



Einführung in Android

Niko Fink, Marco Ziegaus

1. Mai 2015

- ▶ weltweiter Marktanteil: 85 %
- ▶ Linux-Kernel
- ▶ open source
- ▶ Java (Logik) & XML (GUI, ressources, config)

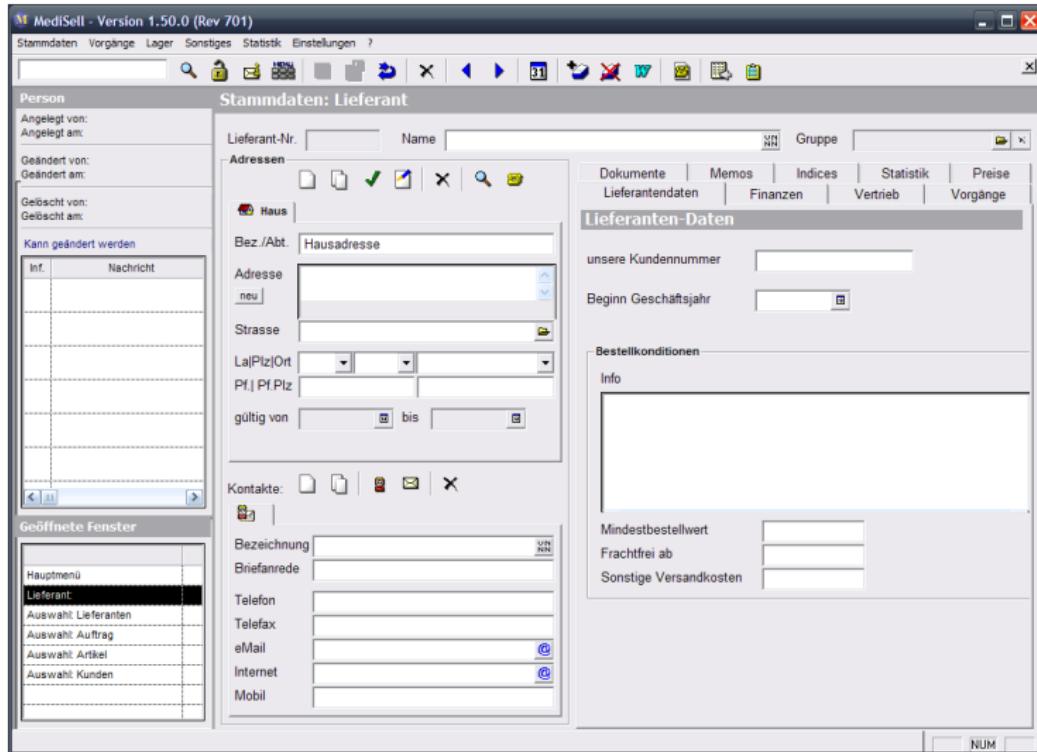
- ▶ 2.3 (Gingerbread)
- ▶ 3.0 (Honeycomb)
- ▶ 4.0 (Ice Cream Sandwich)
- ▶ 5.1 (Lollipop) → aktuell

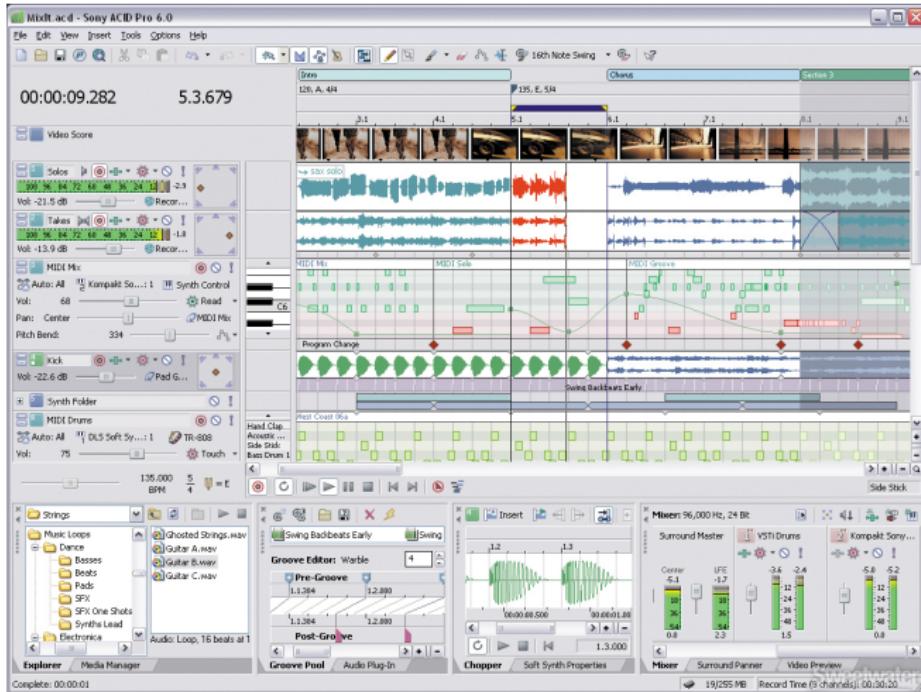
Wetter am aktuellen Ort

- ▶ GUI
- ▶ GPS
- ▶ Internet (openweathermap.org)

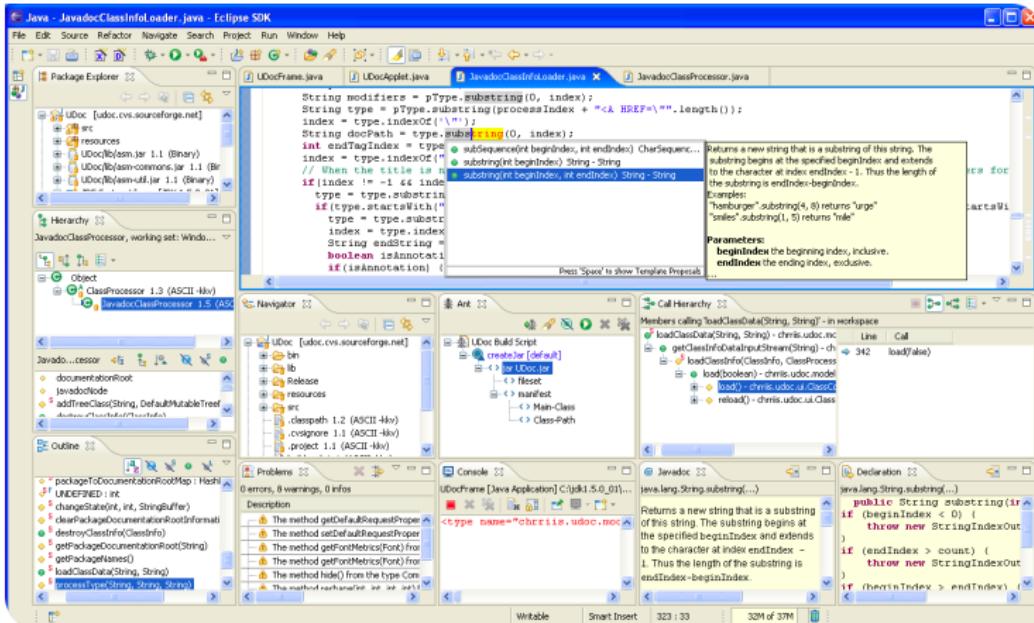
⇒ reduziert auf Minimal-Anforderungen



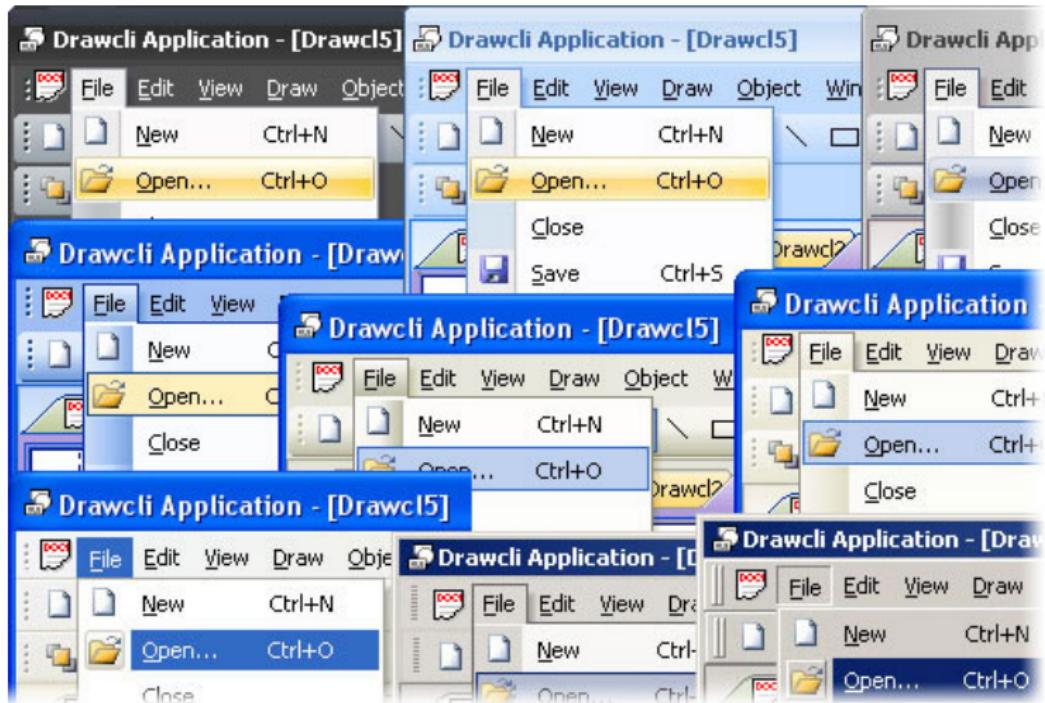




<http://www.superdownloads.com.br/imagens/screenshots/3/0/30730,0.jpg>



<http://www.pouet.net/topic.php?which=9700>



<http://www.mastermagazine.info/termino/wp-content/uploads/GUI.jpg>

- ▶ kleiner Bildschirm

- ▶ kleiner Bildschirm
- ▶ Touch-Bedienung

- ▶ kleiner Bildschirm
- ▶ Touch-Bedienung
- ▶ verschiedene Größen, Auflösungen und Formate

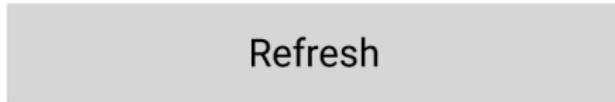
- ▶ kleiner Bildschirm
- ▶ Touch-Bedienung
- ▶ verschiedene Größen, Auflösungen und Formate
 ⇒ Größenangaben in px oder cm ungeeignet

- ▶ kleiner Bildschirm
- ▶ Touch-Bedienung
- ▶ verschiedene Größen, Auflösungen und Formate
 ⇒ Größenangaben in px oder cm ungeeignet
- ▶ Drehen des Bildschirms

- ▶ kleiner Bildschirm
 - ▶ Touch-Bedienung
 - ▶ verschiedene Größen, Auflösungen und Formate
 ⇒ Größenangaben in px oder cm ungeeignet
 - ▶ Drehen des Bildschirms
 - ▶ Anwendungskontext (draußen, im Gehen, etc.)
 - ▶ ...
- ⇒ abstrakte Beschreibung der GUI in XML

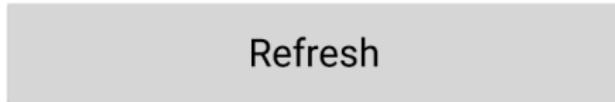
```
<Button  
    android:id="@+id/buttonRefresh"  
    android:layout_width="match_parent"  
    android:layout_height="60dp"  
    android:textSize="20sp"  
    android:text="Refresh" />
```

```
<Button  
    android:id="@+id/buttonRefresh"  
    android:layout_width="match_parent"  
    android:layout_height="60dp"  
    android:textSize="20sp"  
    android:text="Refresh" />
```



Refresh

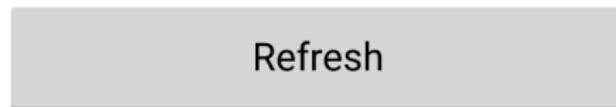
```
<Button  
    android:id="@+id/buttonRefresh"  
    android:layout_width="match_parent"  
    android:layout_height="60dp"  
    android:textSize="20sp"  
    android:text="Refresh" />
```



Refresh

- ▶ 60dp → density-independent pixels
⇒ 1 Pixel auf 160 dpi Screen

```
<Button  
    android:id="@+id/buttonRefresh"  
    android:layout_width="match_parent"  
    android:layout_height="60dp"  
    android:textSize="20sp"  
    android:text="Refresh" />
```



- ▶ 60dp → density-independent pixels
⇒ 1 Pixel auf 160 dpi Screen
- ▶ 20sp → scale-independent pixels
⇒ persönliche Skalierung des Users

```
<TextView  
    android:id="@+id/textViewTemperature"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textSize="50sp"  
    android:text="23 C" />
```

```
<TextView  
    android:id="@+id/textViewTemperature"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textSize="50sp"  
    android:text="23 °C" />
```

23 °C

```
<TextView  
    android:id="@+id/textViewTemperature"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textSize="50sp"  
    android:text="23 °C" />
```

23 °C

- ▶ wrap_content → Breite/Höhe passt auf Inhalt

```
<TextView  
    android:id="@+id/textViewTemperature"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textSize="50sp"  
    android:text="23 °C" />
```

23 °C

- ▶ wrap_content → Breite/Höhe passt auf Inhalt
- ▶ match_parent → Breite/Höhe passt auf Rahmen

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical" >  
  
    <TextView  
        ... />  
    <TextView  
        ... />  
  
</LinearLayout>
```

```
<RelativeLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
<TextView  
    android:id="@+id/textView1"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true"  
    ... />  
<TextView  
    android:id="@+id/textView2"  
    android:layout_below="@+id/textView1"  
    android:layout_centerHorizontal="true"  
    ... />  
</RelativeLayout>
```

```
<TableLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
    <TableRow>  
        <TextView  
            ... />  
        <TextView  
            ... />  
    </TableRow>  
  
    <TableRow> ... </TableRow>  
  
</TableLayout>
```

```
// define the layout for this activity
setContentView(R.layout.activity_main);

// get a reference to the view elements
buttonRefresh = (Button)
    findViewById(R.id.buttonRefresh);
textViewTemperature = (TextView)
    findViewById(R.id.textViewTemperature);
```

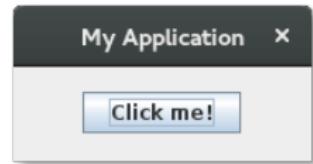
```
// manipulate the UI  
textViewTemperature.setText("30")  
  
//react to UI actions  
buttonRefresh.setOnClickListener(  
    new View.OnClickListener() {  
        public void onClick(View v) {  
            //refresh  
        }  
    } );
```

```
public static void main(String[] args) {  
    JFrame myFrame = new JFrame();  
    myFrame.addComponent(  
        new JButton("Click me!"));  
    myFrame.show();  
    ...  
}
```

Einschub: Swing Button App



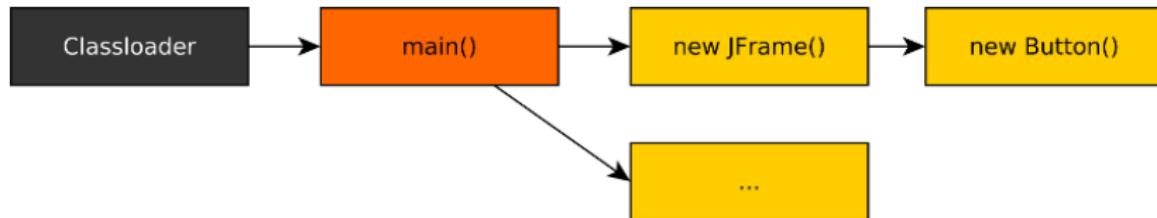
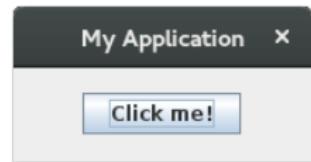
```
public static void main(String[] args) {  
    JFrame myFrame = new JFrame();  
    myFrame.addComponent(  
        new JButton("Click me!"));  
    myFrame.show();  
    ...  
}
```



Einschub: Swing Button App



```
public static void main(String[] args) {  
    JFrame myFrame = new JFrame();  
    myFrame.addComponent(  
        new JButton("Click me!"));  
    myFrame.show();  
    ...  
}
```



- ▶ nur ein Fenster

- ▶ nur ein Fenster
- ▶ weniger Prozessor- und Akkuleistung

- ▶ nur ein Fenster
- ▶ weniger Prozessor- und Akkuleistung
⇒ System muss mit Ressourcen haushalten

- ▶ nur ein Fenster
- ▶ weniger Prozessor- und Akkuleistung
⇒ System muss mit Ressourcen haushalten
- ▶ stärkere Systemintegration
(Notifications, Sensoren,...)

- ▶ nur ein Fenster
- ▶ weniger Prozessor- und Akkuleistung
⇒ System muss mit Ressourcen haushalten
- ▶ stärkere Systemintegration
(Notifications, Sensoren,...)

⇒ **Activities**

Container für graphische Anwendungen

- ▶ nur ein Fenster
- ▶ weniger Prozessor- und Akkuleistung
⇒ System muss mit Ressourcen haushalten
- ▶ stärkere Systemintegration
(Notifications, Sensoren,...)

⇒ Activities

Container für graphische Anwendungen

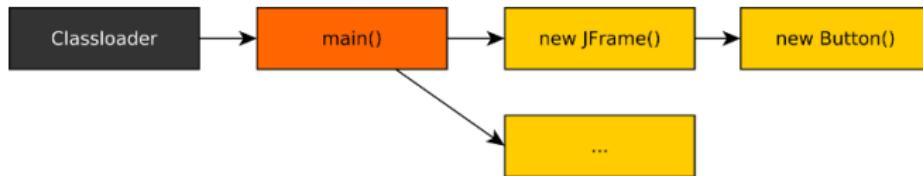


```
class MainActivity extends Activity {  
    @Override  
    public void onCreate(...) {  
        super.onCreate(...);  
        setContentView(R.id.layout);  
        //TODO configure Button  
    }  
}
```

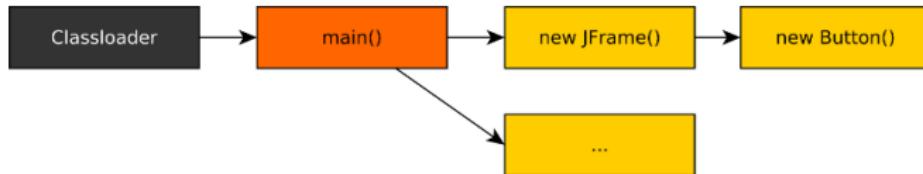
```
class MainActivity extends Activity {  
    @Override  
    public void onCreate(...) {  
        super.onCreate(...);  
        setContentView(R.id.layout);  
        //TODO configure Button  
    }  
}
```

```
<manifest>  
    <application label="ButtonApp">  
        <activity name=".MainActivity"  
                 label="My Activity">  
            ...  
        </...>
```

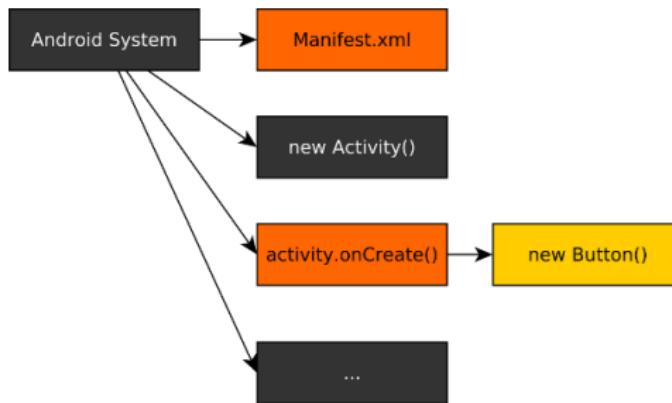
Swing



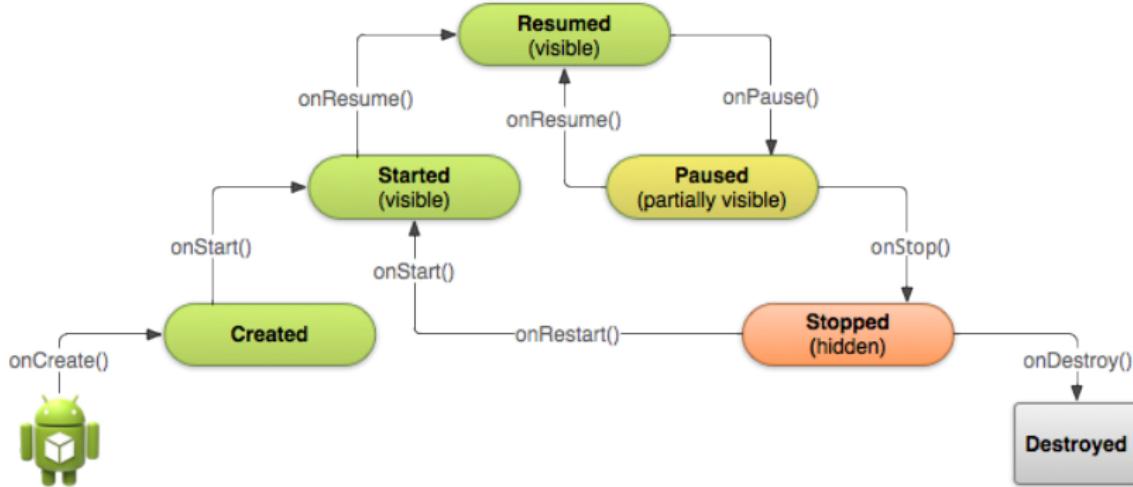
Swing



Android



Activity Lifecycle



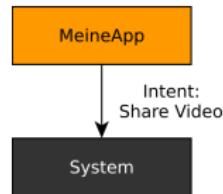
- ▶ Hauptkommunikationsmittel zwischen Komponenten

- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. “Öffne MeineAndereActivity” oder “Send Mail”

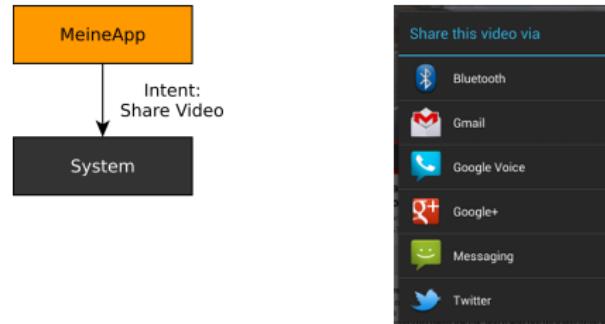
- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. "Öffne MeineAndereActivity" oder "Sende Mail"
- ▶ ... können noch weitere Daten enthalten
z.B. Titel der Mail, URL einer Website, ...

- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. "Öffne MeineAndereActivity" oder "Sende Mail"
- ▶ ... können noch weitere Daten enthalten
z.B. Titel der Mail, URL einer Website, ...
- ▶ ... werden vom System an das jeweilige Ziel weitergeleitet

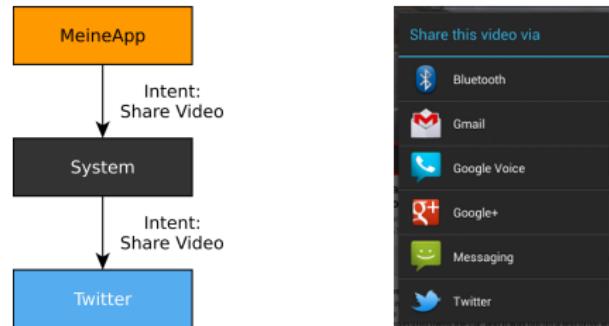
- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. “Öffne MeineAndereActivity” oder “Sende Mail”
- ▶ ... können noch weitere Daten enthalten
z.B. Titel der Mail, URL einer Website, ...
- ▶ ... werden vom System an das jeweilige Ziel weitergeleitet



- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. “Öffne MeineAndereActivity” oder “Sende Mail”
- ▶ ... können noch weitere Daten enthalten
z.B. Titel der Mail, URL einer Website, ...
- ▶ ... werden vom System an das jeweilige Ziel weitergeleitet



- ▶ Hauptkommunikationsmittel zwischen Komponenten
- ▶ ... enthalten explizites oder implizites Ziel
z.B. “Öffne MeineAndereActivity” oder “Sende Mail”
- ▶ ... können noch weitere Daten enthalten
z.B. Titel der Mail, URL einer Website, ...
- ▶ ... werden vom System an das jeweilige Ziel weitergeleitet



- ▶ Physikalisch:
 - ▶ GPS
 - ▶ Accelerometer/Beschleunigung
 - ▶ Helligkeit
 - ▶ Mikrofon
- ▶ Virtuell:
 - ▶ Schrittzähler
 - ▶ Lineare Beschleunigung
 - ▶ Lage/Rotation

```
LocationListener listener;
listener = new LocationListener() {
    public void onLocationChanged(Location loc){
        // fetch the weather using the provided
        // location
        fetchWeather(loc.getLatitude(),
                     loc.getLongitude());
    }

    public void onStatusChanged(...) {...}
    public void onProviderEnabled(...) {...}
    public void onProviderDisabled(...) {...}
};
```

```
// get a location manager
LocationManager locationManager =
    (LocationManager) getSystemService(
        LOCATION_SERVICE);

// request a single location update
locationManager.requestSingleUpdate(
    LocationManager.GPS_PROVIDER, listener,
    getMainLooper());
// getMainLooper() used for executing the
// callback (ignore for now)
```

```
// request continuous location updates each
// hour with a minimum distance of 1 km
locationManager.requestLocationUpdates(
    LocationManager.GPS_PROVIDER ,
    1 * 60 * 60 * 1000, 1000, listener);
```

```
// request continuous location updates each
// hour with a minimum distance of 1 km
locationManager.requestLocationUpdates(
    LocationManager.GPS_PROVIDER ,
    1 * 60 * 60 * 1000, 1000, listener);

// just get the last known location
Location location = locationManager.
    getLastKnownLocation(
        LocationManager.PASSIVE_PROVIDER);
```

Darf ich das?



App permissions

Storage

Modify or delete the contents of your USB storage

Phone calls

Read phone status and identity

Network communication

Full network access

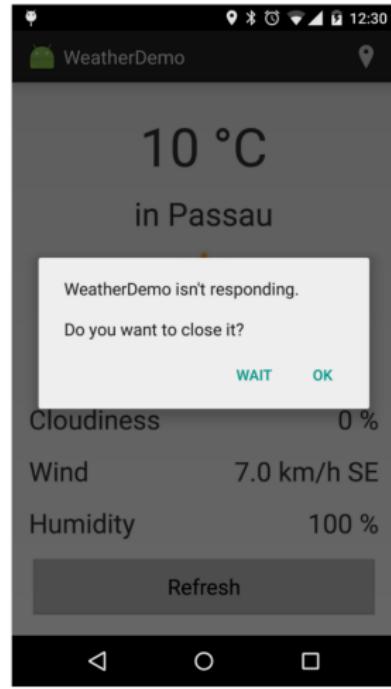
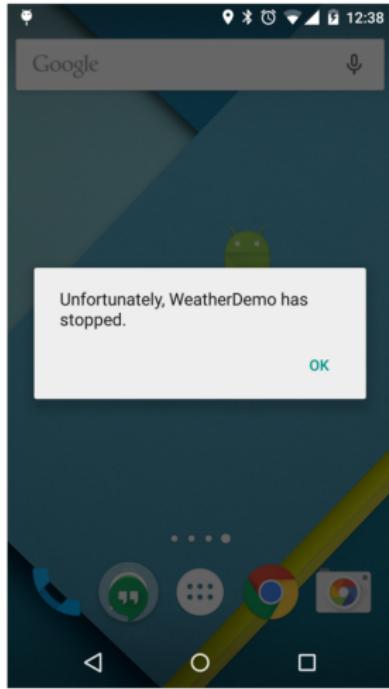
Your location

Approximate location (network-based)

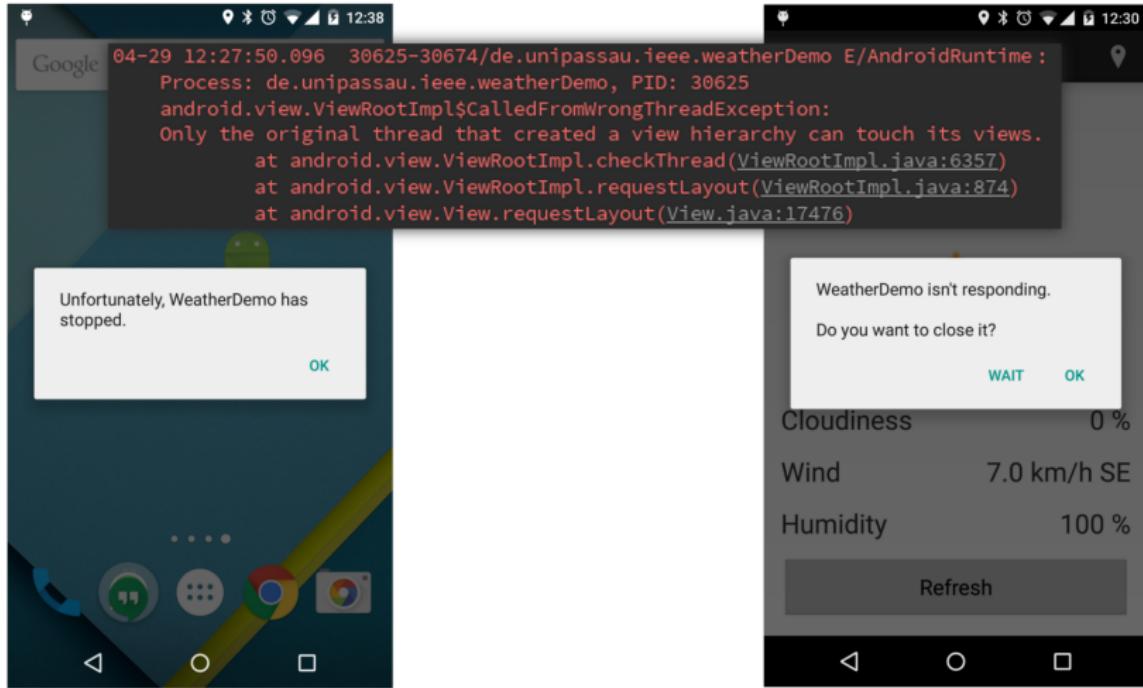
ACCEPT

```
<uses-permission android:name=
    "android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name=
    "android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name=
    "android.permission.INTERNET" />
```

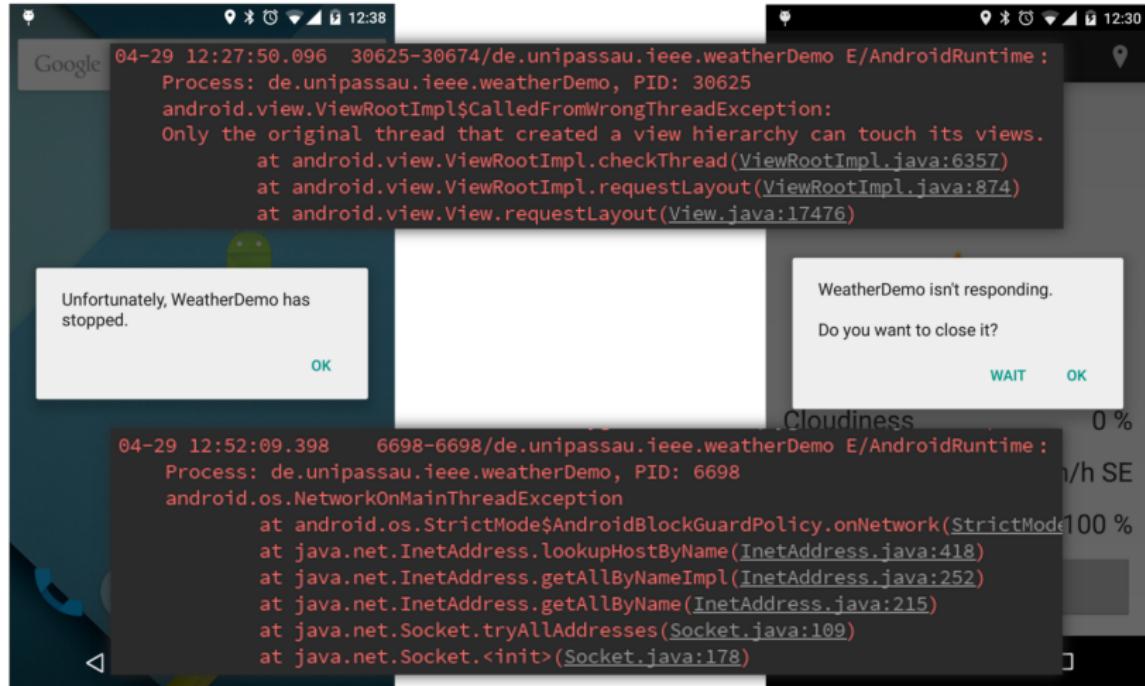
Threading Problems



Threading Problems



Threading Problems



- ▶ **Android UIs sind single-threaded!**

- ▶ **Android UIs sind single-threaded!**
- ▶ Tasks auf dem UI Thread blockieren die App
⇒ *App-Not-Responding* Dialog wird angezeigt

- ▶ **Android UIs sind single-threaded!**
- ▶ Tasks auf dem UI Thread blockieren die App
⇒ *App-Not-Responding* Dialog wird angezeigt
- ▶ IO ist im UI Thread verboten
⇒ sonst `NetworkOnMainThreadException`

- ▶ **Android UIs sind single-threaded!**
- ▶ Tasks auf dem UI Thread blockieren die App
⇒ *App-Not-Responding* Dialog wird angezeigt
- ▶ IO ist im UI Thread verboten
⇒ sonst `NetworkOnMainThreadException`
- ▶ UI kann nur vom UI Thread aus aktualisiert werden
⇒ sonst `CalledFromWrongThreadException`

- ▶ **Android UIs sind single-threaded!**
- ▶ Tasks auf dem UI Thread blockieren die App
⇒ *App-Not-Responding* Dialog wird angezeigt
- ▶ IO ist im UI Thread verboten
⇒ sonst `NetworkOnMainThreadException`
- ▶ UI kann nur vom UI Thread aus aktualisiert werden
⇒ sonst `CalledFromWrongThreadException`
- ▶ Threads, die von einer Activity aus gestartet wurden,
werden beendet sobald die Activity geschlossen wird

- ▶ **Android UIs sind single-threaded!**
- ▶ Tasks auf dem UI Thread blockieren die App
⇒ *App-Not-Responding* Dialog wird angezeigt
- ▶ IO ist im UI Thread verboten
⇒ sonst `NetworkOnMainThreadException`
- ▶ UI kann nur vom UI Thread aus aktualisiert werden
⇒ sonst `CalledFromWrongThreadException`
- ▶ Threads, die von einer Activity aus gestartet wurden,
werden beendet sobald die Activity geschlossen wird
- ▶ Activities (und damit auch alle Threads) werden neu erzeugt,
wenn der Bildschirm gedreht wird

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*

- ▶ Verwende AsyncTask

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*

- ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*
- ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread
 - ▶ `doInBackground(...)` asynchron

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*
- ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread
 - ▶ `doInBackground(...)` asynchron
 - ▶ Parameter und Rückgabewerte

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*
- ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread
 - ▶ `doInBackground(...)` asynchron
 - ▶ Parameter und Rückgabewerte
 - ▶ Fortschritt mit `onProgressUpdate(...)`

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*
- ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread
 - ▶ `doInBackground(...)` asynchron
 - ▶ Parameter und Rückgabewerte
 - ▶ Fortschritt mit `onProgressUpdate(...)`
 - ▶ Abbrechen mit `cancel()`

- ▶ Starte anderen Thread und aktualisiere UI mit
`Activity.runOnUiThread(Runnable)`
⇒ *Kommunikation mit UI trotzdem sehr aufwändig*
 - ▶ Verwende AsyncTask
 - ▶ `onPre/PostExecute()` auf dem UI Thread
 - ▶ `doInBackground(...)` asynchron
 - ▶ Parameter und Rückgabewerte
 - ▶ Fortschritt mit `onProgressUpdate(...)`
 - ▶ Abbrechen mit `cancel()`
- ⇒ *wird immernoch zusammen mit Activity beendet*

- ▶ Eigenständige Komponente neben Activities

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities
- ▶ Aufgabenbereiche
 - ▶ Dateiup- und download

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities
- ▶ Aufgabenbereiche
 - ▶ Dateiup- und download
 - ▶ Datensynchronisierung

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities
- ▶ Aufgabenbereiche
 - ▶ Dateiup- und download
 - ▶ Datensynchronisierung
 - ▶ Notifications

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities
- ▶ Aufgabenbereiche
 - ▶ Dateiup- und download
 - ▶ Datensynchronisierung
 - ▶ Notifications
 - ▶ Datenlogging

- ▶ Eigenständige Komponente neben Activities
- ▶ läuft über längere Zeit im Hintergrund
- ▶ asynchron und unabhängig von Activities
- ▶ Aufgabenbereiche
 - ▶ Dateiup- und download
 - ▶ Datensynchronisierung
 - ▶ Notifications
 - ▶ Datenlogging
 - ▶ Zeitgesteuerte Abläufe
 - ▶ ...

- ▶ **IntentService**

- ▶ Arbeitspakete in Form von Intents

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch
- ▶ Wertrückgabe in Form von Intents

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch
- ▶ Wertrückgabe in Form von Intents

► Started Service

- ▶ durch Intent(s) gestartet

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch
- ▶ Wertrückgabe in Form von Intents

► Started Service

- ▶ durch Intent(s) gestartet
- ▶ unbeschränkte Lebensdauer

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch
- ▶ Wertrückgabe in Form von Intents

► Started Service

- ▶ durch Intent(s) gestartet
- ▶ unbeschränkte Lebensdauer

► Bound Service

- ▶ hält Verbindung mit Activities, mit diesen direkte Kommunikation

► IntentService

- ▶ Arbeitspakete in Form von Intents
- ▶ arbeitet der Reihe nach ab und beendet sich automatisch
- ▶ Wertrückgabe in Form von Intents

► Started Service

- ▶ durch Intent(s) gestartet
- ▶ unbeschränkte Lebensdauer

► Bound Service

- ▶ hält Verbindung mit Activities, mit diesen direkte Kommunikation
- ▶ Lebenszeit an die der Activities gebunden

Vielen Dank für eure Aufmerksamkeit!

Source Code: github.com/N-Coder/android-talk-app