http://www.android.com/

SMS/MMS/Email

Alarms and Notifications

Services and App widgets

Alternative resources and localization

Test the app and Android NDK

Android Security

Intent and intent-filter

- Allows the application to request and/or provide services i.e. start Activities, Services or Broadcast Receivers
- Intents are system messages that notify applications of various events/actions, transfer various data etc.
  - Activity events (launch app, start activity, pressing widgets, etc.)
  - Hardware state changes (battery status, screen off, etc.)
  - Incoming data (call received, SMS received, etc.)
- Applications are registered via an intent-filter allowing to create loosely coupled applications
- You can create your own to launch applications
  - In the AndroidManifest.xml below the ”SmsReceiver extends BroadCastReceiver” class can intercept incoming SMS action

```
<receiver android:name=".SmsReceiver">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

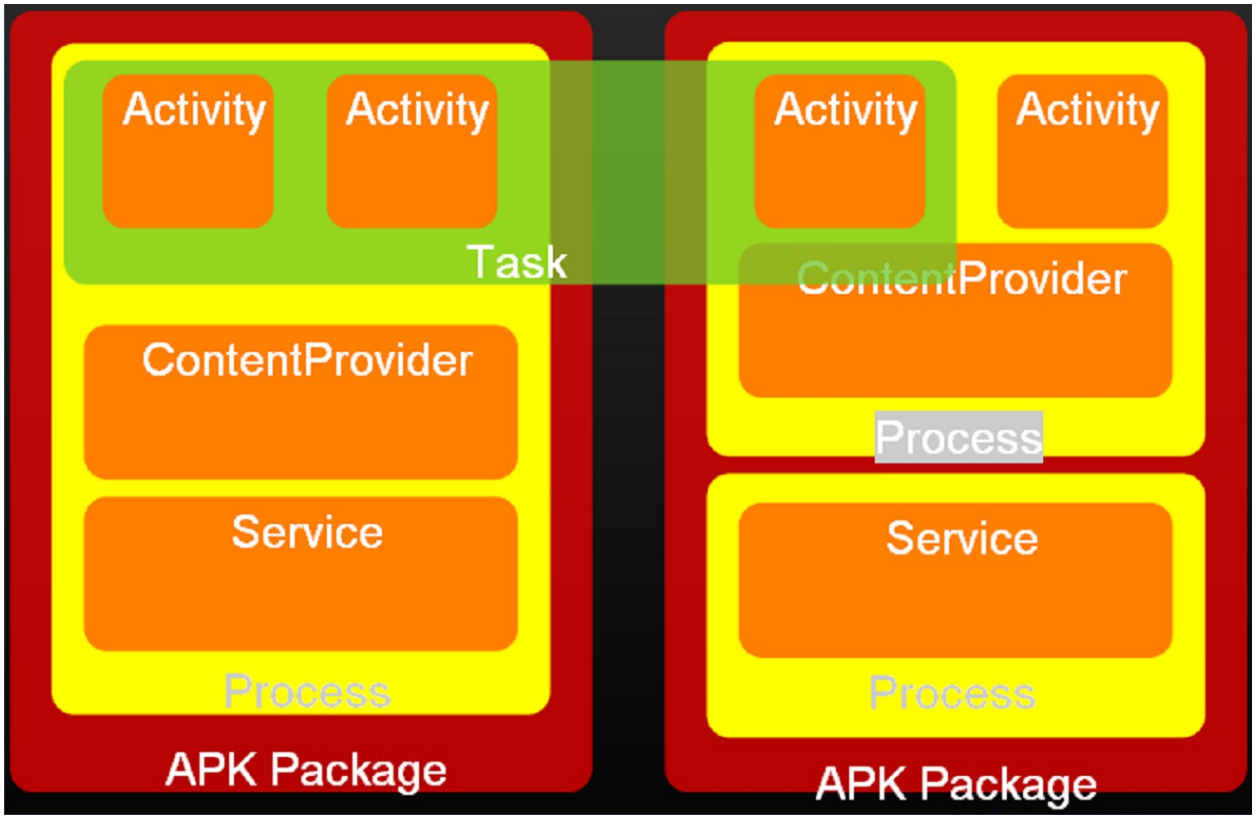
BroadcastReceiver

- By registering a broadcast receiver in the AndroidManifest (class xyz extends BroadcastReceiver) or dynamically in the source code the application can listen and respond to broadcast Intents that match a specific filter criteria
- By calling batteryLevel() a Toast will show the battery level when onReceive() is called by the system
- When onRecive() is done the lifecycle has ended for a broadcast receiver

```
private void batteryLevel()
{
    BroadcastReceiver batteryLevelReceiver = new BroadcastReceiver()
    {
        public void onReceive(Context context, Intent intent)
        {
            // unregistration of the reciever
            context.unregisterReceiver(this);
            int rawlevel = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
            int scale = intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
            int level = -1;
            if (rawlevel >= 0 && scale > 0) {
                level = (rawlevel * 100) / scale;
           }
           showToastMessage("Battery Level Remaining: " + level + "%");
        }
    };

    // Intent and a dynamic registration of a receiver via registerReciver
    IntentFilter batteryLevelFilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
    registerReceiver(batteryLevelReceiver, batteryLevelFilter);
}
```

Android application model

- A task is what the user sees as an application
  - The borders between executable, process and app icon are blurry
  - Default 1 thread/process, additional threads are created only if the app itself create them
  - http://developer.android.com/guide/topics/fundamentals.html



SMS intro

- How mobile phones work (GSM)

BTS

Call received by nearest base station

BSC

Base station controller determines power of transmitters and manages handover of calls made from one cell to the next if you are moving

MSC

MSSC patches the call into the Public Telephone Network which directs call to the mobile or landline

PSTN or other PLMN

Public Telephone Network

Base station controller passes call to Mobile Services Switching Centre (MSSC)

AuC

EIR

VLR

Visitor Location Register

Requests user details from home location register. Tells network where phone is

HLR

Home Location Register

Holds all information on every user

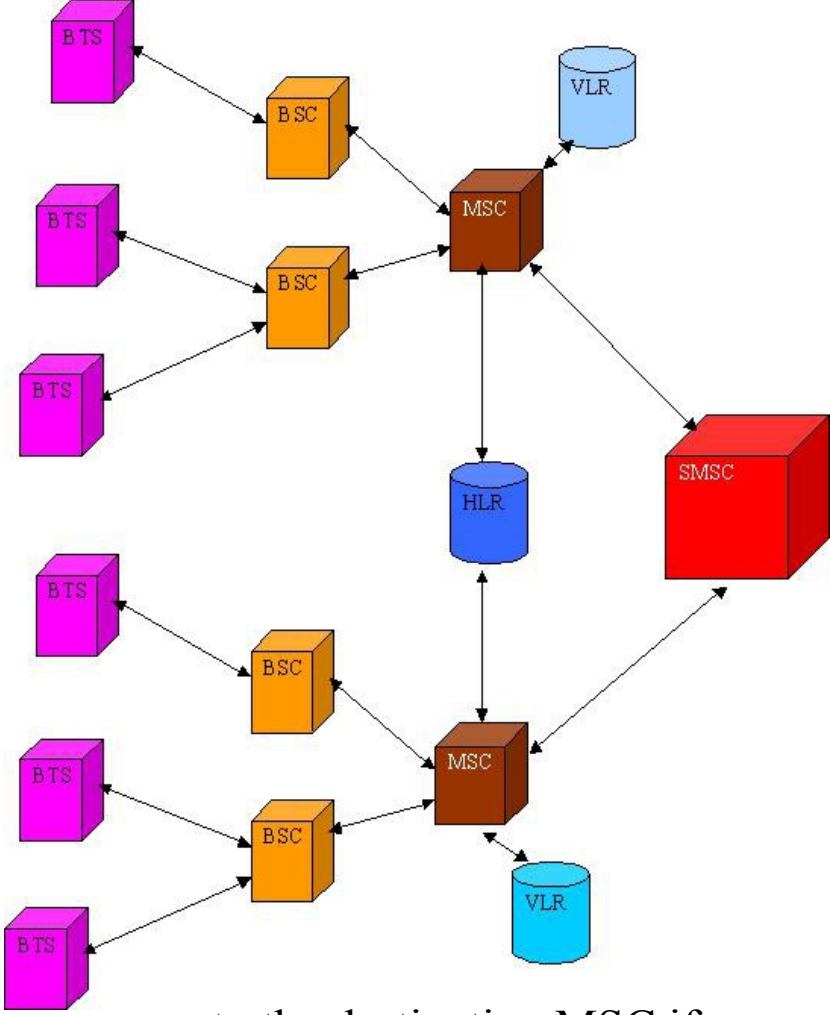
Authentication Centre

Security checks caller ID on SIM Generates encryption key for the call

Equipment Identity Register

Requests IMEI from phone and bars call if blacklisted

SMS intro



- BTS - Base Transceiver Station (antenna)
- BSC - Base Station Controller
- MSC - Mobile Switching Center
- HLR- Home Location Register
- VLR - Visitor Location Register
- SMSC - Short Message Service Center
- When a user sends an SMS, the request is placed via the MSC
- The MSC forwards the SMS to the SMSC where it gets stored
- The SMSC queries the HLR to find out where the destination mobile is and forwards the message to the destination MSC if the destination mobile is available
- If the mobile is not available the message gets stored in the current SMSC itself. In most installations If a mobile is not available for SMS delivery the SMSC will not retry. Instead the destination MSC will inform the SMSC when the mobile comes back in range

http://services.eng.uts.edu.au/userpages/kumbes/public_html/ra/sms/

SMS 1

- PendingIntent is a description of an Intent and target action to perform with it
  - By giving a PendingIntent to another application, you are granting it the right to perform the operation you have specified as if the other application was yourself (with the same permissions and identity)
- SmsManager manages SMS operations
  - Most SMSs are sent via the PDU (Protocol Description Unit) format: http://www.dreamfabric.com/sms/
- To receive SMS we must set up a receiver in AndroidManifest and create the BroadcastReceiver class which override the onReceive(Context context, Intent intent) method (next slide)

```
private void sendSMS(String phoneNumber, String message)
{
      PendingIntent pi = PendingIntent.getActivity(this, 0, new Intent(this, SMS.class), 0);
      // Get the default instance of the SmsManager
      SmsManager sms = SmsManager.getDefault();
      // sendTextMessage (String destinationAddress, String scAddress, String text,
      // PendingIntent sentIntent, PendingIntent deliveryIntent)
      sms.sendTextMessage(phoneNumber, null, message, pi, null);
}
```

If we want to listen for the sent and delivery intents we need to set up receivers for these aswell

Permissions are needed in AndroidManifest
android.permission.SEND_SMS
android.permission.RECEIVE_SMS

SMS 2

- The incoming SMS broadcast receiver uses a bundle to retrieve the PDU (Protocol Description Unit), which contains the SMS text and any additional SMS meta-data, and parses it into an Object array

```
public class SmsReceiver extends BroadcastReceiver
{
      @Override
      public void onReceive(Context context, Intent intent)
      {
            Bundle bundle = intent.getExtras(); //---get the SMS message passed in---
            SmsMessage[] msgs = null;
            String str = "";

            if (bundle != null)
            {
                  //---retrieve the SMS message received---
                  Object[] pdus = (Object[]) bundle.get("pdus");
                  msgs = new SmsMessage[pdus.length];
                  //---for every SMS message received---
                  for (int i=0; i<msgs.length; i++)
                  {
                        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]); // convert Object array
                        str += "SMS from " + msgs[i].getOriginatingAddress(); //sender's phone number
                        str += " :";
                        str += msgs[i].getMessageBody().toString(); // get the text message
                        str += "\n";
                  }//---display the new SMS message---
                  Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
            }
      }
}
```

```
<receiver android:name=".SmsReceiver">
      <intent-filter>
            <action android:name="android.provider.Telephony.SMS_RECEIVED" />
      </intent-filter>
</receiver>
```

SMS 3

- Most SMSes are restricted to 140 characters per text message. To make sure the message is within this limitation, use the **divideMessage()** method that divides the text into fragments in the maximum SMS message size. Then, the method **sendMultipartTextMessage()**

```
private void sendTextSMSMulti(String destination, String message)
{
	SmsManager mySMS = SmsManager.getDefault();
	Intent sentIn = new Intent("SENT_SMS");
	PendingIntent sentPIn = PendingIntent.getBroadcast(this, 0, sentIn, 0);
	Intent deliverIn = new Intent("DELIVER_SMS");
	PendingIntent deliverPIn = PendingIntent.getBroadcast(this, 0, deliverIn, 0);
	ArrayList<String> multiSMS = mySMS.divideMessage(message);
	ArrayList<PendingIntent> sentIns = new ArrayList<PendingIntent>();
	ArrayList<PendingIntent> deliverIns = new ArrayList<PendingIntent>();
	for(int i=0; i< multiSMS.size(); i++){
		sentIns.add(sentPIn);
		deliverIns.add(deliverPIn);
	}
	mySMS.sendMultipartTextMessage(destination, null, multiSMS, sentIns, deliverIns);
	BroadcastReceiver sentReceiver = new BroadcastReceiver(){
		@Override public void onReceive(Context c, Intent in) {
			switch(getResultCode()){
			case Activity.RESULT_OK:
				Break; //sent SMS message successfully;
			default:
				Break; //sent SMS message failed
			}
		}
	};
	BroadcastReceiver deliverReceiver = new BroadcastReceiver(){
		@Override public void onReceive(Context c, Intent in) {
			//SMS delivered actions
		}
	};
	registerReceiver(sentReceiver, new IntentFilter("SENT_SMS"));
	registerReceiver(deliverReceiver, new IntentFilter("DELIVER_SMS"));
}
```

Binary SMS and MMS

- To send binary SMS we need a destination port
- Sending MMS using the built-in SMS/MMS manager (the ones who listen for ACTION_SEND)

```
private void sendBinarySMS(String phoneNumber, byte[] data)
{
     short destinationPort = 2948;
     PendingIntent pi = PendingIntent.getActivity(this, 0, new Intent(this, SMS.class), 0);
     SmsManager sms = SmsManager.getDefault(); // Get the default instance of the SmsManager
     // Send a data based SMS to a specific application port.
     sms.sendDataMessage(phoneNumber, null, destinationPort, data, pi, null);
}
```

```
// Send MMS via a broadcast intent to the UE built in action send components
private void sendMMS(String phoneNo, String subject, String message)
{
      String url = "file:////sdcard//DCIM//07.jpg";
      /* The url being passed to the Uri.parse method should be of the form used to access the media store
       * such as content://media/external/images/media/23 or file://sdcard/dcim/Camera/off2.jpg
      */
      Intent sendIntent = new Intent(Intent.ACTION_SEND);
      sendIntent.putExtra(Intent.EXTRA_PHONE_NUMBER, phoneNo);
      sendIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
      sendIntent.putExtra(Intent.EXTRA_TEXT, message);
      sendIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse(url));
      sendIntent.setType("image/jpg"); // specify explicit, normally type is set automatically from the data
      startActivity(sendIntent);    // broadcast intent for all apps listening to ACTION_SEND
}
```

Binary SMS receiver

```
<receiver android:name="BinarySmsReceiver">
      <intent-filter>
            <action android:name="android.intent.action.DATA_SMS_RECEIVED"
                  android:scheme="sms" android:host="localhost" android:port="2948">
            </action>
      </intent-filter>
</receiver>
```

```
public class BinarySmsReceiver extends BroadcastReceiver
{
      @Override
      public void onReceive(Context context, Intent intent)
      {
            Bundle bundle = intent.getExtras();
            SmsMessage[] msgs = null;
            String info = "Binary SMS from ";
            if (bundle != null){
                  //---retrieve the binary SMS message received---
                  Object[] pdus = (Object[]) bundle.get("pdus");
                  msgs = new SmsMessage[pdus.length];
                  byte[] data = null;
                  for (int i=0; i<msgs.length; i++)
                  {
                        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                        info += msgs[i].getOriginatingAddress();
                        info += "\n*****BINARY MESSAGE*****\n";
                        // returns the user data section minus the user data header if one was present.
                        data = msgs[i].getUserData();
                        for(int index=0; index<data.length; index++)
                              info += Byte.toString(data[index]);
                  }
                  //---display the new binary SMS message---
                  Toast.makeText(context, info, Toast.LENGTH_LONG).show();
            }
      }
}
```

Send SMS and Email with built-in clients (user interaction)

```
public void sendSMS(String phoneNumber, String message)
{
      // sendSMS(”5554”, ”Hello my friends!”);
      Intent i = new Intent(android.content.Intent.ACTION_VIEW);
      i.putExtra("address", phoneNumber);
      i.putExtra("sms_body", message);
      i.setType("vnd.android-dir/mms-sms");
      startActivity(i);
}
```

```
String[] to = {"hjo@du.se"};
String[] cc = {""};
sendEmail(to, cc, "Cellid", mMessage, mFileUrl);

// check http://www.openintents.org/en/uris for MIME types
// http://developer.android.com/reference/android/content/Intent.html

//---sends an Email message to another device---
private void sendEmail(String[] emailAddresses, String[] carbonCopies,
      String subject, String message, String url)
{
    Intent emailIntent = new Intent(Intent.ACTION_SEND);
    emailIntent.setData(Uri.parse ("mailto:"));
    String[] to = emailAddresses;
    String[] cc = carbonCopies;
    emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
    emailIntent.putExtra(Intent.EXTRA_CC, cc);
    emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
    emailIntent.putExtra(Intent.EXTRA_TEXT, message);
    emailIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse("file:///" + url));
    emailIntent.setType("message/rfc822");
    startActivity(Intent.createChooser(emailIntent, "Email"));
}
```

Event Handlers 1

- Most user interaction with an Android device is captured by the system and sent to a corresponding callback method
  - For example, if the physical Back button is pressed, the onBackPressed() method is called
  - Event listeners as setOnClickListener() etc. are however the preferred method when available because they avoid the class extension overhead
- The system first sends any KeyEvent to the appropriate callback method in the in-focus activity or view. Callbacks:
  - onKeyUp(), onKeyDown(), onKeyLongPress() — Physical key press callbacks
  - onTrackballEvent(), onTouchEvent() — Trackball and touchscreen press callbacks
  - OnFocusChanged() — Called when the view gains or loses focus

Event Handlers 2

Physical buttons are most for programming games and other specific usages when events listners are not available or usable

The Power button and HOME key are intercepted by the system and do not reach the application.
The BACK, MENU, HOME, and SEARCH keys should intercept the **onKeyUp()**. Because these buttons might not be physical keys

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
      if (keyCode == KeyEvent.KEYCODE_CAMERA) {
            return true;
// consume event, hence do nothing on camera button
      }
// let event propagate in class tree
      return super.onKeyDown(keyCode, event);
}
```

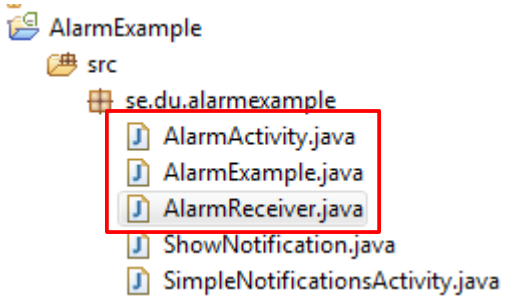
Example

Table 5.1 **The Possible Physical Keys on an Android Device**

Physical Key	KeyEvent	Description
Power button	KEYCODE_POWER	Turns on the device or wakes it from sleep; brings UI to the lock screen
BACK key	KEYCODE_BACK	Navigates to the previous screen
MENU key	KEYCODE_MENU	Shows the menu for the active application
HOME key	KEYCODE_HOME	Navigates to the home screen
SEARCH key	KEYCODE_SEARCH	Launches a search in the active application
Camera button	KEYCODE_CAMERA	Launches the camera
Volume button	KEYCODE_VOLUME_UP KEYCODE_VOLUME_DOWN	Controls volume of the media by context (voice when in a phone call, music when in media playback, or ringer volume)
DPAD	KEYCODE_DPAD_CENTER KEYCODE_DPAD_UP KEYCODE_DPAD_DOWN KEYCODE_DPAD_LEFT KEYCODE_DPAD_RIGHT	Directional pad on some devices
Trackball	-	Directional joystick on some devices
Keyboard	KEYCODE_0, ..., KEYCODE_9, KEYCODE_A, ..., KEYCODE_Z	Pull-out keyboard on some devices
Media button	KEYCODE_HEADSETHOOK	Headset Play/Pause button

Alarm and notofication example 1

- The Activity AlarmExample executes and use the AlarmManager to set a wakeup intent with a message within 5 seconds
- The AlarmReceiver will get the intent and message which in turn starts AlarmActivity

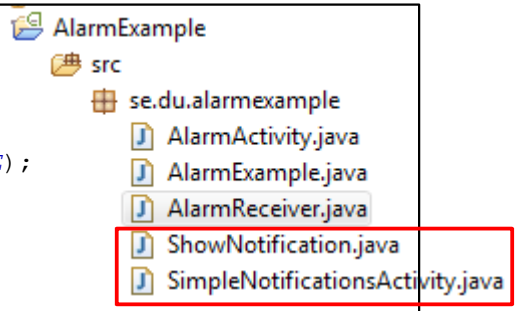


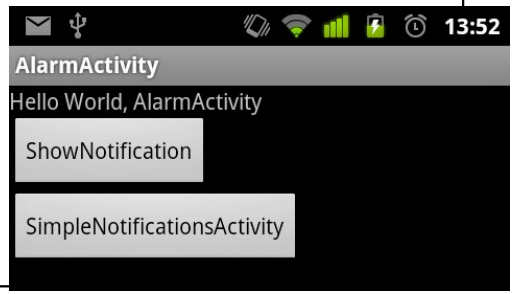

```
private void alarmTest()
{
    int requestCode = 192837;
    // get a Calendar object with current time
    Calendar cal = Calendar.getInstance();
    // add 5 seconds to the calendar object
    cal.add(Calendar.SECOND, 5);
    Intent intent = new Intent(getApplicationContext(), AlarmReceiver.class);
    intent.putExtra("alarm_message", "Android Notifications Rules!");
    // Private request code for the sender (currently not used) according to
    // http://developer.android.com/reference/android/app/PendingIntent.html
    // Retrieve a PendingIntent that will perform a broadcast
    PendingIntent sender = PendingIntent.getBroadcast(this, requestCode,
        intent, PendingIntent.FLAG_UPDATE_CURRENT);

    // Get the AlarmManager service
    AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);
    // http://developer.android.com/reference/android/app/AlarmManager.html
    am.set(AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(), sender);
}
```

Alarm and notofication example 2

- The AlarmReceiver got the intent - created a new intent/message and started the AlarmActivity class which will toast the message and now can start execute different kinds of notification tests

```
private NotificationManager mNManager;
public void onCreate(Bundle savedInstanceState) {
Button start = (Button)findViewById(R.id.button1);
Button cancel = (Button)findViewById(R.id.button2);
mNManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
final Notification msg = new Notification(R.drawable.icon,
            "New event of importance", System.currentTimeMillis());
// start button in ShowNotification class
start.setOnClickListener(new OnClickListener() {
      public void onClick(View v)
      {
            Context context = getApplicationContext();
            CharSequence contentTitle = "ShowNotification Example";
            CharSequence contentText = "Browse SweDroid";
            Intent msgIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.swedroid.se"));
            PendingIntent intent = PendingIntent.getActivity(ShowNotification.this,
                        0, msgIntent, Intent.FLAG_ACTIVITY_NEW_TASK);
            msg.defaults |= Notification.DEFAULT_SOUND;
            msg.flags |= Notification.FLAG_AUTO_CANCEL;
            msg.setLatestEventInfo(context, contentTitle, contentText, intent);
            mNManager.notify(NOTIFY_ID, msg);
      }
});

cancel.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
            mNManager.cancel(NOTIFY_ID);
      }
});
```





Notofication example 3.1

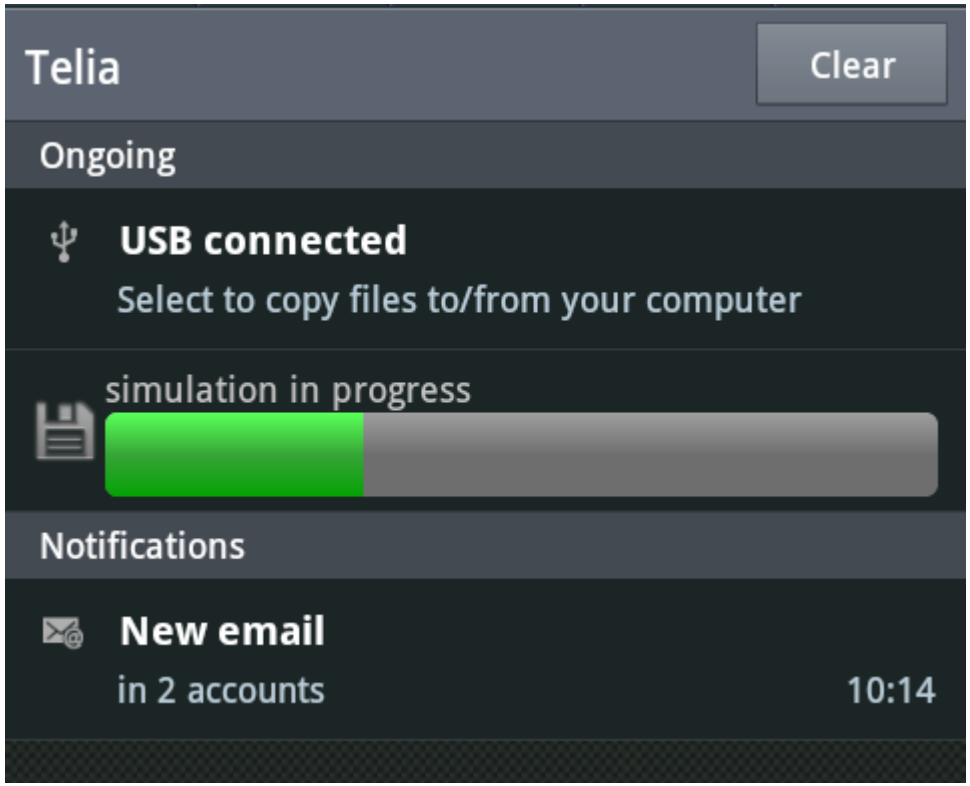
- Sometimes you want to perform longer work in the background, you can use an ongoing notification for this with possibly some kind of progress with a remote view

```
final int NOTIFY_ID = 434;
int progress = 10;
Context context = getApplicationContext();
Intent intent = new Intent(this, DownloadProgress.class); // configure the intent
final PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);
// configure the notification
final Notification notification = new Notification(R.drawable.icon, "simulating a download", System.currentTimeMillis());
notification.flags = notification.flags | Notification.FLAG_ONGOING_EVENT;
notification.contentView = new RemoteViews(context.getPackageName(), R.layout.download_progress);
notification.contentIntent = pendingIntent;
notification.contentView.setImageViewResource(R.id.status_icon, R.drawable.ic_menu_save);
notification.contentView.setTextViewText(R.id.status_text, "simulation in progress");
notification.contentView.setProgressBar(R.id.status_progress, 100, progress, false);
final NotificationManager notificationManager =
             (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(NOTIFY_ID, notification);

// simulate progress
Thread download = new Thread() {
    @Override
    public void run() {
        for (int i = 1; i < 100; i++) {
            progress++;
            notification.contentView.setProgressBar(R.id.status_progress, 100, progress, false);
            notificationManager.notify(NOTIFY_ID, notification); // inform the progress bar of updates in progress
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        notificationManager.cancel(NOTIFY_ID);  // remove the notification (we're done)
    }
};
download.run();
```

Notofication example 3.2

- The layout for the notification in notification drop-down

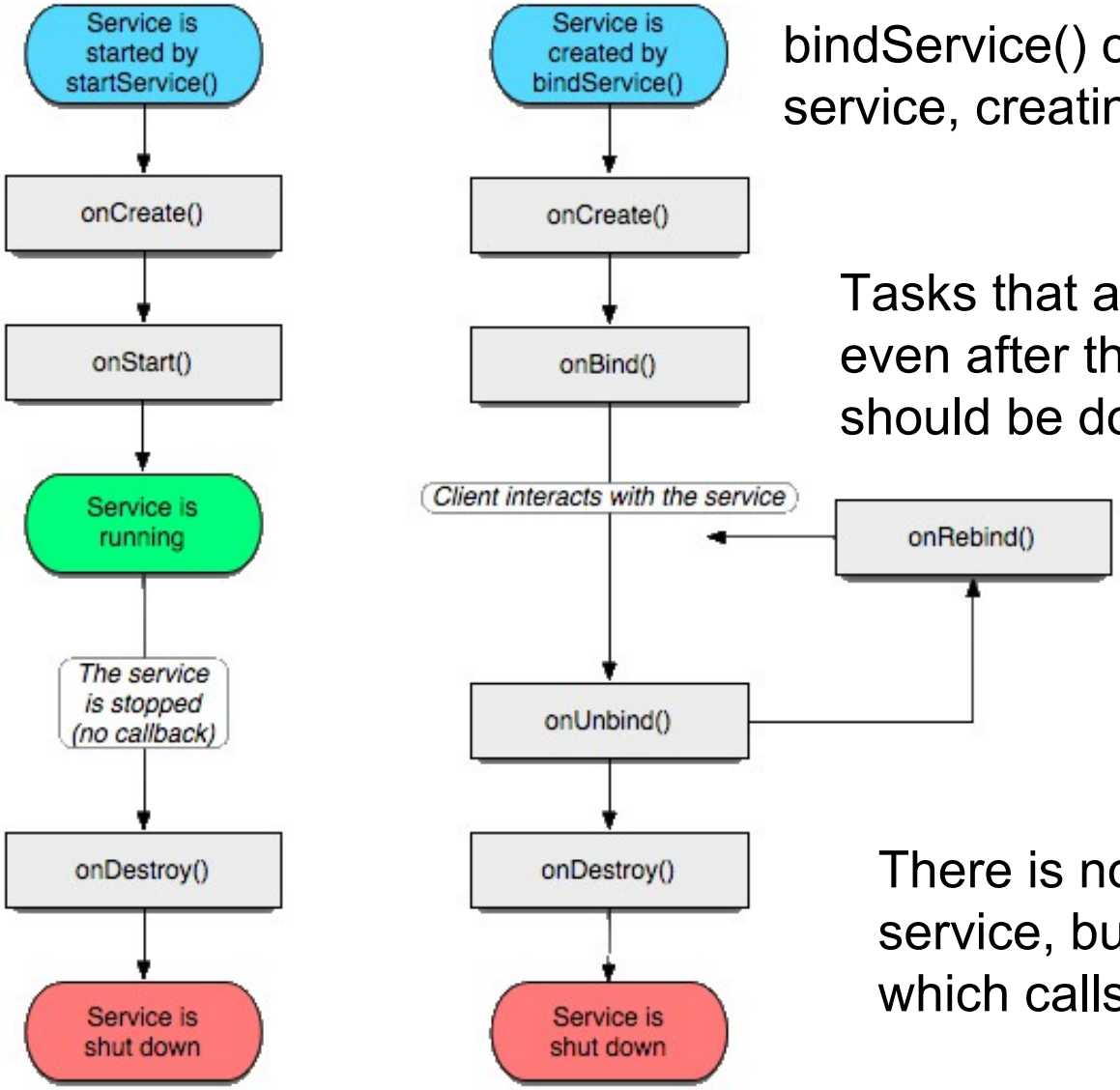
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dp" >

    <ImageView
        android:id="@+id/status_icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentLeft="true" />

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_toRightOf="@id/status_icon" >

        <TextView
            android:id="@+id/status_text"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true" />

        <ProgressBar
            android:id="@+id/status_progress"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/status_text"
            android:indeterminate="false"
            android:indeterminateOnly="false"
            android:progressDrawable="@android:drawable/progress_horizontal" />
    </RelativeLayout>
</RelativeLayout>
```



Services

- A service is an Android component that runs in the background without any user interaction
- Other components can start and stop the service, it can also stop itself
- While it is running other components can bind to it (if implemented)
- Some illustrative scenarios are
  - An activity provides the user a way to select a set of music files, which then starts a service to play back the files. During playback, a new activity starts and binds to the existing service to allow the user to change songs or stop playback.
  - An activity starts a service to upload a set of pictures to a website. A new activity starts and binds to the existing service to determine which file is currently being uploaded and displays the picture to the screen.
  - A broadcast receiver receives a message that a picture was taken and launches a service to upload the new picture to a website.The broadcast receiver then goes in-active and is eventually killed to reclaim memory, but the service continues until the picture is uploaded. Then, the service stops itself.

Service Life cycles



bindService() connect to an application service, creating it if needed

Tasks that are meaningful to continue even after the component stops should be done by launching a service

There is no concept of pausing a service, but it can be stopped, which calls the onDestroy() method

Creating a simple service

- Declare the service in your Androidmanifest inside the application tag

```
<service android:name=".SimpleService"></service>
```

- Create the source file and override the onCreate() and onDestroy() methods
  - In Eclipse, this can be done by right-clicking on the java class file, choosing Source > Override/Implement methods...
- Also override the onBind() method for cases when a new component binds to this service after it have been created
- Start, stop or bind to the service from an external trigger

```
startService(new Intent(LocationTracker.this, SimpleService.class));
stopService(new Intent(LocationTracker.this, SimpleService.class));
bindService(Intent service, ServiceConnection conn, int flags);
```

- Remember that threads etc. have to be created in in the service in order to not block the UI
- The service will not stop when the activity is destroyed, paused or the screen orientation is changed

Simple service

```
public class SimpleService extends Service
{
	int[] notes =
		{R.raw.c5, R.raw.b4, R.raw.a4, R.raw.g4};
	int NOTE_DURATION = 500; //millisec
	MediaPlayer m_mediaPlayer;
	boolean paused = false;

	@Override
	public IBinder onBind(Intent arg0) {
		return null;
	}

	@Override
	public void onCreate() {
		super.onCreate();
		Toast.makeText(this,"Service created ...",
			Toast.LENGTH_LONG).show();
		paused = false;
		Thread initBkgdThread =
			new Thread(new Runnable() {
			public void run() {
				play_music();
			}
		});
		initBkgdThread.start();
	}


	@Override
	public void onDestroy() {
		super.onDestroy();
		Toast.makeText(this, "Service destroyed ...",
			Toast.LENGTH_LONG).show();
		paused = true;
	}
```

```
// SimpleService class continue here

	private void play_music()
	{
		int i=0;
		for(int i=0; i<120; i++) {
			//check to ensure main activity not paused
			if(!paused)
			{
				if(m_mediaPlayer != null)
					m_mediaPlayer.release();

				m_mediaPlayer =
				MediaPlayer.create(this, notes[i%4]);
				m_mediaPlayer.start();

				try {
					Thread.sleep(NOTE_DURATION);
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
				i++;
			}
		}
	}
} // end SimpleService class
```

http://developer.android.com/reference/android/app/Service.html

URL above got more advanced examples and information how to bind to running services etc.

App Widgets

http://developer.android.com/guide/topics/appwidgets/index.html

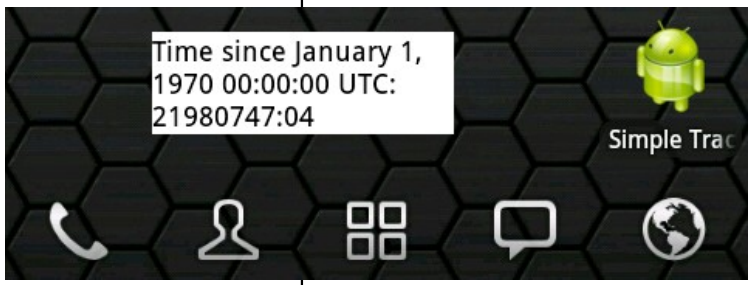
- App Widgets are usually small icon-like views in an application. They implement a subclass of the broadcast receiver for use in updating this view.
- Called widgets for short, they can be embedded into other applications, such as the home screen. In all, they require the following
  - A view describing the appearance of the widget.This is defined in an XML layout resource file and contains text, background, and other layout parameters.
  - An App Widget provider that receives broadcast events and interfaces to the widget to update it.
  - Detailed information about the App Widget, such as the size and update frequency. Note that the home screen is divided into 4x4 cells and so a widget is often a multiple of a single cell size (which is 80x100dp in Portrait mode and 106x74dp in Landscape mode).
    - http://developer.android.com/guide/practices/ui_guidelines/widget_design.html
  - Optionally, an App Widget configuration activity can be defined to properly set any parameters of the Widget. This activity is launched upon creation of the Widget.

```xml
<receiver android:name=".SimpleWidgetProvider">
        <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
        </intent-filter>
        <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/widget_info" />
</receiver>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
        android:minWidth="146dp" android:minHeight="72dp"
        android:updatePeriodMillis="1800000" android:initialLayout="@layout/widget_layout">
</appwidget-provider>
```

Simple App widget

```java
public class SimpleWidgetProvider extends AppWidgetProvider {
// Note! Updates requested with updatePeriodMillis will not be delivered more than once every 30 minutes
        @Override
        public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
                super.onUpdate(context, appWidgetManager, appWidgetIds);
                Log.v(Consts.TAG, "SimpleWidgetProvider > onUpdate()");
                // Perform this loop procedure for each App Widget that belongs to this provider
                final int N = appWidgetIds.length; // i.e. user have created
                for (int i=0; i<N; i++) {
                        int appWidgetId = appWidgetIds[i];
                        String titlePrefix = "Time since January 1, 1970 00:00:00 UTC:";
                        updateAppWidget(context, appWidgetManager, appWidgetId, titlePrefix);
                }
        }

        static void updateAppWidget(Context context, AppWidgetManager
                appWidgetManager, int appWidgetId, String titlePrefix) {
                Long millis = System.currentTimeMillis();
                int seconds = (int) (millis / 1000);
                int minutes = seconds / 60;
                seconds = seconds % 60;
                String text = titlePrefix;
                text += " " + minutes + ":" + String.format("%02d", seconds);
                Log.v(Consts.TAG, "updateAppWidget(): " + text);
                // Construct the RemoteViews object.
                RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget_layout);
                // String implements CharSequence, but CharSequence doesn't implement String
                views.setTextViewText(R.id.widget_example_text, text);
                // Tell the AppWidgetManager to perform an update on the current app widget
                appWidgetManager.updateAppWidget(appWidgetId, views);
        }
}
```



Alternative resources and localization

http://developer.android.com/guide/topics/resources/index.html

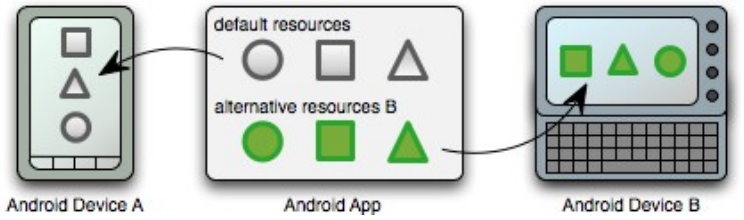
- Android run on many devivces and in many regions. To reach the most users, your application should handle text, audio files, numbers, currency, and graphics in ways appropriate to the locales and screen size etc.
- The default resources are required and should define every string etc.

```
res/values/strings.xml (required directory)
The default resource set must also include any default drawables and layouts, and can include other types
of resources such as animations.
res/drawable/(required directory holding at least one graphic file, for the application's icon in the Market)
res/layout/ (required directory holding an XML file that defines the default layout)
res/anim/ (required if you have any res/anim-<qualifiers> folders)
res/xml/ (required if you have any res/xml-<qualifiers> folders)
res/raw/ (required if you have any res/raw-<qualifiers> folders)
```

- When the user runs your program the Android system selects which resources to load, based on the device's locale etc.
  - If not found or partially not found it will fallback to the default resources
- Examples for lang, dimensions and styles

res/layout-se/main.xml, res/values-se/strings.xml

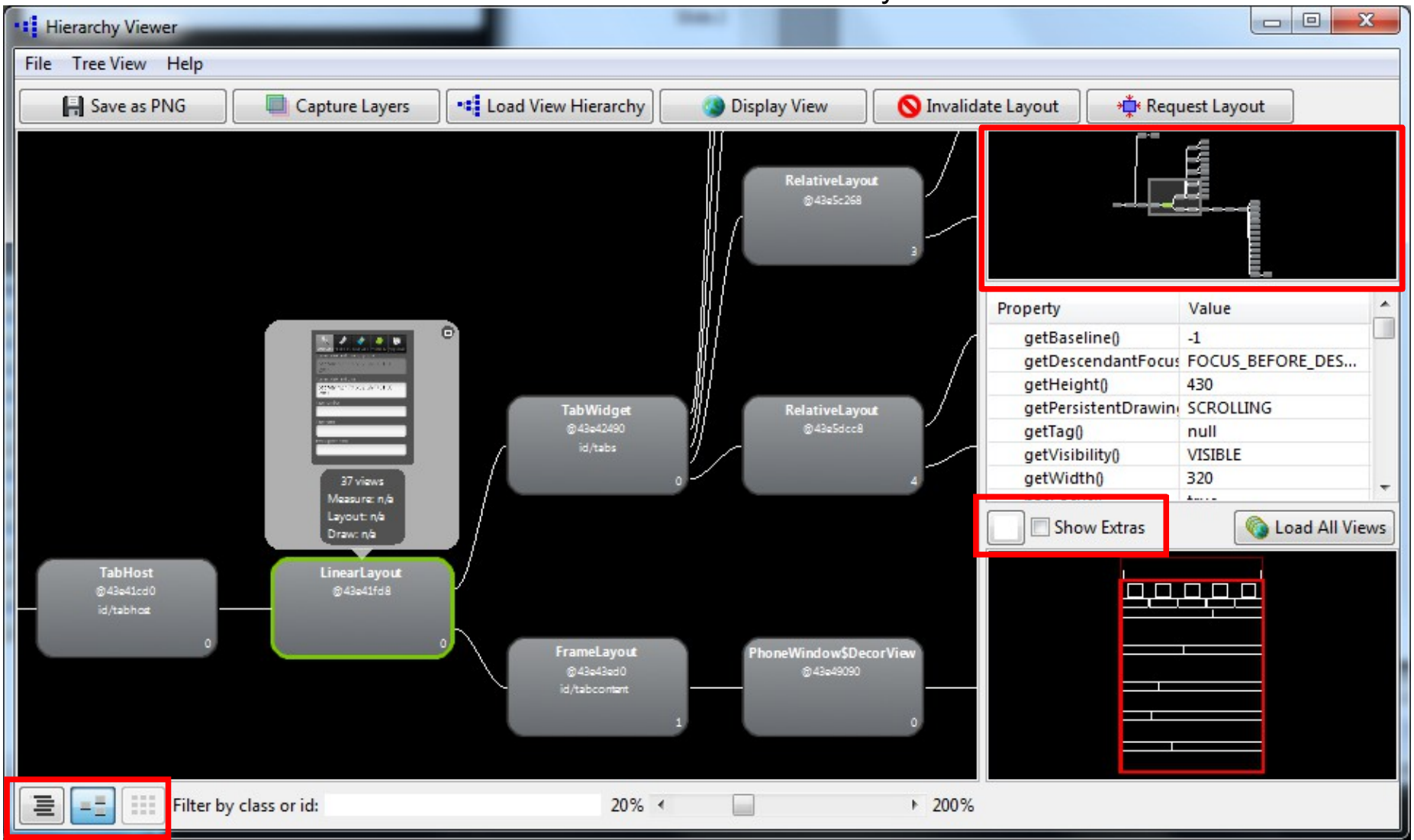
res/values-ldpi/dimens.xml, res/values-9/styles.xml



Hierarchy Viewer

View the layout in a program. Only works with the current app in emulator.

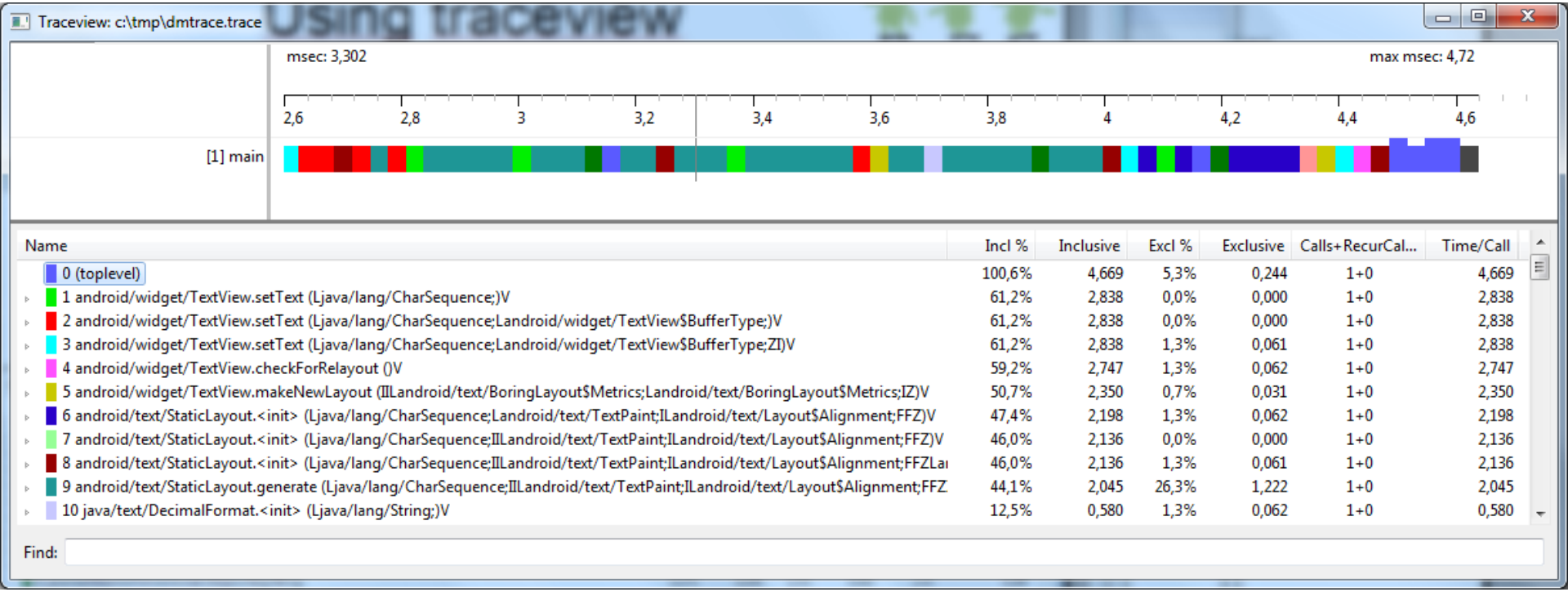
C:\android-sdk-windows\tools\hierarchyviewer.bat



Using traceview

- TraceView is a tool to optimize performance (profile the program)
- A dmtrace.trace file will be created on the SD-card
- Also possible via DDMS (Start Method Profiling button)

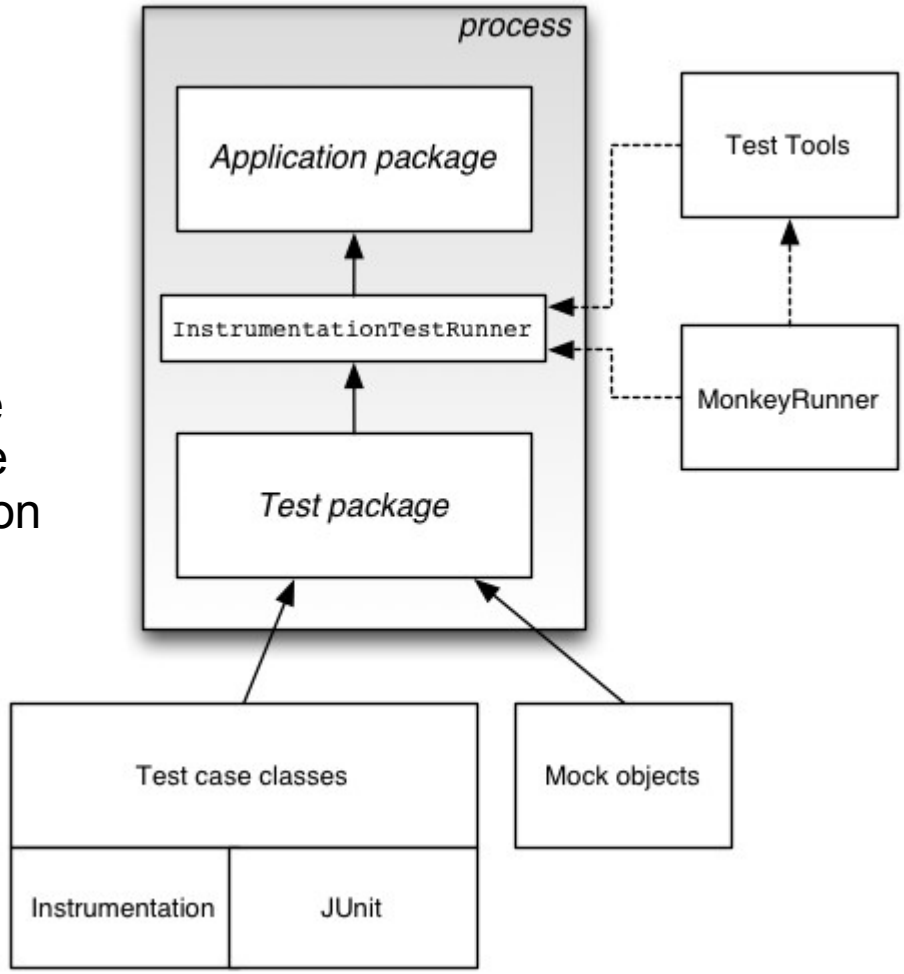
```
Debug.startMethodTracing("tag")
doHeavyWorkHere();
Debug.stopMethodTracing();
```



Testing

http://developer.android.com/guide/developing/testing/index.html

- Android testing framework is a part of the SDK/Eclipse ADT
- MonkeyRunner for stress-test of UI sending pseudo-random events
- JUnit - http://www.junit.org/
  - TestCase and AndroidTestCase classes do unit testing on a plain Java objects and Android objects
  - The Assert class methods compare values you expect from a test to the actual results and throw an exception if the comparison fails
- Android instrumentation is a set of control methods or "hooks" in the Android system. These hooks control an Android component independently of its normal lifecycle. They also control how Android loads applications.



What to test?

- Change in orientation
  - Is the screen re-drawn correctly? Any custom UI code you have should handle changes in the orientation.
  - Does the application maintain its state?
- Change in configuration
  - Change in the device's configuration, such as a change in the availability of a keyboard or a change in system language.
- Battery life
  - You need to write your application to minimize battery usage, you need to test its battery performance, and you need to test the methods that manage battery usage.
- Dependence on external resources
  - If your application depends on network access, SMS, Bluetooth, or GPS, then you should test what happens when the resource or resources are not available or limited.

Android native components

- Android NDK, is not hard to install or to use (limited C++ support)
- It have a rather limited API and is intended for performace with
  - OpenGL, including support for some newer versions that the (Java) SDK supports
  - Math (some, but not all, calculation-intensive algorithms might benefit from being done on the native layer)
  - 2D graphics – pixelbuffer support (only starting with 2.2)
  - libc – it's there for compatibility and perhaps to allow you to port existing native code
- You need to install C/C++ support in Eclipse mainly for syntax coloring and checking
- In Windows you need Cygwin 1.7.x and all devel branch packages
- The Android NDK

http://mindtherobot.com/blog/452/android-beginners-ndk-setup-step-by-step/

Using the NDK with Java Native Interface

- The idea is to put your native pieces of code into libraries/modules that you can then consume from the Java code via JNI

```
// C/C++ files are placed in the <project>/jni/ folder
#include <string.h>
#include <jni.h>

jint factorial(jint n){
      if(n == 1){
            return 1;
      }
      return factorial(n-1) * n;
}
// Java + package name + activity + function
jint Java_com_cookbook_advance_ndk_ndkact_factorial(
            JNIEnv* env, jobject javaThis, jint n ) {
      return factorial(n);
}
```

Type mapping Java <-> Native

Java Type in C/C++	Native Type	Description
jboolean	unsigned char	unsigned 8 bits
jbyte	signed char	signed 8 bits
jchar	unsigned short	unsigned 16 bits
jshort	short	signed 16 bits
jint	long	signed 32 bits
jfloat	float	32 bits
jlong	long long _int64	signed 64 bits
jdouble	double	64 bits

```
public class ndkact extends Activity {
            // loading the library/module – must match jni/android.mk
      static {
            System.loadLibrary("ndkcookbook");
      }
      // declaring the native function – must match cookbook.c
      private static native int factorial(int n);

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            TextView tv = new TextView(this);
            tv.setText(" native calculation on factorial :"+factorial(30));
            setContentView(tv);
      }
}
```

```
# the <project>/jni/Android.mk file
# a make file needed to build the lib
#
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE    := ndkcookbook
LOCAL_SRC_FILES := cookbook.c
include $(BUILD_SHARED_LIBRARY)
```

Compability

- New Android versions are generally additive and forward compatible at the API level. A device can be called an Android device only if it passes compatibly tests with the Android APIs.
  - Do not use internal or unsupported APIs.
  - Do not directly manipulate settings without asking the user
  - Do not go overboard with layouts. This is rare, but complicated layouts (more than 10 deep or 30 total) can cause crashes.
  - Do not make bad hardware assumptions. Be sure to check for the hardware needed
  - Ensure device orientations do not disrupt the application or result in unpredictable behavior.
- Note that backward compatibility is not guaranteed with Android! Use the minimum SDK version

```
<uses-sdk android:minSdkVersion="8" />
```

Robustness

- In the same vein as compatibility support, applications should be designed and tested for robustness.
  - Use the Android libraries before Java libraries. Android libraries are constructed specifically for embedded devices and cover many of the requirements needed in an application.
  - Take care of memory allocation. Initialize variables. Try to reuse objects rather than reallocate. This speeds up application execution and avoids excessive use of garbage collection.
  - Utilize the LogCat tool for debugging and check for warnings or errors
  - Test thoroughly, including different environments and devices if possible

Effective use of Java

- Make good use of static methods and scalar types
- Compare against 0 or null, ex. **for(int i=s.size(); i>=0; i--)**
- Avoid operations on String objects - Use the StringBuilder class for efficient manipulation of strings
- Limit the use of inner classes
- Use an obfuscator to reduce class file size
- Set object references to null as soon as they are no longer needed
- Avoid unnecessary re-initialization of variables that are automatically set to 0 or null by the VM
- Use synchronization sparingly, it is costly and is only needed in multi-threaded applications
- **Design is most important as usual!**
- Use native bridged code as: System.arraycopy()
- Profiling/tracing to reduce bottle necks…

Reversing an Android app

- Task – get rid of the 10 minute lockout time nag in the Android Bluetooth GPS output program
  - http://www.meowsbox.com/btgps/index.html
- Tools
  - Apktool, dex2jar, Java Decompiler and jarsigner (Java JDK)

- Get hold of the apk file e.g. /data/app/com.meowsbox.btgps.apk, (can also be in /data/app-private/) from the Android phone or Internet
- Unzip the com.meowsbox.btgps.apk file and grab classes.dex file
  - Run ”dex2jar classes.dex” which will convert the dex file into a ordinary jar file which can be opened with the **Java Decompiler** program
- Run ”apktool d com.meowsbox.btgps.apk” which will decompress and disassembly the apk file
  - A folder is created with the resources and .smali dalvik code etc.
- Using Java Decompiler try to localize where the time nag is in the java code and find the corresponding code in the dalvik ”assemblies”
- To learn more search on: ”android reversing”
  - Good reversing site: **http://androidreversing.blogspot.com/**

Java (decompiler) vs. dalvik code

I changed the opcode from **if-eqz** to **if-nez** in BluetoothChat.smali

```
public void sendNMEAString(String paramString)
{
  if (this.mChatService.getState() == 3)
  {
    if (!this.isRegistered)
      break label44;
    byte[] arrayOfByte1 = paramString.getBytes();
    this.mChatService.write(arrayOfByte1);
    int i = this.limit_nmeaCount + 1;
    this.limit_nmeaCount = i;
  }
  while (true)
  {
    return;
    label44: if (this.limit_nmeaCount < 3000)
    {
      byte[] arrayOfByte2 = paramString.getBytes();
      this.mChatService.write(arrayOfByte2);
      int j = this.limit_nmeaCount + 1;
      this.limit_nmeaCount = j;
      continue;
    }
  TextView localTextView = (TextView)findViewById(2131099664);
  localTextView.setText(
    "***Trial time limit reached: GPS output disabled.");
```

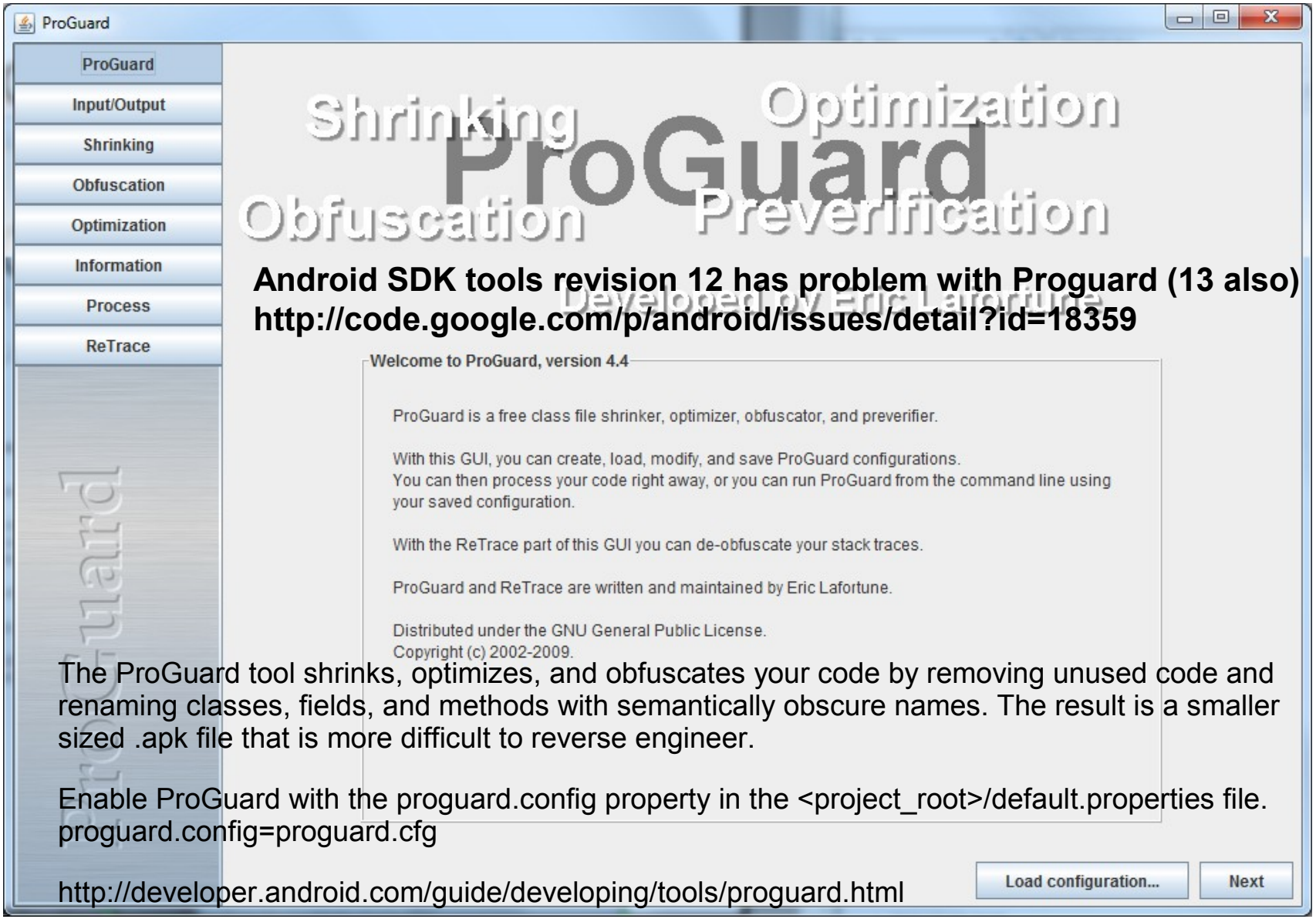
```
.method public sendNMEAString(Ljava/lang/String;)V
    .locals 8
    .parameter "nmeaString"

    .prologue
    const/4 v7, 0x1

    .line 954
    iget-object v4, p0, Lcom/meowsbox/btgps/BluetoothChat;->mChatService:Lcom/meowsbox/btgps/BluetoothChatService;

    invoke-virtual {v4}, Lcom/meowsbox/btgps/BluetoothChatService;->getState()I

    move-result v4

    const/4 v5, 0x3

    if-ne v4, v5, :cond_0

    .line 955
    iget-boolean v4, p0, Lcom/meowsbox/btgps/BluetoothChat;->isRegistered:Z

    if-nez v4, :cond_1

    .line 956
    invoke-virtual {p1}, Ljava/lang/String;->getBytes()[B
```

Repackaging and protection

- When the dalvik code changes is saved run
  - ”apktool b com.meowsbox.btgps com.meowsbox.btgps_new.apk”
- Now the application needs to be signed, run
  - "C:\Program Files\Java\jdk1.6.0_24\bin\jarsigner" -keystore C:\Users\hjo\.android\debug.keystore com.meowsbox.btgps_new.apk androiddebugkey
  - Use the password: **android**
- After this you can install the cracked app with ADB etc.

- To protect your code enable proguard in the default.properties file
  - proguard.config=proguard.cfg, make sure you got a proguard.cfg file!
  - Note that Proguard never runs when you compile ”debug” code!
- To obfuscate your android program (debuggable must be off in AndroidManifest.xml) and create ”release” code
  - In eclipse mark your project and select File > Export > Android > Export Android Application which will compile and align your code
  - Then follow the wizard and point out your debug keystore (as above) or your registered developer keystore and enter the password

ProGuard



Android SDK tools revision 12 has problem with Proguard (13 also)
http://code.google.com/p/android/issues/detail?id=18359

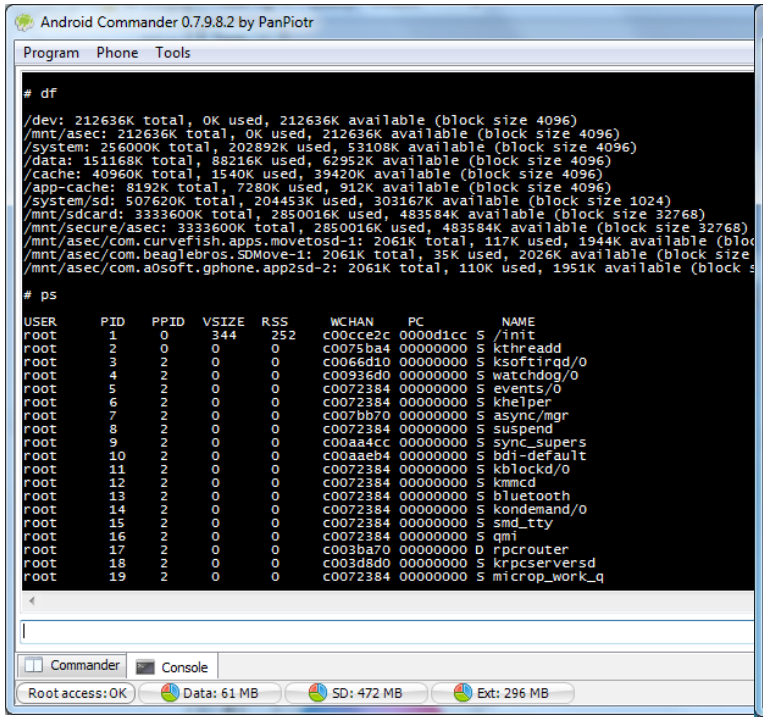
The ProGuard tool shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names. The result is a smaller sized .apk file that is more difficult to reverse engineer.

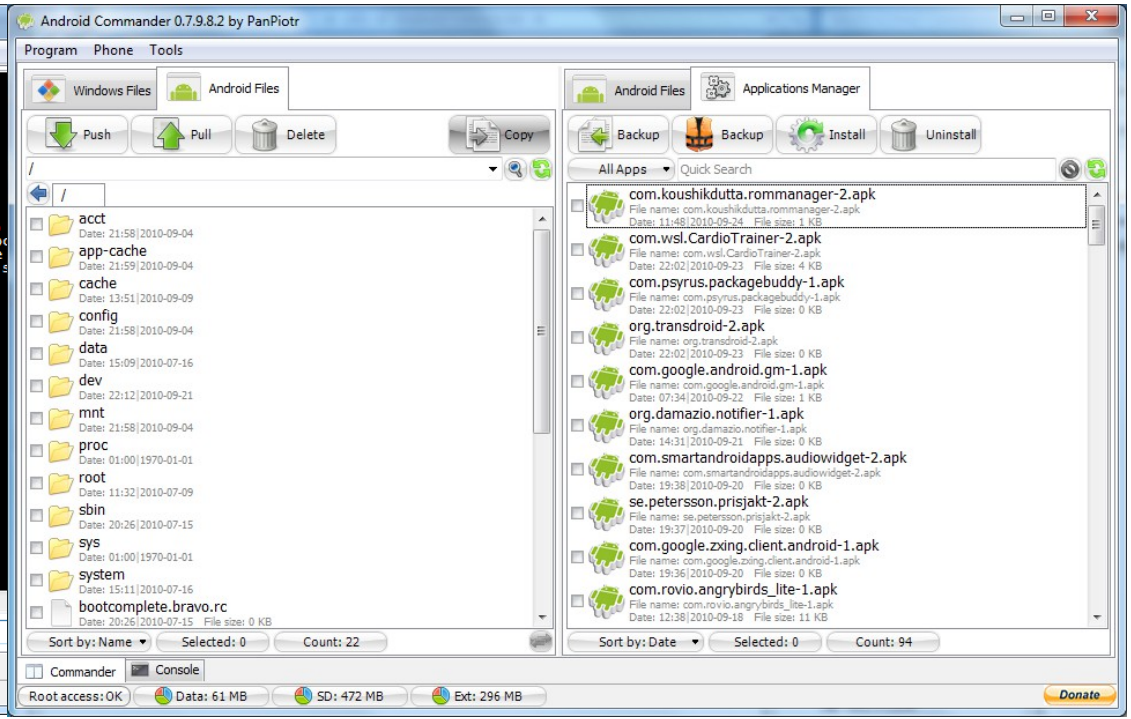
Enable ProGuard with the proguard.config property in the <project_root>/default.properties file.
proguard.config=proguard.cfg

http://developer.android.com/guide/developing/tools/proguard.html

Andra verktyg

- Android Commander, QtADB
- MyPhoneExplorer
  - Agent APK och PC program
- Reverse engineering verktyg
  - dex2jar: http://code.google.com/p/dex2jar/ i kombination med **Java Decompiler**
  - APK tool: http://code.google.com/p/android-apktool/



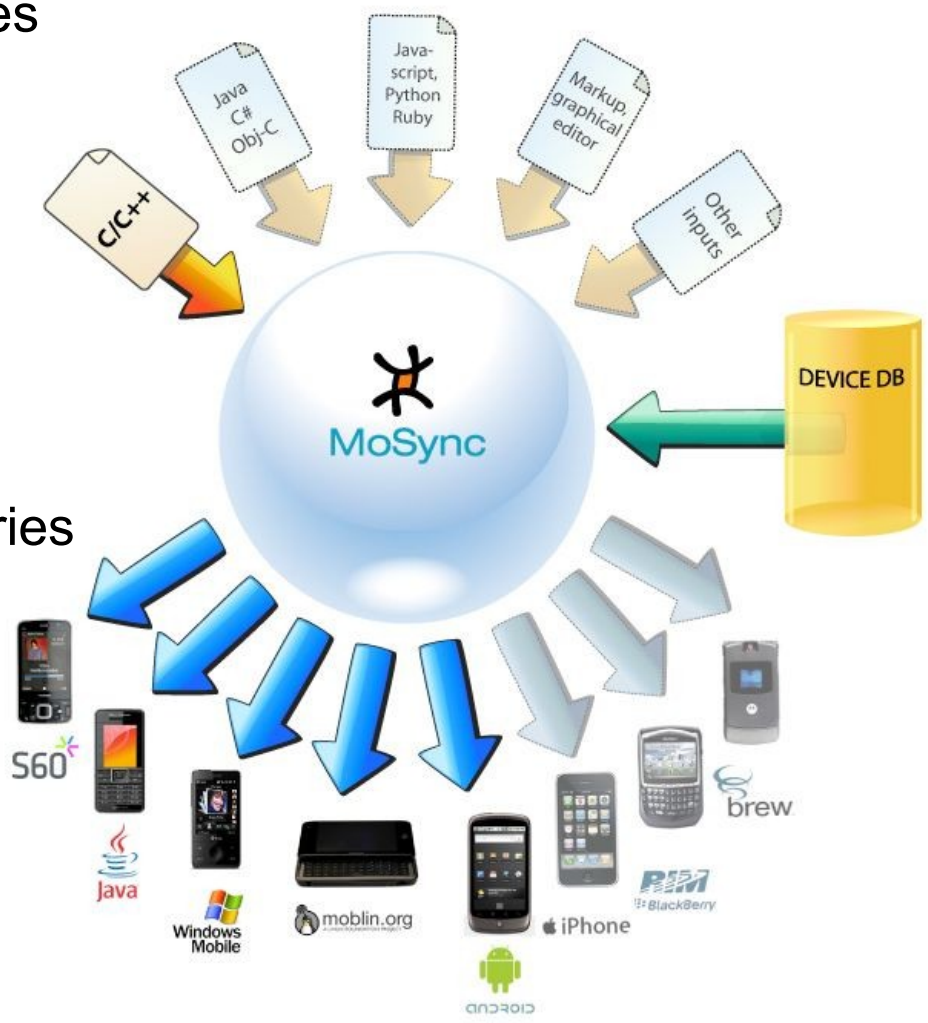


Libraries and other tips

- Android Libraries
  - More than 70 different libraries
  - Game engines
  - Charts
  - Social (Facebook, Twitter)
  - RPC (JSON/XML)
  - Contacts
  - NFC
  - …

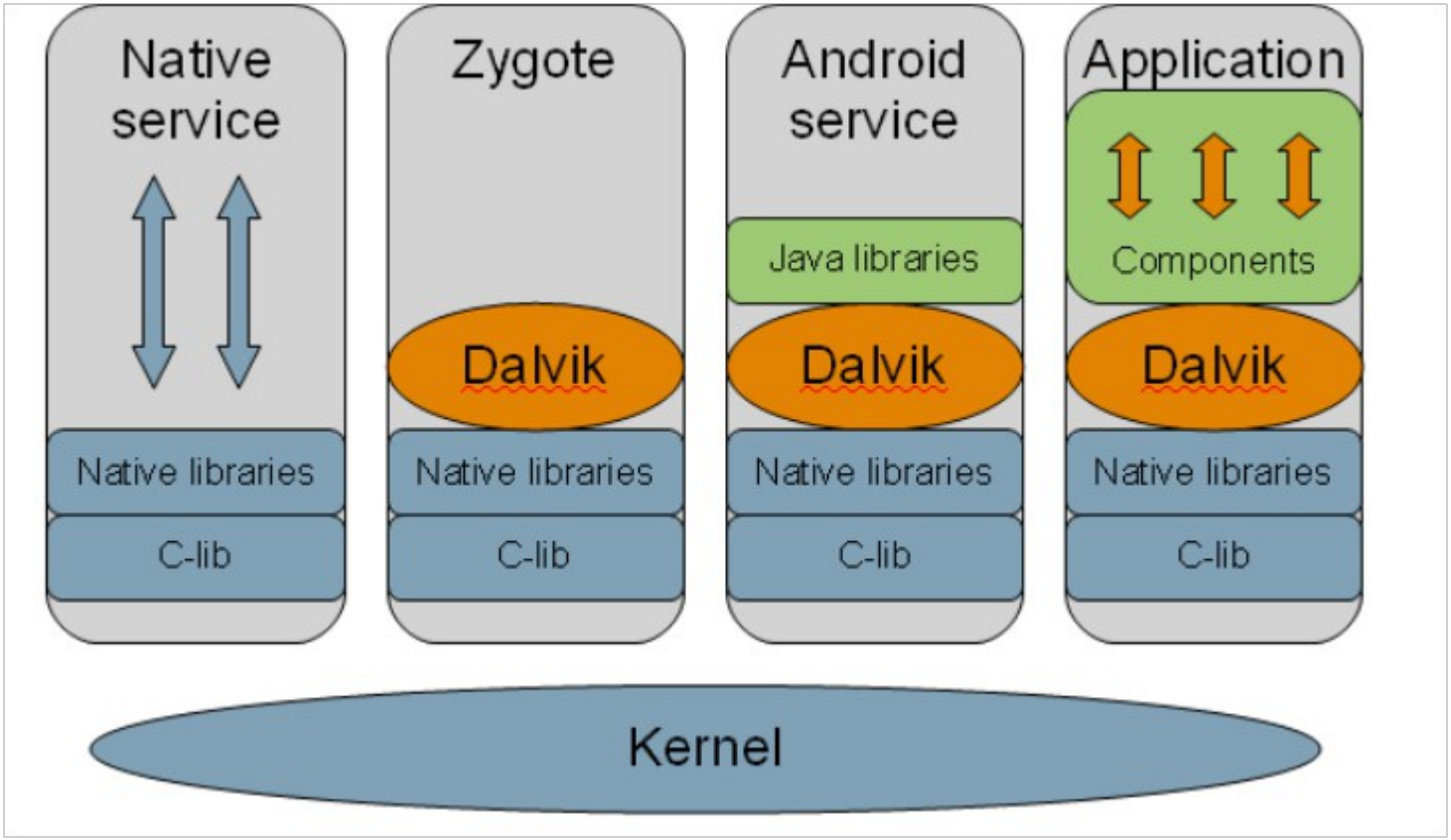
http://www.openintents.org/en/libraries

http://www.mosync.com



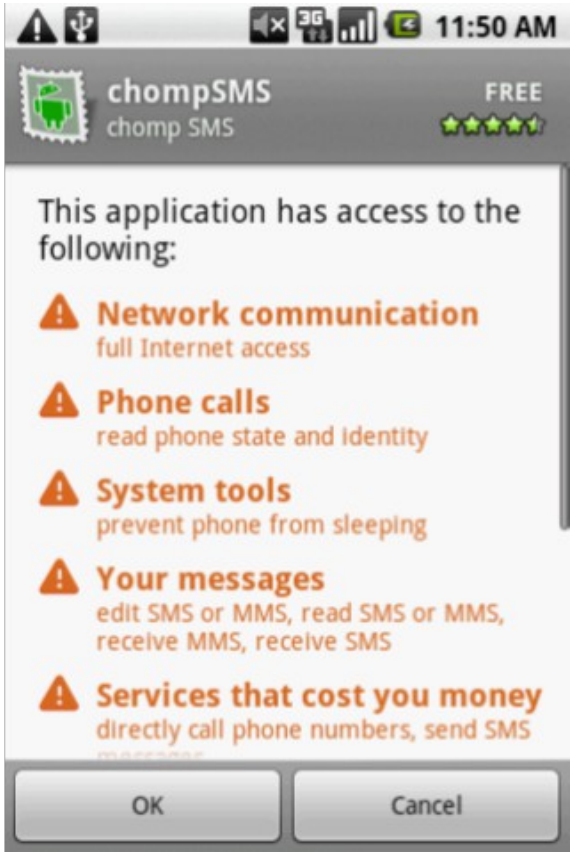
Android security approach 1

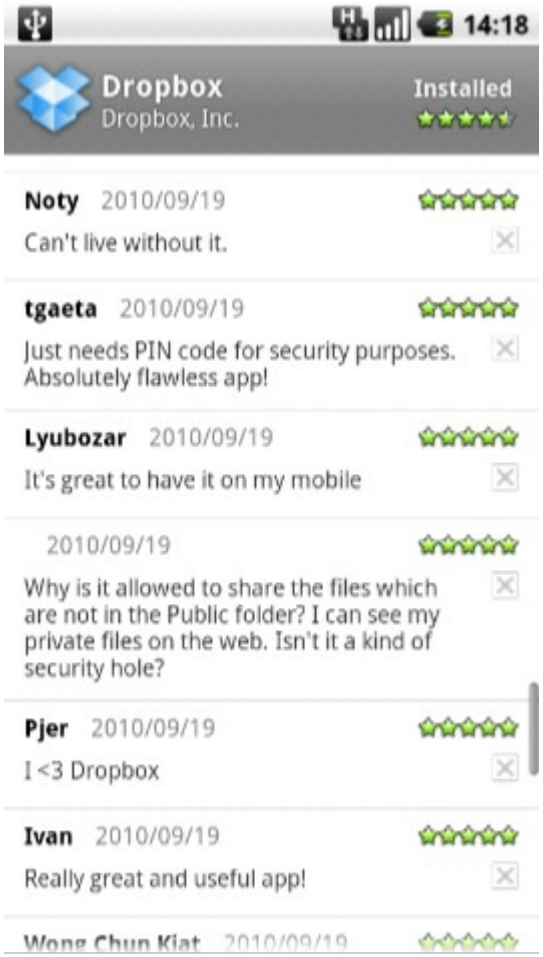
Strong base – The Linux level sandbox



Android security approach 2

Permissions and Community/Peer review
- checked at install time





Android - Room for improvement

- Fine grained permission system
  - Risk of next->-next->finish behaviour
  - Vulnerable to Trojans
- Basing some features on Linux file system rights
  - Very good for individual users with non-root devices
  - Does not provide enough piracy protection for developers, some devices will get rooted
  - SD-card uses FAT file system, no concept of ownership
- Update cycles
  - Kernel vulnerabilities are sometimes discovered and need fast updates
- No support for hardware assisted key stores and trusted computing
  - Useful for banking applications
  - Secure authentication