Fachhochschule Nordwestschweiz

Android Application Model

Content

- Activities

- Intent

- Tasks / Applications

- Lifecycle

- Processes and Thread

- Services

- Content Provider

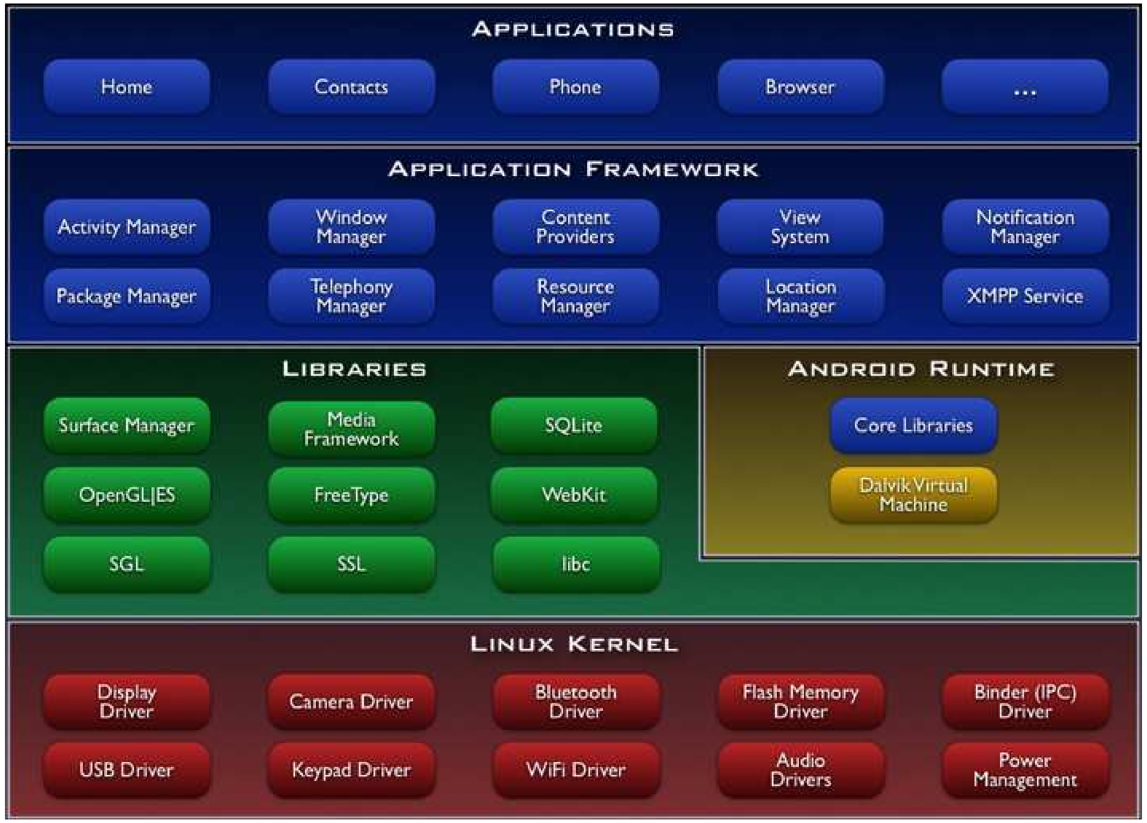
Dominik Gruntz IMVS

dominik.gruntz@fhnw.ch

Fachhochschule
Nordwestschweiz

Android Software Stack

- Java
- C/C++
- Kernel



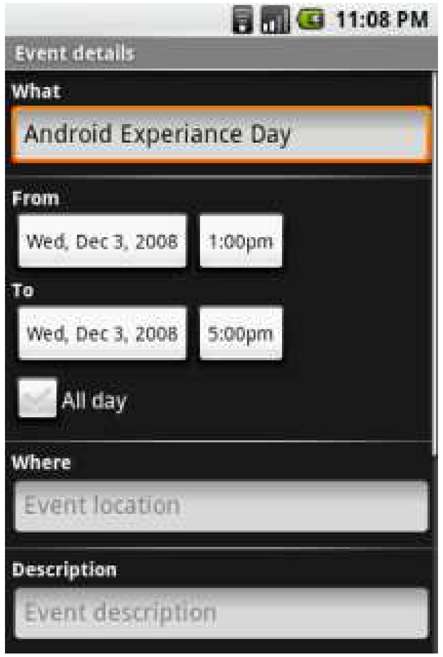
Fachhochschule
Nordwestschweiz

Android Building Blocks

- **Activity** **[User Interaction]**
  - UI component typically corresponding of one screen
    - E.g. Contacts: 3 activities: View contacts, Send message, Edit contact
- **Service** **[Service Provider]**
  - Background process without UI (e.g. mp3 player)
    - Messages can be sent from and to a service
- **Content Provider** **[Data Provider]**
  - Enables applications to share data
    - E.g. Contacts are provided to all applications
- **Broadcast Intent Receiver**
  - Responds to external events, can wake up your process
    - Phone rings, network activity established, time controlled

Activity

- **Activity is usually a single screen**
  - Implemented as a single class extending Activity
  - Displays user interface controls (views)
  - Reacts on user input / events
- **An application typically consists of several screens**
  - Each screen is implemented by one activity
  - Moving to the next screen means starting a new activity
  - An activity may return a result to the previous activity


11:08 PM

Event details

What: Android Experiance Day

From: Wed, Dec 3, 2008 1:00pm

To: Wed, Dec 3, 2008 5:00pm

All day

Where: Event location

Description: Event description

Intents and Intent Filters

- **Intent**
  - Intents are used to move from activity to activity
  - Intent describes what the application wants to do
  - Consists of
    - Action to be performed (MAIN / VIEW / EDIT / PICK / DELETE / …)
    - Data to act on (URI)

```
startActivity(new Intent(Intent.VIEW_ACTION,
   Uri.parse("http://www.fhnw.ch"));

startActivity(new Intent(Intent.VIEW_ACTION,
   Uri.parse("geo:47.480843,8.211293"));

startActivity(new Intent(Intent.EDIT_ACTION,
   Uri.parse("content://contacts/people/1"));
```

- *Comparable to HTTP protocol*

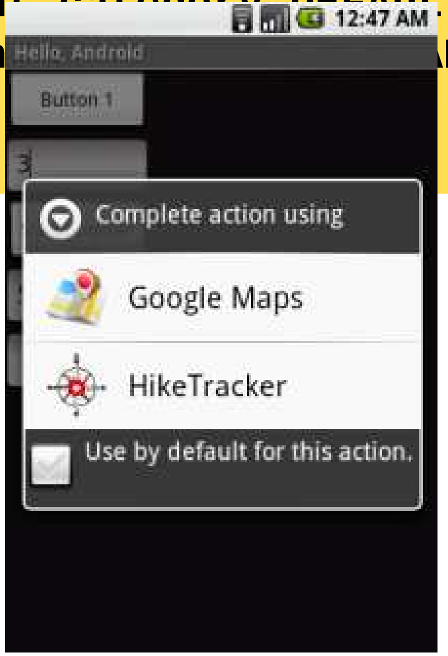
Fachhochschule
Nordwestschweiz

Intents and Intent Filters

- **Intent Filters**
  - Description of what intents an activity can handle
  - Activities publish their intent filters in a manifest file

```
<intent-filter android:priority="0">
   <action android:name="android.intent.action.VIEW"/>
   <category android:name="android.intent.category.DEFAULT"/>
   <category android:name="android.inten                  BLE"/>
   <data android:scheme="geo"/>
</intent-filter>
```

  - Upon invocation of startActivity(intent) the system looks at the intent filters of all installed applications a

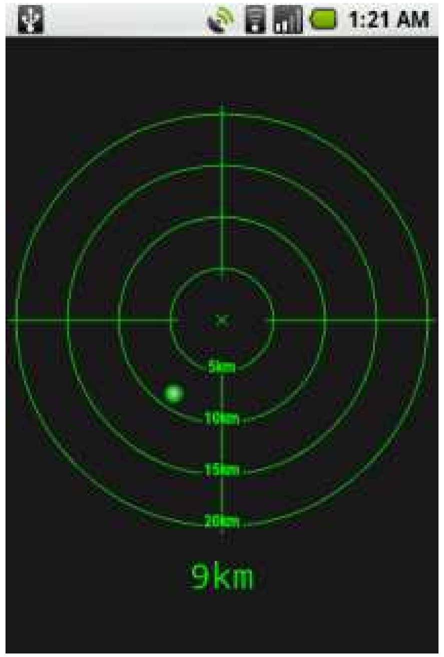


Fachhochschule
Nordwestschweiz

Android Component Model

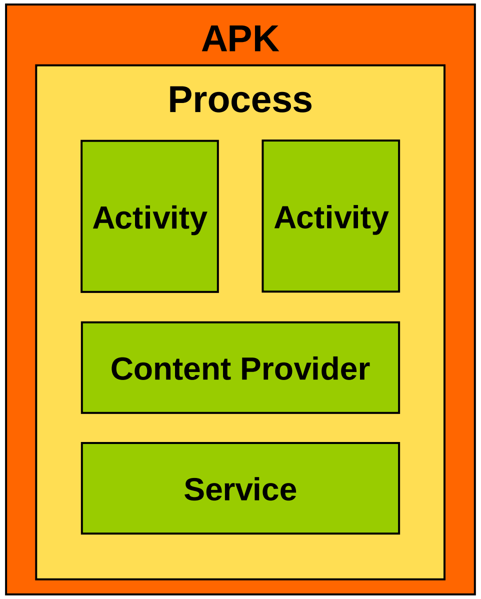
- **Component Software**
  - Activities can reuse functionality from other components simply by making a request in form of an Intent
  - Activities can be replaced at any time by a new Activity with an equivalent Intent Filter

```
Intent i = new Intent(
   "com.google.android.radar.SHOW_RADAR");
i.putExtra("latitude", 47.6f);
i.putExtra("longitude", 8.23f);
startActivity(i);
```



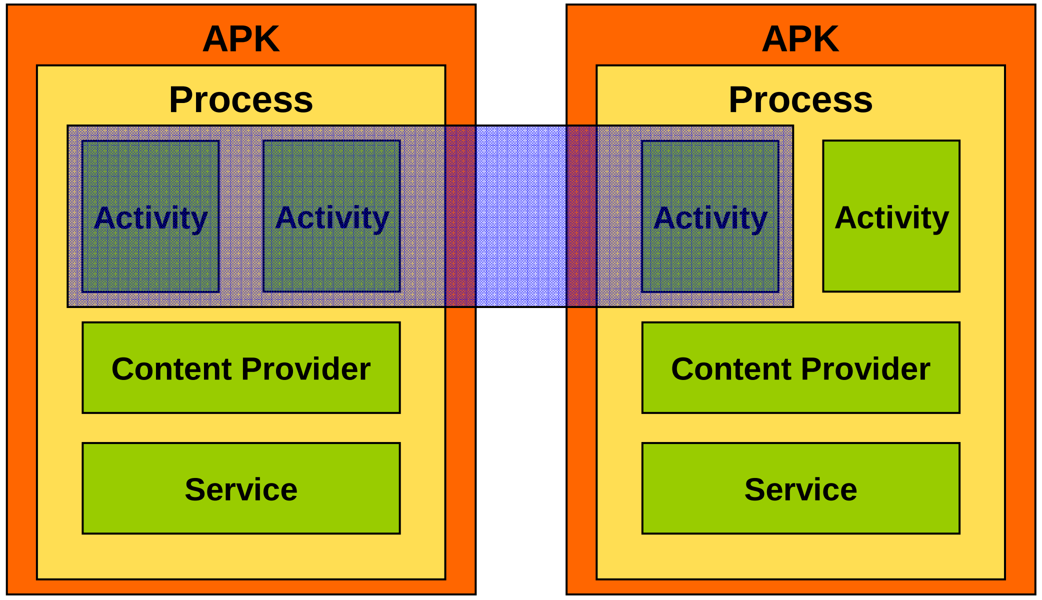
Android Component Model

- **Packaging: APK File (Android Package)**
  - Collection of components
  - Components share a set of resources
    - Preferences, Database, File space
  - Components share a Linux process
    - By default, one process per APK
  - APKs are isolated
    - Communication via Intents or AIDL
  - Every component has a managed lifecycle



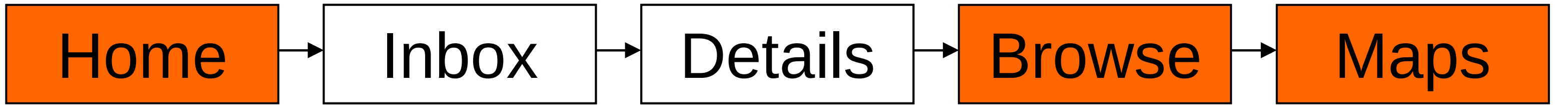
Task / Application / Process

- **Task (what users know as applications)**
  - Collection of related activities
  - Capable of spanning multiple processes
  - Associated with its own UI history stack



Task / Application / Process

- **Tasks**
  - Processes are started & stopped as needed
  - Processes may be killed to reclaim resources
  - Upon Invocation of another activity, the view state can be saved

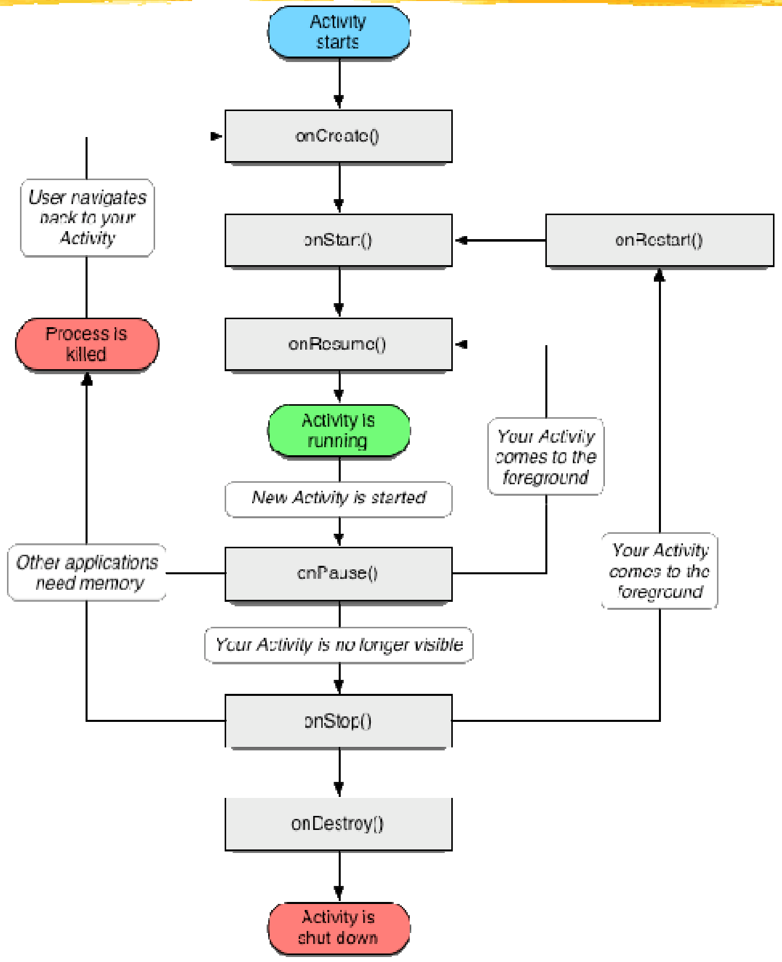


  - Comparable with EJBs stateful session beans (SFSB)
  - Each Android component has a managed lifecycle

Fachhochschule Nordwestschweiz

Activity Life Cycle

- **Active / Running**
  - Activity is in foreground
  - Activity has focus
- **Paused**
  - Still visible, partially overlaid
  - Lost focus
- **Stopped**
  - Activity is not visible
- **Dead**
  - Activity was terminated or was never started


Activity starts
onCreate()
User navigates back to your Activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
Your Activity comes to the foreground
New Activity is started
Other applications need memory
onPause()
Your Activity comes to the foreground
Your Activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Activity Life Cycle (1/2)

- **onCreate**
  - Called when activity is first created (with null parameter) or when activity was killed (called with a bundle)
  - Initialization of views
- **onRestart**
  - Called when activity was stopped only
- **onStart**
  - Activity becomes visible to user, animations could be started
- **onRestoreInstanceState**
  - Restore view state
- **onResume**
  - New activity is visible, TOS, camera might be used here

Activity Life Cycle (2/2)

- **onSaveInstanceState**
  - Save UI state of a complex dialog
    - => onCreate
    - => onRestoreInstanceState
  - If application is explicitly finished, this method is not called
  - Called before or after onPause
- **onPause**
  - Activity no longer TOS
  - New activity is not started until onPause returns
- **onStop**
  - Activity no longer visible
- **onDestroy**
  - Release resources; it is not guaranteed that this method is called

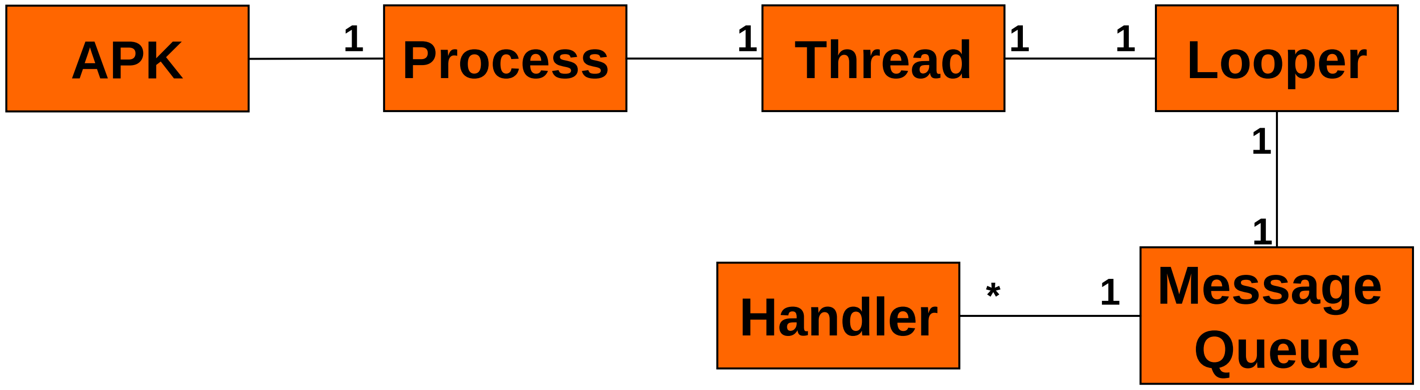
Activity Life Cycle Sample

- **Child Activity**
  - onCreate(null) -> onStart -> onResume()
  - onSaveInstanceState() -> onPause() -> onStop()
  - onRestart() -> onStart() -> onResume()
- **Transparent View**
  - onCreate(null) -> onStart -> onResume()
  - onSaveInstanceState() -> onPause()
  - onResume()
- **Turn Display**
  - onCreate(null) -> onStart -> onResume()
  - onSaveInstanceState() -> onPause() -> onStop() -> onDestroy() -> onCreate() -> onStart() -> onRestoreInstanceState() -> onResume()

Fachhochschule
Nordwestschweiz

Process / Thread

- **Threading Overview**
  - Each process has one thread (by default)
    - => Single Threaded Model
- **Threads and Loopers**
  - Each thread has a Looper to handle a message queue
  - Events from all components are interleaved into the looper/Queue

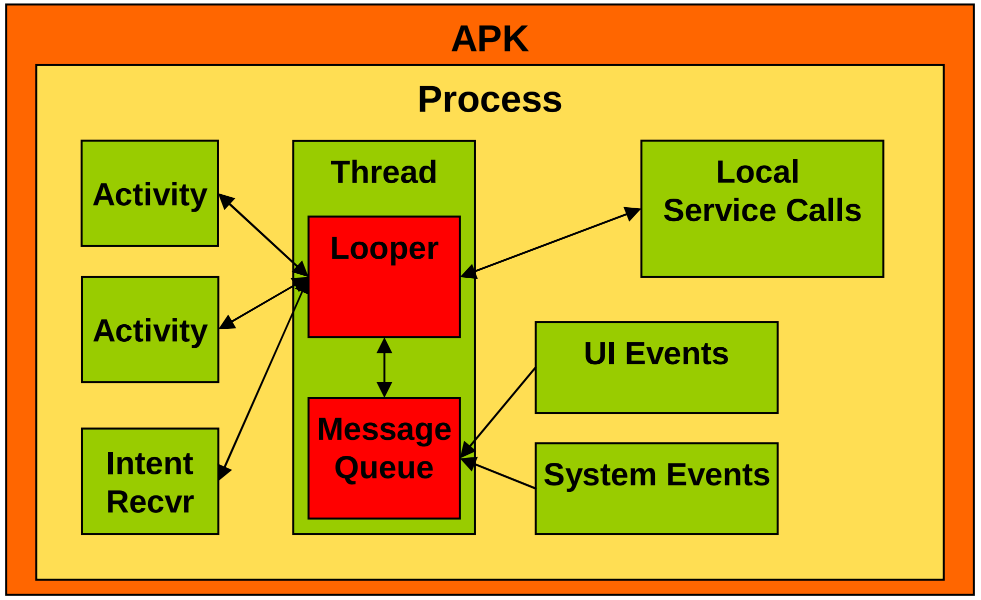


Process / Thread

- **ActivityThread**
  - Manages the main thread in an application process
  - Calls Looper.loop

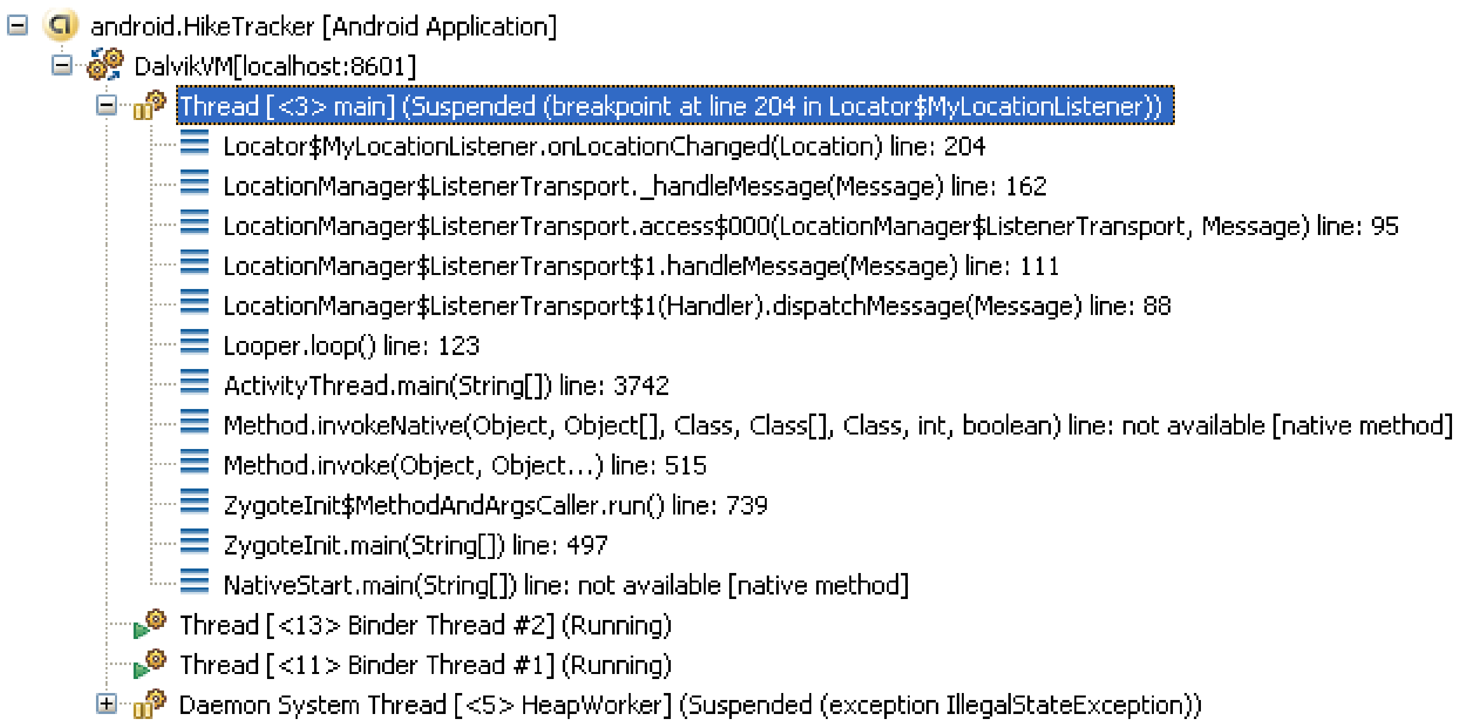
- **Looper.loop**

```
while(true){
   Message m=queue.next();
   // may block
   if(m!=null){
      m.target.dispatch-
               Message(m);
      m.recycle();
   }
}
```



Process / Thread

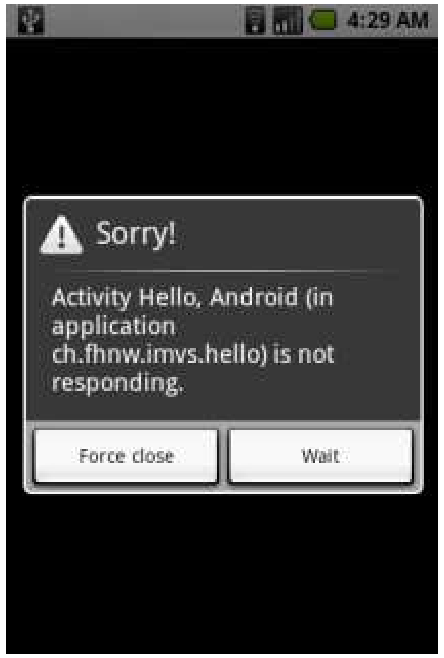
- **Location Update in HikeTracker**



```
android.HikeTracker [Android Application]
  DalvikVM[localhost:8601]
    Thread [<3> main] (Suspended (breakpoint at line 204 in Locator$MyLocationListener))
      Locator$MyLocationListener.onLocationChanged(Location) line: 204
      LocationManager$ListenerTransport._handleMessage(Message) line: 162
      LocationManager$ListenerTransport.access$000(LocationManager$ListenerTransport, Message) line: 95
      LocationManager$ListenerTransport$1.handleMessage(Message) line: 111
      LocationManager$ListenerTransport$1(Handler).dispatchMessage(Message) line: 88
      Looper.loop() line: 123
      ActivityThread.main(String[]) line: 3742
      Method.invokeNative(Object, Object[], Class, Class[], Class, int, boolean) line: not available [native method]
      Method.invoke(Object, Object...) line: 515
      ZygoteInit$MethodAndArgsCaller.run() line: 739
      ZygoteInit.main(String[]) line: 497
      NativeStart.main(String[]) line: not available [native method]
    Thread [<13> Binder Thread #2] (Running)
    Thread [<11> Binder Thread #1] (Running)
    Daemon System Thread [<5> HeapWorker] (Suspended (exception IllegalStateException))
```

Process / Thread

- **Inactive Activities**
  - If an activity does not consume events, the system assumes that the activity has a problem

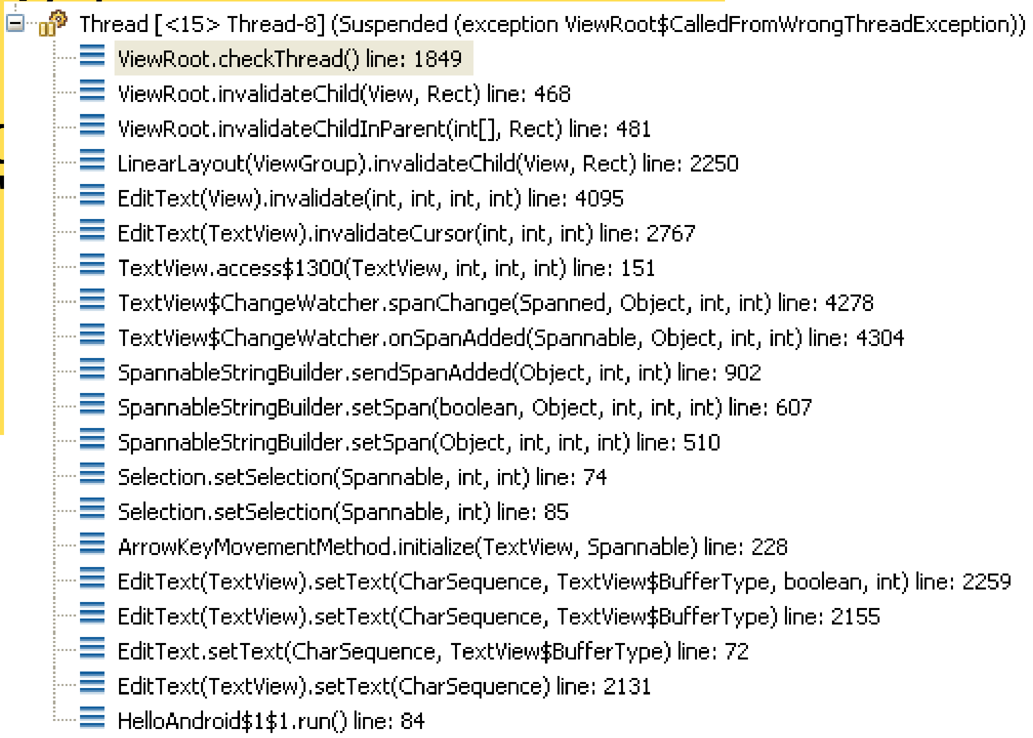


Sorry!

Activity Hello, Android (in application ch.fhnw.imvs.hello) is not responding.

Dealing with Threads

```
button.setOnClickListener(new OnClickListener() {
   @Override
   public void onClick(View v) {
      new Thread() {
         @Override
         public void run() {
            input1.setText("
         }
      }.start();
   }
});
```



```
Thread [<15> Thread-8] (Suspended (exception ViewRoot$CalledFromWrongThreadException))
    ViewRoot.checkThread() line: 1849
    ViewRoot.invalidateChild(View, Rect) line: 468
    ViewRoot.invalidateChildInParent(int[], Rect) line: 481
    LinearLayout(ViewGroup).invalidateChild(View, Rect) line: 2250
    EditText(View).invalidate(int, int, int, int) line: 4095
    EditText(TextView).invalidateCursor(int, int, int) line: 2767
    TextView.access$1300(TextView, int, int, int) line: 151
    TextView$ChangeWatcher.spanChange(Spanned, Object, int, int) line: 4278
    TextView$ChangeWatcher.onSpanAdded(Spannable, Object, int, int) line: 4304
    SpannableStringBuilder.sendSpanAdded(Object, int, int) line: 902
    SpannableStringBuilder.setSpan(boolean, Object, int, int, int) line: 607
    SpannableStringBuilder.setSpan(Object, int, int, int) line: 510
    Selection.setSelection(Spannable, int, int) line: 74
    Selection.setSelection(Spannable, int) line: 85
    ArrowKeyMovementMethod.initialize(TextView, Spannable) line: 228
    EditText(TextView).setText(CharSequence, TextView$BufferType, boolean, int) line: 2259
    EditText(TextView).setText(CharSequence, TextView$BufferType) line: 2155
    EditText.setText(CharSequence, TextView$BufferType) line: 72
    EditText(TextView).setText(CharSequence) line: 2131
    HelloAndroid$1$1.run() line: 84
```

- **checkRoot**
  - Compares current thread with thread which created the view

Dealing with Threads

- **Activity.runOnUiThread(Runnable)**
  - Runs the specified action on the UI thread, i.e. the action is posted into the event queue of the UI thread

- **Handler**
  - Associated with a thread and its message queue
  - Used to add messages in the message queue

▪ sendMessage	postRunnable
▪ sendMessageAtFrontOfQueue	postAtFrontOfQueue
▪ sendMessageAtTime	postAtTime
▪ sendMessageDelayed	postDelayed

  - Used to handle the request (called by associated thread)

Fachhochschule
Nordwestschweiz

Process & Security

- **Security Model**
  - Each application runs in its own process
    - Has its own unique Linux User ID
    - Each application has access to its own data

```
USER     PID   PPID  VSIZE RSS   WCHAN    PC         NAME
root     23    1      69508 18668 c008be9c afe0b874 S zygote
radio    87    23    100940 16320 ffffffff afe0c824 S com.android.phone
app_2    92    23    101792 17900 ffffffff afe0c824 S android.process.acore
app_14   120   23     93772 11444 ffffffff afe0c824 S com.google.process.gapps
app_8    158   23    100088 11860 ffffffff afe0c824 S com.android.mms
app_21   160   23     99740 13064 ffffffff afe0c824 S ch.fhnw.imvs.hello
app_0    175   23     90580 11116 ffffffff afe0c824 S com.android.alarmclock
app_3    183   23     94784 12080 ffffffff afe0c824 S android.process.media
```

  - Other resources are only available by defined interfaces

Services	[exposes functionality]
Content Provider	[exposes data]

Service

- **Characteristics**
  - Execution of long running tasks and business logic outside an activity
    - E.g. a background task that has to download data periodically
  - Services can explicitly be started and stopped
  - Communication with service
    - In-process if service runs in same APK
    - Inter-Process Communication across APKs (AIDL)

Service Sample (1/2)

```
public class CounterService extends Service {
   private static final long UPDATE_INTERVAL = 1000;
   private Timer timer = new Timer();
   private static long counter = 0;

   private static CounterListener listener;
   public static void setCounterListener(CounterListener l) {
      listener = l;
   }

   public IBinder onBind(Intent intent) { return null; }
   public void onCreate() { super.onCreate(); startTimer(); }
   public void onDestroy(){ super.onDestroy(); stopTimer(); }
```

Service Sample (2/2)

```
   private void startTimer() {
      timer.scheduleAtFixedRate(new TimerTask() {
         public void run() {
            counter++;
            if (CounterService.listener != null) {
               CounterService.listener.
                                    counterValueChanged(counter);
            }
         }
      }, 0, UPDATE_INTERVAL);
   }

   private void stopTimer() { timer.cancel(); }
}
```

Service Example: Invocation

- **Activity.onCreate: register call-back interface**

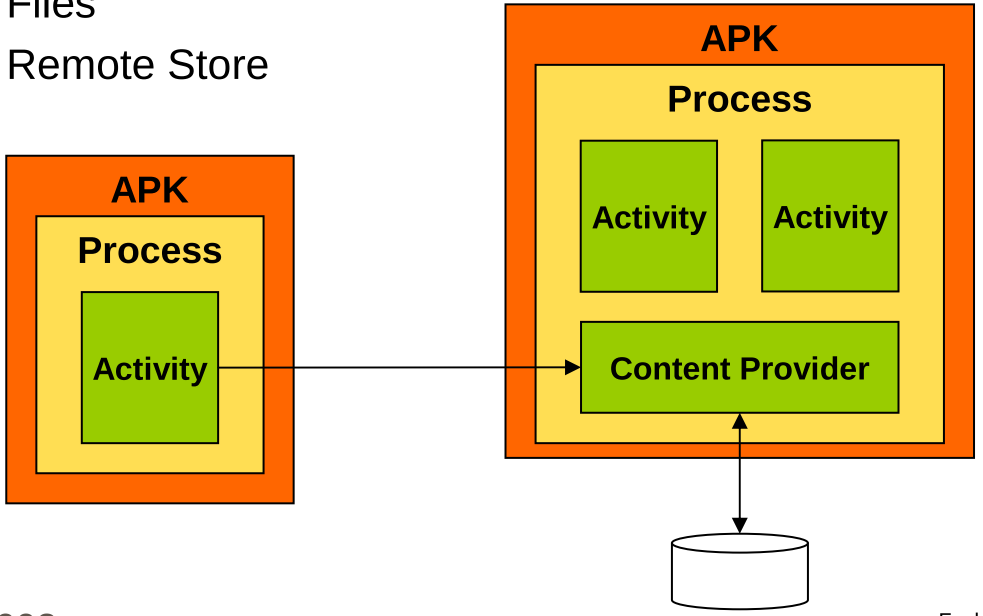
```
CounterService.setCounterListener(new CounterListener() {
   public void counterValueChanged(final long value) {
      HelloAndroid.this.runOnUiThread(new Runnable() {
         public void run() {
            input.setText("" + value);
         }
      });
   }
});
```

- **Start/Stop service**

```
startService(new Intent(this, CounterService.class));
stopService(new Intent(this, CounterService.class));
```

Content Provider

- **Content Provider**
  - The only way to share data between Android Packages
  - Implements a standard set of methods to provide access to data
  - Any form of storage can be used
    - SQLite DB
    - Files
    - Remote Store



Content Provider

- **Content Provider Interface**

```
abstract class ContentProvider {
   public Cursor query(Uri uri, String[] projection,
        String selection, String[] selectionArgs,
        String sortOrder);

   public Uri insert(Uri uri, ContentValues values)
   public int delete(Uri uri, String selection,
        String[] selectionArgs);

   String getType(Uri uri);

   public int update(ContentURI uri,
        ContentValues values, String selection,
        String[] selectionArgs)
}
```

Fachhochschule Nordwestschweiz

Content Provider

- **Access to Data Providers**
  - Comparable to a DB Access (Cursor)
  - Identification via URI
    - content://contacts/phones

```
String[] projections = new String[]{
                "number", "name", "_id", "photo"};

Cursor cur = managedQuery(
   new ContentURI("content://contacts/phones"),
   projections, null, "name ASC");
```

Fachhochschule
Nordwestschweiz

Content Provider Sample

```
public class Provider extends android.content.ContentProvider {
   public static final android.net.Uri CONTENT_URI =
        Uri.parse("content://ch.fhnw.imvs.fibonacci/numbers");

   public static final String ID    = BaseColumns._ID;
   public static final String VALUE = "value";

   public boolean onCreate() { return true; }

   public Cursor query(Uri uri, String[] projection,
      String selection, String[] selectionArgs, String sortOrder){

      MatrixCursor c =new MatrixCursor(new String[]{ID,VALUE},10);
      for(int i=0; i<10; i++)
         c.addRow(new Object[]{i, getNumber(i)});
      return c;
   }
```

Summary

- **Scalability**
  - Model well suited for mobile devices
    - Reduced memory
    - Phone calls have higher priority than other applications

- **Component Model**
  - Interesting programming model
  - Existing activities may be reused / extended