " + array[2];

indexBList.add(buildingName);

latitude = array[3];

longitude = array[4];

}

**double** lat = Double.*parseDouble*(latitude); // since the data is

// read as strings,

// we need to parse

// that data

**double** lon = Double.*parseDouble*(longitude); // into doubles so

// we can use the

// LatLng data type

LatLng buildingCoords = **new** LatLng(lat, lon);

Marker m = sncMap.addMarker(**new** MarkerOptions()

// sets marker properties and adds each one to the map

// according

.position(buildingCoords)

// to the data from the .txt file

.icon(BitmapDescriptorFactory

.*defaultMarker*(BitmapDescriptorFactory.*HUE\_BLUE*))

.title(buildingName)

.snippet("Click for Info"));

mArray.add(m);

/\*

\* new AlertDialog.Builder(this) .setTitle("Index Name Value")

\* .setMessage(buildingmarker.getName()) .show();

\*/

}

br.close(); // buffered reader

is.close(); // input stream | don't forget to close everything!

isr.close(); // input stream reader

} **catch** (IOException e1) {

**new** AlertDialog.Builder(**this**).setTitle("Read Error")

.setMessage("Something went wrong").show();

}

}

// This function is used to inhabit the map with Residential Building

// markers

// The same format applies as the Academic Buildings

**public** **void** PopulateResidentialBuildings() {

AssetManager mgr;

String line = **null**;

**try** {

mgr = getAssets();

InputStream is = mgr.open("residentialbuildings.txt");

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

String[] array;

String buildingName, latitude, longitude;

**while** ((line = br.readLine()) != **null**) {

array = line.split(" ");

// Accommodate for building name lengths here

**if** (array.length == 3) { // One name

buildingName = array[0];

indexBList.add(buildingName);

latitude = array[1];

longitude = array[2];

} **else** **if** (array.length == 4) { // Two names

buildingName = array[0] + " " + array[1];

indexBList.add(buildingName);

latitude = array[2];

longitude = array[3];

} **else** { // three names, array.length == 5

buildingName = array[0] + " " + array[1] + " " + array[2];

indexBList.add(buildingName);

latitude = array[3];

longitude = array[4];

}

**double** lat = Double.*parseDouble*(latitude);

**double** lon = Double.*parseDouble*(longitude);

LatLng buildingCoords = **new** LatLng(lat, lon);

Marker m = sncMap.addMarker(**new** MarkerOptions()

.position(buildingCoords)

.icon(BitmapDescriptorFactory

.*defaultMarker*(BitmapDescriptorFactory.*HUE\_GREEN*))

.title(buildingName)

.snippet("Click for Info"));

mArray.add(m);

/\*

\* new AlertDialog.Builder(this) .setTitle("Index Name Value")

\* .setMessage(buildingmarker.getName()) .show();

\*/

}

br.close();

is.close();

isr.close();

} **catch** (IOException e1) {

**new** AlertDialog.Builder(**this**).setTitle("Read Error")

.setMessage("Something went wrong").show();

}

}

// Inflate the menu; this adds items to the action bar if it is present.

// and will provide the user with the Action Overflow Icon,, indicated by

// the three vertical dots in the upper right-hand corner of the device.

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.*sncmap\_menu*, menu);

**return** **true**;

}

//This function will create the map if it has not already been created and center

// the view of the map in the middle of Campus

**private** **void** setUpMapIfNeeded() {

// Do a null check to confirm that we have not already instantiated the

// map.

**if** (sncMap == **null**) {

// Try to obtain the map from the SupportMapFragment.

sncMap = ((SupportMapFragment) getSupportFragmentManager()

.findFragmentById(R.id.*map*)).getMap();

// Check if we were successful in obtaining the map.

**if** (sncMap != **null**) {

setUpMap();

sncMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(

centercampus, 17.0f)); // This nifty little function begins the map in the middle of

// campus without the animation process, so the user doesn't

//have to watch the map move from its default position.

}

}

}

// This guy adds the markers to designate the different buildings

// across campus by calling the PopulateAcademicBuildings and

// PopulateResidentialBuildings functions.

// GREEN == RESIDENTIAL BUILDINGS

// BLUE == ACADEMIC BUILDINGS

**private** **void** setUpMap() {

sncMap.setOnMapClickListener(**this**);

PopulateAcademicBuildings();

PopulateResidentialBuildings();

//When a marker on the map is clicked, it will display an info window with the building's

// name and the snippet "Click for Info." This function allows that info window to be clickable

// and sends the information from the window to the BuildingInfoActivity.

sncMap.setOnInfoWindowClickListener(**new** OnInfoWindowClickListener() {

@Override

**public** **void** onInfoWindowClick(Marker m){

Intent sendtitle = **new** Intent(SncMapActivity.**this**,BuildingInfoActivity.**class**); // If the programmer wants the intent to send more than just

Bundle extras = **new** Bundle(); // one piece of data, he/she can use the Bundle data type to

extras.putStringArrayList("buildingList", indexBList); // store multiple variables and then just stick the bundle in

extras.putString("selectedbuilding", m.getTitle()); // the intent.

sendtitle.putExtras(extras);

startActivity(sendtitle);

}

});

sncMap.setMapType(GoogleMap.*MAP\_TYPE\_HYBRID*);

}

// this function will show an alert onscreen that will display

// the coordinates (Latitude/Longitude) of where the user

// has touched the map. A boolean can set it on or off.

@Override

**public** **void** onMapClick(LatLng point) {

**if** (touchCoords == **true**) {

**new** AlertDialog.Builder(**this**).setTitle("Location")

.setMessage("Location Touched:" + point).show();

}

}

// This determines if any item in the action overflow has been clicked

// ("Where am I?", "Filter Buildings", "Settings")

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.*action\_findme*:

currentLocation = mLocationClient.getLastLocation();

sncMap.setMyLocationEnabled(**true**);

LatLng pos = **new** LatLng(currentLocation.getLatitude(),

currentLocation.getLongitude());

sncMap.animateCamera(CameraUpdateFactory.*newLatLng*(pos));

**return** **true**;

**case** R.id.*action\_buildingIndex*:

Intent intent = **new** Intent(**this**, BuildingIndexActivity.**class**);

intent.putStringArrayListExtra("buildingNameList", indexBList);

// This clever little method will pass

// the names of all of the buildings

startActivity(intent); // read from the file to the

// buildingIndexActivity so we won't have to

**return** **true**; // read from the file again.

**case** R.id.*action\_settings*:

// openSettings();

**return** **true**;

**case** android.R.id.*home*:

NavUtils.*navigateUpFromSameTask*(**this**);

**return** **true**;

**default**:

**return** **super**.onOptionsItemSelected(item);

}

}

// This chunk of code will begin the process of finding a users location

// This section was borrowed from an Android Development Tutorial

// https://developer.android.com/training/location/retrieve-current.html#DefineCallbacks

**public** **static** **class** ErrorDialogFragment **extends** DialogFragment {

**private** Dialog mDialog;

**public** ErrorDialogFragment() {

**super**();

mDialog = **null**;

}

**public** **void** setDialog(Dialog dialog) {

mDialog = dialog;

}

@Override

**public** Dialog onCreateDialog(Bundle savedInstanceState) {

**return** mDialog;

}

}

@Override

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {

**switch** (requestCode) {

**case** *CONNECTION\_FAILURE\_RESOLUTION\_REQUEST*:

**switch** (resultCode) {

**case** Activity.*RESULT\_OK*:

**break**;

}

}

}

// Next define location services callbacks!!!!

@Override

**public** **void** onConnected(Bundle dataBundle) {

// this will display the connection status

// Toast.makeText(this, "Connected", Toast.LENGTH\_SHORT).show();

}

@Override

**public** **void** onDisconnected() {

// Display the connection status

Toast.*makeText*(**this**, "Disconnected. Please re-connect.",

Toast.*LENGTH\_SHORT*).show();

}

@Override

**public** **void** onConnectionFailed(ConnectionResult connectionResult) {

// Google Play Services can resolve some errors it detects.

// If the error has a resolution, try sending an Intent to

// Start a Google Play Services activity that can resolve

// Error.

**if** (connectionResult.hasResolution()) {

**try** {

connectionResult.startResolutionForResult(**this**,

*CONNECTION\_FAILURE\_RESOLUTION\_REQUEST*);

//

// Thrown if Google Play Services canceled the original

// PendingIntent

//

} **catch** (IntentSender.SendIntentException e) {

// Log the error

e.printStackTrace();

}

} **else** {

// If no resolution is available, display a dialog to the user with

// the error.

//

connectionResult.getErrorCode();

}

}

}

Map\_snc.xml ------------------------------------------------------------------------------------------------------------

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

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

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

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<!-- This view will allow the Android Device to display the Google Map

provided from the Google Play Services Library -->

<fragment

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

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

android:layout\_height=*"match\_parent"*

class=*"com.google.android.gms.maps.SupportMapFragment"* />

</LinearLayout>

BuildingIndexActivity.java --------------------------------------------------------------------------------------------

**package** com.quarry.thetour;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.InputStreamReader;

**import** java.util.ArrayList;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.content.res.AssetManager;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.view.ViewGroup;

**import** android.widget.ArrayAdapter;

**import** android.widget.Button;

**import** android.widget.ListView;

**import** android.widget.TextView;

**public** **class** BuildingIndexActivity **extends** Activity {

**private** Button btnmap, btninfo;

**private** ArrayList<String> buildingNames; // 'buildingNames' - this variable will inherit the building

// names passed by the intent from the SncMapActivity

@Override

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*building\_index*);

PopulateList(); // As soon as the Activity begins we want to populate the ListView

// with the building names.

}

// This function's purpose is to populate the ListView with all of the buildings read

// in from the text files. It requires an Array Adapter data type to do so/

//The Array adapter takes each of the items we want to be in the list and adapts them

// to the view.

**private** **void** PopulateList() {

buildingNames = **new** ArrayList<String>();

ArrayAdapter<String> adapter = **new** MyListAdapter(**this**, // The Array Adapter's parameters include,, the Context, What view to display on, and

R.id.*building\_title*, buildingNames); // a list of items to display.

ListView list = (ListView) findViewById(R.id.*building\_index\_lv*);

AssetManager mgr; // The AssetManager allows us to retrieve items from the Assets folder

String line = **null**; // like our text files containing the information on the buildings.

**try** { // issue a try/catch in case any errors occur with the .txt file

mgr = getAssets();

InputStream is = mgr.open("academicbuildings.txt");

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr); // this will read the

// data from the

// file

String[] array; // will hold the data read from the file after it

// has been split

String buildingName;

**while** ((line = br.readLine()) != **null**) { // read one line at a time

// until end of file has

// been reached

array = line.split(" ");

// must accommodate for building name lengths here

**if** (array.length == 3) { // one name

buildingName = array[0];

buildingNames.add(buildingName); // indexBList will later be

} **else** **if** (array.length == 4) { // two names

buildingName = array[0] + " " + array[1];

buildingNames.add(buildingName);

} **else** { // three names, array.length == 5

buildingName = array[0] + " " + array[1] + " " + array[2];

buildingNames.add(buildingName);

}

}

br.close(); // buffered reader

is.close(); // input stream | don't forget to close everything!

isr.close(); // input stream reader

} **catch** (IOException e1) {

**new** AlertDialog.Builder(**this**).setTitle("Read Error")

.setMessage("Something went wrong").show();

}

**try** { // issue a try/catch in case any errors occur with the .txt file

mgr = getAssets();

InputStream is = mgr.open("residentialbuildings.txt");

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr); // this will read the

// data from the

// file

String[] array; // will hold the data read from the file after it

// has been split

String buildingName;

**while** ((line = br.readLine()) != **null**) { // read one line at a time

// until end of file has

// been reached

array = line.split(" ");

// must accommodate for building name lengths here

**if** (array.length == 3) { // one name

buildingName = array[0];

buildingNames.add(buildingName); // indexBList will later be

} **else** **if** (array.length == 4) { // two names

buildingName = array[0] + " " + array[1];

buildingNames.add(buildingName);

} **else** { // three names, array.length == 5

buildingName = array[0] + " " + array[1] + " " + array[2];

buildingNames.add(buildingName);

}

}

br.close(); // buffered reader

is.close(); // input stream | don't forget to close everything!

isr.close(); // input stream reader

} **catch** (IOException e1) {

**new** AlertDialog.Builder(**this**).setTitle("Read Error")

.setMessage("Something went wrong").show();

}

list.setAdapter(adapter); // This function executes the adapter.

}

// The listView used in this app is not a default listView given by the android package. It is a shiny custom

// listView I created with an individual layout called index\_item\_view in the resources folder. In order to

// use a custom listView we need to implement a custom ArrayAdapter to handle it.

**private** **class** MyListAdapter **extends** ArrayAdapter<String> {

// as mentioned above, this is the constructor for the customArrayAdapter, it takes the context,

// the view to populate and the items to populate with.

**public** MyListAdapter(Context context, **int** textViewResourceID,

ArrayList<String> objects) {

**super**(BuildingIndexActivity.**this**, R.layout.*index\_item\_view*,

buildingNames);

}

//This function will retrieve and initialize the custom view (index\_item\_view)

// the important variable here is the 'position' variable, which will return

// what position in the listView each element is located.

@Override

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

View itemView = convertView;

**if** (itemView == **null**) {

itemView = getLayoutInflater().inflate(

R.layout.*index\_item\_view*, parent, **false**);

}

btnmap = (Button) itemView.findViewById(R.id.*btn\_to\_map*); // we'll initialize the buttons located in the custom view here

btnmap.setTag(position);

btnmap.setOnClickListener(onBtnMapClickListener);

btninfo = (Button) itemView.findViewById(R.id.*btn\_to\_info*);

btninfo.setTag(position);

btninfo.setOnClickListener(onBtnInfoClickListener);

// Find the building Titles to work with

String currentBuilding = buildingNames.get(position);

// Fill the view

TextView tv = (TextView) itemView.findViewById(R.id.*building\_title*);

tv.setText(currentBuilding);

itemView.setTag(btnmap); // setting the tag will allow the device to locate which

// index a button has been pressed at. could have used either btnmap or

**return** itemView; // btninfo because they are located in the same index.

}

**private** OnClickListener onBtnMapClickListener = **new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

**final** **int** position = (Integer) v.getTag(); // call this to retrieve what position in the index the button was located

Intent throwbuilding = **new** Intent(BuildingIndexActivity.**this**,

SncMapActivity.**class**);

throwbuilding.putExtra("buildingtitle",

buildingNames.get(position));

startActivity(throwbuilding);

}

};

**private** OnClickListener onBtnInfoClickListener = **new** OnClickListener(){

@Override

**public** **void** onClick(View v){

**final** **int** position = (Integer) v.getTag();

Intent throwdata = **new** Intent(BuildingIndexActivity.**this**,

BuildingInfoActivity.**class**);

Bundle extras = **new** Bundle(); // we will need to send a bundle to the InfoActivity

extras.putStringArrayList("buildingList", buildingNames);

extras.putString("selectedbuilding", buildingNames.get(position));

throwdata.putExtras(extras);

startActivity(throwdata);

}

};

}

}

Building\_index.xml ------------------------------------------------------------------

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

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

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

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<!-- This is the base ListView contained in the buildingIndexActivity, it will hold

an Array of views contained in the layout folder. It is to be noted that this will

contain the same view over and over again -->

<ListView

android:id=*"@+id/building\_index\_lv"*

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

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

</ListView>

</LinearLayout>

Index\_item\_view.xml --------------------------------------------------------------------------------------------------

<?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=*"match\_parent"*

android:background=*"#000000"* >

<!-- This is the view that the Index uses to populate itself.

The index view is just a repetition of the code below,, one

for each of the buildings listed in the app -->

<!-- This button will return the user to the map and send the name

of the building that has been selected to the sncMap activity

for further computing-->

<Button

android:id=*"@+id/btn\_to\_map"*

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

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

android:layout\_below=*"@id/building\_title"*

android:layout\_marginLeft=*"8dip"*

android:layout\_marginTop=*"6dip"*

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

android:layout\_toRightOf=*"@id/btn\_to\_info"*

android:background=*"@drawable/ibtn\_begintour\_selector"*

android:minHeight=*"24dip"*

android:minWidth=*"32dip"*

android:text=*"@string/btn\_to\_map"* />

<!-- This button will determine which position in the index the button was

located and then begin the Building Info Activity -->

<Button

android:id=*"@+id/btn\_to\_info"*

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

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

android:layout\_below=*"@id/building\_title"*

android:layout\_marginLeft=*"8dip"*

android:layout\_marginTop=*"6dip"*

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

android:background=*"@drawable/ibtn\_begintour\_selector"*

android:minHeight=*"24dip"*

android:minWidth=*"32dip"*

android:text=*"@string/btn\_to\_info"* />

<!-- This textview allows the index view to display the Title of each of the

buildings. Each building name is read from the files and added to the Array -->

<TextView

android:id=*"@+id/building\_title"*

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

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

android:layout\_alignParentLeft=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginTop=*"4dip"*

android:text=*"@string/index\_building\_title"*

android:textAppearance=*"?android:attr/textAppearanceMedium"*

android:textColor=*"#FFFFFF"*

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

</RelativeLayout>

BuildingInfoActivity.java -----------------------------------------------------------

package com.quarry.thetour;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Intent;

import android.content.res.AssetManager;

import android.os.Bundle;

import android.text.method.ScrollingMovementMethod;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.\*;

public class BuildingInfoActivity extends Activity {

private ArrayList<Images> imageArray = new ArrayList<Images>();

private ImageView iv;

private TextView tv, titleView; // these will display the name of the building and its description

private Button btnmap, btnindex;

private Intent catchdata; // will be used to retrieve the data from the incoming intent, containing building names

private Bundle extras; // this will be used to extract the bundled data from the incoming intent

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.building\_info\_layout);

catchdata = getIntent(); // caught the incoming intent

extras = catchdata.getExtras(); // retrieve the bundle of data from the intent

GetBuildingNames();

MatchImages();

AddDescription();

DisplayLayout();

}

//Used to retrieve the building names from the incoming intent

// and add them to the Name variable of the custom class : Images

private void GetBuildingNames() {

ArrayList<String> names = extras.getStringArrayList("buildingList");

for(int i = 0; i < names.size(); i++)

{

Images im = new Images(); //

im.setName(names.get(i));

imageArray.add(im);

}

}

// This function matches the picture of the building to the corresponding

// building name. It can do this because the program reads the buildings

// from the text file in the order in which they were added.

//The pictures of the buildings are extracted from the Drawables folder,

// A simpler way might be to read them in from an asset file but I do not know

// how.

private void MatchImages() {

imageArray.get(0).setID(R.drawable.jms);

imageArray.get(1).setID(R.drawable.cofrin);

imageArray.get(2).setID(R.drawable.bemis);

imageArray.get(3).setID(R.drawable.toddwehr);

imageArray.get(4).setID(R.drawable.boyle);

imageArray.get(5).setID(R.drawable.mulva);

imageArray.get(6).setID(R.drawable.bush);

imageArray.get(7).setID(R.drawable.madlor);

imageArray.get(8).setID(R.drawable.bergstrom);

imageArray.get(9).setID(R.drawable.mmm);

imageArray.get(10).setID(R.drawable.vanderzanden);

imageArray.get(11).setID(R.drawable.riverside);

imageArray.get(12).setID(R.drawable.kress);

imageArray.get(13).setID(R.drawable.gries);

imageArray.get(14).setID(R.drawable.burke);

imageArray.get(15).setID(R.drawable.sense);

imageArray.get(16).setID(R.drawable.vmc);

imageArray.get(17).setID(R.drawable.carriage);

imageArray.get(18).setID(R.drawable.townhouse);

imageArray.get(19).setID(R.drawable.michels);

}

//This function sets the description of the buildings by reading in each

// description from the file in the Asset folder. There were complications

// reading in chars such as ' " '.

private void AddDescription(){

AssetManager mgr;

String line = null;

int i = 0;

try { // issue a try/catch in case any errors occur with the .txt file

mgr = getAssets();

InputStream is = mgr.open("buildingdescriptions.txt");

InputStreamReader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr); // this will read the

// data from the

// file

// has been split

while ((line = br.readLine()) != null) { // read one line at a time

if(line.contains("'")) // Android does not know how to display some characters so they must

line = line.replaceAll("'", "\'"); // be manually handled.

imageArray.get(i).setDescription(line);

i++;

}

br.close(); // buffered reader

is.close(); // input stream | don't forget to close everything!

isr.close(); // input stream reader

} catch (IOException e1) {

new AlertDialog.Builder(this).setTitle("Read Error")

.setMessage("Something went wrong").show();

}

}

// This will initialize the entire layout and set all of the required variables

private void DisplayLayout(){

iv = (ImageView) findViewById(R.id.building\_imageview); // Sets the imageView to the corresponding view in the XML Layout File

tv = (TextView) findViewById(R.id.building\_desc\_tv);

titleView = (TextView) findViewById(R.id.building\_title\_tv);

tv.setMovementMethod(new ScrollingMovementMethod()); // This will allow the textView to scroll if it runs off the screen

String name = catchdata.getStringExtra("selectedbuilding");

btnmap = (Button) findViewById(R.id.btn\_info\_tomap);

btnmap.setOnClickListener(onBtnMapClickListener);

btnindex = (Button) findViewById(R.id.btn\_info\_toindex);

btnindex.setOnClickListener(onBtnIndexClickListener);

for(Images i : imageArray) // this 'for each' loop will run thru the Array of Images class and set each object's variables

{

if(i.getName().equalsIgnoreCase(name))

{

iv.setImageResource(i.getID());

tv.setText(i.getDescription());

titleView.setText(i.getName());

}

}

}

// This Listener will send the name of the building back to the Map Activity so the Map may

// animate to the desired buildling's position

private OnClickListener onBtnMapClickListener = new OnClickListener() {

@Override

public void onClick(View v) {

String name = catchdata.getStringExtra("selectedbuilding");

Intent throwbuilding = new Intent(BuildingInfoActivity.this,

SncMapActivity.class);

throwbuilding.putExtra("buildingname",name);

startActivity(throwbuilding);

}

};

// This Listener will simply send the user back to the index of buildings

private OnClickListener onBtnIndexClickListener = new OnClickListener() {

@Override

public void onClick(View v){

startActivity(new Intent(BuildingInfoActivity.this, BuildingIndexActivity.class));

}

};

// This custom class allows the images, building names, and descriptions to be grouped into one

// object. It also plays a major roll in the layout of the Activity as it can be implemented into

// an ArrayList for easy handling.

public class Images{

private String name;

private String description;

private int imageid;

public Images()

{

}

public Images(String iname, int id)

{

name = iname;

imageid = id;

}

String getName()

{

return name;

}

String getDescription()

{

return description;

}

int getID()

{

return imageid;

}

void setName(String n)

{

name = n;

}

void setDescription(String d)

{

description = d;

}

void setID(int id)

{

imageid = id;

}

}

}

Building\_info\_layout.xml

<?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=*"match\_parent"*

android:background=*"#000000"* >

<!-- This ImageView will hold the picture of the building, the default building image

is Bergstrom,, don't ask me why, I just needed a placeholder. The image that this

view holds will actually be determined dynamically in the BuildingInfo Activity -->

<ImageView

android:id=*"@+id/building\_imageview"*

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

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

android:contentDescription=*"@string/content\_buildingimage"*

android:scaleType=*"centerCrop"*

android:adjustViewBounds=*"true"*

android:src=*"@drawable/bergstrom"* />

<!-- This TextView contains the title of the building that will be displayed on the

Activity. Android does not appreciate it much if the user uses "Direct" text (android:text="Title")

instead of referencing a String from the values folder but the title is determined dynamically

as with everything else in this layout so it can piss off -->

<TextView

android:id=*"@+id/building\_title\_tv"*

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

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

android:layout\_below=*"@+id/btn\_info\_toindex"*

android:layout\_centerHorizontal=*"true"*

android:text=*"Title"*

android:textColor=*"#FFFFFF"*

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

<!-- This button button will return the user to the snc map activity

it uses the same selector as the "Begin Tour" button in the Main Activity -->

<Button

android:id=*"@+id/btn\_info\_tomap"*

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

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

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

android:layout\_alignParentRight=*"true"*

android:layout\_marginTop=*"20dp"*

android:layout\_marginRight=*"25dp"*

android:background=*"@drawable/ibtn\_begintour\_selector"*

android:minHeight=*"24dip"*

android:minWidth=*"32dip"*

android:text=*"@string/btn\_to\_map"* />

<!-- This button will return the user to the building index activity -->

<Button

android:id=*"@+id/btn\_info\_toindex"*

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

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

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

android:layout\_alignParentLeft=*"true"*

android:layout\_marginTop=*"20dp"*

android:layout\_marginLeft=*"25dp"*

android:background=*"@drawable/ibtn\_begintour\_selector"*

android:minHeight=*"24dip"*

android:minWidth=*"32dip"*

android:text=*"@string/btn\_to\_index"* />

<!-- This textview contains the description of the building, it is dynamically

determined once the activity is begun -->

<TextView

android:id=*"@+id/building\_desc\_tv"*

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

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

android:layout\_below=*"@+id/building\_title\_tv"*

android:layout\_centerHorizontal=*"true"*

android:layout\_marginTop=*"17dp"*

android:autoLink=*"web"*

android:maxLines=*"6"*

android:scrollbars=*"vertical"*

android:text=*"I&apos;m right here!"*

android:textAppearance=*"?android:attr/textAppearanceMedium"*

android:textColor=*"#FFFFFF"*

android:textSize=*"16sp"* />

</RelativeLayout>

Ibtn\_begintour\_selector.xml-----------------------------------------------------------------------------------------

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

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

<!-- This xml file serves as the graphic background for the buttons

used in the Application. Each button has three forms, it's idle

look, it's focused look, and it's pressed look, a different image

must be used for each of the three states. It is to be noted that

when creating a selector for buttons the three states must appear as

they are displayed in the order below -->

<item android:state\_pressed=*"true"*

android:drawable=*"@drawable/btnpressed"* />

<item android:state\_focused=*"true"*

android:drawable=*"@drawable/btnfocused"* />

<item android:drawable=*"@drawable/btndrawable"* />

<!-- If they user were to add a new graphic button they would need to input

the code sequentially for the three states of the button. ButtonPressed,

ButtonFocused, ButtonIdle as the order depicts above -->

</selector>

AndroidManifest.xml---------------------------------------------------------------------------------------------------

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

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

package=*"com.quarry.thetour"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<!-- This is where the user can change the targeted operating version and the minimum version after initially creating the project -->

<uses-sdk

android:minSdkVersion=*"11"*

android:targetSdkVersion=*"19"* />

<uses-permission android:name=*"android.permission.INTERNET"*/>

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"*/>

<uses-permission android:name=*"com.google.android.providers.gsf.permission.READ\_GSERVICES"*/>

<!-- The following two permissions are not required to use

Google Maps Android API v2, but are recommended. -->

<!--uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/-->

<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"*/>

<uses-feature

android:glEsVersion=*"0x00020000"*

android:required=*"true"*/>

<!-- The Android Manifest is a file that the programmer shouldn't modify outside of adding

permissions and activities -->

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/touriconfool"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.quarry.thetour.MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

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

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

</intent-filter>

</activity>

<meta-data

android:name=*"com.google.android.gms.version"*

android:value=*"@integer/google\_play\_services\_version"* />

<meta-data

android:name=*"com.google.android.maps.v2.API\_KEY"*

android:value=*"AIzaSyCfmBdDgbgIT5jEp3sLqnS2rHCvUXruiPU"*/>

<activity

android:name=*".SncMapActivity"*

android:label=*"@string/campus\_map\_label"*

android:parentActivityName=*"com.quarry.thetour.MainActivity"* />

<activity

android:name=*".BuildingIndexActivity"*

android:label=*"@string/buildingIndex\_label"*

android:parentActivityName=*".SncMapActivity"* />

<activity

android:name=*".BuildingInfoActivity"*

android:label=*"@string/buildinginfo\_label"*

android:parentActivityName=*".BuildingIndexActivity"* />

</application>

</manifest>