# Android

## Using SMS

2010.05.14

***Database Laboratory***
***Hanyang Univ.***

# Let's Start

- Android offers **full access to SMS functionality**, letting you send and receive SMS messages from within application
- In this example you'll create an **SMS application** that turns an Android phone into an emergency response beacon
- You can set your phone to respond to your friends' and family members' pleas for a status update with a friendly message (or a desperate cry for help).
- The robustness of SMS network infrastructure makes SMS an excellent option for applications like this where reliability and accessibility are critical.

# Emergency Responder SMS Example

1. Start by creating a new *EmergencyResponder* project that features an *EmergencyResponder* Activity.

```
package com.paad.emergencyresponder;

import java.io.IOException;

public class EmergencyResponder extends Activity {

    @Override
    public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);
    }
```

# Emergency Responder SMS Example

## 2. Add permissions for finding your location as well as sending and receiving incoming SMS messages to the project manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.paad.emergencyresponder">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".EmergencyResponder" android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-permission android:name="android.permission.ACCESS_GPS"/>
  <uses-permission android:name="android.permission.ACCESS_LOCATION"/>
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
```

## 3. Modify the main.xml layout resource

- Include a List View to show the people requesting a status update and a series of buttons that users can press to send response SMS messages
- Use external resource references to fill in the button text; you'll create them in Step 4.

# Emergency Responder SMS Example

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView android:id="@+id/labelRequestList"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="These people want to know if you're ok"
    android:layout_alignParentTop="true"/>
  <LinearLayout android:id="@+id/buttonLayout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="5px"
    android:layout_alignParentBottom="true">
    <CheckBox android:id="@+id/checkboxSendLocation"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Include Location in Reply"/>
    <Button android:id="@+id/okButton"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/respondAllClearButtonText"/>
    <Button android:id="@+id/notOkButton"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/respondMaydayButtonText"/>
  </LinearLayout>
  <ListView android:id="@+id/myListView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_below="@id/labelRequestList"
    android:layout_above="@id/buttonLayout"/>
</RelativeLayout>
```

# Emergency Responder SMS Example

4. **Update the external strings.xml resource to include the text for each button and default response messages to use when responding, with "I'm safe" or "I'm in danger" messages**
   - You should also define the incoming message text to use when detecting requests for status responses.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Emergency Responder</string>

  <string name="respondAllClearButtonText">I am Safe and Well</string>
  <string name="respondMaydayButtonText">MAYDAY! MAYDAY! MAYDAY!</string>

  <string name="respondAllClearText">I am safe and well. Worry not!</string>
  <string name="respondMaydayText">Tell my mother I love her.</string>

  <string name="querystring">"are you safe?"</string>
</resources>
```

# Emergency Responder SMS Example

## 5. Create a new Array List of Strings within the *EmergencyResponder* Activity to store the phone numbers of the incoming requests for your status

- Bind the Array List to the List View, using an Array Adapter in the Activity's *onCreate* method
- Create a new *ReentrantLock* object to ensure thread safe handling of the Array List
- Take the opportunity to get a reference to the Check Box and to add Click Listeners for each of the response buttons. Each button should call the respond method

```
ReentrantLock lock;
CheckBox locationCheckBox;
ArrayList<String> requesters;
ArrayAdapter<String> aa;

@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);

  lock = new ReentrantLock();
  requesters = new ArrayList<String>();

  wireUpControls();
}
```

# Emergency Responder SMS Example

```
/** Hook up the ListView and Buttons */
private void wireUpControls() {
  locationCheckBox = (CheckBox)findViewById(R.id.checkboxSendLocation);
  ListView myListView = (ListView)findViewById(R.id.myListView);

  int layoutID = android.R.layout.simple_list_item_1;
  aa = new ArrayAdapter<String>(this, layoutID, requesters);

  myListView.setAdapter(aa);

  Button okButton = (Button)findViewById(R.id.okButton);
  okButton.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
      respond(true, locationCheckBox.isChecked());
    }
  });

  Button notOkButton = (Button)findViewById(R.id.notOkButton);
  notOkButton.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) {
      respond(false, locationCheckBox.isChecked());
    }
  });
}
```

# *Emergency Responder SMS Example*

## 6. Next, implement a Broadcast Receiver that will listen for incoming SMS messages

### 1) Start by creating a new static string variable to store the incoming SMS message intent action.

```
public static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
```

### 2) Then create a new Broadcast Receiver as a variable in the *EmergencyResponder* Activity

- The receiver should listen for incoming SMS messages and call the *requestRecieved* method when it sees SMS messages containing the “are you safe” String you *defined* as an external resource in Step 4.

# Emergency Responder SMS Example

```
BroadcastReceiver emergencyResponseRequestReceiver = new BroadcastReceiver() {
@Override
  public void onReceive(Context _context, Intent _intent) {
    if (_intent.getAction().equals(SMS_RECEIVED)) {
      String queryString = getString(R.string.querystring);

      Bundle bundle = _intent.getExtras();
      if (bundle != null) {
        Object[] pdus = (Object[]) bundle.get("pdus");
        SmsMessage[] messages = new SmsMessage[pdus.length];
        for (int i = 0; i < pdus.length; i++)
          messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);

        for (SmsMessage message : messages) {
          if (message.getMessageBody().toLowerCase().contains(queryString)) {
            requestReceived(message.getOriginatingAddress());
          }
        }
      }
    }
  }
};
```

# Emergency Responder SMS Example

7. Update the *onCreate* method of the Emergency Responder Activity to register the Broadcast Receiver created in Step 6

```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);

  lock = new ReentrantLock();
  requesters = new ArrayList<String>();

  wireUpControls();

  IntentFilter filter = new IntentFilter(SMS_RECEIVED);
  registerReceiver(emergencyResponseRequestReceiver, filter);
}
```

# Emergency Responder SMS Example

9. Update the *requestReceived* method stub so that it adds the originating number of each status request's SMS to the "requesters" Array List

```
    public void requestReceived(String _from) {
      if (!requesters.contains(_from)) {
            lock.lock();
            requesters.add(_from);
            aa.notifyDataSetChanged();
            lock.unlock();
          }
    }
}
```

# Emergency Responder SMS Example

## 10. Now update the Activity to let users respond to these status requests

- Start by completing the respond method stub you created in Step 5. It should iterate over the Array List of status requesters and send a new SMS message to each
- The SMS message text should be based on the response strings you defined as resources in Step 4
- Fire the SMS using an overloaded respond method that you'll complete in the next step

```
/** Respond to all requests for status */
  @SuppressWarnings("unchecked")
public void respond(boolean _ok, boolean _includeLocation) {
    String okString = getString(R.string.respondAllClearText);
    String notOkString = getString(R.string.respondMaydayText);

    String outString = _ok ? okString : notOkString;

    ArrayList<String> requestersCopy = (ArrayList<String>)requesters.clone();

    for (String to : requestersCopy)
      respond(to, outString, _includeLocation);
  }
```

# Emergency Responder SMS Example

## 11.Update the respond method that handles the sending of each response SMS

- Start by removing each potential recipient from the “requesters” Array List before sending the SMS
- If you are responding with your current location, use the Location Manager to find it before sending a second SMS with your current position as raw longitude/latitude points and a geocoded address

# Emergency Responder SMS Example

```
private void respond(String _to, String _response, boolean _includeLocation) {
  // Remove the target from the list of people we need to respond to.
  lock.lock();
  requesters.remove(_to);
  aa.notifyDataSetChanged();
  lock.unlock();

  SmsManager sms = SmsManager.getDefault();

  // Send the message
  sms.sendTextMessage(_to, null, _response, null, null);

  StringBuilder sb = new StringBuilder();

  // Find the current location and send it as SMS messages if required.
  if (_includeLocation) {
    String ls = Context.LOCATION_SERVICE;
    LocationManager lm = (LocationManager)getSystemService(ls);
    Location l = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);

    sb.append("I'm @:\n");
    sb.append(l.toString() + "\n");
```

# Emergency Responder SMS Example

```
    List<Address> addresses;
    Geocoder g = new Geocoder(getApplicationContext(), Locale.getDefault());
    try {
      addresses = g.getFromLocation(l.getLatitude(), l.getLongitude(), 1);
      if (addresses != null) {
        Address currentAddress = addresses.get(0);
        if (currentAddress.getMaxAddressLineIndex() > 0) {
          for (int i = 0; i < currentAddress.getMaxAddressLineIndex(); i++) {
            sb.append(currentAddress.getAddressLine(i));
            sb.append("\n");
          }
        }
        else {
          if (currentAddress.getPostalCode() != null)
            sb.append(currentAddress.getPostalCode());
        }
      }
    } catch (IOException e) {}

    ArrayList<String> locationMsgs = sms.divideMessage(sb.toString());
    for (String locationMsg : locationMsgs)
      sms.sendTextMessage(_to, null, locationMsg, null, null);
  }
}
```

# *Emergency Responder SMS Example*

## 12. In emergencies, it's important that messages get through

- Improve the robustness of the application by including auto-retry functionality
- Monitor the success of your SMS transmissions so that you can re-broadcast a message if it doesn't successfully send

## 1) Start by creating a new public static String in the Emergency Responder Activity to be used as a local "SMS Sent" action.

```
public static final String SENT_SMS = "com.paad.emergencyresponder.ER_SMS_SENT";
```

# Emergency Responder SMS Example

2) Update the *respond* method to include a new *PendingIntent* that broadcasts the action created in the previous step when the SMS transmission has completed

```
private void respond(String _to, String _response, boolean _includeLocation) {
  // Remove the target from the list of people we need to respond to.
  lock.lock();
  requesters.remove(_to);
  aa.notifyDataSetChanged();
  lock.unlock();

  SmsManager sms = SmsManager.getDefault();

  Intent intent = new Intent(SENT_SMS);
  intent.putExtra("recipient", _to);

  PendingIntent failed = PendingIntent.getBroadcast(getApplicationContext(), 0, intent, 0);

  // Send the message
  sms.sendTextMessage(_to, null, _response, failed, null);

  StringBuilder sb = new StringBuilder();

  // Find the current location and send it as SMS messages if required.
  if (_includeLocation) {
    String ls = Context.LOCATION_SERVICE;
    LocationManager lm = (LocationManager)getSystemService(ls);
    Location l = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

# Emergency Responder SMS Example

```
    sb.append("I'm @:\n");
    sb.append(l.toString() + "\n");

    List<Address> addresses;
    Geocoder g = new Geocoder(getApplicationContext(), Locale.getDefault());
    try {
      addresses = g.getFromLocation(l.getLatitude(), l.getLongitude(), 1);
      if (addresses != null) {
        Address currentAddress = addresses.get(0);
        if (currentAddress.getMaxAddressLineIndex() > 0) {
          for (int i = 0; i < currentAddress.getMaxAddressLineIndex(); i++) {
            sb.append(currentAddress.getAddressLine(i));
            sb.append("\n");
          }
        }
        else {
          if (currentAddress.getPostalCode() != null)
            sb.append(currentAddress.getPostalCode());
        }
      }
    } catch (IOException e) {}

    ArrayList<String> locationMsgs = sms.divideMessage(sb.toString());
    for (String locationMsg : locationMsgs)
      sms.sendTextMessage(_to, null, locationMsg, failed, null);
  }
}
```

# Emergency Responder SMS Example

## 3) Then implement a new Broadcast Receiver to listen for this broadcast Intent

- Override its *onReceive* handler to confirm that the SMS was successfully delivered; if it wasn't, then add the intended recipient back on to the requesters Array List

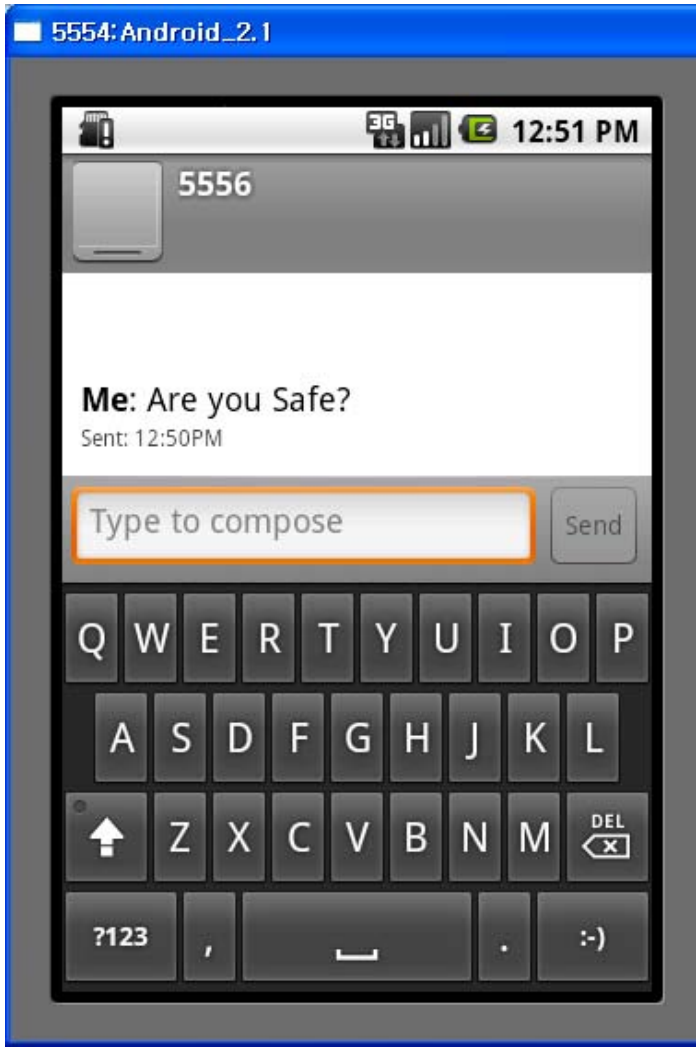
```
private BroadcastReceiver attemptedDeliveryReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context _context, Intent _intent) {
    if (_intent.getAction().equals(SENT_SMS)) {
      if (getResultCode() != Activity.RESULT_OK) {
        String recipient = _intent.getStringExtra("recipient");
        requestReceived(recipient);
      }
    }
  }
};
```

# Emergency Responder SMS Example

4. Finally, register the new Broadcast Receiver by extending the *onCreate* method of the Emergency Responder Activity

```
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.main);

  lock = new ReentrantLock();
  requesters = new ArrayList<String>();

  wireUpControls();

  IntentFilter filter = new IntentFilter(SMS_RECEIVED);
  registerReceiver(emergencyResponseRequestReceiver, filter);

  IntentFilter attemptedDeliveryfilter = new IntentFilter(SENT_SMS);
  registerReceiver(attemptedDeliveryReceiver, attemptedDeliveryfilter);
}
```

# Emergency Responder SMS Example




# More things

## Some areas where it could be improved

- The Broadcast Receiver created and registered in Step 6 and 7 would be better registered within the manifest to allow the application to respond to incoming SMS messages even when it isn't running
- The parsing of the incoming SMS messages performed by the Broadcast Receiver in Steps 6 and 8 should be moved into service, and executed on a background thread.
- Similarly, Step 12, sending the response SMS messages, would be better executed on a background thread within a Service