

Lab 4: Intents, Linkify, and Google Maps

Dr. Kanda Runapongsa Saikaew
Computer Engineering Department
<http://twitter.com/krunapon>

Agenda

- Introducing Intents
- Starting new Activities using implicit and explicit Intents
- Using linkify

Introducing Intents

- Intents are used as a message-passing mechanism that works both within your application and between applications
- Intents can be used to
 - Declare your intention that an Activity or Service be started to perform an action, usually with (or an) a particular piece of data
 - Broadcast that an event (or action) has occurred
 - Explicitly start a particular Service or Activity
- You can use Intents to support interaction among any of the application components installed on an Android device
 - This turns your device from a platform containing a collection of independent components into a single interconnected system

Using Intents to Start New Activities

- Explicitly
 - Specifying the class to load
- Implicitly
 - Requesting that an action be performed on a piece of data
 - The action need not be performed by an Activity within the calling application

Using Intents to Broadcast

- Intents can also be used to broadcast messages across the system
 - Any application can register Broadcast Receivers to listen for, and react to, these broadcast Intents
 - This lets you create event-driven applications based on internal, system, or third-party-application events
- Android broadcasts Intents to announce system events, like changes in Internet connection status or battery charge levels
- The native Android applications, such as the SMS manager, simply register components that listen for specific broadcast Intents such as incoming phone call

Using Intents to Propagate Actions

- Using Intents to propagate actions -- even within the same application -- is a fundamental Android design principle
- It encourages the decoupling of components, to allow the seamless replacement of application elements
- It also provides the basis of a simple model for extending an application's functionality

Using Intents to Launch Activities

- To open an Activity, call `startActivity`, passing in an Intent as shown in the following snippet:

```
Intent myIntent = new Intent(MyActivity.
```

```
this, MyOtherActivity.class);startActivity(myIntent);
```

- After `startActivity` is called, the new Activity (in this example `MyOtherActivity`) will be created and become visible and active, moving to the top of the Activity stack
- Calling `finish` on the new Activity, or pressing the hardware back button, will close it and remove it from the stack

Implicit Intents and Late Runtime Binding

- An implicit Intent is a mechanism that lets anonymous application components service action requests
- You can ask the system to launch an Activity that can perform a given action without knowing which application, or Activity, will do so
- When constructing a new implicit Intent to use with `startActivity`, you nominate an action to perform and, optionally, supply the URI of the data to perform that action on
- You can also send additional data to the target Activity by adding extras to the Intent

Implicitly Starting an Activity Sample

```
if (somethingWeird && itDontLookGood) {  
    Intent intent = new Intent(Intent.ACTION_DIAL,  
        Uri.parse("tel:555-2368"));  
    startActivity(intent);  
}
```

- Android resolves this Intent and starts an Activity that provides the dial action on a telephone number -- in this case the dialer Activity
- In circumstances where multiple Activities are capable of performing a given action, the user is presented with a choice

Sub-Activity

- You can start an Activity as a sub-Activity that's inherently connected to its parent
- A sub-Activity triggers an event handler within its parent Activity when it closes
- Sub-Activities are perfect for situations in which one Activity is providing data input (such as a user's selecting an item from a list) for another
- Sub-Activities are really just Activities opened in a different way. As such they must be registered in the application-manifest
 - In fact any manifest-registered Activity can be opened as a sub-Activity

Starting an Activity for a Result

- Explicit

```
private static final int SHOW_SUBACTIVITY = 1;
```

```
Intent intent = new Intent(this, MyOtherActivity.class);  
startActivityForResult(intent, SHOW_SUBACTIVITY);
```

- Implicit

```
private static final int PICK_CONTACT_SUBACTIVITY = 2;  
Uri uri = Uri.parse("content://contacts/people");  
Intent intent = new Intent(Intent.ACTION_PICK, uri);  
startActivityForResult(intent,  
PICK_CONTACT_SUBACTIVITY);
```

Returning Results

- When your sub-Activity is ready to return, call `setResult` before `finish` to return a result to the calling Activity
- The `setResult` method takes two parameters: the result code and the result itself, represented as an Intent
- The result code is the 'result' of running the sub-Activity
 - Generally either `Activity.RESULT_OK` or `Activity.RESULT_CANCELED`
 - In some circumstances you'll want to use your own response codes to handle application specific choices; `setResult` supports any integer value
- The Intent returned as a result often includes a URI to a piece of content (such as the selected contact, phone number, or media file)

Returning Results Sample Code (1/2)

```
Button okButton = (Button) findViewById(R.id.ok_button);
okButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        Uri data = Uri.parse("content://horses/"
+         select_horse_id);
        Intent result = new Intent(null,data);
        result.putExtra(IS_INPUT_CORRECT, inputCorrect);
        result.putExtra(SELECTED_PISTOL, selectedPistol);
        setResult(RESULT_OK, result);
        finish();
    }
});
```

Returning Results Sample Code (2/2)

```
Button cancelButton = (Button) findViewById(R.id.  
cancel_button);  
cancelButton.setOnClickListener(new View.OnClickListener() {  
    setResult(RESULT_CANCELED, null);  
    finish();  
})  
});
```

IntentActionDemo

- After writing a single activity, there comes a need to transition to another activity to perform another task either with or without information from the first activity.
- Android platform allows transition by means of Intent Interface.
- In this example there are two activities - IntentActionDemo.java and IntentA.java that both extend the super class Activity
- Do not forget to declare any new activity in the AndroidManifest.xml with permission.

IntentActionDemo Output



IntentActionDemo Implementation

- **Simple intent example;**

Note that optional step 2 was not used in our demo.

Step 1: `Intent i = new Intent(context, NameOfClassToTransitionTo.class)`

Step 2:(Optional)Intents can take various forms that make it even carry data in key/name pairs ie i.

`putExtra("key1", "My first Info")`

`i.putExtra("key2", "My second Info")`

Step 3: `startActivity(i)`

- Please see the code at <http://marakana.com/forums/android/examples/65.html>

IntentActionDemo.java

```
public class IntentActionDemo extends Activity implements OnClickListener {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        ...  
        Button button = (Button) findViewById(R.id.intentButton);  
button.setOnClickListener(this);  
    }  
    @Override  
    public void onClick(View src) {  
Intent i = new Intent(this, IntentA.class);  
startActivity(i);  
    }  
}
```

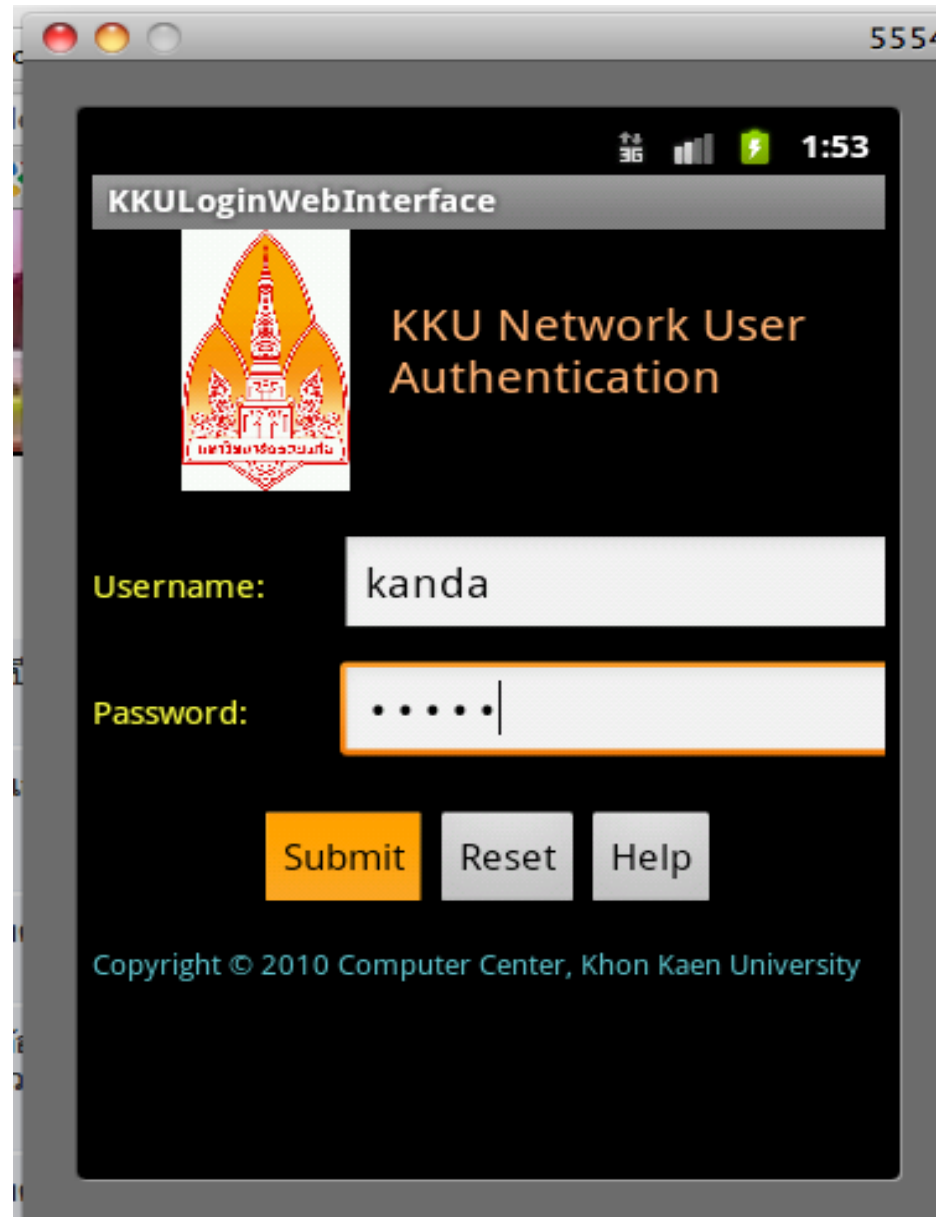
IntentA.java

```
public class IntentA extends Activity implements
OnClickListener{
@Override
public void onClick(View src) {
Intent i = new Intent(this, IntentActionDemo.class);
startActivity(i);
}
public void onCreate(Bundle savedInstanceState) {
....
Button button = (Button) findViewById(R.id.ButtonIntentA);
button.setOnClickListener(this);
}
}
```

AndroidManifest.xml


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.
com/apk/res/android"
package="com.marakana.com"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".IntentActionDemo"
android:label="@string/app_name">
...
</activity>
<activity android:name="IntentA"></activity>
</application>
<uses-sdk android:minSdkVersion="9" />
</manifest>
```

KKULogin: Submit (1)



5554

KKULoginWebInterface

 KKU Network User Authentication

Username:

Password:

Copyright © 2010 Computer Center, Khon Kaen University

KKULogin:Go to Web Page (2)



KKULogin: Web Browser (3)



Introducing Linkify

- Linkify is a helper class that automatically creates hyperlinks within Text View (and Text View-derived) classes through RegEx pattern matching
- Text that matches a specified RegEx pattern will be converted into a clickable hyperlink that implicitly fires `startActivity (new Intent(Intent.ACTION_VIEW, uri))`, using the matched text as the target URI
- You can specify any string pattern you want to turn into links; for convenience; the Linkify class provides presets for common content types (like phone numbers and email/web addresses)

The Native Linkify Link Types

- The static `Linkify.addLinks` method accepts the `View` to linkify, and a bitmask of one or more of the default content types supported and supplied by the `Linkify` class:
 - `WEB_URLS`
 - `EMAIL_ADDRESSES`
 - `PHONE_NUMBERS`
 - `ALL`

Using Linkify Samples

- Using Linkify in code

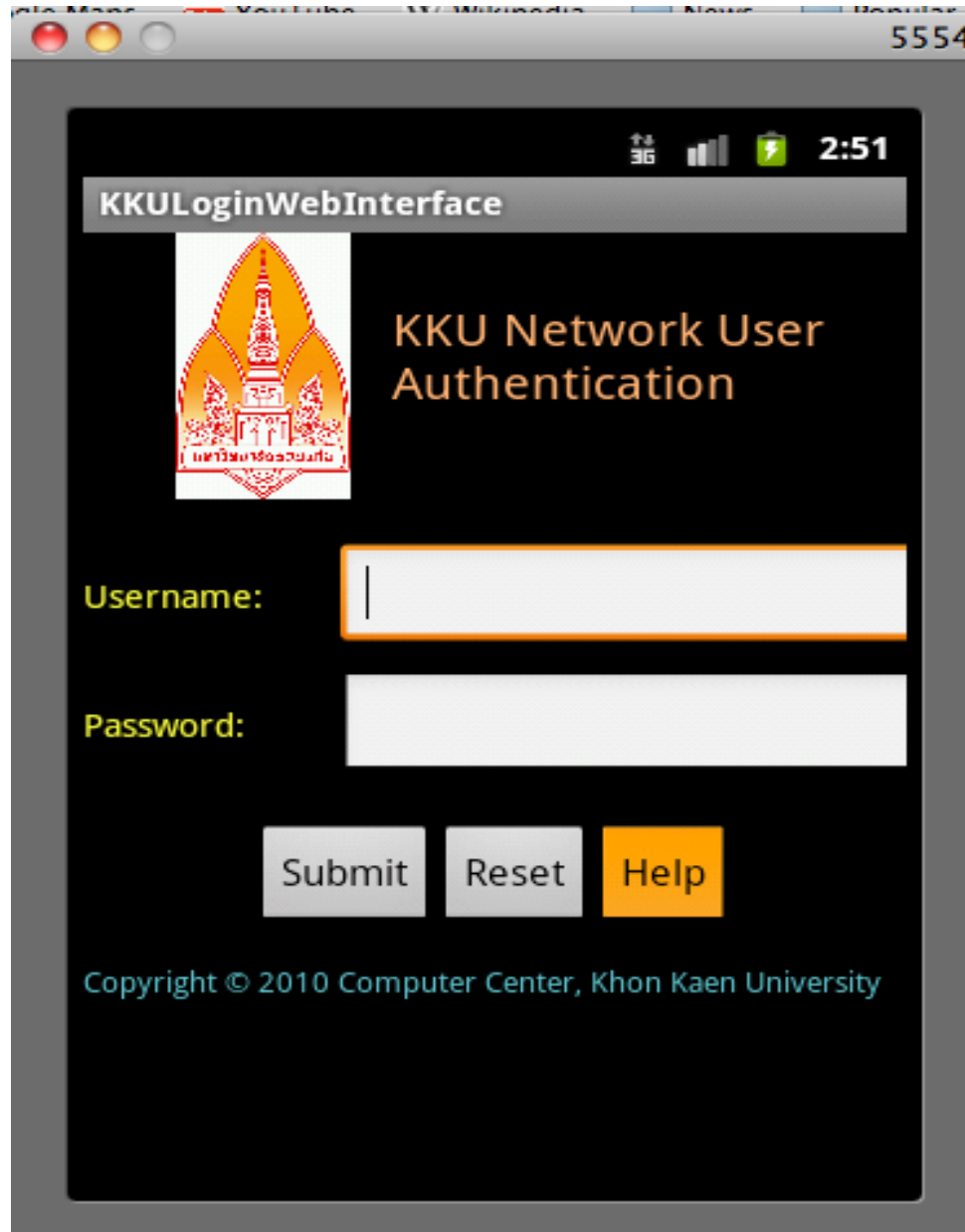
```
TextView textView = (TextView) findViewById(R.id.  
myTextView);
```

```
Linkify.addLinks(textView, Linkify.WEB_URL|Linkify.  
EMAIL_ADDRESSES);
```

- Using Linkify in XML


```
<TextView  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:text="@string/linkify_me"  
    android:autoLink="phone|email"  
>
```

KKU Login:Help (4)



The image shows a mobile application window titled "KKULoginWebInterface" with a status bar at the top displaying "3G", signal strength, battery, and the time "2:51". The interface has a black background and features the KKU logo on the left. The main heading is "KKU Network User Authentication". Below this, there are two input fields: "Username:" and "Password:". At the bottom, there are three buttons: "Submit", "Reset", and "Help". The "Help" button is highlighted in orange. At the very bottom, a copyright notice reads "Copyright © 2010 Computer Center, Khon Kaen University".

KKULoginWebInterface



KKU Network User Authentication

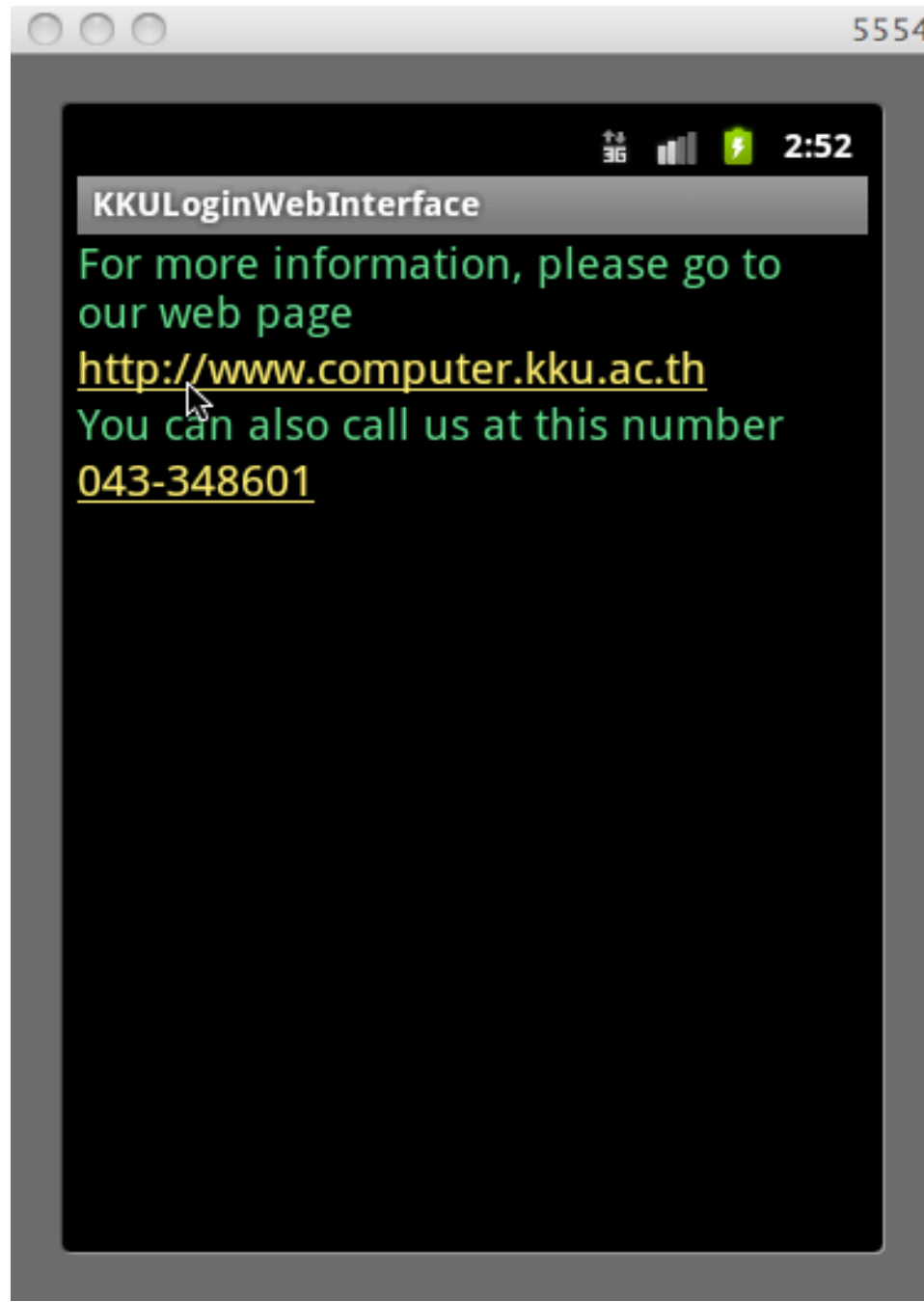
Username:

Password:

Submit Reset Help

Copyright © 2010 Computer Center, Khon Kaen University

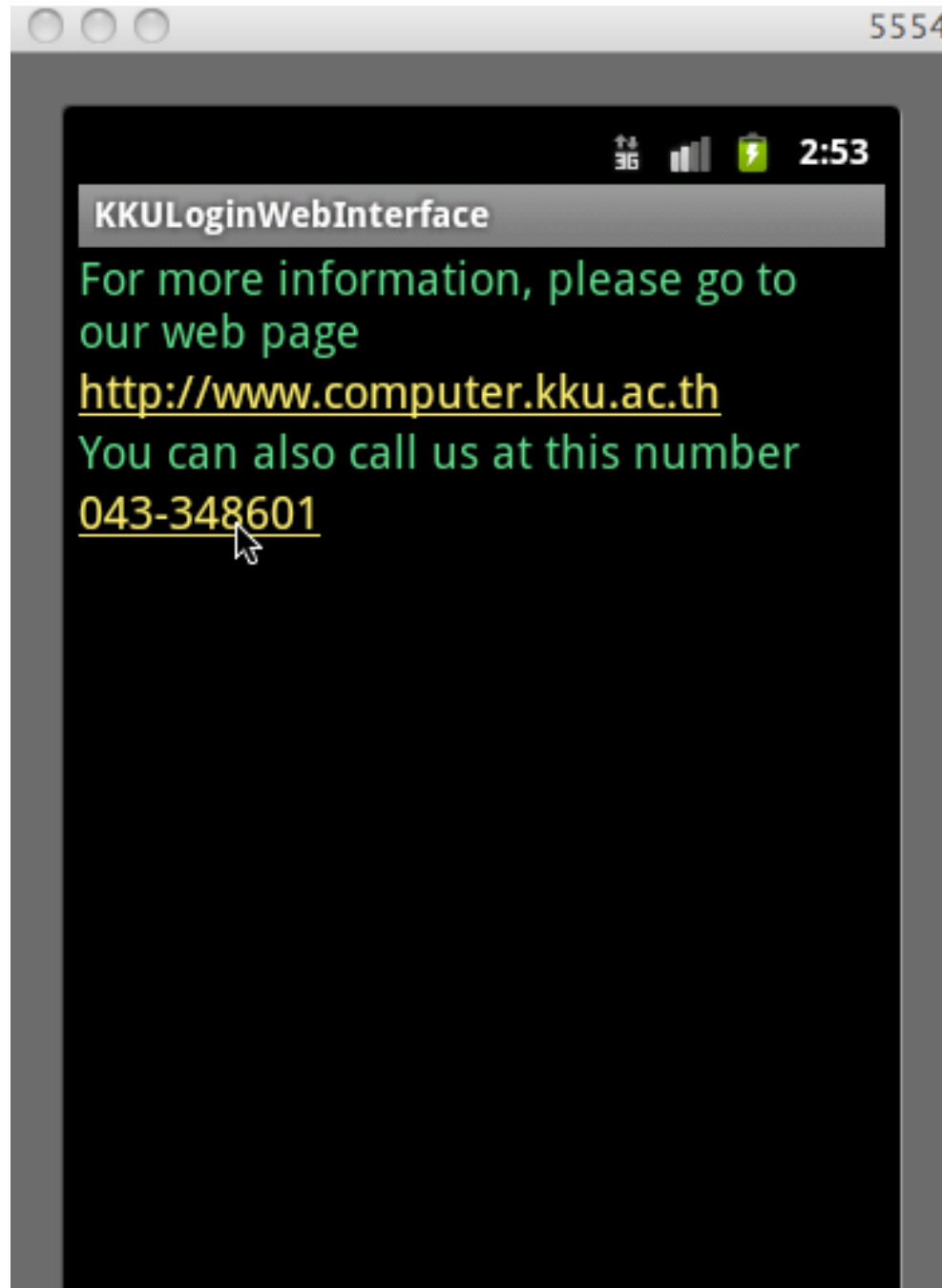
KKU Login: Click Web Link (5)



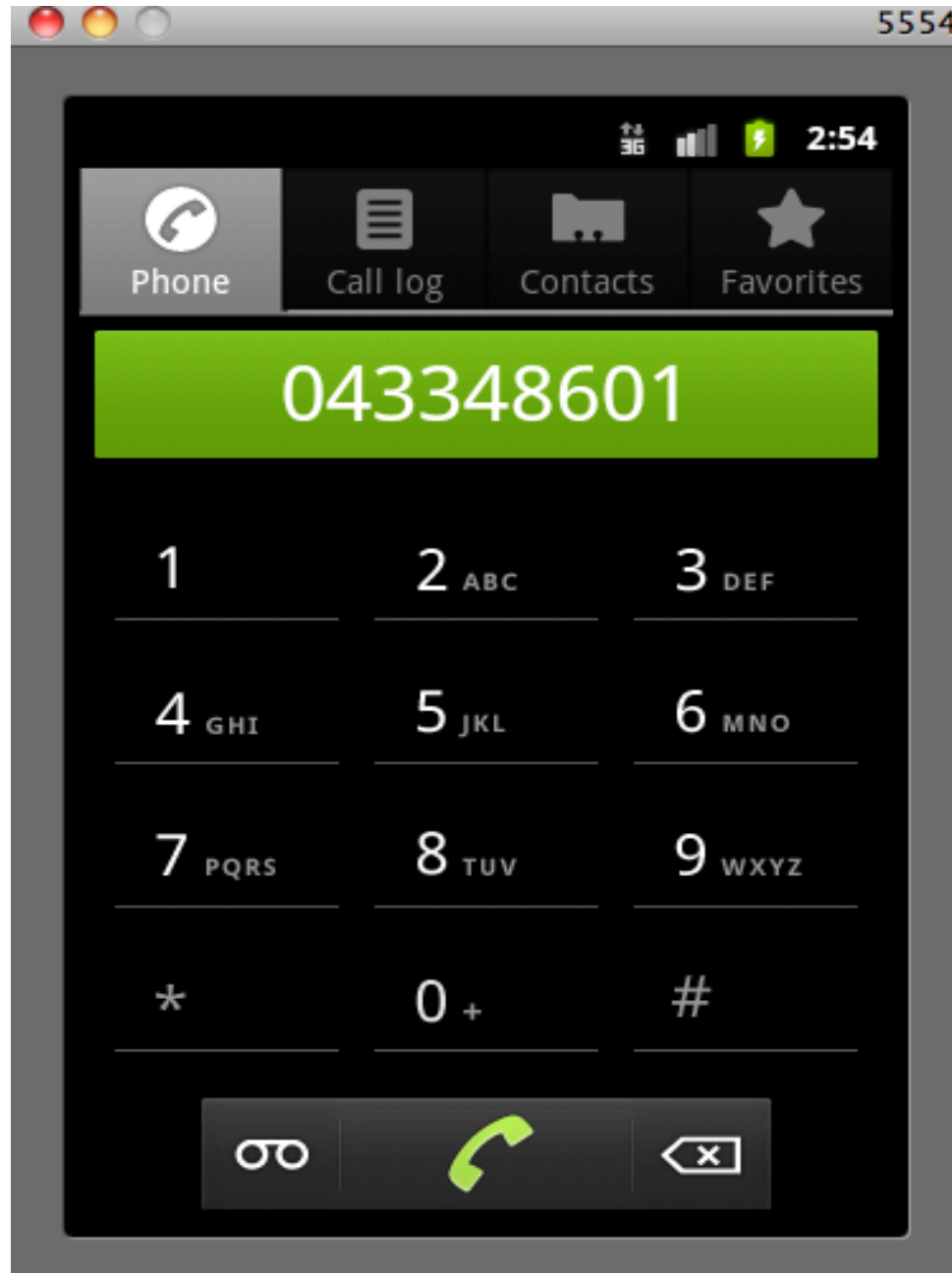
KKU Login: Web Link Open (6)



KKU Login: Click Phone Link (7)



KKU Login: Phone to be Dialed (8)



Intents: Starting a New Activity

- Dial a number

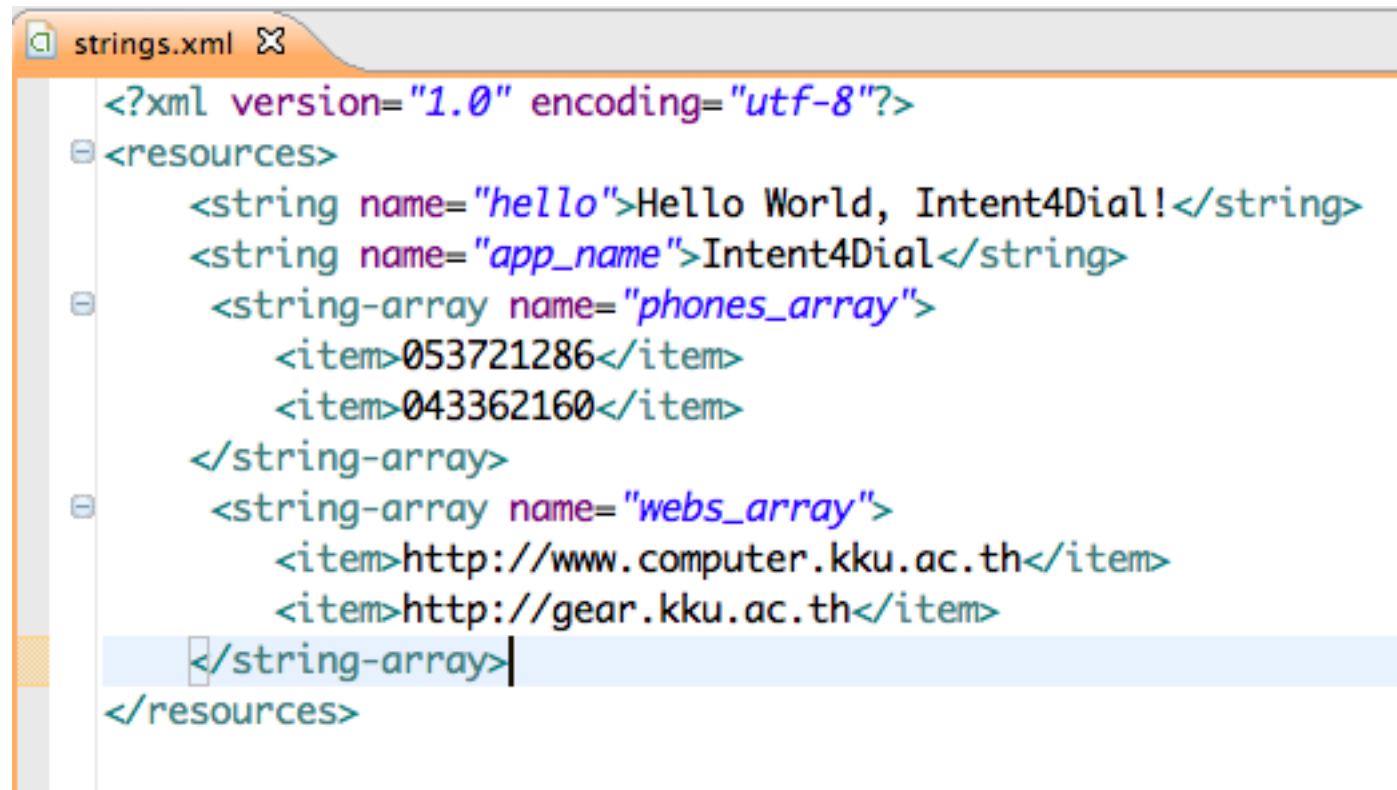
```
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:
043362160"));
startActivity(intent);
```

- Launch a website

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http:
//gear.kku.ac.th"));
startActivity(intent);
```

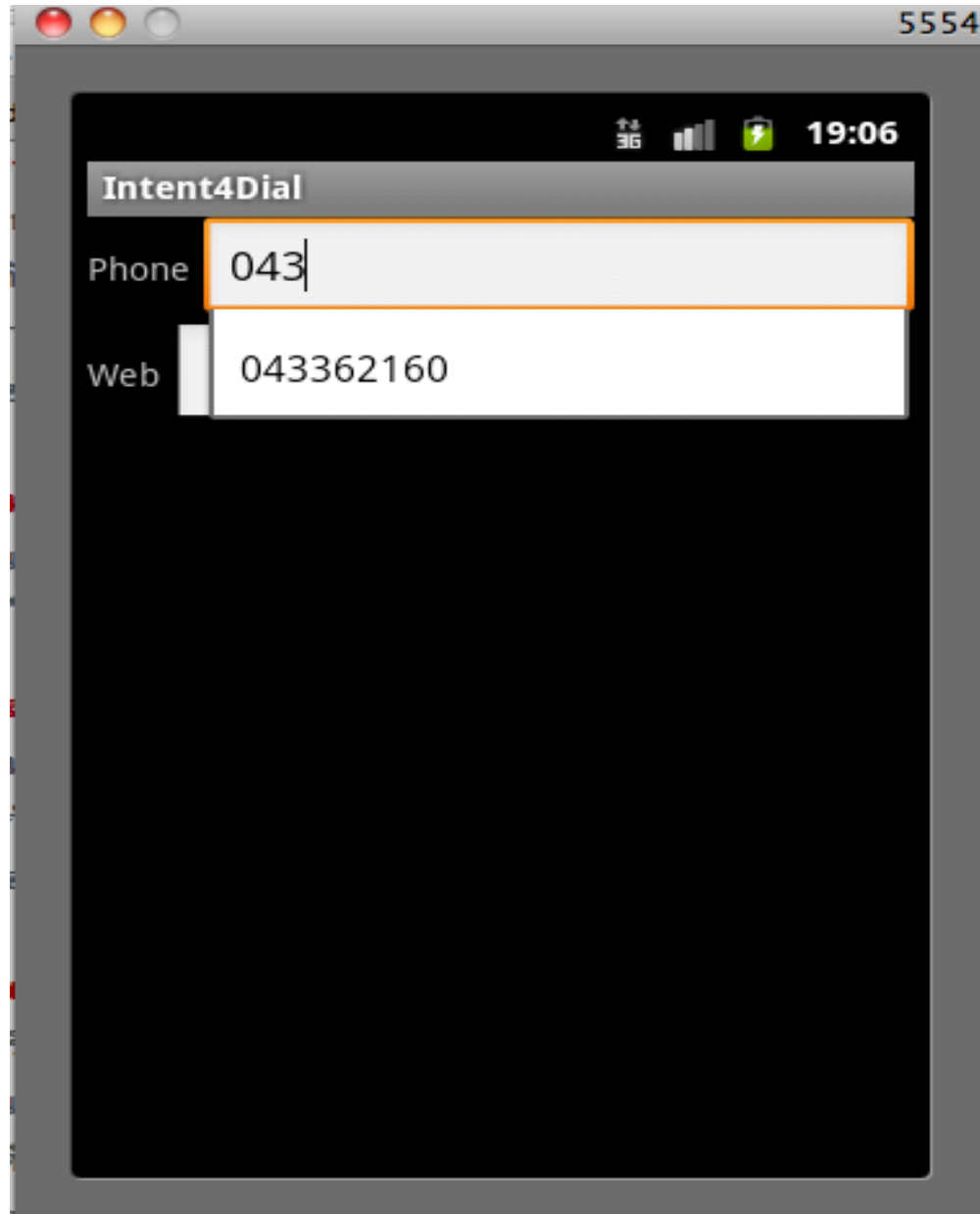

Defining the Arrays of Values

- The values that appear at AutoComplete components are defined in strings.xml

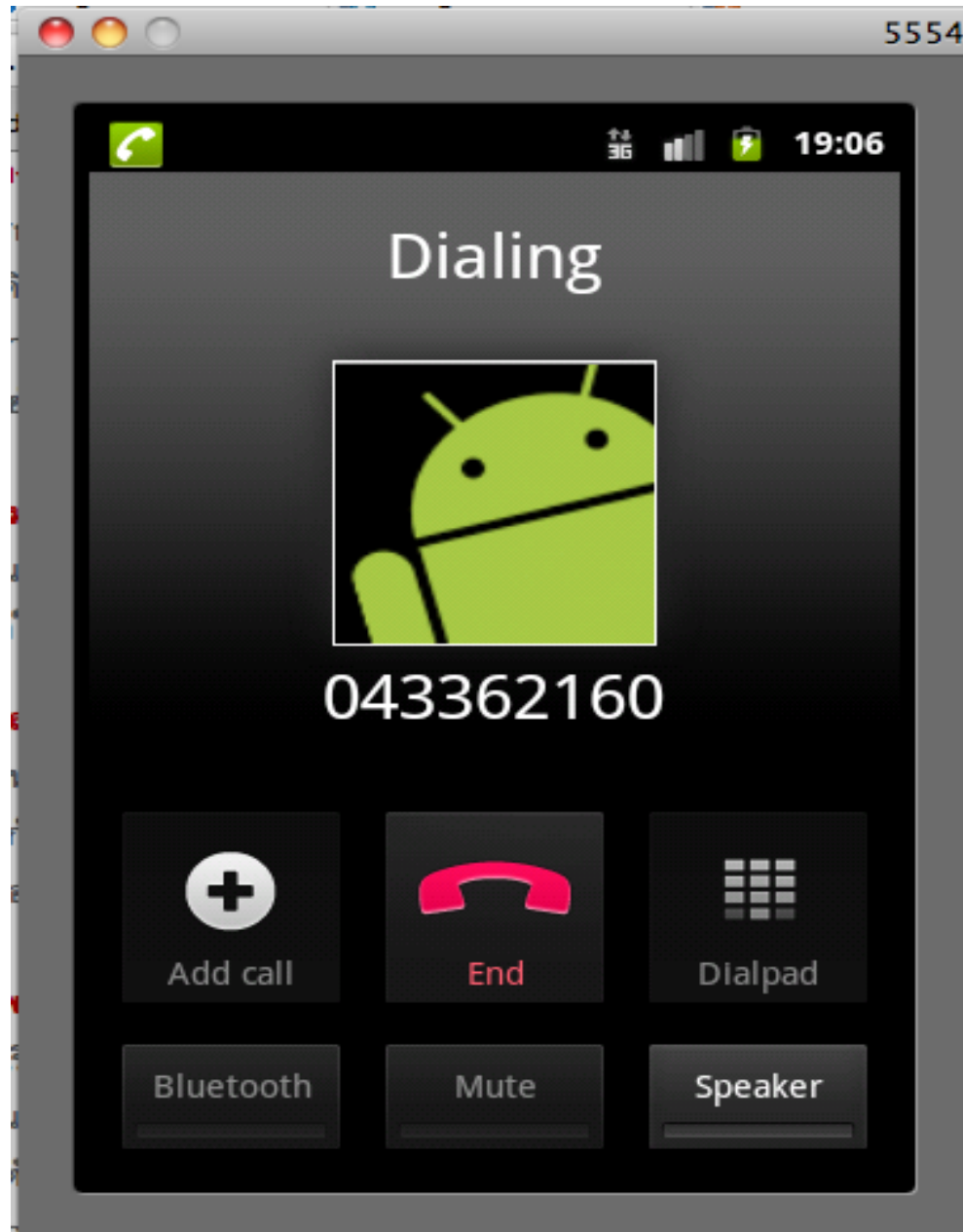
A screenshot of an IDE window titled 'strings.xml'. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello World, Intent4Dial!</string>
  <string name="app_name">Intent4Dial</string>
  <string-array name="phones_array">
    <item>053721286</item>
    <item>043362160</item>
  </string-array>
  <string-array name="webs_array">
    <item>http://www.computer.kku.ac.th</item>
    <item>http://gear.kku.ac.th</item>
  </string-array>
</resources>
```

Intents Starting Activity (1)



Intents Starting Activity (2)



Intents Starting Activity (3)



Intents Starting Activity (4)

5554

36 19:07

http://gear.kku.ac.th/

Department of Computer Engineering
Faculty of Engineering, Khon Kaen University

ติดต่อ
ภาควิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยขอนแก่น
ถนนพหลโยธิน
เขตเมือง
ขอนแก่น 40002

ติดต่อ
ภาควิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยขอนแก่น
อาคาร 105 ชั้น 2
พหลโยธิน 40002
KhonKaen
Thailand
Video

เปิดพิธีการ

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะ 2535 ได้ถูกคัดเลือกเข้าร่วมพิธีการ Thailand's Network Security Center 2019-12-08 11:19:58

พิธีการ Thailand's Network Security Center 2019-12-08 11:19:58

1. ทีม Security Center
2. ทีม Speed & Accuracy
3. ทีม Patch
4. ทีม TESA

#สมาคมดิจิทัลประเทศไทย
สมาคมวิศวกรรมคอมพิวเตอร์แห่งประเทศไทย

TESA Top Gun Rally 2019

Displaying Google Maps

- By default, the Google Maps displays the map of the United States when it is first loaded.
- However, you can also set the Google Maps to display a particular location. In this case, you can use the `animateTo()` method of the `MapController` class

```
mapView = (MapView) findViewById(R.id.mapView); mc = mapView.  
getController(); double lat = Double.parseDouble("16.466"); double lng = Double.  
parseDouble("102.478"); p = new GeoPoint((int) (lat * 1E6), (int) (lng * 1E6)); mc.  
animateTo(p); mc.setZoom(17); mapView.invalidate();
```

Adding a Marker (1/2)

To add a marker to the map, you first need to define a class that extends the Overlay class:

```
class MapOverlay extends com.google.  
android.maps.Overlay { @Override public boolean draw(Canvas canvas,  
MapView mapView, boolean shadow, long when) { super.draw(canvas,  
mapView, shadow); //---translate the GeoPoint to screen pixels--- Point  
screenPts = new Point(); mapView.getProjection().toPixels(p,  
screenPts); //---add the marker--- Bitmap bmp = BitmapFactory.  
decodeResource( getResources(), R.drawable.pushpin); canvas.  
drawBitmap(bmp, screenPts.x, screenPts.y-50, null); return true; } }
```

Adding a Marker (2/2)

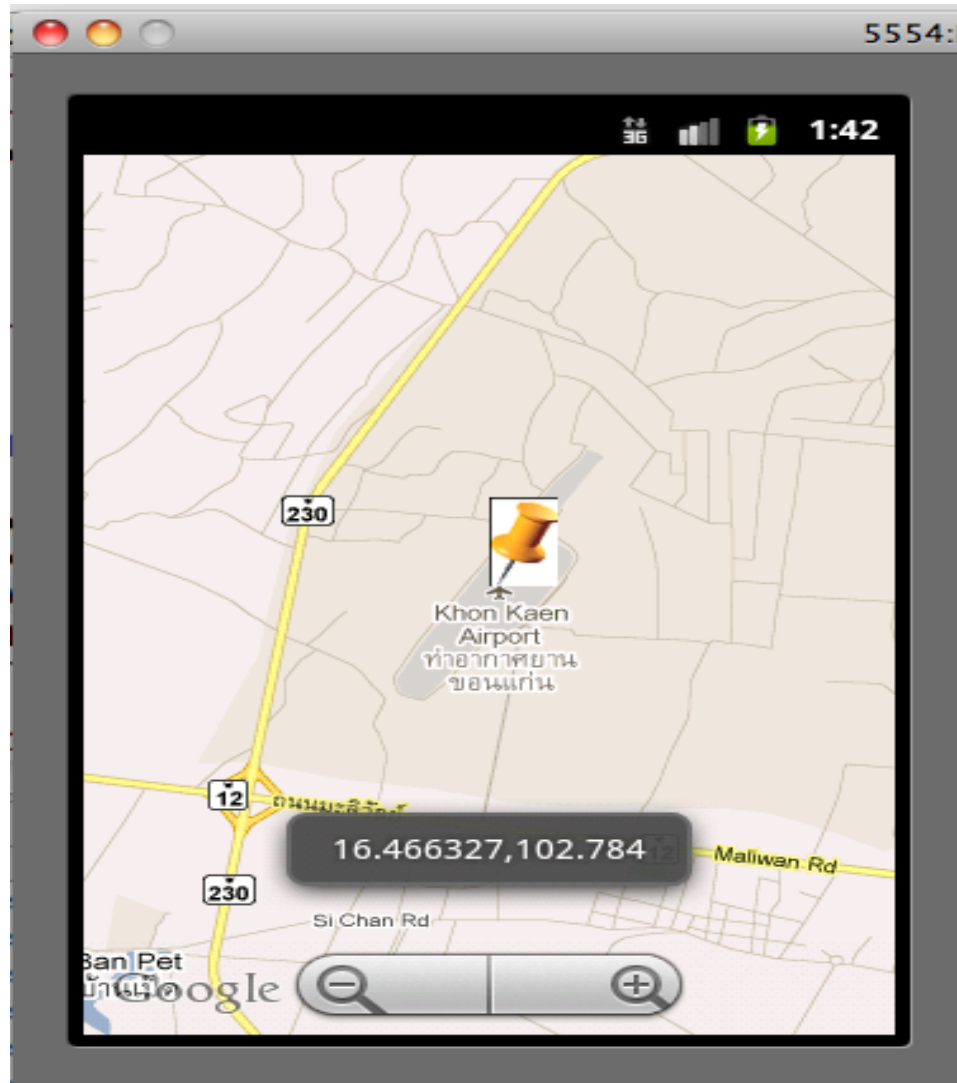
- Create an instance of the `MapOverlay` class and add it to the list of overlays available on the `MapView` object:
- `@Override public void onCreate(Bundle savedInstanceState) { //... mc.animateTo(p); mc.setZoom(17); ///---Add a location marker--- MapOverlay mapOverlay = new MapOverlay(); List<Overlay> listOfOverlays = mapView.getOverlays(); listOfOverlays.clear(); listOfOverlays.add(mapOverlay); mapView.invalidate(); }`

Getting the Location that was touched

```
class MapOverlay extends com.google.android.maps.Overlay ... {  
    @Override public boolean onTouchEvent(MotionEvent event,  
    MapView mapView) { //---when user lifts his finger--- if (event.getAction() == 1) { GeoPoint p = mapView.  
    getProjection().fromPixels( (int) event.getX(), (int) event.getY()); Toast.makeText(getApplicationContext(), p.  
    getLatitudeE6() / 1E6 + "," + p.getLongitudeE6() / 1E6 , Toast.LENGTH_SHORT).show(); } return false; } }
```

GoogleMaps with a Marker

We we press the mouse at the marker corner, the program will display the location



Getting Variables from One Screen to Another

- Sending information by using method putExtras

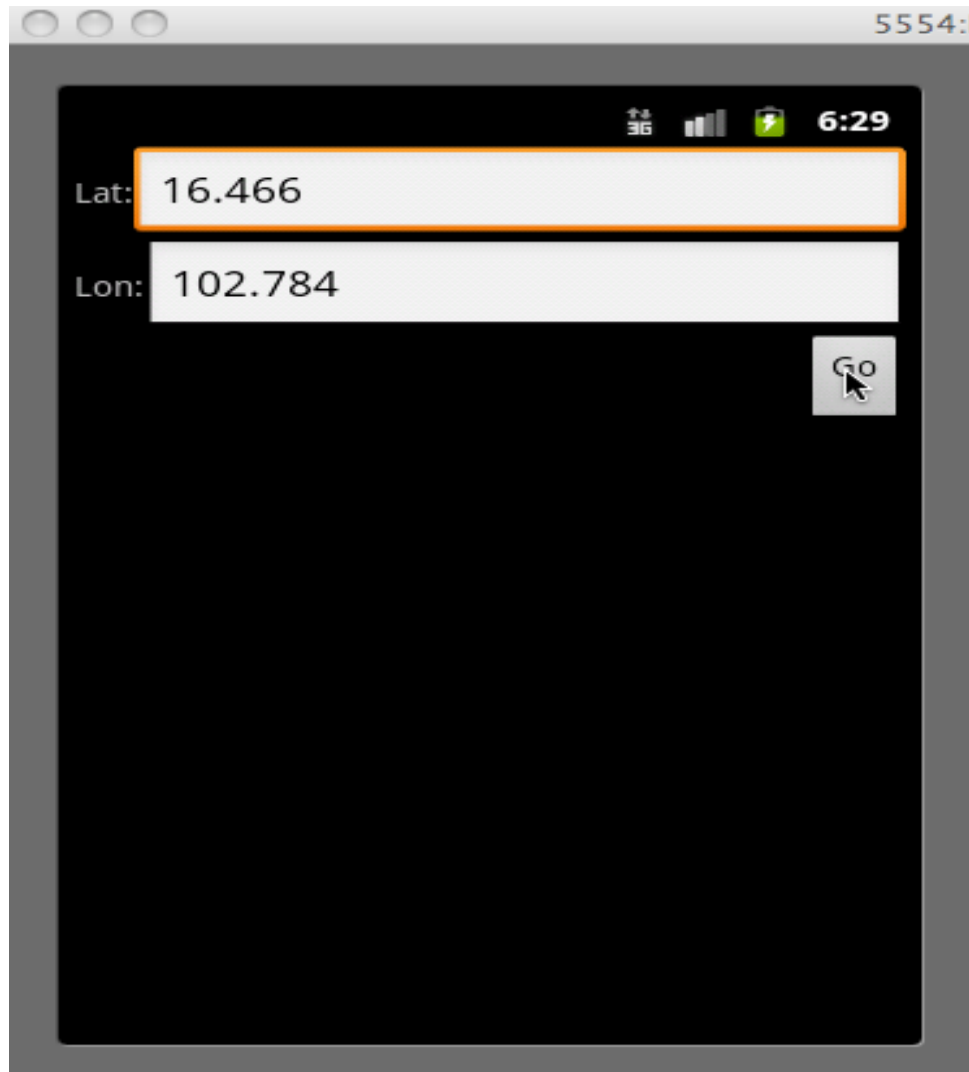
```
int array[] = {1,2,3};  
Intent i = new Intent(A.this, B.class);  
i.putExtras("numbers", array);  
i.putString("key", "value");  
startActivity(i);
```

- Getting information by using method getExtras

```
Bundle extras = getIntent().getExtras();  
int[] arrayB = extras.getInt("numbers");  
String value = extras.getString("key");  
// Alternatively, you can also do  
// String value = getIntent().getStringExtra("key");
```

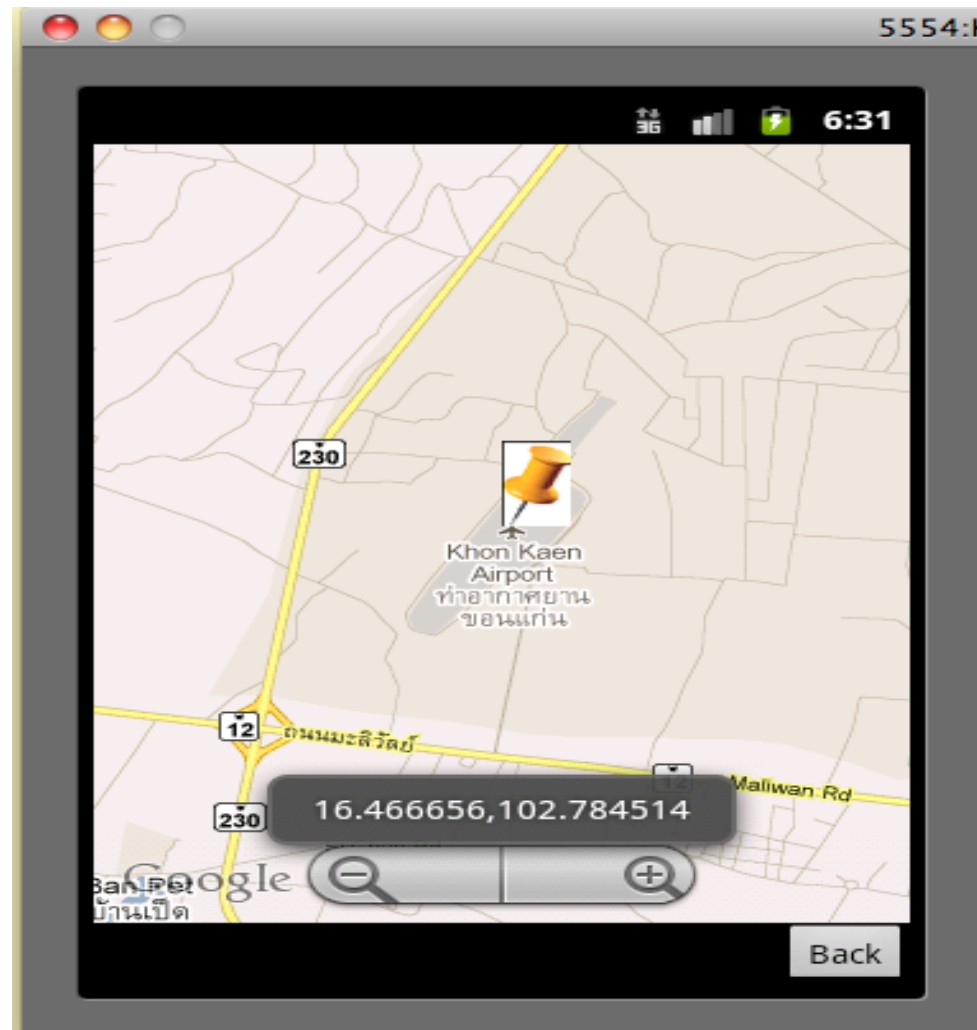
Google Maps with Intents (1/4)

- The program initializes values at EditText components
- The user just clicks button "Go"



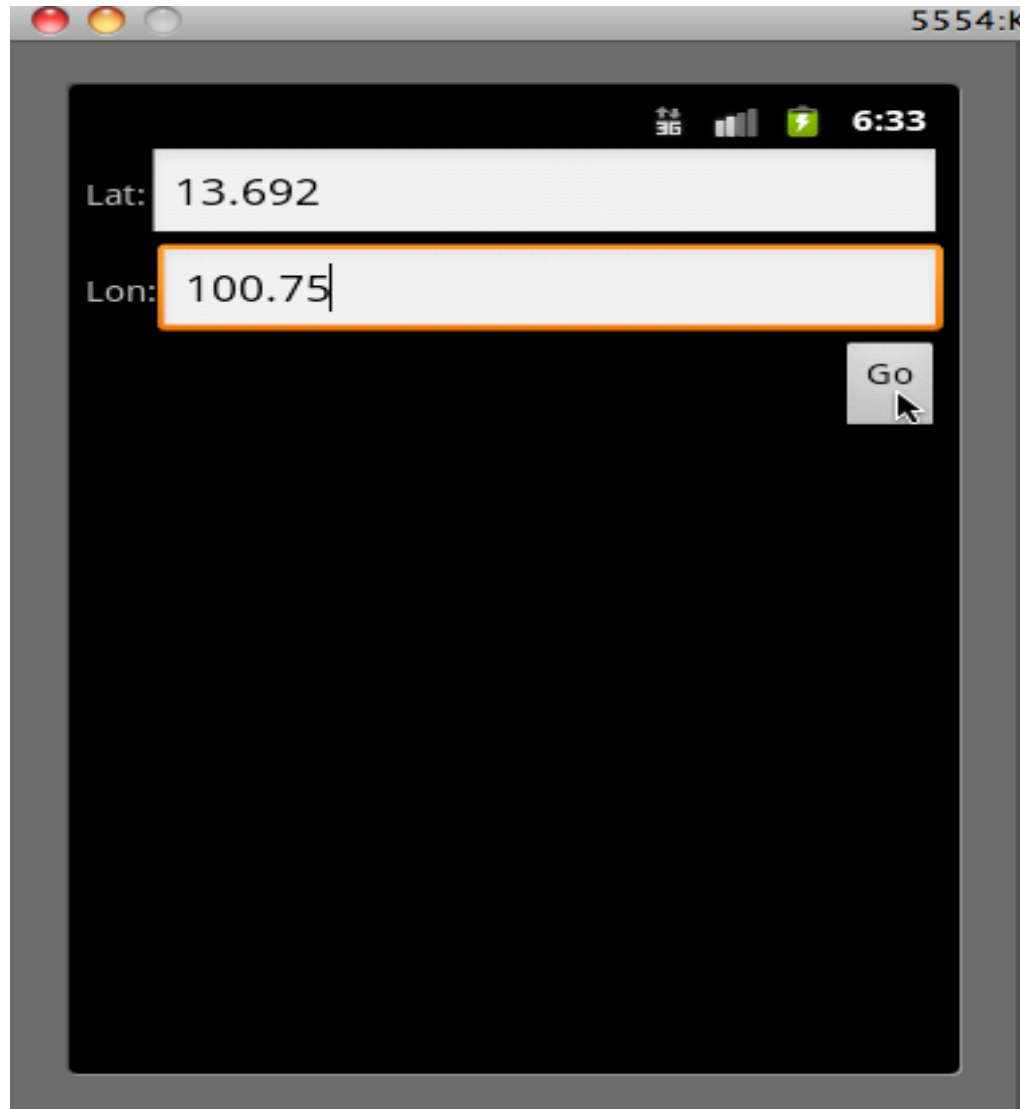
Google Maps with Intents (2/4)

- The program then goes to another screen with map view and the marker at the specified location
- The user then clicks button "Back"

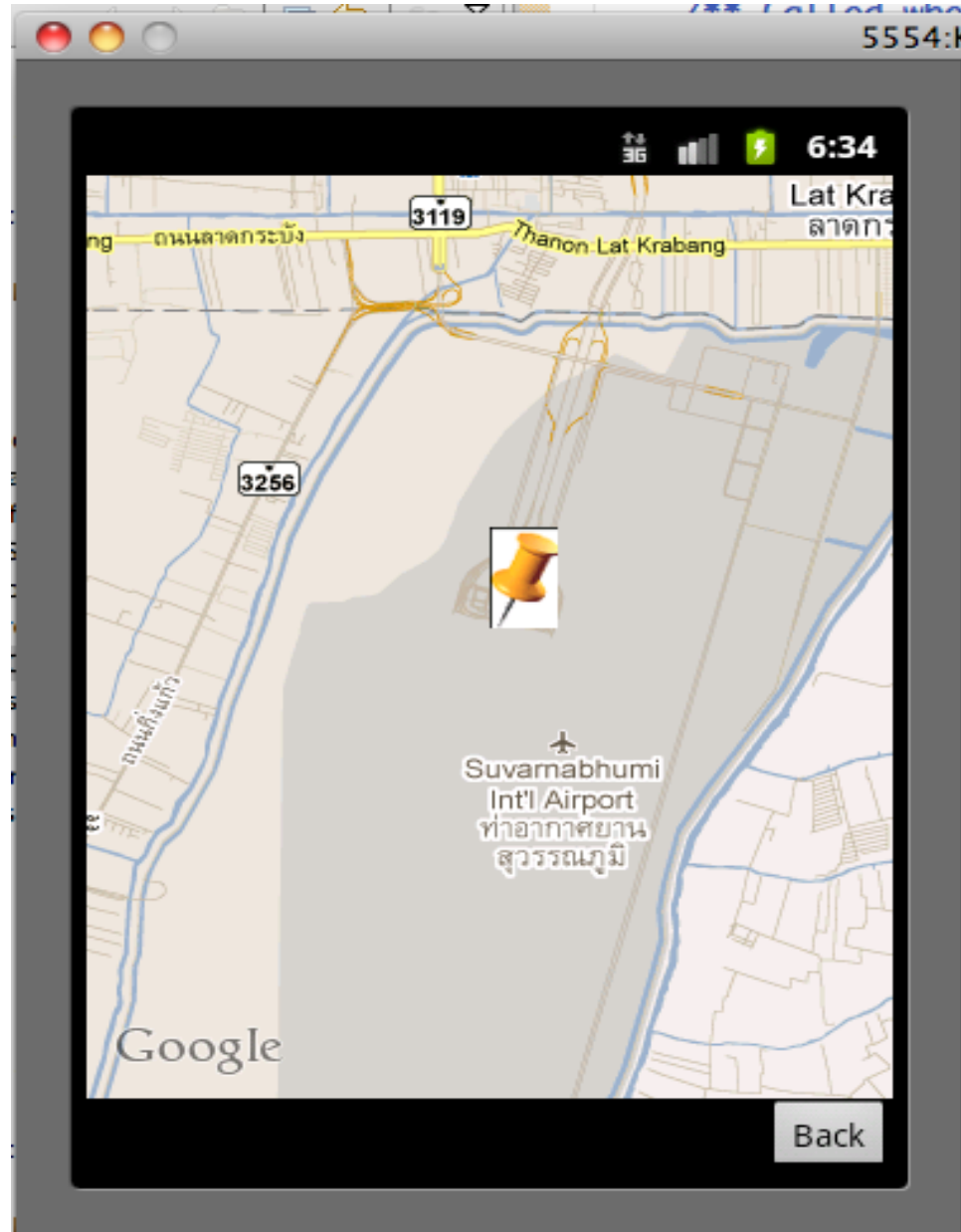


Google Maps with Intents (3/4)

- Now the user changes the latitude and the longitude and clicks button "Go"



Google Maps with Intents (4/4)



References

- <http://marakana.com/forums/android/examples/65.html>
- <http://www.slideshare.net/CodeAndroid/android-intent-intent-filter-broadcast-receivers>
- <http://mobiforge.com/developing/story/using-google-maps-android>
- <http://stackoverflow.com/questions/3848148/sending-information-with-intent-putextra>