EntwickeIn für Android OS

Am Beispiel der WAPLA
Wissenschaftliche **A**rbeitstechniken und **P**räsentation **L**ern **A**pplikation

Christian Kain

Kevin Kain

Wolfgang Kremser

Gregor Bankhamer

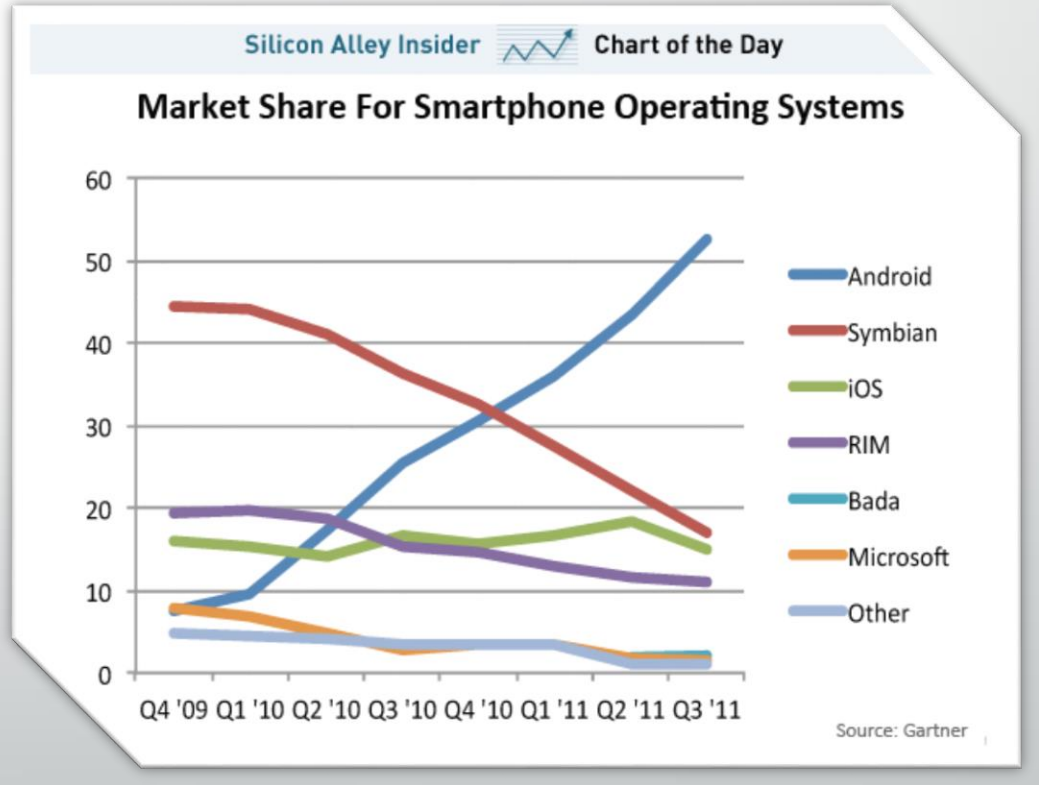
Warum Android?

Verbreitung von Android Geräten

- 48 Milliarden Apps wurden bis Mai 2013 installiert.
- Davon 2.5 Milliarden alleine im Monat April.
- Im 3. Quartal 2013 wurden geschätzte 211 Android Geräte weltweit verkauft.
- Das entspricht 80% des gesamten Smartphone Umsatzes.

Gefragt am Arbeitsmarkt

- Das rasante Wachstum der Plattform erzeugte großen Bedarf an Android-Entwicklern.
- Mehr und mehr Unternehmen wollen auf die Smartphones und Tablets ihrer Kunden und Mitarbeiter.
- Jeder User gibt heute 2.5 mal mehr für Apps aus als noch 2012.



Die Tools

„Können wir das schaffen?“

- Bob

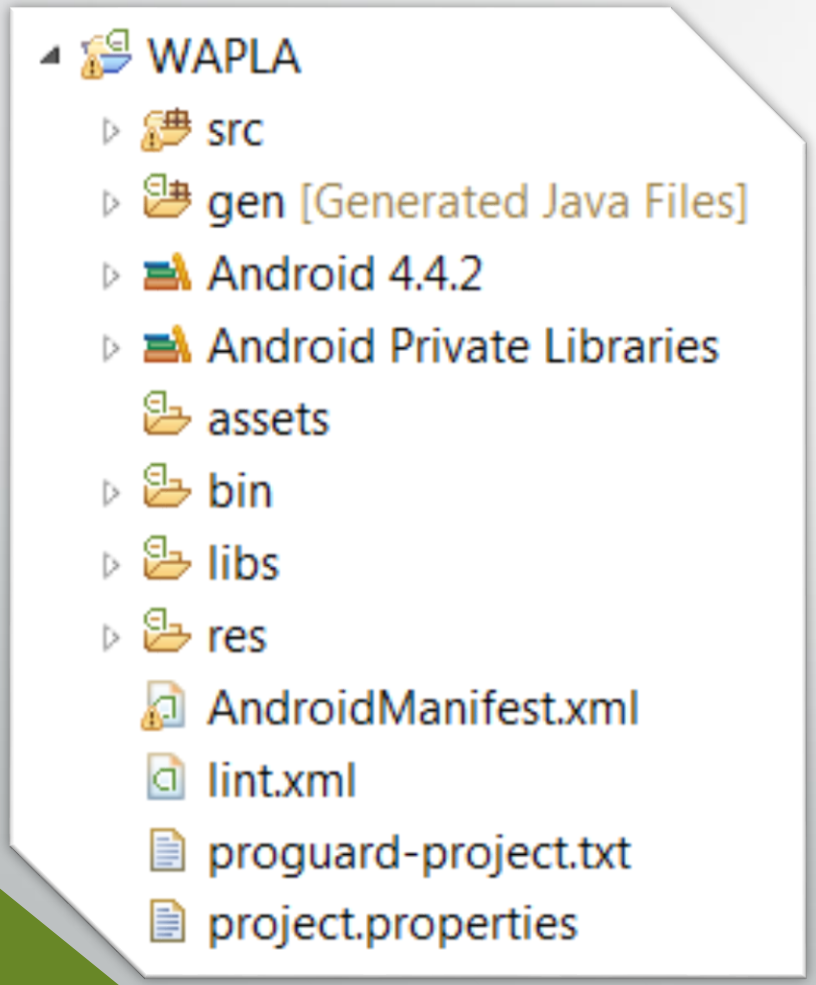
Was brauche ich zum Programmieren?

- Java
- Android Developer Tools (ADT)
- Empfehlenswert für Tests: Android-Device(s)

Was muss der Developer beachten?

- Große Anzahl an verschiedenen Android-Geräten im Umlauf
- die noch dazu auf verschiedenen Android Versionen laufen.
- Weniger Ressourcen als bei Desktop-PCs
  - Akku
  - Speicher, Rechenleistung
- Nutzerverhalten anders als „am Schreibtisch".
- Positionierung der Elemente

Grundlegende Ordnerstruktur



src	Source Code (.java)
gen	Vom Android Builder automatisch generierte Dateien (.java)
assets	Raum für Zusatzdateien (z.B. Texturen)
bin	Kompilierte Dateien
libs	Libraries
res	Ressourcen, die von der R-Klasse automatische verwaltet werden (XML, Bilder, Icons)
Manifest	Beinhaltet Metadaten und Aufbau der App

Aufbau einer App

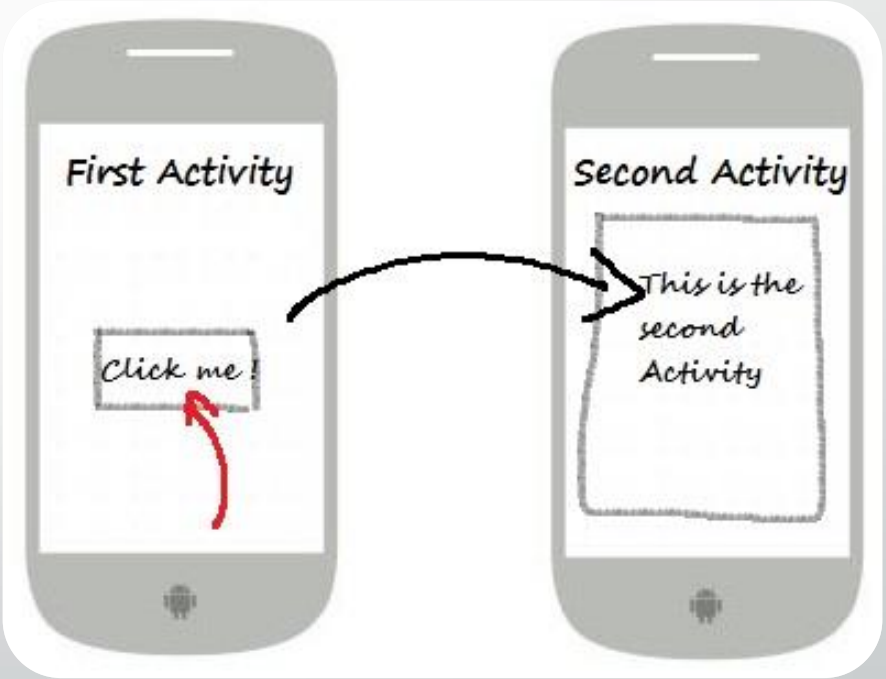
1. Activities
2. GUI
3. Intents
4. Manifest

Activities

Ohne Activities ist in der App nichts los

Activity – Was ist das?

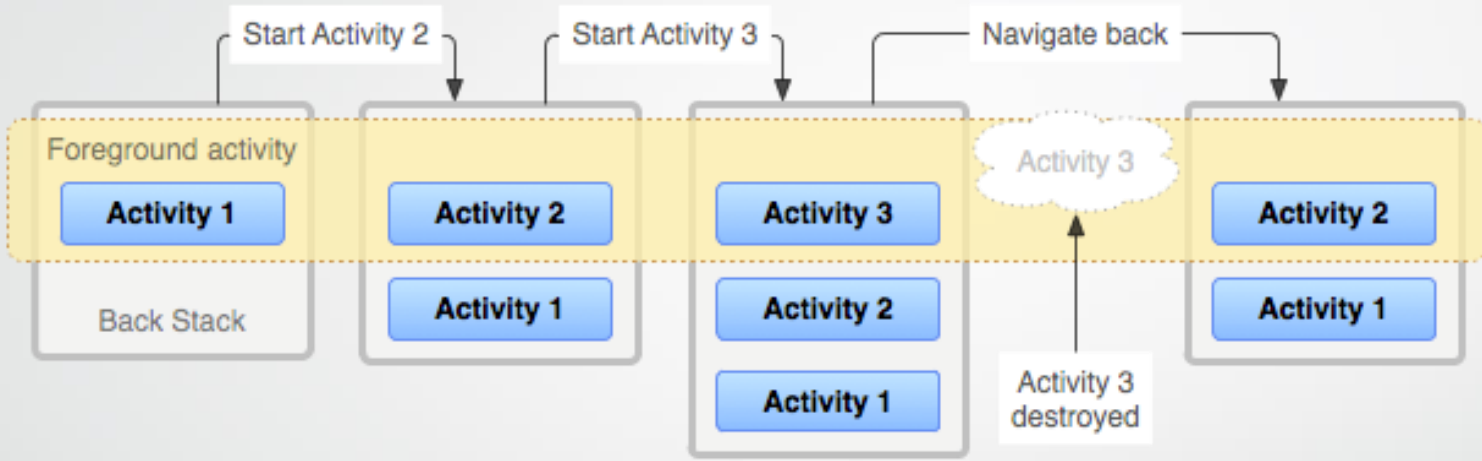
- Stellt dem Benutzer einen Bereich zur Interaktion zur Verfügung
- Eine App besteht meistens aus mehreren Activities.
- Jede Activity hat ein Fenster, in dem ihr Interface gezeichnet wird.
- In diesem Fenster nimmt sie Benutzerinput entgegen.



Beispiel-Activity

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Der Activity-Backstack



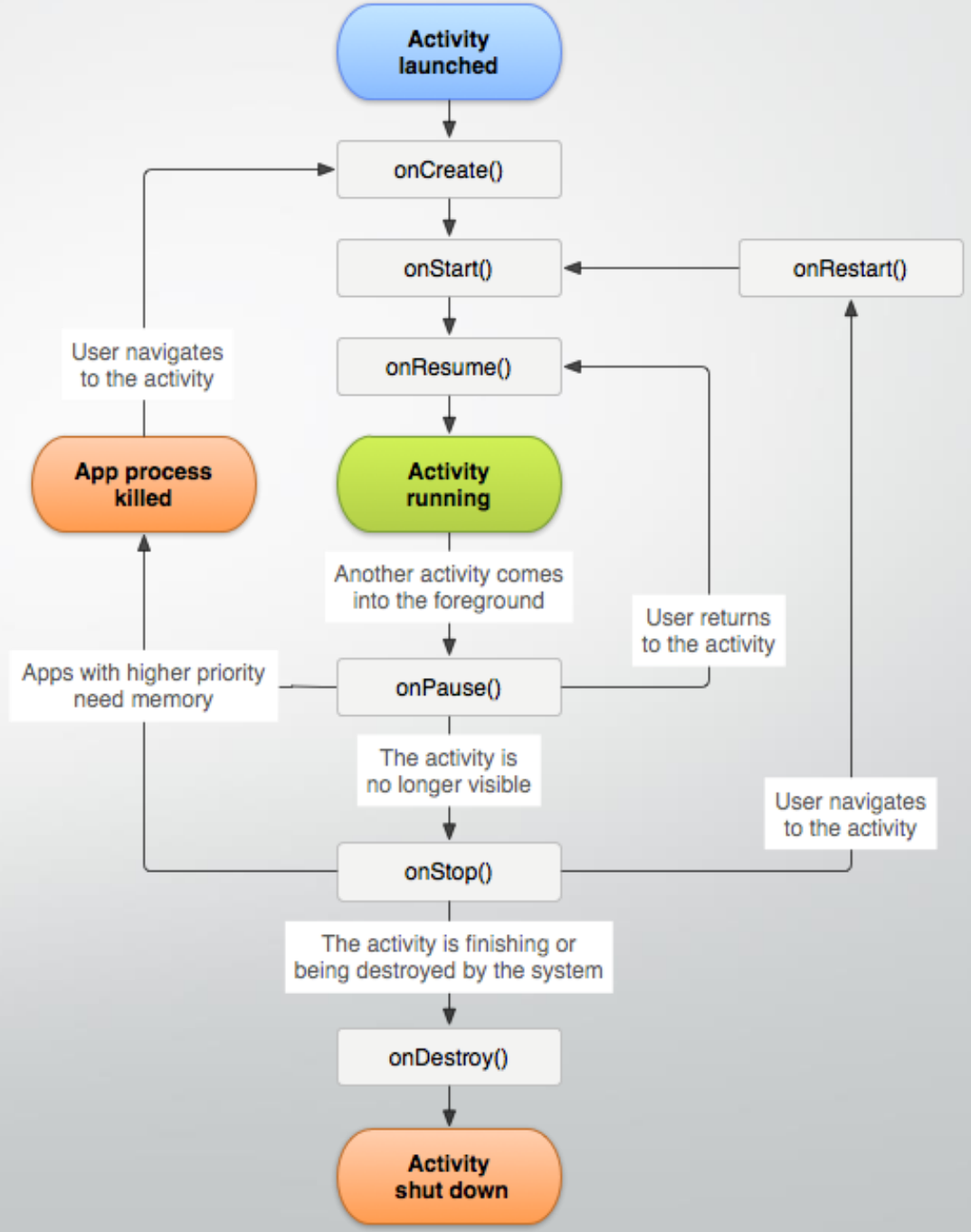
- Neue Activities werden auf den Backstack gelegt
- Die Activity ganz oben auf dem Stack ist im Vordergrund
- Rückt eine Activity in den Hintergrund, wird ihr Zustand gespeichert
- Im Hintergrund laufen Activities noch weiter -> Ressourcen freigeben!

Activity foreground vs. background

Activity Lifecycle

Jede Statusveränderung löst auch das dazugehörige Event aus und resultiert in einem Methodenaufruf.

create	Bei Erzeugung
start	Acitivity wird Sichtbar
resume	Activity rückt in den Vordergrund
pause	Andere Activity im Vordergrund, aber diese Activity noch sichtbar
stop	Activity vollständig im Hintergrund
restart	Rückkehr aus gestopptem Zustand
destroy	Activity wird beendet



Events

onSmartphoneDrop();

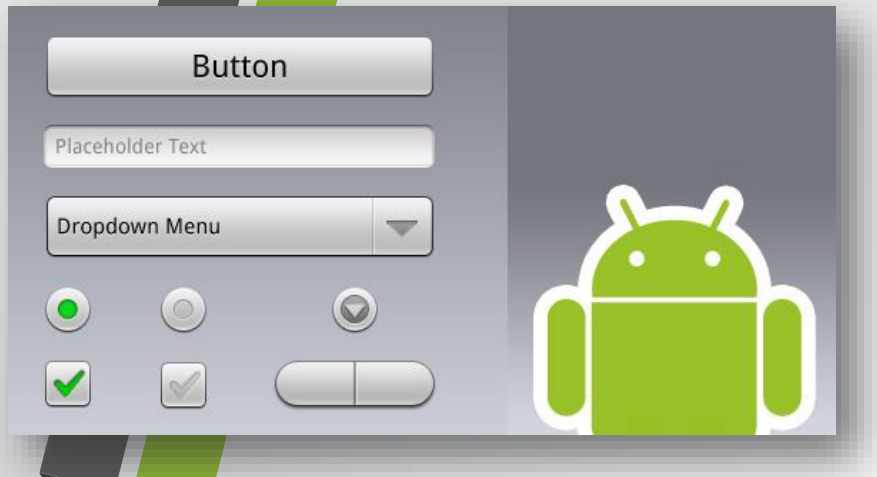
Events

- sind vom System erfasste Vorgänge
  - Touch Input
  - Laden einer Ressource
  - State Changes
- Ist an einer Stelle ein entsprechender Listener, wird das Event an den Event-Handler weitergegeben.
- Eine Activity lauscht bei ihrer Erstellung bereits auf viele dieser Events.

Beispiel für vordefinierte Event-Handler

```
@Override
public boolean onTouchEvent(MotionEvent event)
{
        // Touch verarbeiten return false;
}

@Override
protected void onCreate(Bundle savedInstanceState)
{
    // Konstanten definieren
    // Objekte initiialisieren
    // setContentView(someView);
}
```

```
@Override
public void onDestroy() {
        // Daten sichern
        // Ressourcen freigeben
}

@Override
public void onLowMemory() {
        // Speicher freigeben
        // Kritische Daten sichern
}
```



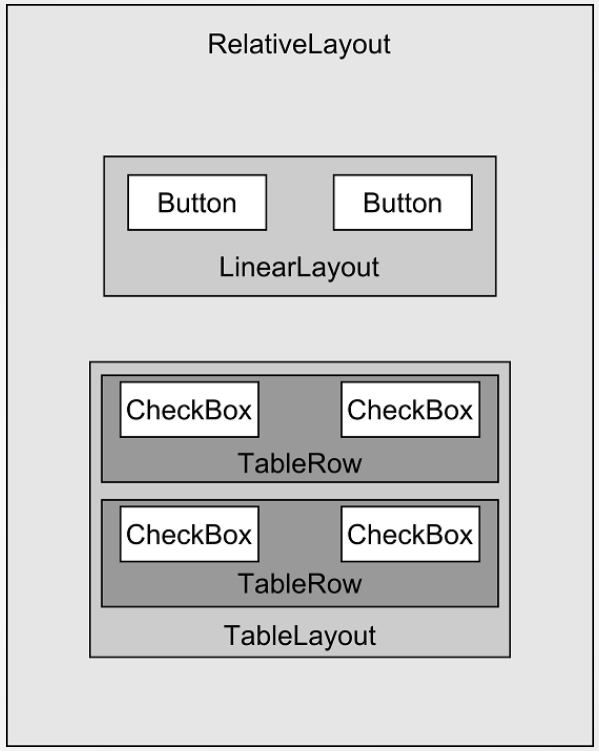
GUI mit Hilfe der Views

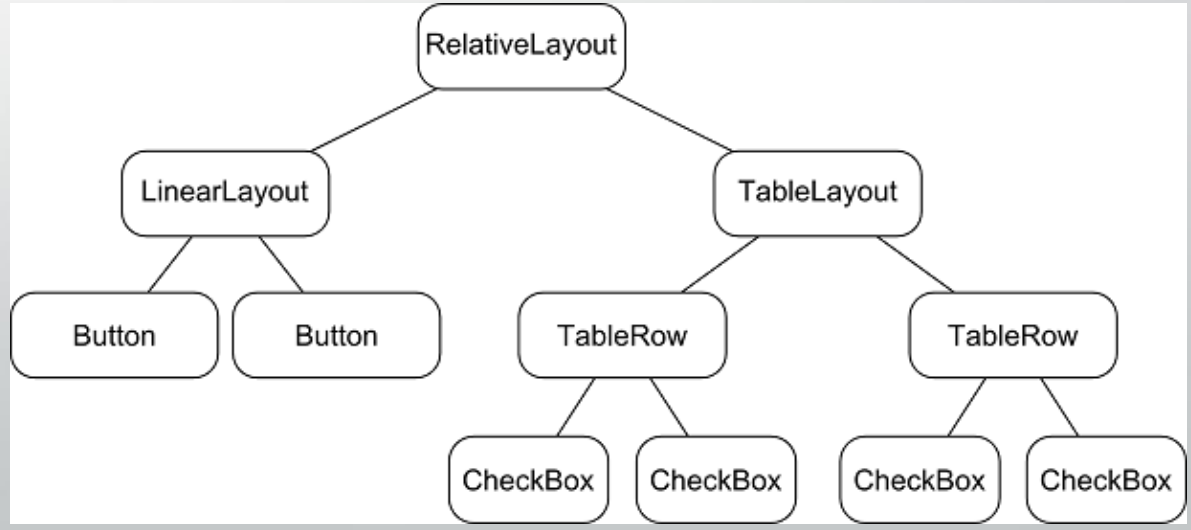
Knöpfe, Eingabefelder, Dropdowns –

Alles was das Herz begehrt!

GUI

- Um mit einer Activity interagieren zu können, empfiehlt sich eine GUI.
- Zusammengesetzt aus View-Objekten.
- Diese Views können wiederrum andere Views enthalten.
- Der View, der von einer Activity angezeigt wird, ist der ContentView



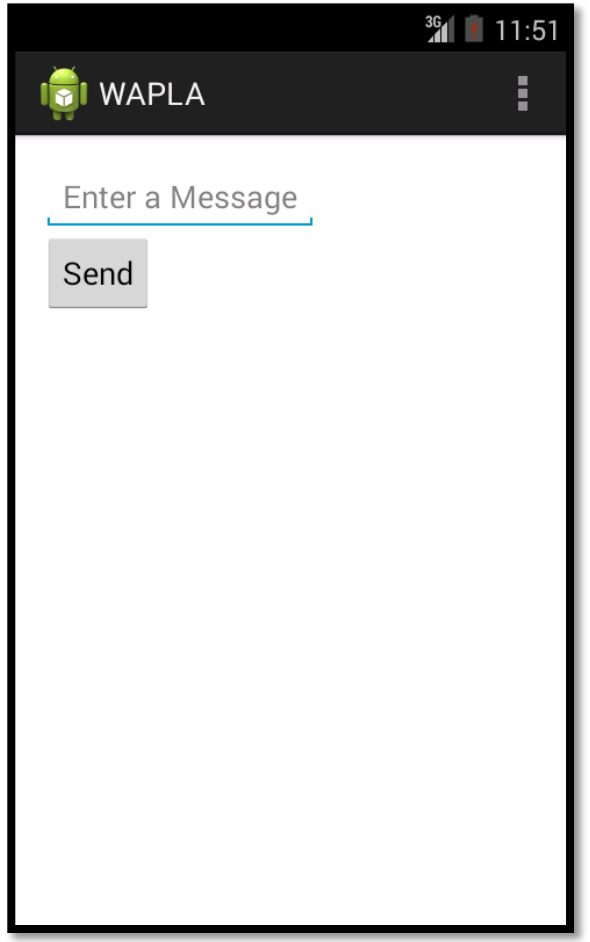


Wie erstelle ich View Objekte?
Ansatz 1 – Im Java Code

```
protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);

        // Linear Layout
        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
                                                      LayoutParams.MATCH_PARENT));
        linearLayout.setOrientation(LinearLayout.VERTICAL);

        // EditText
        EditText editText = new EditText(this);
        editText.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                                      LayoutParams.WRAP_CONTENT));
        editText.setHint(„Enter a message“);

        // Button
        Button button = new Button(this);
        button.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                                                LayoutParams.WRAP_CONTENT));
        button.setText("Send");
        button.setOnClickListener(new OnClickListener() {
                @Override public void onClick(View v) { //DO SOMETHING }
        });
```

```
        // Button
        Button button = new Button(this);
        button.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT,
                                                LayoutParams.WRAP_CONTENT));
        button.setText("Send");
        button.setOnClickListener(new OnClickListener() {
              @Override public void onClick(View v) { //DO SOMETHING }
        });

        // Verschachteln
        linearLayout.addView(editText);
        linearLayout.addView(button);
        setContentView(linearLayout);
}
```

Ergebnis



Ansatz 2Views in XML

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
        xmlns:tools=http://schemas.android.com/tools
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity" >

        <EditText android:id="@+id/edit_message"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:hint="@string/edit_message" />

        <Button android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/button_send"
                android:onClick="sendMessage" />
</LinearLayout>
```

Ressourcen und die R-Klasse

- Ressourcen befinden sich im *res* Folder
- Um auf eine bestimmte Ressource zugreifen zu können, benötigen wir eine Referenz zu ihr.
- Diese Referenzen verwaltet die Klasse *R*, im Folder *res*.
- *R* ist voller Konstanten, die Verweise auf die Ressourc Zahl beinhalten.

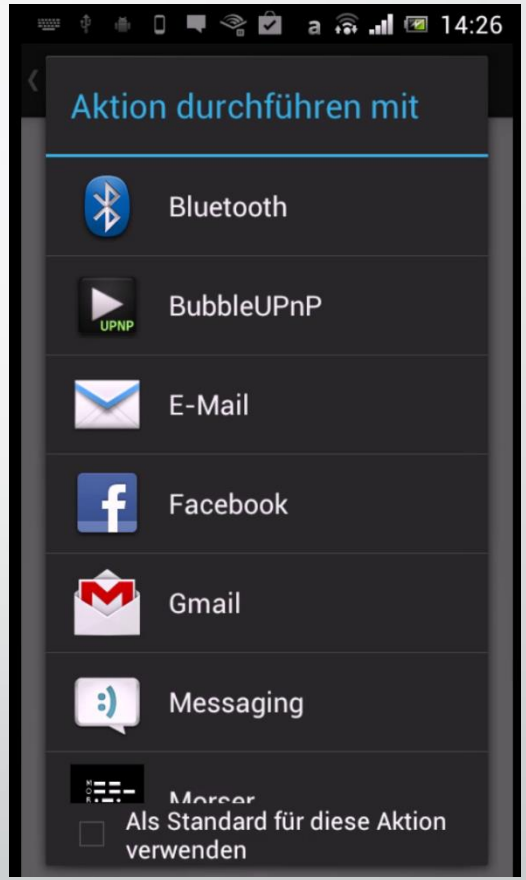
```
public static final class string {
        public static final int action_settings=0x7f070001;
        public static final int app_name=0x7f070000;
        public static final int hello_world=0x7f070003;
}

public static final class drawable {
        public static final int ic_launcher=0x7f020000;
        public static final int icon=0x7f020001;
}
```

Wozu ist R jetzt gut?

- Man möchte aus einer Activity heraus einen bestimmten View ansprechen:
  - findViewById(R.id.some_view);
- Dazu muss die ID aber in R registriert sein, das erreicht man mit folgendem XML-Attribut:
  - android:id="@+id/some_view„
  - Das + erzeugt in diesem Falle eine neue ID in R

Intents

- ist ein Messaging-Objekt, mit dem man Funktionalität anderer Komponenten anfragen kann.
- Explizite Intents sind zielgerichtet
  - Intent intent = new Intent(this, someActivity.class);
  - startActivity(intent)
- Implizite Intents werden an alle Komponenten des Geräts geschickt, die solche Intents filtern
  - Intent intent = new Intent();
  - intent.setAction(Intent.*ACTION_SEND);*
  - *startActivity(intent)*



<intent-filter> für implizite Intents

- Implizite Intents müssen gefiltert werden
- Diese Filter definiert man im Manifest der App

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Die AndroidManifest.xml

- Jede App braucht eine Manifest Datei.
- In dieser Datei sind der Aufbau und Metadaten der App beschrieben.
  - Mindestanforderungen (Version, Hardware, …)
  - Activities und deren Hierarchie
  - Name, Icon, Versionsnummer
  - Intent Filter
- Die Hardware-Anforderungen werden hier angegeben.
- Will die App Telefonhardware verwenden, muss sie erst um Erlaubnis fragen

Jetzt gibt es eine Live-Demo!

Veröffentlichen der App

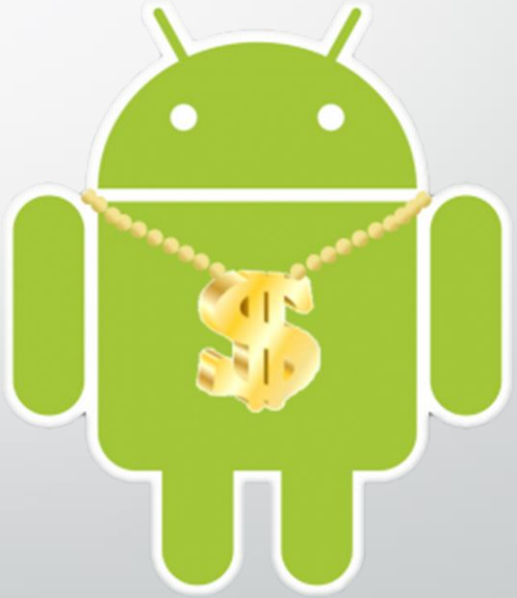
Der Android Market von Google

Veröffentlichen in Google Play

- Notwendig: Google Developer Konto
- Einmalzahlung: 25$ mit Kreditkarte
- Schlüssel-File von Google
- Mit diesem Schlüsselfile und dem Developer-Passwort kann man seine App signiert exportieren
- Das Resultat ist ein .apk File, das in den Store hochgeladen wird
- Zusätzlich werden in der Developer Console Screenshots, Beschreibung, Icon usw. angegeben

Rechtliches für Google Play

- Einhalten der Content-Richtlinien
  - Sexuelle Inhalte, Gewaltdarstellung, Glücksspiel, …
- App darf keine Geräte oder Netzwerke stören
- Entwickler ist alleine verantwortlich für den Support der App
- Google darf die App für Werbezwecke verwenden
- Die App kann für die Verbesserung von Android verwendet werden



Einnahmen
(oder was davon übrig bleibt)

- Apps dürfen gratis sein, oder der Preis ist frei wählbar, muss aber zwischen
  - 0.50 € und 100 € bzw.
  - 0.99 $ und 200 $liegen
- 70% des Erlöses der App gehen an den Developer
- davon müssen aber noch die fälligen Steuern gezahlt werden
- 30% behält sich Google für die Distribution im Play Store und die Zahlungsabwicklung

Wir wünschen frohes Appen!

- Informationen Android Development
  - http://developer.android.com/
  - „Android-Apps entwickeln für Einsteiger: Eigene Spiele-Apps für Leser mit Programmierkenntnissen!“, Uwe Post, Galileo Computing, 3. Auflage, August 2013
- Informationen zu Developer Guidelines
  - https://support.google.com/googleplay/android-developer/
- Wirtschaftsinformationen
  - http://www.gartner.com/technology/home.jsp