

# Android User Interface

## Android Smartphone Programming

University of Freiburg

**Matthias Keil**  
Institute for Computer Science  
Faculty of Engineering  
University of Freiburg

20. Oktober 2014



**UNI  
FREIBURG**



## 1 Android User Interface

## 2 Multi-Language Support

## 3 Summary





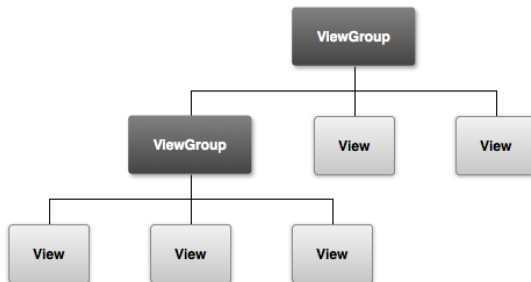
**Activity** Application component that provides a screen [1].

User interface of an activity is build using View and ViewGroup objects [5].

**View** Basis unit for user interface, base for subclasses called *widgets*.

**ViewGroup** Base for subclasses called *layouts*.





Android View Hierarchy containing ViewGroup objects as nodes and View objects as leafs.





- Can be defined in an **XML** layout file [7].
- Similar to HTML layout development.
- Each element is a View or ViewGroup object or a subclass of these.
- ViewGroup objects contain more Views or ViewGroup objects.



# Android User Interface

## Example XML layout

University of Freiburg



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.
   android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical" >
6 <TextView android:id="@+id/text"
7     android:layout_width="wrap_content"
8     android:layout_height="wrap_content"
9     android:text="Hello , I am a TextView" />
10 <Button android:id="@+id/button"
11     android:layout_width="wrap_content"
12     android:layout_height="wrap_content"
13     android:text="Hello , I am a Button" />
14 </LinearLayout>
```





- Subclass of `View`.
- Serves as interaction interface with user.
- Many fully implemented widgets available.
  - Examples: `Button`, `Checkbox`, `EditText` and many more.
  - Advanced Example `WebView`: Displays web pages and can use JavaScript [6].
- Own implementation enables full customization of behavior.





- Many ways to intercept events from user interaction.
- Approach for user interface events: Capture events from View objects the user interacts with [2].
- Two ways of implementation:
  - Overwrite existing callback method.
  - Define own event listener.
- Mostly used: Defining event listeners.





# Android User Interface

## Example: Overwriting Callback Method

University of Freiburg



UNI  
FREIBURG

```
1 public class MyActivity extends Activity {  
2     ...  
3     @Override  
4     public boolean onKeyDown (int keyCode,  
5         KeyEvent event) {  
6         // Do something.  
7     }  
}
```



# Android User Interface

Example: Defining own Event Listener

University of Freiburg



```
1 public class MyActivity extends Activity {
2     private OnClickListener myListener = new
        OnClickListener() {
3         public void onClick(View v) {
4             // Do something.
5         }
6     };
7
8     public void onCreate(Bundle state) {
9         ...
10        Button button = (Button)findViewById(R.id.
            myButton);
11        button.setOnClickListener(myListener);
12        ...
13    }
14 }
```





**Intent** Message to communicate between components. [3].

Can connect components in the same or in different applications.

Starts activities, background processes or notifies broadcast receivers.

**Broadcast Receiver** Can be registered to receive certain intents.

Example: Intent sent from system indicates incoming call and application stops playing music.





- Intent starts activity by specifying what action should be performed.
- Note: Activity only implicitly given though action.

```
1 Intent intent = new Intent(Intent.ACTION_DIAL,  
    Uri.parse("tel:5905-5635"));  
2 startActivity(intent);
```



- Step 1: Create broadcast receiver as a new class.

```
1 public class MyPhoneReceiver extends
    BroadcastReceiver {
2     @Override
3     public void onReceive(Context context, Intent
        intent) {
4         // Do something.
5     }
6 }
```



- Step 2: Extend *AndroidManifest.xml* to register broadcast receiver to intents.

```
1 <application ... >
2   <receiver android:name="MyPhoneReceiver" >
3     <intent-filter>
4       <action android:name="android.intent.
5         action.PHONE_STATE" >
6     </action>
7   </intent-filter>
8 </receiver>
9 </application>
```





- Done though localization: Switch language according to locale settings of the device [4].
- Helps reaching more users.
- Easy though separation of string resources and application code.
- Refer to string names in code and define strings in resource files.





- Default resources in *res/values/strings.xml* provides all strings used.
- Special language resource files like e.g. *res/values-de/strings.xml* provides adjusted strings.
- If no special resource file exists, default is used.







### ■ In Activity

```
1 tv = new TextView(this);  
2 tv.setText(R.string.example);
```

### ■ In *res/values/strings.xml*

```
1 <string name="example">Example</string>
```








### ■ In *res/values-de/strings.xml*

```
1 <string name="example">Beispiel</string>
```



- User interfaces of activities are build through *View* and *ViewGroup* objects.
- *ViewGroup* subclasses are *layouts* that group other *ViewGroup* or *View* objects.
- *View* subclasses are *widgets* for user interaction like button, checkbox and so on.
- Enabling user interaction is implemented by *capturing input events*.
- *Intents* are messages and can be received through broadcast receivers.
- Multi-language support is implemented through *resource files* for strings.



-  ANDROID DEVELOPERS.  
Activities.  
<http://developer.android.com/guide/topics/fundamentals/activities.html>.
-  ANDROID DEVELOPERS.  
Input Events.  
<http://developer.android.com/guide/topics/ui/ui-events.html>.
-  ANDROID DEVELOPERS.  
Intents and Intent Filters.  
<http://developer.android.com/guide/topics/intents/intents-filters.html>.
-  ANDROID DEVELOPERS.  
Localization.  
<http://developer.android.com/guide/topics/resources/localization.html>.
-  ANDROID DEVELOPERS.  
User Interface.  
<http://developer.android.com/guide/topics/ui/index.html>.
-  ANDROID DEVELOPERS.  
WebView.  
<http://developer.android.com/reference/android/webkit/WebView.html>.
-  ANDROID DEVELOPERS.  
XML Layouts.  
<http://developer.android.com/guide/topics/ui/declaring-layout.html>.

