CS378 - Mobile Computing

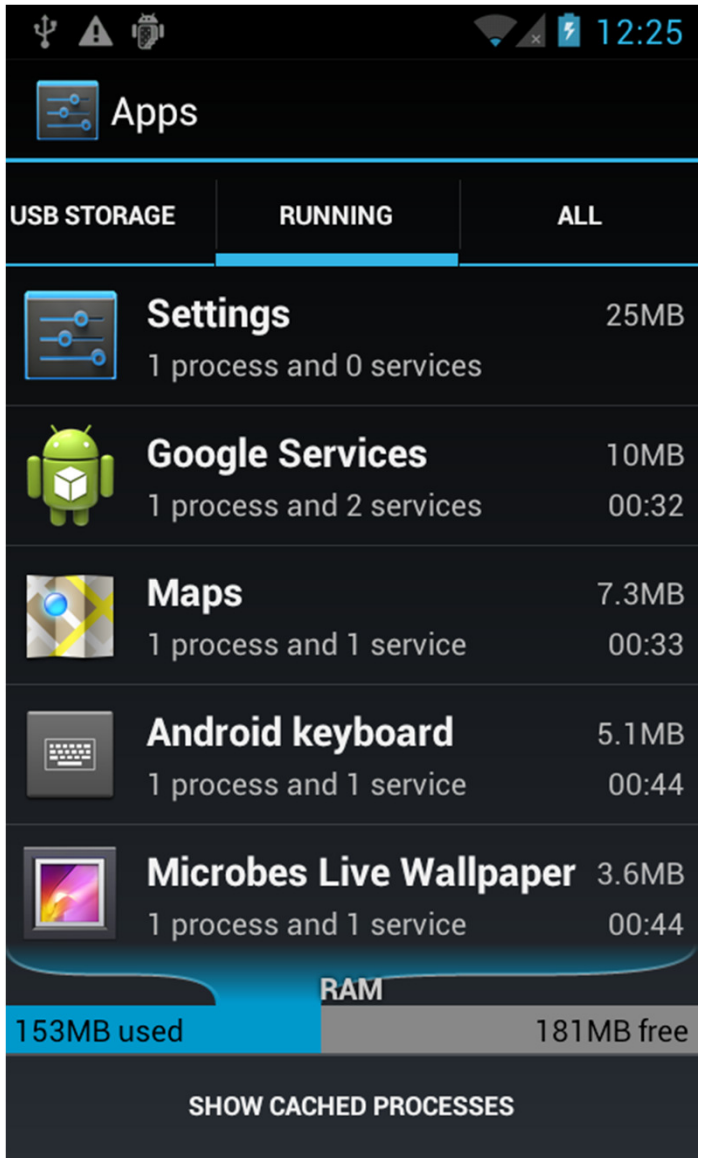
Services and Broadcast Receivers

Services

- One of the four primary application components:
  - activities
  - content providers
  - services
  - broadcast receivers

Services

- Application component that performs long-running operations in background with no UI
- application starts service and service continues to run even if original application ended or user moves to another application



Forms of Services

- Stated:
  - application component, such as an Activity, starts the service with the method call startService()
  - once started service can run in background indefinitely
  - generally services do not return a result (see bound service)
  - service should stop itself when done

Forms of Services

- Bound
  - application component binds itself to existing service via the bindService() method
  - bound service provides client-server interface that allows application component to interact with service
  - interact with service, send requests, get result via IPC (inter process communication
  - service runs as long as one or more applications bound to it
  - destroyed when no applications bound

Forms of Services

- Service can be started and later bound to other applications
- private service (manifest) cannot be bound by other applications

Service or Thread

- Past examples, kept UI thread responsive with other threads of execution, especially AsyncTask
- Should services be used for this?
- Service for actions that need to take place even if user not interacting with UI or has closed application
- Example, do complex rendering of image to display to user.
  - Not a job for a service

Creating a Service

- create subclass of Android Service class or one of its existing subclasses
- override callback methods that handle important aspects of service lifecycle
- most important of these are:
  - onStartCommand
  - startService
  - onBind
  - onCreate
  - onDestroy
  - stopSelf
  - stopService

public abstract class

Service

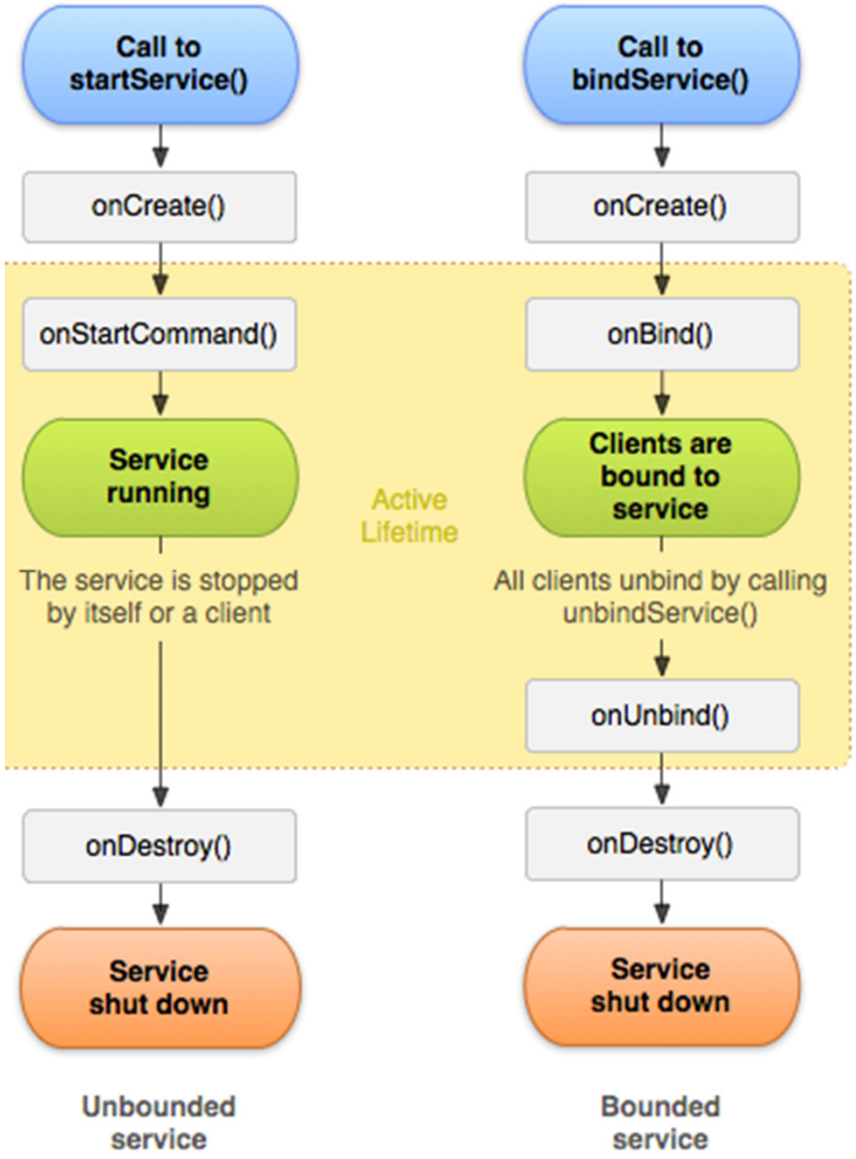
extends ContextWrapper
implements ComponentCallbacks2

java.lang.Object
↳ android.content.Context
↳ android.content.ContextWrapper
↳ android.app.Service

Service Lifecycle

- If component starts service with startService method (leads to call to onStartCommand) service runs until it calls stopSelf or another activity calls stopService
- if component calls bindService (onStartCommand no called) service runs as long as at least one component bound to it

Service Lifecycle

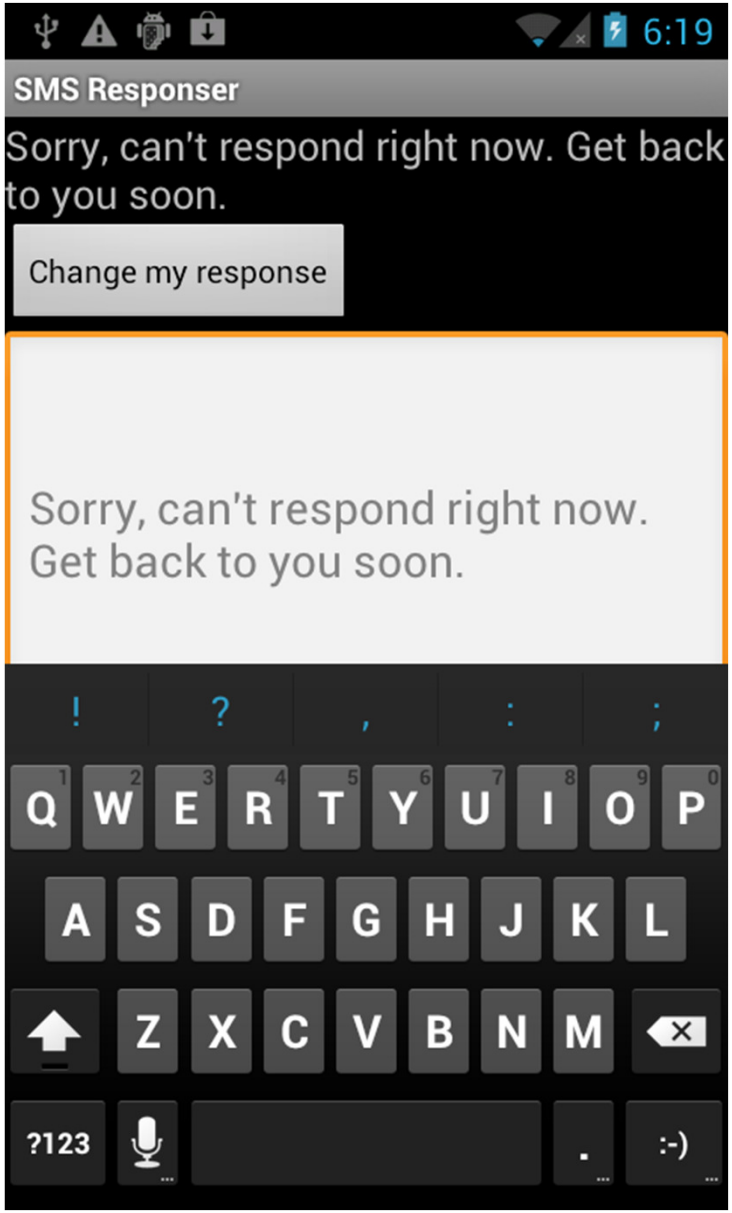


Service Example

- From Roger Wallace
  - wanted an app that would respond to texts (SMS) received when driving and respond with a message ("Driving - Get back to you soon.")
  - Initial version simply auto responds to all texts
  - how to change it so it responds only when driving?

Example Service Application

- From *The Android Developer's Cookbook*
- SMSResponder Application
- Response stored in shared preferences
- App simply allows changes to message



Using SMS

- Permission in manifest file to send and / or receive SMS messages

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
```

ResponseSMS Basic App

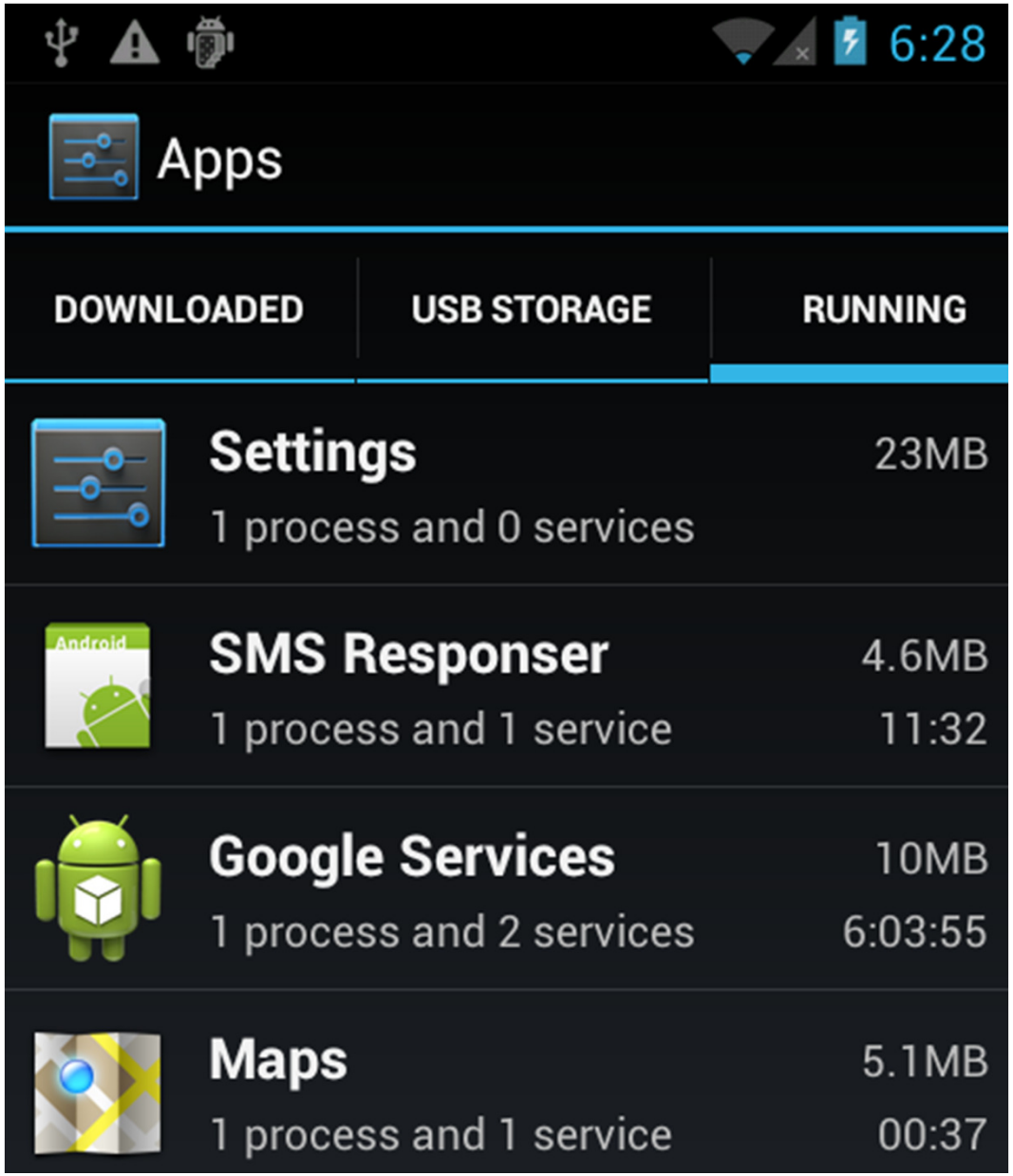
- All work done in onCreate method

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    myprefs
        = PreferenceManager.getDefaultSharedPreferences(this);
    tv1 = (TextView) this.findViewById(R.id.display);
    ed1 = (EditText) this.findViewById(R.id.editText);
    bt1 = (Button) this.findViewById(R.id.submit);
    reply = myprefs.getString("reply", "Thank you " +
            "for your message. " +
            "I am busy now. I will call you later");
    tv1.setText(reply);
    updater = myprefs.edit();
    ed1.setHint(reply);
```

ResponseSMS onCreate

```
bt1.setOnClickListener(new OnClickListener(){
    public void onClick(View view){
        updater.putString("reply", ed1.getText().toString());
        updater.commit();
        SMSResponser.this.finish();

    }
});

try {
    // start Service
      Intent svc = new Intent(this, ResponserService.class);
      startService(svc);
}
catch (Exception e) {
    Log.e("onCreate", "service creation problem", e);
}
```

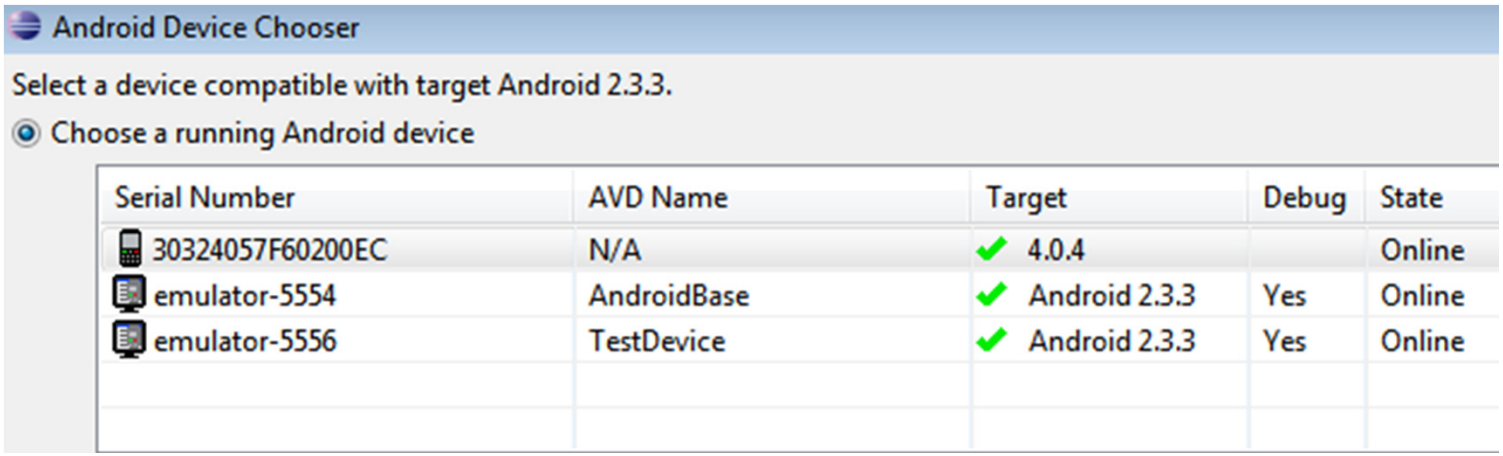
Service Running



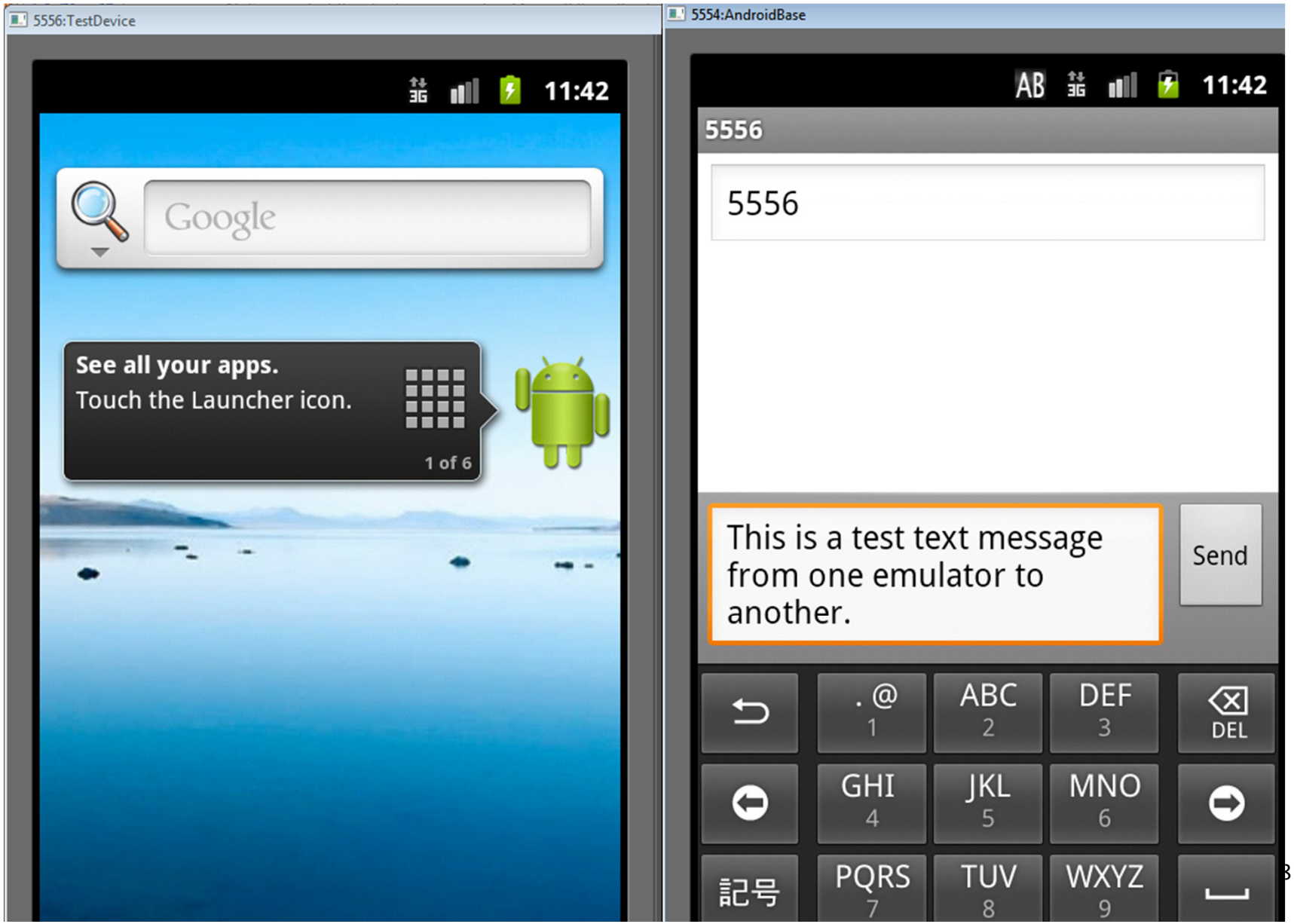
app still running, and service has started

Simulating Texts

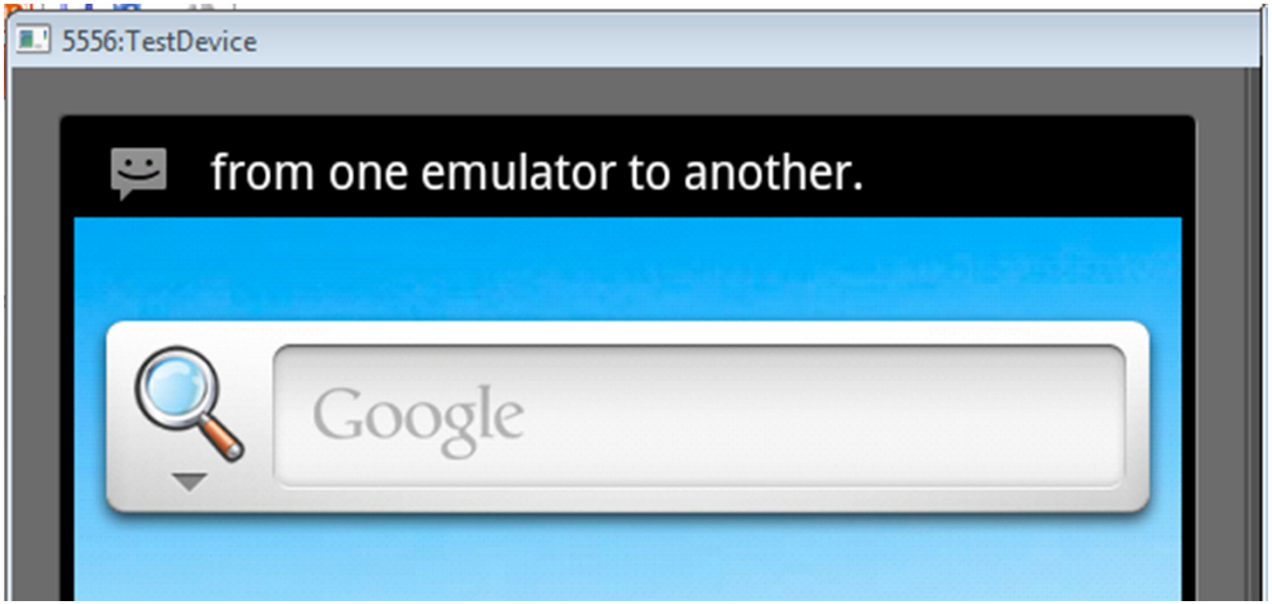
- Calls and texts can be simulated between emulators
- Start two emulators
- Use messaging app to send text
- Phone number is simply the emulator port number (visible at top of the emulator or in eclipse)

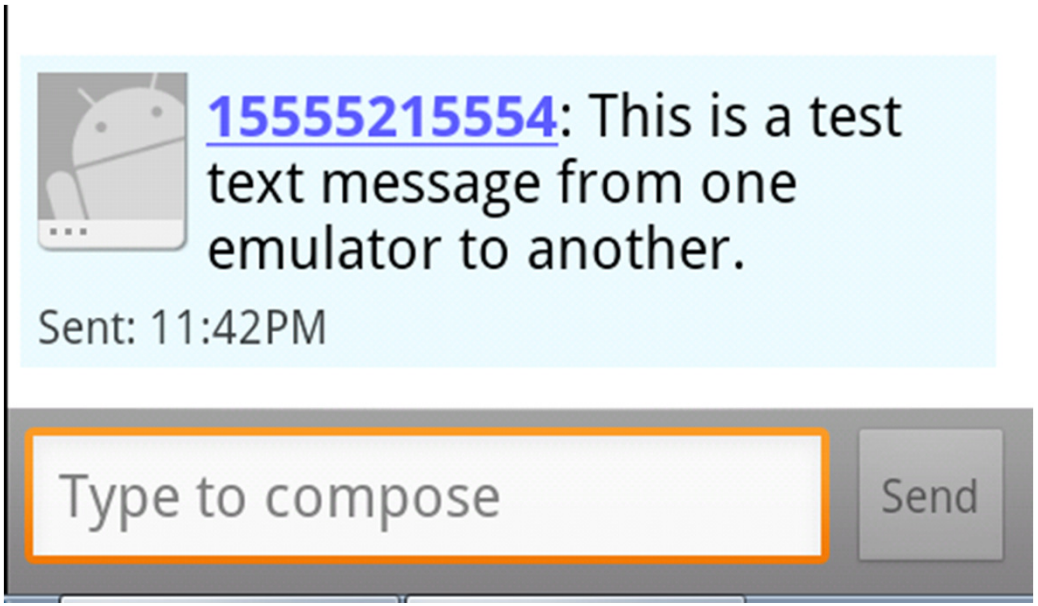


Dual Emulators

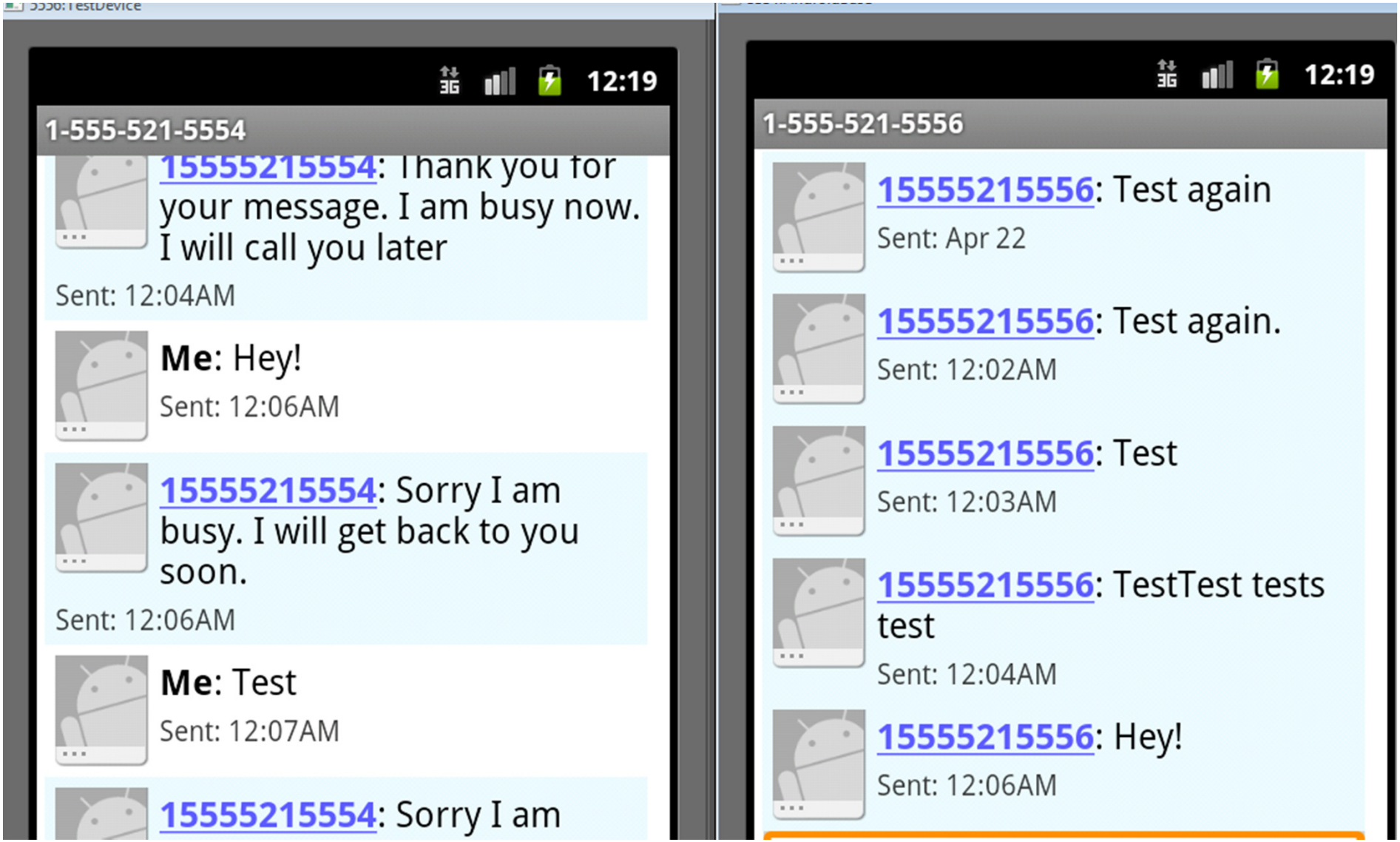


Emulator Texts





Testing Service



Creating a Service

- Extend the Service class
  - adapter class exists, IntentService that handles a lot of the details
- override onStartCommand
  - return an int describing what system should do for starting service
  - START_NOT_STICKY, if system kills service don't restart
  - START_STICKY, if system kills service then recreate, but does not redeliver intent
  - START_REDELIVER_INTENT, if system kills service then recreate and redeliver last intent

SMS Responder

```
//The Action fired by the Android-System when a SMS was received.
private static final String RECEIVED_ACTION
        = "android.provider.Telephony.SMS_RECEIVED";

private static final String SENT_ACTION = "SENT_SMS";
private static final String DELIVERED_ACTION = "DELIVERED_SMS";

private String requester;
private String reply;
private SharedPreferences myprefs;
```

SMS Responder - onCreate

```
@Override
public void onCreate() {
    super.onCreate();
    myprefs = PreferenceManager.getDefaultSharedPreferences(this);

    registerReceiver(sentReceiver, new IntentFilter(SENT_ACTION));
    registerReceiver(deliverReceiver, new IntentFilter(DELIVERED_ACTION));

    IntentFilter receiverfilter = new IntentFilter(RECEIVED_ACTION);
    registerReceiver(receiver, receiverfilter);

    IntentFilter sendfilter = new IntentFilter(SENT_ACTION);
    registerReceiver(sender,sendfilter);
}
```

Broadcast Receivers

- The fourth main application component
- "A *broadcast receiver* is a component that responds to system-wide broadcast announcements."
- Android system sends multiple kinds of broadcasts
  - screen turned off, battery low, picture captured, SMS received, SMS sent

Broadcast Receivers

- Applications can initiate broadcasts to inform other applications of status or readiness
- Don't display UI
  - may create status bar notifications
- Usually just a gateway to other components and does very minimal work
  - initiate service to perform based on some event
- Broadcasts are delivered as Intents

Broadcast Receivers

- receive intents sent by sendBroadcast() method
- LocalBroadcastManager to send Broadcasts within your application only
- In SMS responder register receivers
- unregister when service destroyed
- **key point: override the onReceive method for BroadcastReceiver subclass**

BroadcastReceivers

- What broadcasts are available?
- Check the Intent class
- http://developer.android.com/reference/android/content/Intent.html
  - search for "Broadcast Action"
- Also look in android-sdk\platforms\<number>\data\broadcast_actions.txt

Broadcasts

String	ACTION_CAMERA_BUTTON	Broadcast Action: The "Camera Button" was pressed.
String	ACTION_CHOOSER	Activity Action: Display an activity chooser, allowing the user to pick what they want to before proceeding.
String	ACTION_CLOSE_SYSTEM_DIALOGS	Broadcast Action: This is broadcast when a user action should request a temporary system dialog to dismiss.
String	ACTION_CONFIGURATION_CHANGED	Broadcast Action: The current device `Configuration` (orientation, locale, etc) has changed.
String	ACTION_CREATE_SHORTCUT	Activity Action: Creates a shortcut.
String	ACTION_DATE_CHANGED	Broadcast Action: The date has changed.
String	ACTION_DEFAULT	A synonym for `ACTION_VIEW`, the "standard" action that is performed on a piece of data.
String	ACTION_DELETE	Activity Action: Delete the given data from its container.
String	ACTION_DEVICE_STORAGE_LOW	Broadcast Action: A sticky broadcast that indicates low memory condition on the device This is a protected intent that can only be sent by the system.

Broadcasts

- from broadcast_actions.txt in sdk files
- platforms-> <api level>-> data\

```
android.intent.action.TIME_SET
android.intent.action.TIME_TICK
android.intent.action.UID_REMOVED
android.intent.action.USER_PRESENT
android.intent.action.WALLPAPER_CHANGED
android.media.ACTION_SCO_AUDIO_STATE_UPDATED
android.media.AUDIO_BECOMING_NOISY
android.media.RINGER_MODE_CHANGED
android.media.SCO_AUDIO_STATE_CHANGED
android.media.VIBRATE_SETTING_CHANGED
android.media.action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION
android.media.action.OPEN_AUDIO_EFFECT_CONTROL_SESSION
android.net.conn.BACKGROUND_DATA_SETTING_CHANGED
android.net.wifi.NETWORK_IDS_CHANGED
android.net.wifi.RSSI_CHANGED
android.net.wifi.SCAN_RESULTS
android.net.wifi.STATE_CHANGE
android.net.wifi.WIFI_STATE_CHANGED
android.net.wifi.p2p.CONNECTION_STATE_CHANGE
android.net.wifi.p2p.PEERS_CHANGED
android.net.wifi.p2p.STATE_CHANGED
android.net.wifi.p2p.THIS_DEVICE_CHANGED
android.net.wifi.supplicant.CONNECTION_CHANGE
android.net.wifi.supplicant.STATE_CHANGE
android.provider.Telephony.SIM_FULL
android.provider.Telephony.SMS_CB_RECEIVED
android.provider.Telephony.SMS_EMERGENCY_CB_RECEIVED
android.provider.Telephony.SMS_RECEIVED
android.provider.Telephony.SMS_REJECTED
android.provider.Telephony.WAP_PUSH_RECEIVED
android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED
android.speech.tts.engine.TTS_DATA_INSTALLED
```

SMS Received - Broadcast Receiver

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context c, Intent in) {
        Log.v(TAG,"On Receive");
        if(in.getAction().equals(RECEIVED_ACTION)) {
            Log.v(TAG,"On SMS RECEIVE");
            Bundle bundle = in.getExtras();
            if(bundle!=null) {
                Object[] pdus = (Object[])bundle.get("pdus");
                SmsMessage[] messages = new SmsMessage[pdus.length];
                for(int i = 0; i<pdus.length; i++) {
                    Log.v(TAG,"FOUND MESSAGE");
                    messages[i]=SmsMessage.createFromPdu((byte[])pdus[i]);
                }
                for(SmsMessage message: messages)
                    requestReceived(message.getOriginatingAddress());
                respond();
            }
```

SMS Data

- The SMS data in the Bundle (map) is under the key "pdus"
  - pdu, protocol data unit (some sources indicate protocol description unit)

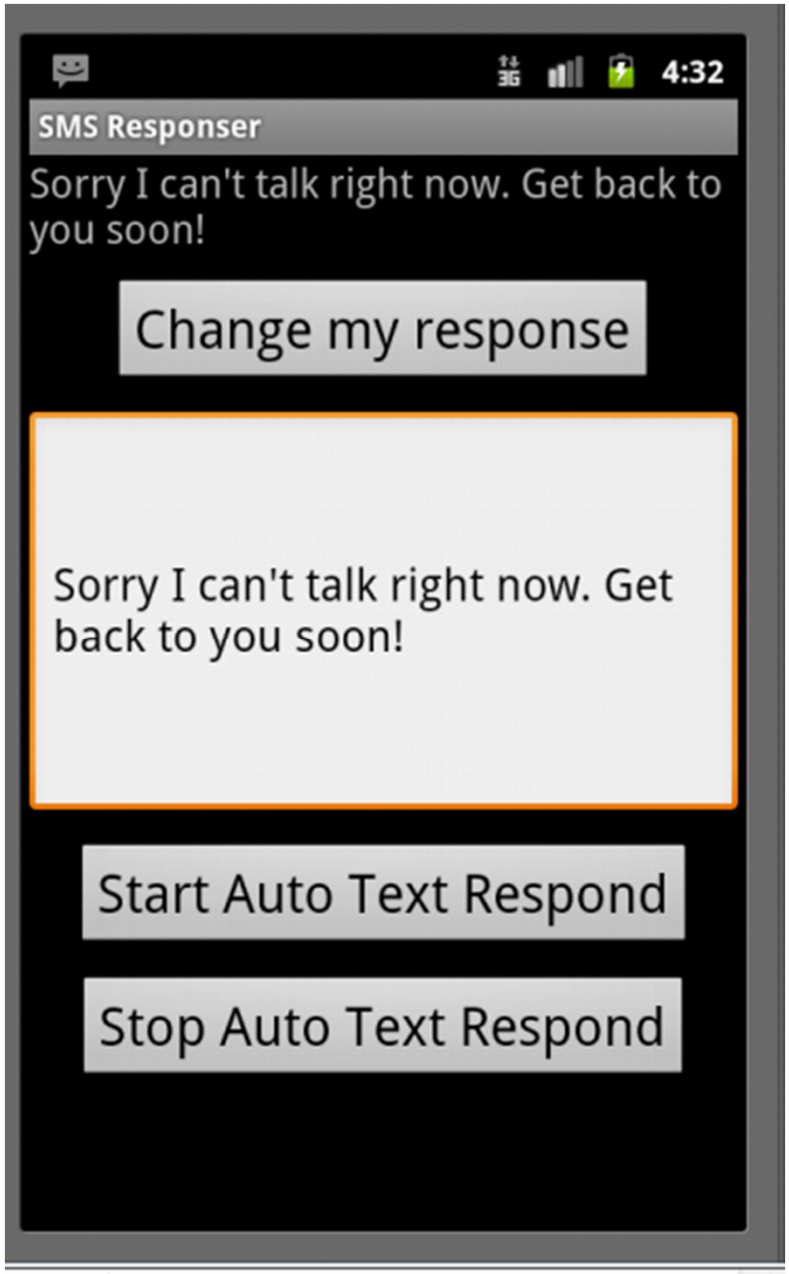
respond method

- incoming SMS messages trigger respond method

```
private void respond() {
    reply = myprefs.getString("reply", "Thank you for your message. I am busy
    if(reply.length() == 0)
        reply = "Thank you for your message. I am busy now. I will call you l
    SmsManager sms = SmsManager.getDefault();
    Intent sentIn = new Intent(SENT_ACTION);
    PendingIntent sentPIn = PendingIntent.getBroadcast(this,0,sentIn,0);

    Intent deliverIn = new Intent(DELIVERED_ACTION);
    PendingIntent deliverPIn = PendingIntent.getBroadcast(this,0,deliverIn,0)

    if(reply.length() > 140)
        reply = reply.substring(0, 140);

    sms.sendTextMessage(requester, null, reply, sentPIn, deliverPIn);
```

Stopping Service

- Once started service runs until device shut down
- Starts again when app started again
- Add option to start and shut down the service



Starting Service

```
private void startMyService() {
    Log.v(TAG,"In startMyService method");
    boolean running = isMyServiceRunning();
    Log.d(TAG, "running: " + running);
    if(!running) {
        try {
            // start Service
            Intent svc = new Intent(this, ResponserService.class);
            startService(svc);
        }
        catch (Exception e) {
            Log.e("onCreate", "service creation problem", e);
        }
    }
}
```

Checking Running Processes

```
private boolean isMyServiceRunning() {
    Log.v(TAG, "checking if service is running");
    ActivityManager manager
        = (ActivityManager) getSystemService(ACTIVITY_SERVICE);
    for (RunningServiceInfo service
                : manager.getRunningServices(Integer.MAX_VALUE)) {
        Log.v(TAG, service.service.getClassName() + "");
        if (serviceName.equals(service.service.getClassName())) {
            return true;
        }
    }
    return false;
}
```

```
checking if service is running
com.android.internal.service.wallpaper.ImageWallpaper
jp.co.omronsoft.openwnn.OpenWnnJAJP
com.android.systemui.statusbar.StatusBarService
com.android.phone.BluetoothHeadsetService
```