Android – Eine Einführung

Layouts, Views & Adapter

Andreas Wilhelm

Institut für Informatik
Georg-August-Universität Göttingen

www.avedo.net

Contents

1. Layouts
2. View-Klassen
3. ViewGroup-Klassen
4. AdapterView-Klassen
5. Adapter

Layouts

Contents

1 Überblick

2 ViewGroups

3 View

4 RequestFocus

5 Include

6 Merge

Allgemeines

- Oberflächen für Aktivitäten, Menüs, Dialogen und Widgets
- Grundelemente Views und ViewGroups (Layouts & AdapterViews)
- Deklaration im XML-Format
- Formatierung eines Elements über Attribute
- Alternativ View-Objekte während der Laufzeit im Quellcode erzeugen und ändern
- Weitestgehend einheitliche Namensgebung für XML-Attribute und Objekt-Methoden

XML-Layouts

Deklaration von Layouts in XML verschlankt den Code, ermöglicht Unterstützung verschiedener Oberflächen, Sprachen und Auflösungen und erleichtert den Debugging-Prozess.

Deklaration

- Erstellung von Layouts erinnert an HTML-Code
- Jedes layout darf nur ein Wurzelelement enthalten (View, ViewGroup, Layout, Merge)
- Darunter beliebig verschachtelte Layout-Hierarchie
- Speichern von Layouts unter */res/layouts*
- Zugriff im Quellcode über automatisch generierte Klasse *R*
- Zugriff in anderen Layouts über *@[package:]layout/filename*

Allgemeines

- Struktur-Elemente, die andere Views anordnen
- LinearLayout, das FrameLayout oder das RelativeLayout
- Weitere ViewGroups ohne Kindelemente (Bsp: AdapterView)
- Basis-Attribute ID (optional), sowie Breite und Höhe

Android-IDs

Eindeutige ID eines Layout-Elements wird über *android:id* in der Form *@+id/name* deklariert. + signalisiert dabei, dass es sich um eine neue ID handelt. Diese ID kann im Quellcode mit *R.id.name* referenziert werden.
Alternativ kann eine ID als Resource deklariert und mit *@id/name* referenziert werden. Die Deklaration einer solchen Resource wird in XML vorgenommen und unter einem beliebigen Dateinamen unter */res/values* abgelegt.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <item type="id" name="id_name" />
</resources>
```

Listing : Deklaration von IDs

Bemaßungen

- Bemaßungen eines ViewGroups mit den Attributen *android:layout_width* und *android:layout_height* entweder explizit oder implizit
- Explizite Angabe erfolgt dabei als Wert oder als Resource
- Implizite über die Schlüsselwörter *fill_parent*, *match_parent* oder *wrap_content*

Zur Bemaßungen können verschiedene Einheiten (px, dp, sp, pt, in, mm) oder Schlüsselwörter verwenden.

Schlüsselwort	Beschreibung
match_parent	Weist aktuellem Objekt die Größe des Eltern-Elements zu (löst *fill_parent* ab)
fill_parent	Weist aktuellen Objekt die Größe des Eltern-Elements zu
wrap_content	Sorgt für Anpassung der Größe, sodass Inhalt umschlossen wird

Alternativ kann man auch eine Resource im Quellcode über die Klasse *R* oder in einer XMl-Datei mit dem Schlüssel *@[package:]dimen/dimension_name* referenziert.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="imagebox_height">64dp</dimen>
    <dimen name="imagebox_width">32dp</dimen>
</resources>
```

Allgemeines

- Allein stehendes Element einer grafischen Oberfläche
- Nimmt rechteckige Fläche des Bildschirms ein
- Kümmert sich um die Ausgabe und das Verarbeiten von Ereignissen
- Textfelder, Buttons und Eingabefelder, wie beispielsweise *TextView*
- Deklaration per XML oder im Quellcode
- Verwaltung der Basis-Attribute, wie bei ViewGroups

RequestFocus

- Gibt View bzw. ViewGroup direkt bei Initialisierung den Fokus
- XML-Deklaration als leeres Element *<requestFocus>*
- Zuweisung eines solchen Elements nur einmal pro Datei

Include

- Verbinden bereits existierender Layouts
- Element <*include*>
- Einziges eigenes Attribut Name des einzubinden Layouts
- Optional eine ID, sowie Breite und Höhe des Layouts
- Werte überschreiben die Werte des Wurzelknotens im eingebundenen Layout
- Änderungen an den Bemaßungen mit *android:layout_width* und *android:layout_height* nur für beide Attribute möglich

ViewStub

Alternativ kann anstatt des <*include*>-Elements auch ein *ViewStub* verwendet werden.

Merge

- Wurzelelement um Layouts zu verbinden
- Sinnvoll wenn Layouts mit *<include>* verbunden werden sollen
- Flacherer und dadurch einfacherer Aufbau der View-Hierarchie

View-Klassen

Contents

7 TextView

8 EditText
- Button

9 CompoundButton

10 CheckBox

11 ToggleButton

12 RadioButton

13 RadioGroup

14 CheckedTextView

15 ImageView

16 ImageButton

17 ProgressBar

18 Spinner

19 Chronometer

20 WebView

21 TimePicker

22 DatePicker

Allgemeines

- Ausgabe von Text auf dem Bildschirm
- Formatierung über Attribute

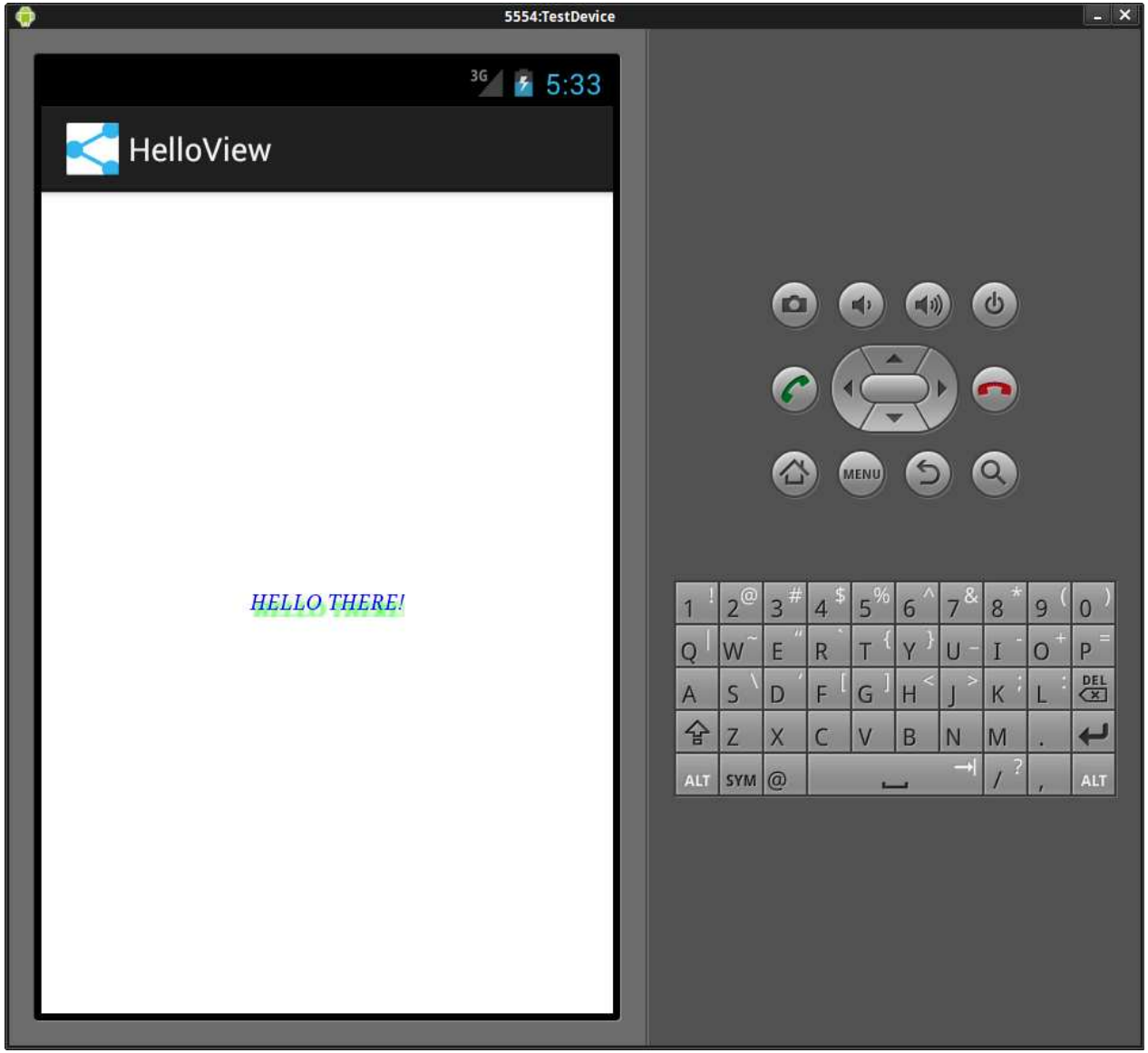
Attribut	Beschreibung
android:text	Der anzuzeigene Text
android:textSize	Schriftgröße [dimension]
android:textColor	Farbe des Textes [color]
android:textStyle	Stil des Texts (bold, italic, bolditalic) [flag]
android:typeface	Schriftart (normal, sans, serif, monospace) in der der Text angezeigt werden soll [enum]
android:textAllCaps	Flag um Text in Großbuchstaben auszugeben [boolean]
android:shadowColor	Farbe des Text-Schattens [color]
android:shadowRadius	Radius des Textschattens [double]
android:shadowDx	Verschiebung des Schattens in x-Richtung [double]
android:shadowDy	Verschiebung des Schattens in y-Richtung [double]

Deklaration im Layout

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello There!"
    android:textSize="12dp"
    android:textColor="#0000ff"
    android:textStyle="italic"
    android:typeface="serif"
    android:textAllCaps="true"
    android:shadowColor="#00ff00"
    android:shadowRadius="2.0"
    android:shadowDx="3.0"
    android:shadowDy="7.0" />
```

Listing : TextView

Screenshot



Allgemeines

- Abgeleitet von der Klasse TextView
- Erlaubt Editieren von Text
- Formatierungen, wie bei TextView möglich

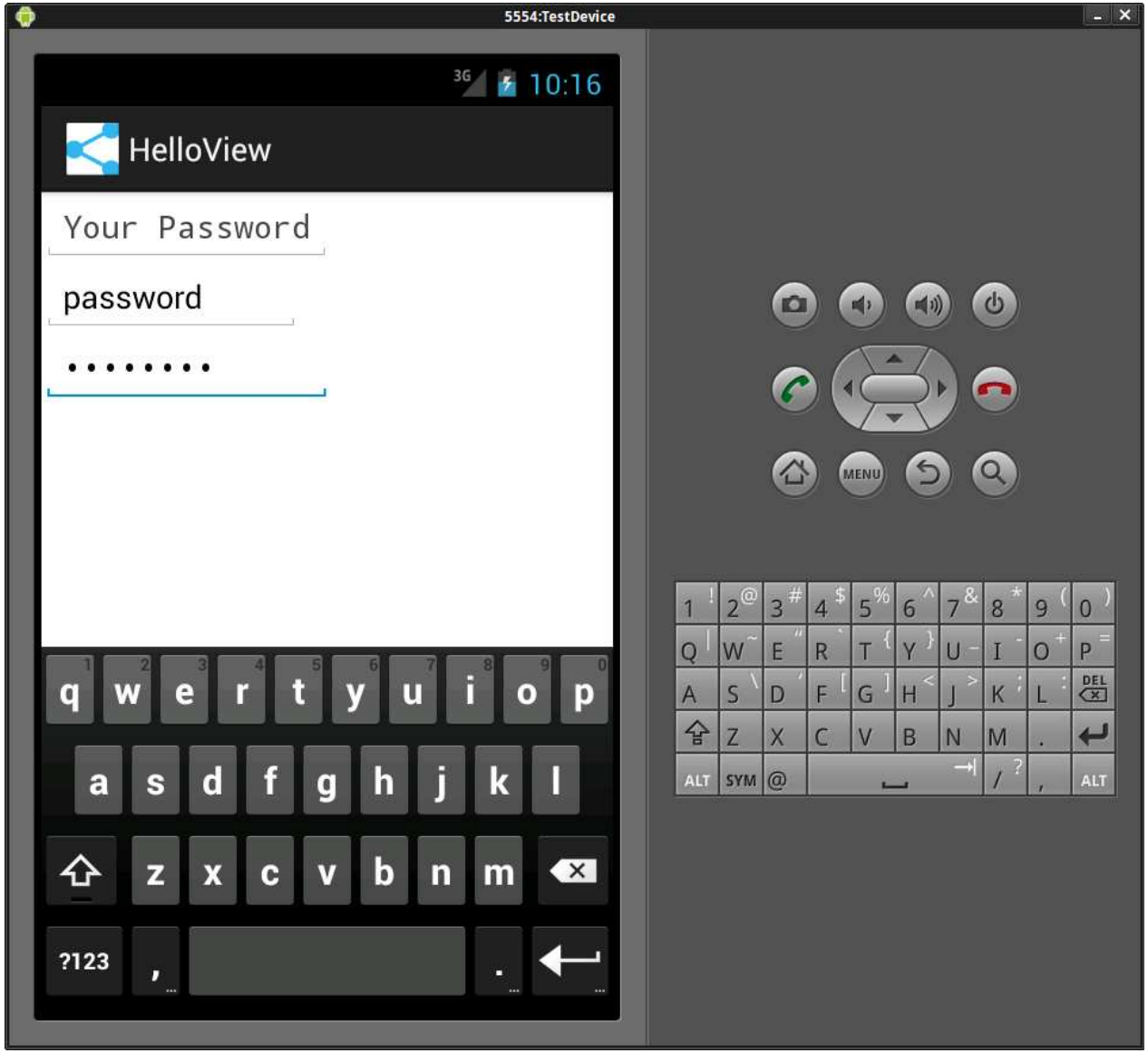
Attribut	Beschreibung
android:hint	Text der angezeigt werden soll, wenn das Eingabefeld leer ist
android:lines	Höhe des Eingabefeldes in Zeilen
android:maxLength	Maximale Länge der EingabeFarbe des Textes
android:password	Erlaubt das verstecken des eingegeben Texts und deaktiviert die Autovervollständigung [boolean]
android:textColorHint	Farbe des Stanard-Textes [color]

Deklaration im Layout

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Your Password"
    android:textColorHint="#454545"
    android:password="true"
    android:maxLength="10" />
```

Listing : EditText

Screenshot


5554:TestDevice
10:16
HelloView
Your Password
password
MENU

Allgemeines

- Abgeleitet von der Klasse TextView
- Erzeugt anklickbare Schaltfläche
- Erlaubt direkte Verlinkung mit Callback-Methode
- Alternativ Verwendung von *OnClickListener*

Attribut	Beschreibung
android:onClick	Definiert eine Methode der Aktivität, die beim klicken auf den Button ausgeführt werden soll

Deklaration im Layout

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="KlickMe"
    android:onClick="klickMeClicked" />
```

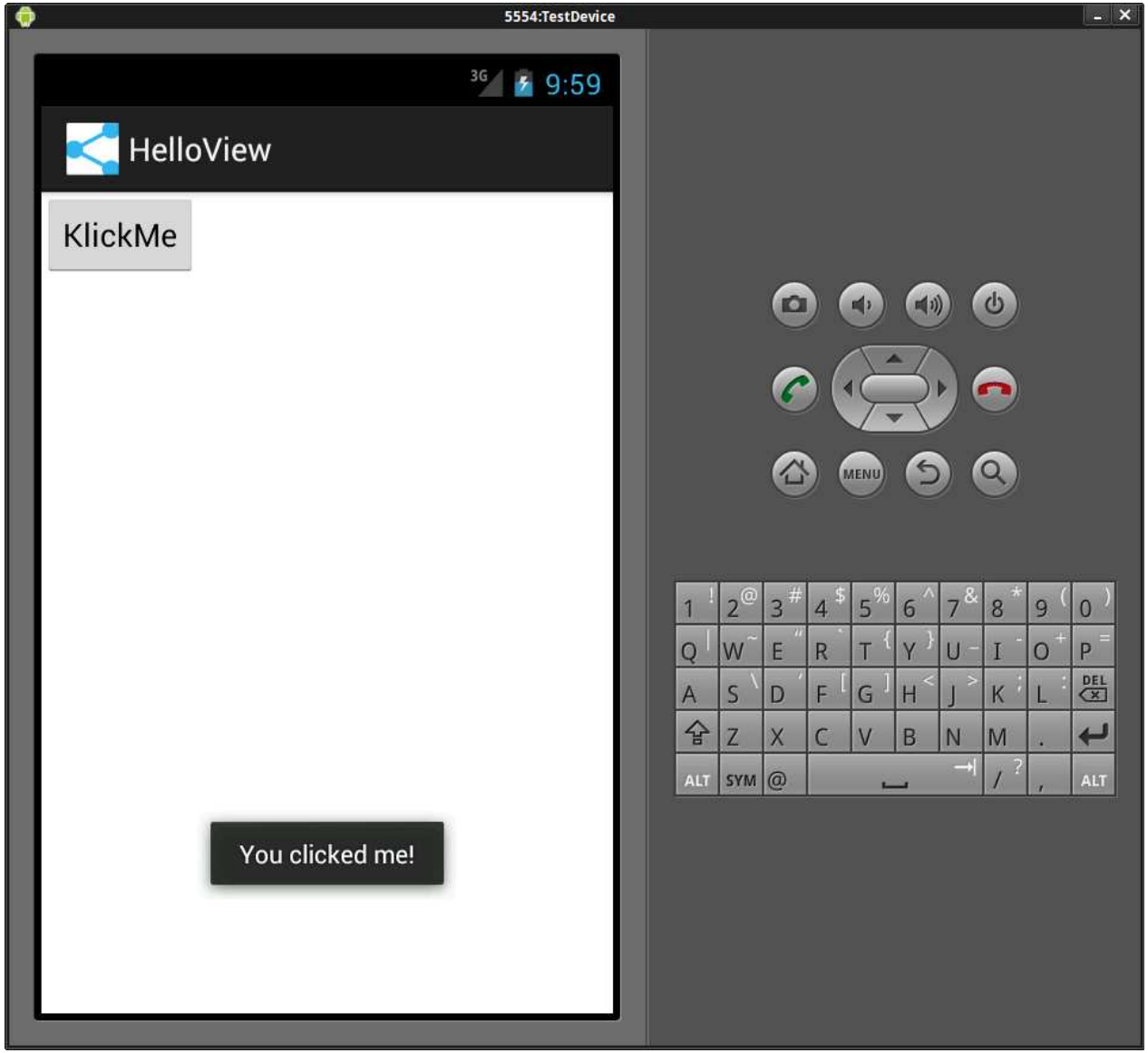
Listing : Button

Implementierung des Callbacks

```
public class HelloView extends Activity {
   ...

   public void onKlickMeClicked(View view) {
      Toast.makeText(this, "You clicked me!", Toast.LENGTH_LONG).show();
   }
}
```

Listing : Button OnClickListener

Screenshot



Allgemeines

- Abstrakte View Klasse – Kann nicht direkt verwendet werden
- Abgeleitet von der Klasse Button
- Kennt zwei Zustände aktiviert (*checked*) und deaktiviert (*unchecked*)
- Basis-Klasse für z.B. CheckBox, RadioButton, ToggleButton

Attribut	Beschreibung
android:button	Legt ein Bild fest, welches auf der Schaltfläche angezeigt werden soll [drawable]
android:checked	Gibt den initialen Zustand der Schaltfläche an [boolean]

Layout Deklaration

```
<CompoundButton
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="Toogle Me!"
    android:button="@drawable/ic_launcher"
    android:checked="true" />
```

Listing : CompoundButton

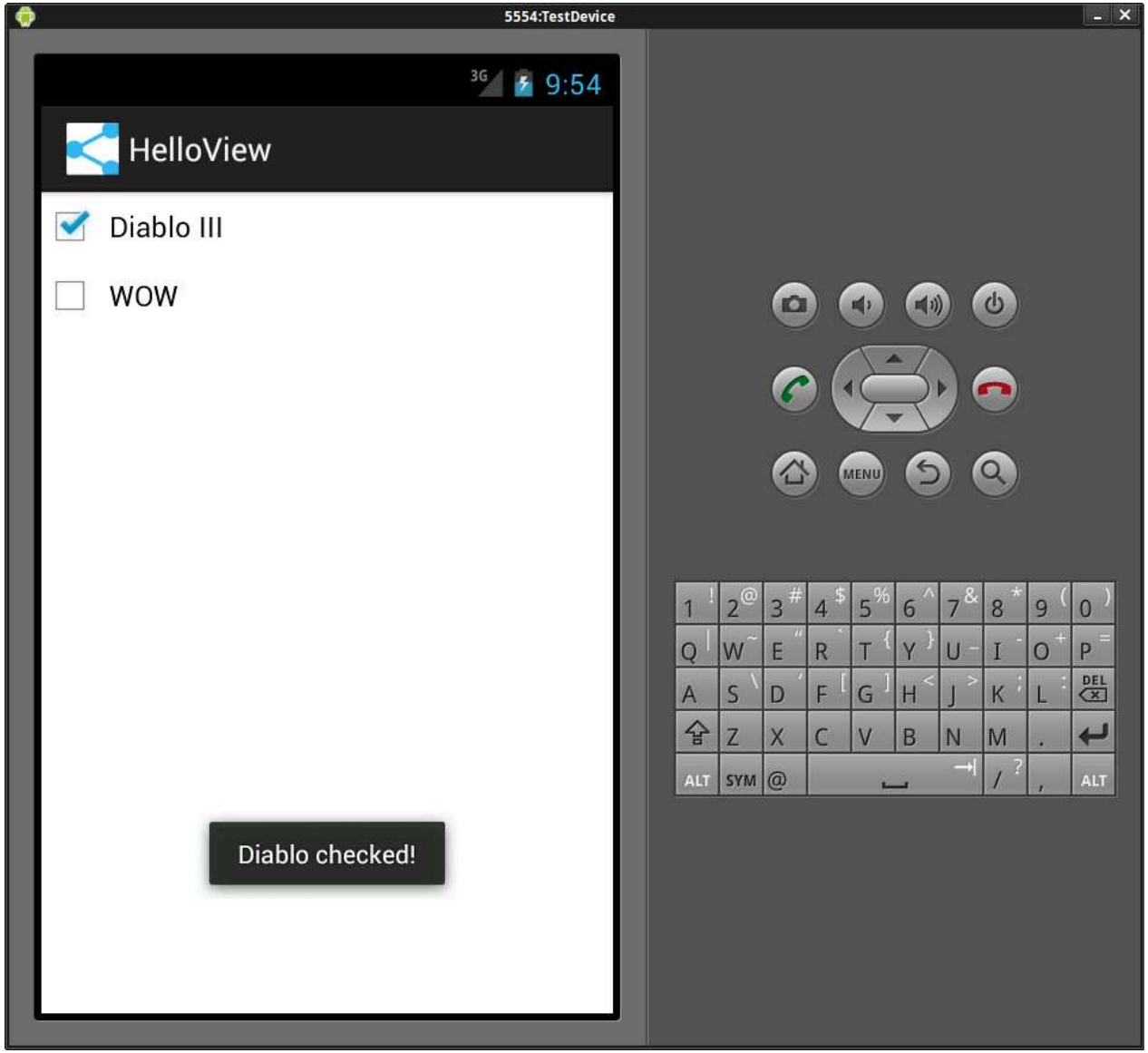
Allgemeines

- Abgeleitet von der Klasse CompoundButton
- Kennt zwei Zustände aktiviert (*checked*) und deaktiviert (*unchecked*)

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <CheckBox
        android:id="@+id/cbxDiablo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Diablo III"
        android:onClick="onDiabloChecked" />
    <CheckBox
        android:id="@+id/cbxWow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="WOW"
        android:onClick="onWowChecked" />
</LinearLayout>
```

Listing : CheckBox

Screenshot


5554:TestDevice
9:54
HelloView
Diablo III
WOW
Diablo checked!
MENU

Callback Implementierung

- Verwendung mehrerer zusammengehöriger Checkboxen möglich
- Zuweisung eines Callbacks über *android:onClick*-Attribut
- Nutzung des gleichen Callbacks möglich

```
public class HelloView extends Activity {
   ...

   public void onGameCbxClicked(View view) {
      // Get the checkbox state ...
      boolean checked = ((CheckBox) view).isChecked();

      // ... and check which checkbox was clicked.
      switch(view.getId()) {
         case R.id.cbxDiablo:
            Toast.makeText(this, "Diablo checked!", Toast.LENGTH_LONG).show();
            break;
         case R.id.cbxWow:
            Toast.makeText(this, "WOW checked!", Toast.LENGTH_LONG).show();
            break;
      }
   }
}
```

Listing : Methode für mehrere CheckBoxen

Allgemeines

- Abgeleitet von der Klasse CompoundButton
- Kennt zwei Zustände aktiviert (*checked*) und deaktiviert (*unchecked*)
- Dient als An- und Aus-Schalter
- Weißt jedem Zustand einen Namen zu

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Sound on"
    android:textOff="Sound off"
    android:onClick="onToggled"/>
```

Listing : ToggleButton

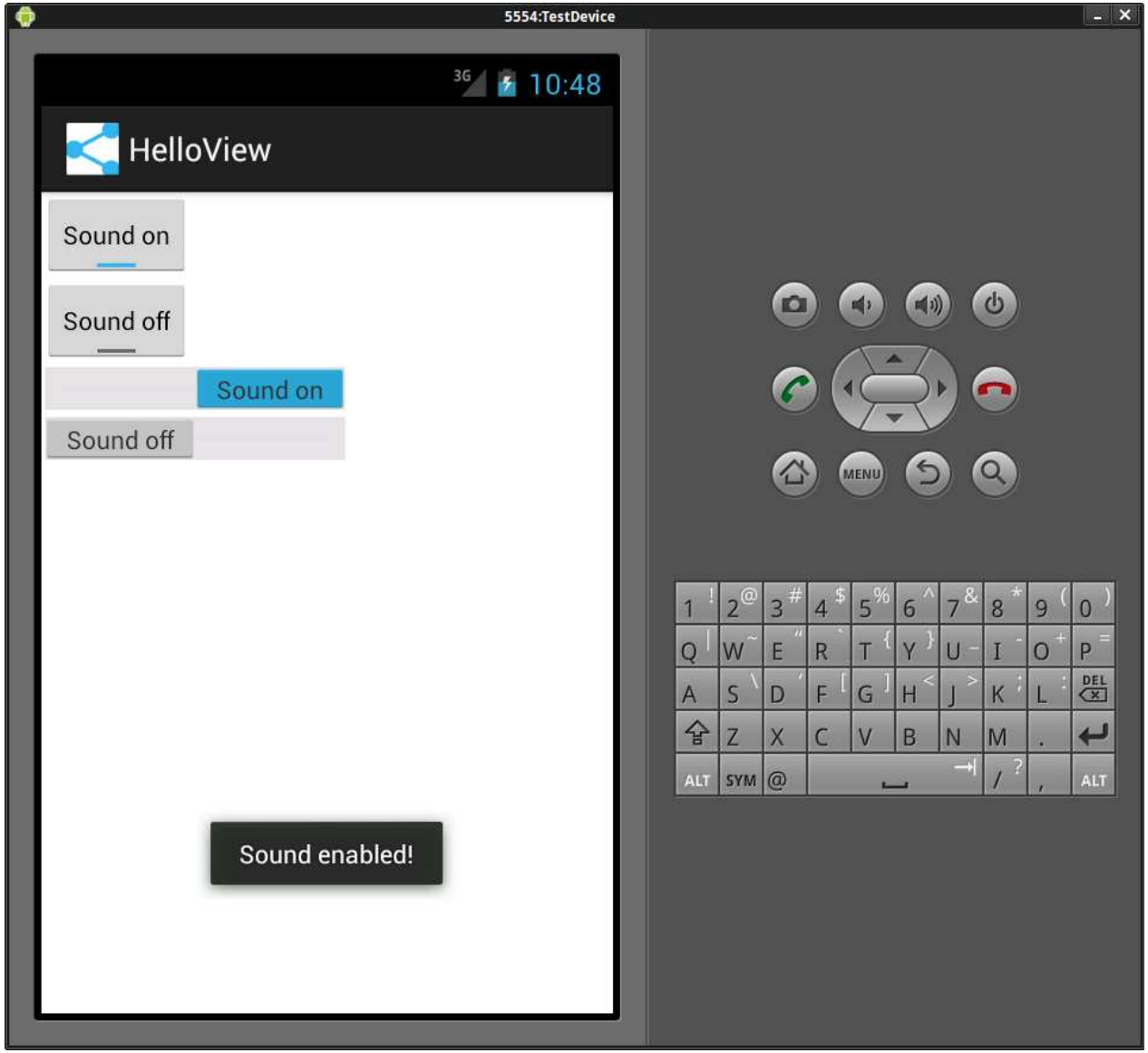
Alternative: Slider

- Eingeführt in Android 4.0
- Abgeleitet von der Klasse CompoundButton
- Verhalten wie ToggleButton
- Ausgabe als Slider

```
<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Sound on"
    android:textOff="Sound off" />
```

Listing : Switch

Screenshot


5554:TestDevice
10:48
HelloView
Sound on
Sound off
Sound on
Sound off
Sound enabled!
MENU

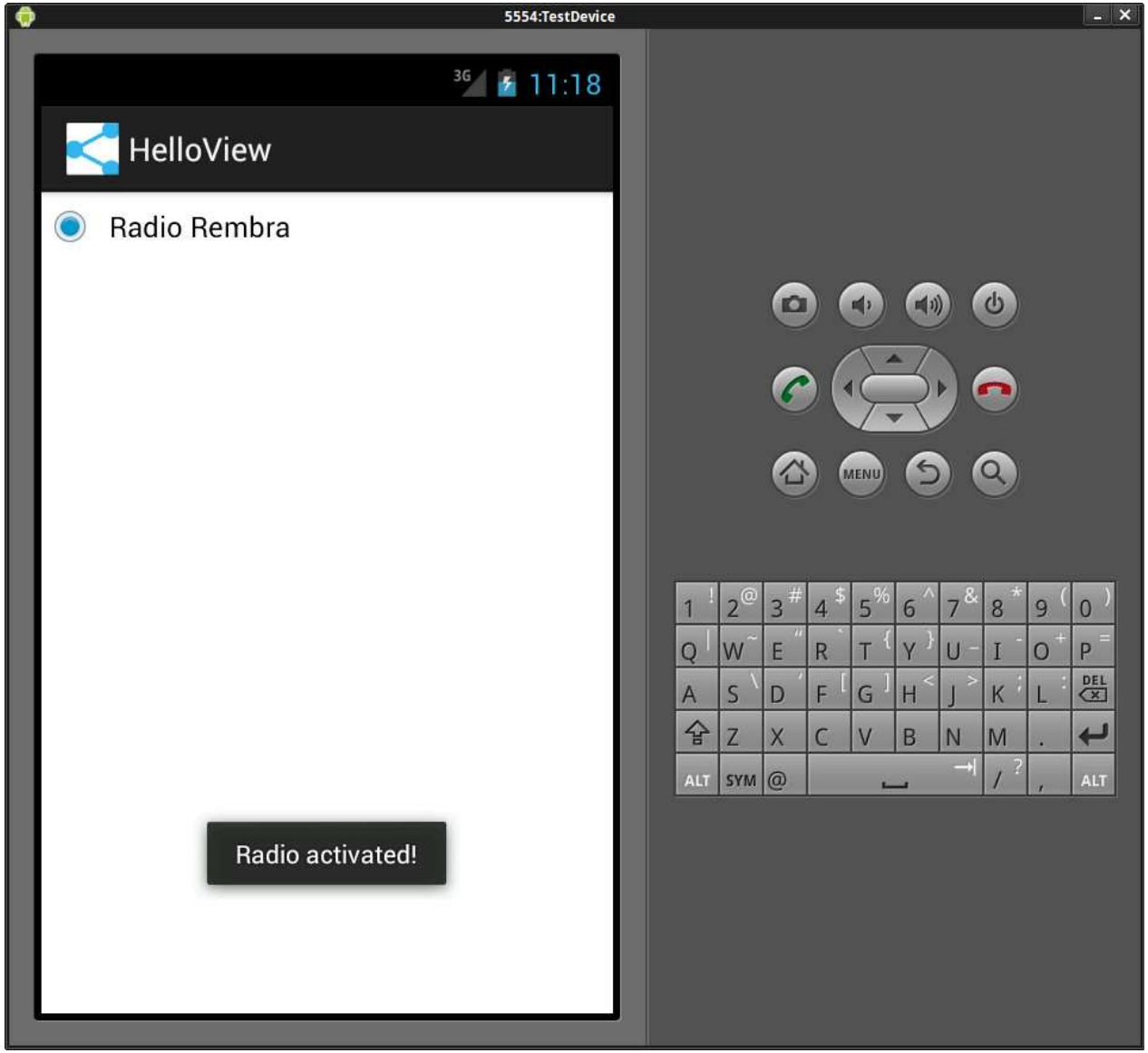
Allgemeines

- Abgeleitet von der Klasse CompoundButton
- Einmal aktiviert immer aktiviert
- Gruppierung in RadioGroup sinnvoll

```
<RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Radio Rembra"
    android:onClick="onToggled"/>
```

Listing : RadioButton

Screenshot


5554:TestDevice
11:18
HelloView
Radio Rembra
Radio activated!
MENU

Allgemeines

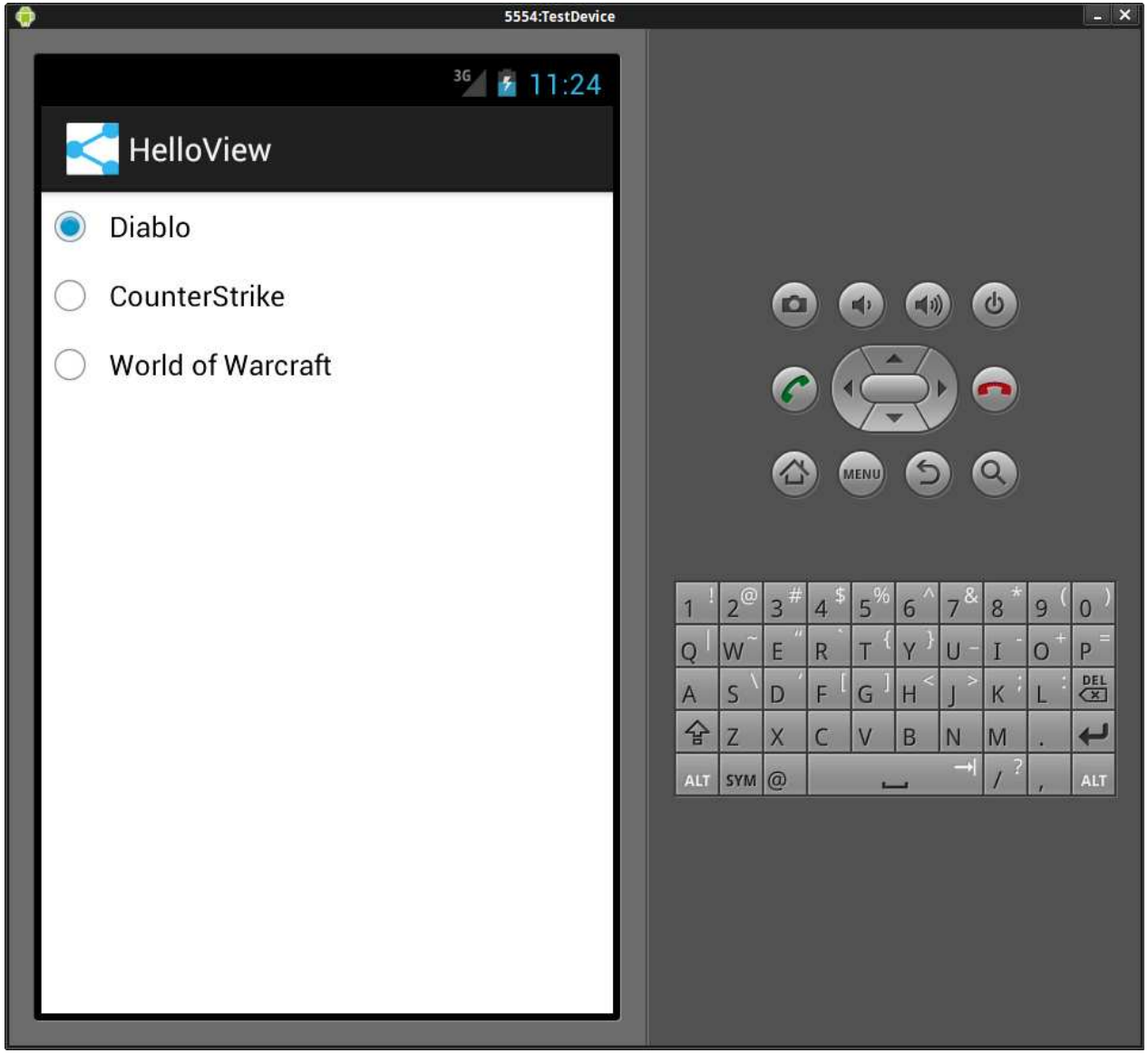
- Abgeleitet von der Klasse LinearLayout
- Dienen der Gruppierung von RadioButtons
- Immer nur ein einzelner RadioButton aktiviert

Deklaration im Layout

```
<RadioGroup
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Diablo"
        android:onClick="onRadio"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CounterStrike"
        android:onClick="onRadio"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="World of Warcraft"
        android:onClick="onRadio"/>
</RadioGroup>
```

Listing : RadioGroup

Screenshot


5554:TestDevice
3G
11:24
HelloView
Diablo
CounterStrike
World of Warcraft
MENU

Allgemeines

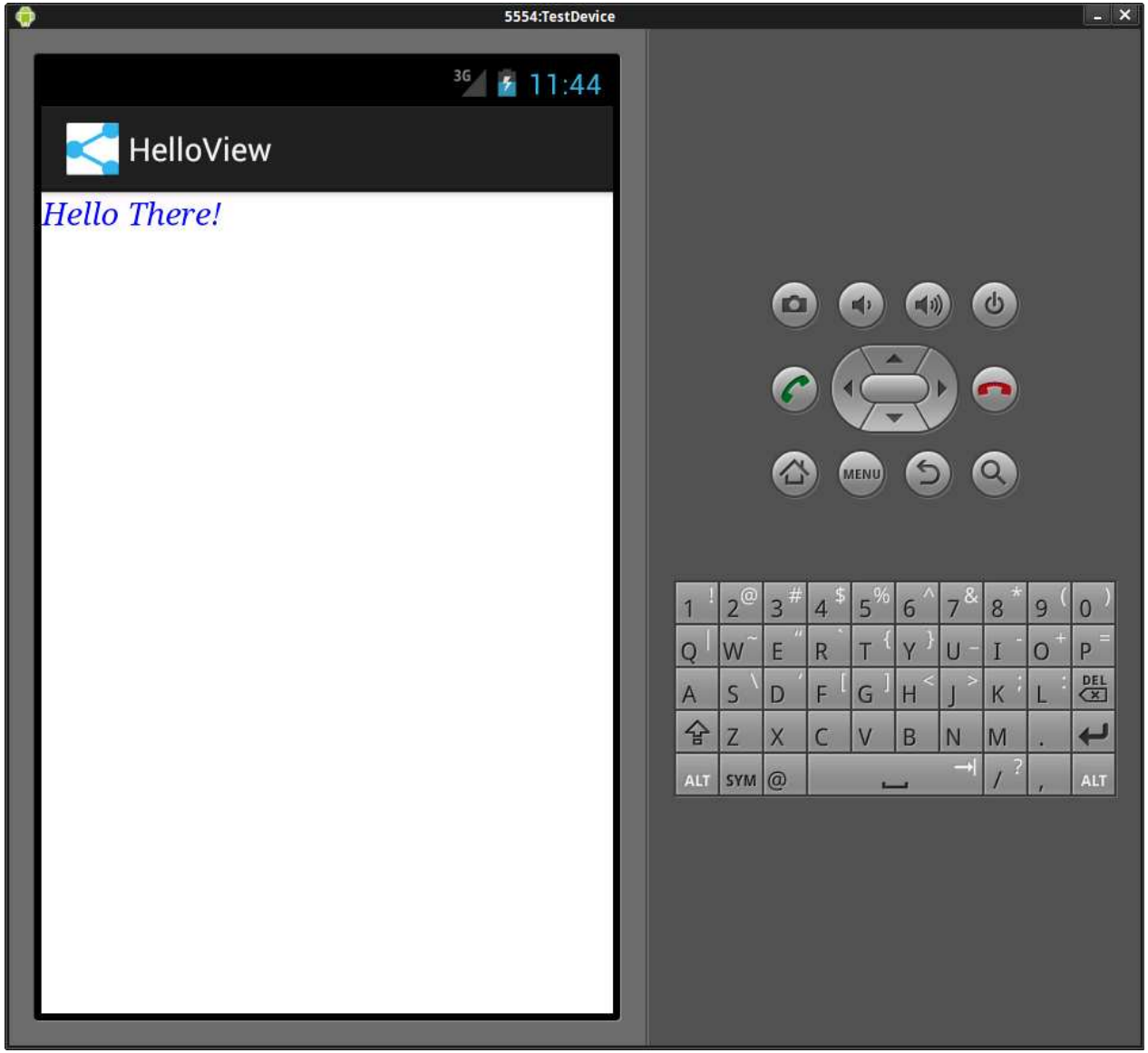
- Abgeleitet von TextView
- Implementiert das Checkable Interface
- Kennt Zustände aktiviert (*checked*) und deaktiviert (*unchecked*)
- Einsatz sinnvoll mit Spinnern und ListViews, wenn *setChoiceMode()* auf *CHOICE_MODE_NONE* gesetzt

Deklaration im Layout

```
<CheckedTextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello There!"
    android:textSize="17dp"
    android:textColor="#0000ff"
    android:textStyle="italic"
    android:typeface="serif"
    android:onClick="onTextChecked" />
```

Listing : CheckedTextView

Screenshot



Allgemeines

- Anzeige eines Bildes
- Kümmert sich um die Bemaßung des Bildes
- Formatierungen (Bsp. Einfärbung) über Attribute möglich

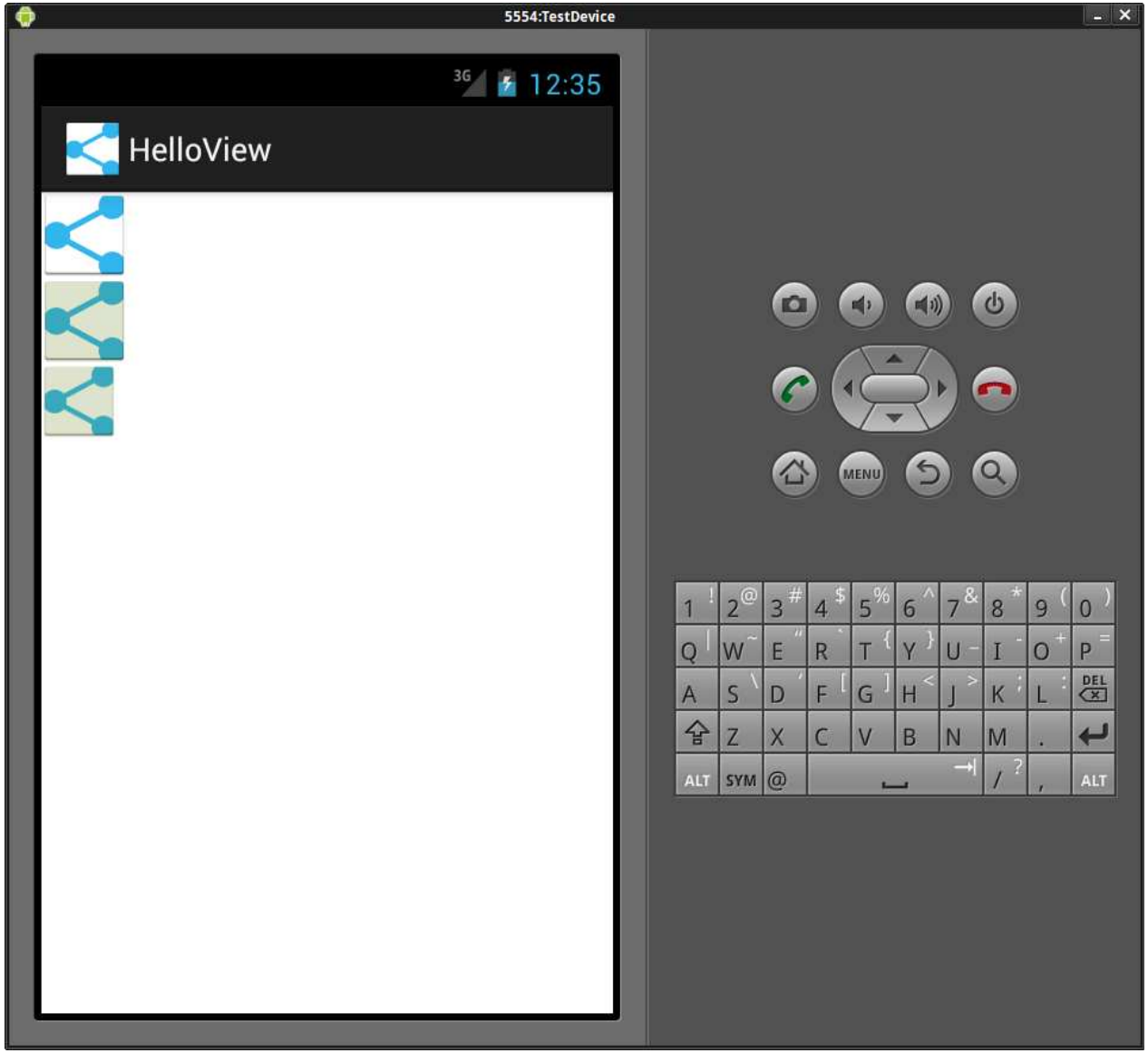
Attribut	Beschreibung
android:src	Die zu verwendende Bilddatei [drawable]
android:tint	Farbe für Bildeinfärbung [color]
android:adjustViewBounds	Legt fest, ob das ImageView seine Bemaßung an die seines Bildes anpasst [boolean]
android:scaleType	Legt fest, wie das Bild skaliert und bewegt werden soll um die erlaubten Bemaßungen einzuhalten [enum]
android:cropToPadding	Legt fest ob ein Bild zugeschnitten werden soll um in seinen Bereich zu passen [boolean]
android:baselineAlignBottom	Legt fest ob das Bild an der Grundlinie basierend auf der unteren kante ausgerichtet wird [boolean]
android:baseline	Legt die Grundlinie des Bildes fest [flag]
android:maxHeight	Maximale Höhe [dimension]
android:maxWidth	Maximale Breite [dimension]

Deklaration im Layout

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher"
    android:maxHeight="7dp"
    android:maxWidth="7dp"
    android:tint="#345678" />
```

Listing : ImageView

Screenshot



Allgemeines

- Abgeleitet von ImageView
- Sieht normalerweise aus wie Button mit Bild
- Angabe von Callback mit *android:onClick*

Deklaration im Layout

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher"
    android:onClick="onImageButtonClicked" />
```

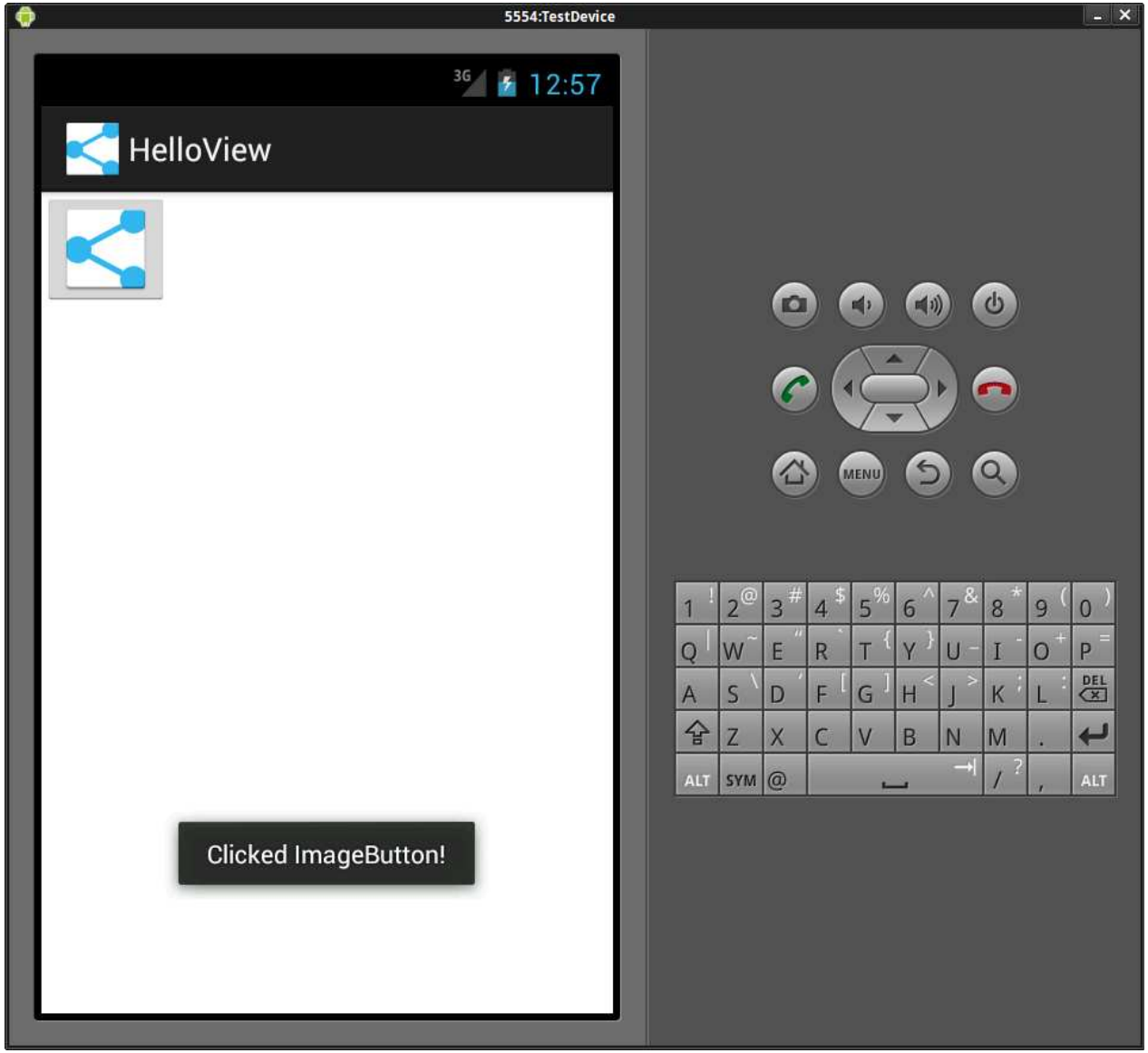
Listing : ImageButton

Implementierung des Callbacks

```
public void onImageButtonClicked(View view) {
    Toast.makeText(this, "Clicked ImageButton!", Toast.LENGTH_LONG).show();
}
```

Listing : Methode onImageButtonClicked

Screenshot


5554:TestDevice
3G
12:57
HelloView
Clicked ImageButton!
MENU

Anmerkung

Event-Hintergründe

Es ist möglich die Änderung der Hintergrundfarbe bei verschiedenen Benutzeraktionen, wie beispielsweise einem Klick, selbst festzulegen. Zu diesem Zweck kann man in XML einen *selector* definieren, der anhand der verschiedenen Button-Zustände ein Bild auswählt.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:state_pressed="true"
      android:drawable="@drawable/button_pressed" />
   <item android:state_focused="true"
      android:drawable="@drawable/button_focused" />
   <item android:drawable="@drawable/button_normal" />
</selector>
```

Listing : Deklaration von Button-Farben

Diese Datei legt man wie ein normales Bild unter */res/drawable* ab.

Allgemeines

- Visualisierung des Fortschritts eines laufenden Prozesses
- Unterscheidung in nicht deterministisch und deterministisch
- Möglichkeit einen zweiten Ladebalken anzuzeigen (Bsp. Buffering)

Android bietet verschiedene Styles, die über das Attribut *style* in der Form *@android:style/style_name* ausgewählt werden:

- Widget.ProgressBar.Inverse
- Widget.ProgressBar.Horizontal
- Widget.ProgressBar.Small
- Widget.ProgressBar.Small.Inverse
- Widget.ProgressBar.Large
- Widget.ProgressBar.Large.Inverse

Attribute

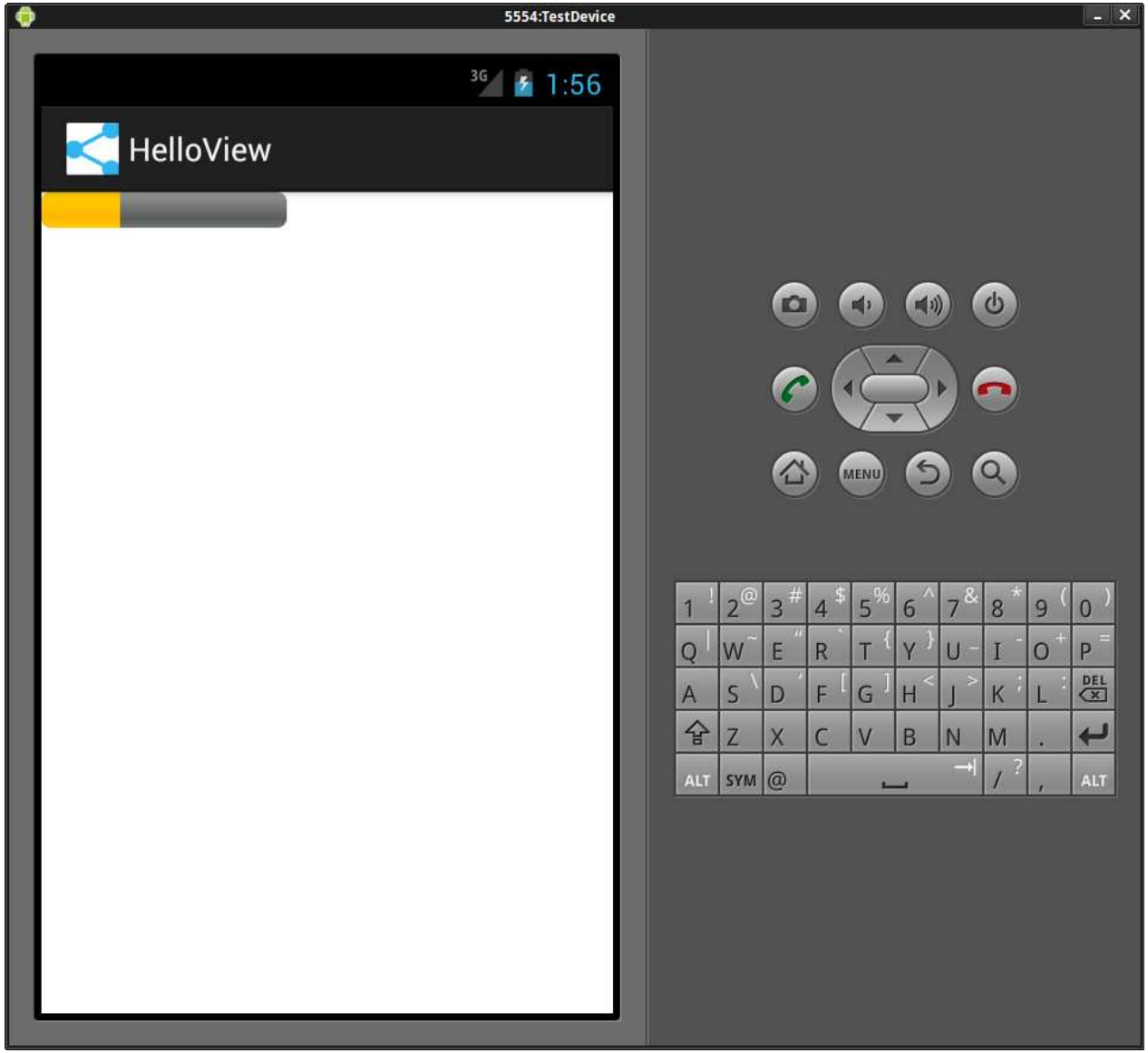
Attribut	Beschreibung
android:indeterminate	Aktiviert den *indeterminate* Modus einer ProgressBar
android:max	Legt den maximalen Wert fest [double]
android:maxHeight	Maximale Höhe der Bar [dimension]
android:maxWidth	Maximale Breite der Bar [dimension]
android:minHeight	Minimale Höhe der Bar [dimension]
android:minWidth	Minimale Breite der Bar [dimension]
android:progress	Der Wert des aktuellen Fortschritts (zwischen 0 und max)
android:progressDrawable	Bild die ProgressBar
android:secondaryProgress	Der Wert des aktuellen sekundären Fortschritts (zwischen 0 und max)

Deklaration im Layout

```
<ProgressBar
    android:id="@+id/progress_bar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:minWidth="137dp"
    android:max="50"
    style="@android:style/Widget.ProgressBar.Horizontal" />
```

Listing : ProgressBar

Screenshot



Implementierung

```
public class HelloView extends Activity {
   private ProgressBar mProgress;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      ...

      mProgress = (ProgressBar) findViewById(R.id.progress_bar);
      new Thread(new Runnable() {
         public void run() {
            for(int stat = 0; stat < 100; stat++) {
               try {
                  Thread.sleep(1000);
                  mProgress.setProgress(stat);
               } catch (InterruptedException e) {
                  e.printStackTrace();
               }
            }
         }
      }).start();
   }

   ...
}
```

Listing : Aktualisierung einer ProgressBar

Allgemeines

- Stellen ein DropDown-Menü bereit
- Alternativ anzeige als Dialog
- Platz sparende Alternative zu RadioGroups
- Initialisierung mit einem Start-Wert möglich

Attribut	Beschreibung
android:dropDownWidth	Breite des DropDowns für Spinner im DropDown-Modus
android:popupBackground	Hintergrundbild für Spinner im DropDown-Modus
android:prompt	Titel des Spinner Dialogs [resource]
android:spinnerMode	Anzeige Modus eines Spinners (dialog, dropdown) [int]

Deklaration im Layout

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical|center_horizontal"
    android:spinnerMode="dialog" />
```

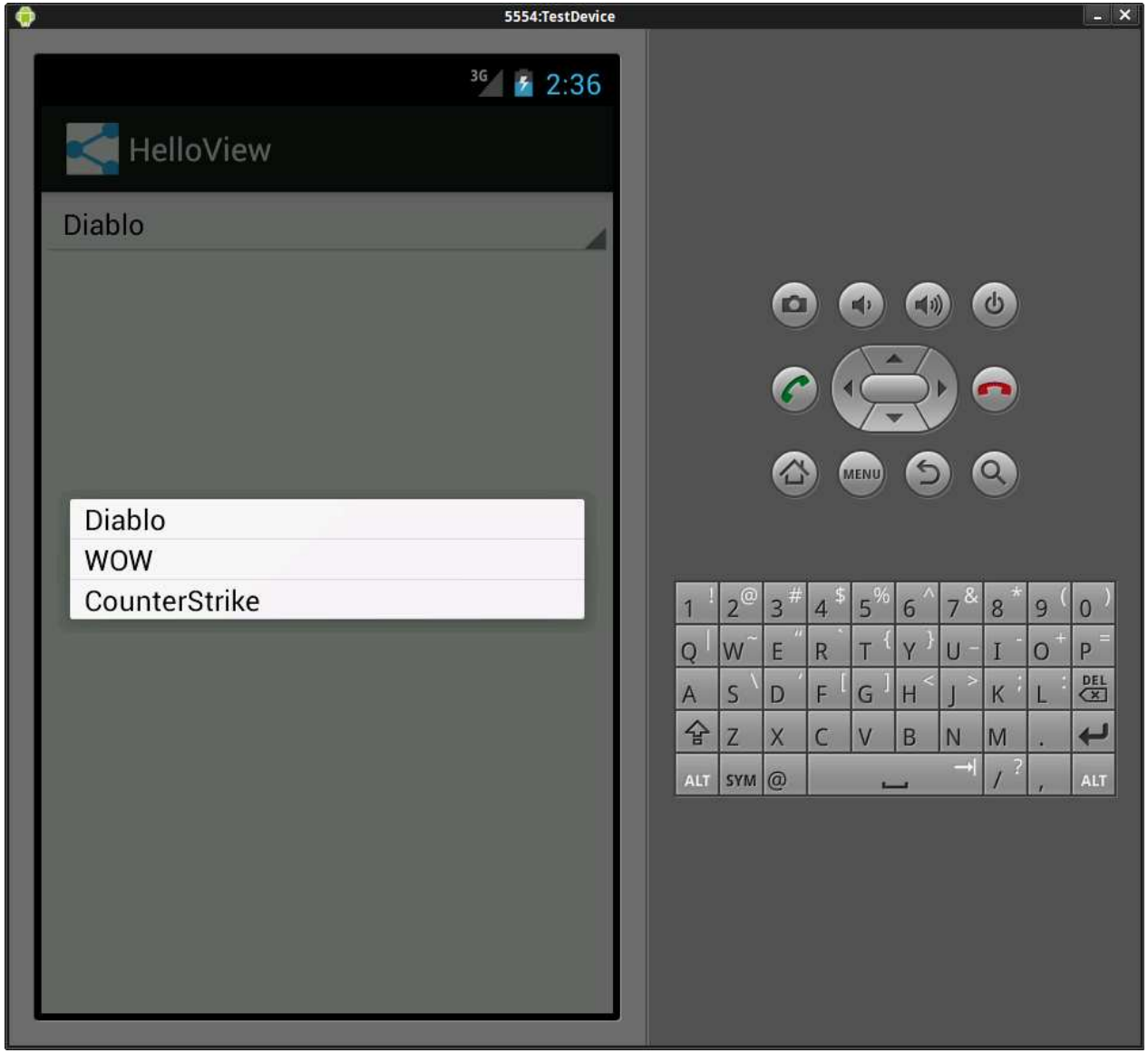
Listing : Spinner

Implementierung

```
public class HelloView extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      ...

      Spinner spinner = (Spinner) findViewById(R.id.spinner);
      ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
         this, R.array.games_array, android.R.layout.simple_spinner_item);
      spinner.setAdapter(adapter);
   }

   ...
}
```

Listing : Ein Spinner-Adapter

Screenshot


5554:TestDevice
3G
2:36
HelloView
Diablo
Diablo
WOW
CounterStrike
MENU

Anmerkung

A

uswahllisten, wie Spinner, können mit String-Array-Resourcen befüllt und sprachunabhängig gestaltet werden. Deklaration in der Sprachdatei *string.xml* unter */res/values*.

```
<resources>
    <string name="app_name">Hello Views</string>

    ...

    <string-array name="games_array">
        <item>Diablo</item>
        <item>WOW</item>
        <item>CounterStrike</item>
    </string-array>
</resources>
```

Listing : Deklaration eines String-Arrays

Allgemeines

- Benutzeroberfläche für Timer
- Abgeleitet von Klasse *TextView*
- Startwert mit *elapsedRealtime()* Mathode aus *SystemClock* Klasse setzen
- Alternativ: Automatisch mit *start()* initialisieren
- Format *MM:SS* oder *H:MM:SS*
- Format kann mit *setFormat(String)* geändert werden

Attribut	Beschreibung
android:format	String der ein Ausgabeformat festlegt. Das erste Vorkommen von *%s* wird durch den aktuellen wert des Timers im Format *MM:SS* oder *H:MM:SS* ersetzt.

Deklaration im Layout

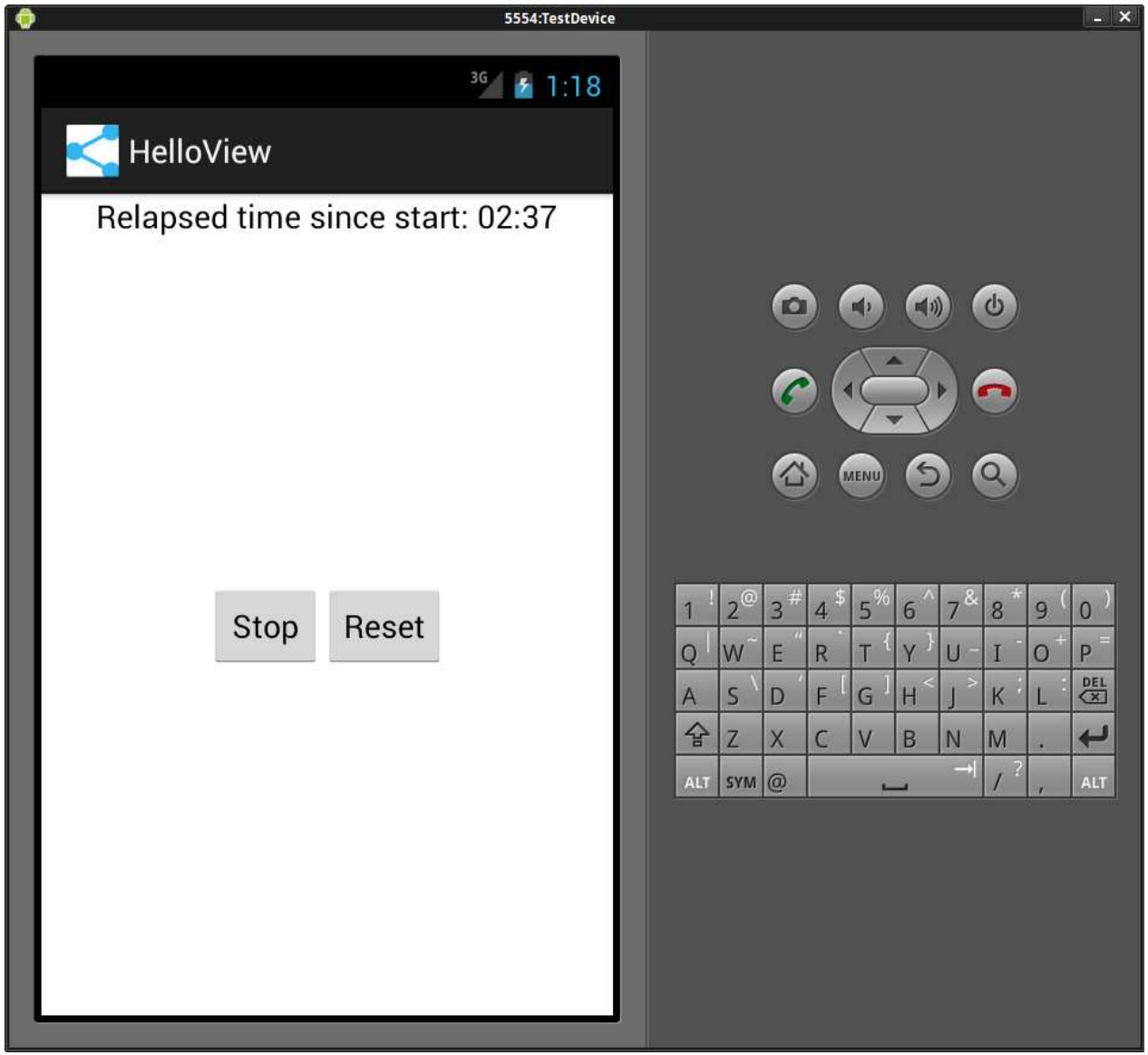
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Chronometer
        android:id="@+id/chronometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:format="Relapsed time since start: %s" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/startStopwatch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Start"
            android:onClick="onStartStop" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reset"
            android:onClick="onReset" />
    </LinearLayout>
</LinearLayout>
```

Listing : Chronometer

Implementierung

```
public class HelloView extends Activity {
    private Chronometer chron;
    private Button startStop;
    private boolean isChronRunnig  = false;
    private long elapsed = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_hello_view);

      // Fetch the chronometer.
      chron = (Chronometer) this.findViewById(R.id.chronometer);

      // Load the start/stop button.
      startStop = (Button) this.findViewById(R.id.startStopwatch);
    }

   public void onStartStop(View v) {
      if (!isChronRunnig) {
         chron.setBase(SystemClock.elapsedRealtime() - elapsed);
         isChronRunnig = true;
         startStop.setText("Stop");
         chron.start();
      } else {
         elapsed = SystemClock.elapsedRealtime() - chron.getBase();
         isChronRunnig = false;
         startStop.setText("Start");
         chron.stop();
      }
   }

   public void onReset(View v) {
      elapsed = 0;
      isChronRunnig = false;
      chron.stop();
```

Screenshot


5554:TestDevice
3G
1:18
HelloView
Relapsed time since start: 02:37
Stop
Reset
MENU

Allgemeines

- Ermöglicht Anzeige von Webseiten innerhalb einer Applikation
- Abgeleitet von der Klasse *ViewGroups* oder genauer von *AbsolutLayout*
- Anzeigen der Webinhalte übernimmt die *WebKit* Engine
- Unterstützt nicht von Haus aus Web typische Besonderheiten, wie JavaScript
- Inhalte mit *loadUrl()* oder HTML-Code mit *loadData()* laden

Zugriffsrechte für das Internet

Um einer Applikation den Zugriff auf das Internet zu ermöglichen, benötigt sie die entsprechenden Rechte. Das bedeutet, dass in der Android Manifest-Datei diese Rechte gesetzt werden müssen. Dazu muss die Zeile aus Listing eingebunden werden.

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

Listing : Internet Zugriffsrechte

Web 2.0 Funktionen

JavaScript einschalten

Auch wenn ein WebView nicht standardmäßig JavaScript unterstützt kann diese Funktionalität aktiviert werden. Zu diesem Zweck verwaltet jedes WebView seine Einstellungen in einem *WebSettings* Objekt, auf das mit der Methode *getSettings()* zugegriffen werden kann. Mit einem Aufruf der Methode *setJavaScriptEnabled(true)* der Klasse WebSettings kann die Unterstützung von JavaScript für das entsprechende WebView aktiviert werden.

HTML-Standard

Man muss bei der Verwendung von WebView immer darauf achten, dass es nicht nur Probleme mit komplexeren Webseiten hat, die JavaScript oder Flash einbinden, sondern dass es auch Einschränkungen in Bezug auf den HTML-Code macht.

Funktionalitäten erweitern

WebChromeClient Kümmert sich um die Verarbeitung von Events, die die grafische Oberfläche beeinflussen.

WebViewClient Kümmert sich um die Verarbeitung von Events, die die Anzeige des Inhalts beeinflussen.

WebSettings Änderungen an den Einstellungen, wie Aktivierung von JavaScript.

Web-Applikationen

Mit WebView ist es möglich bereits existierende Web-Applikationen direkt in Android zu integrieren.

Dazu wird es ermöglicht Teile des oftmals in Web-Applikationen verwendeten JavaScript-Codes an Android-Code zu binden. So ist es beispielsweise möglich alle Ausgaben, die durch ein JavaScript-Alert erzeugt würden in Android-Toasts ausgeben zu lassen. Android ermöglicht dies durch die Implementierung von *JavaScriptInterfaces*.

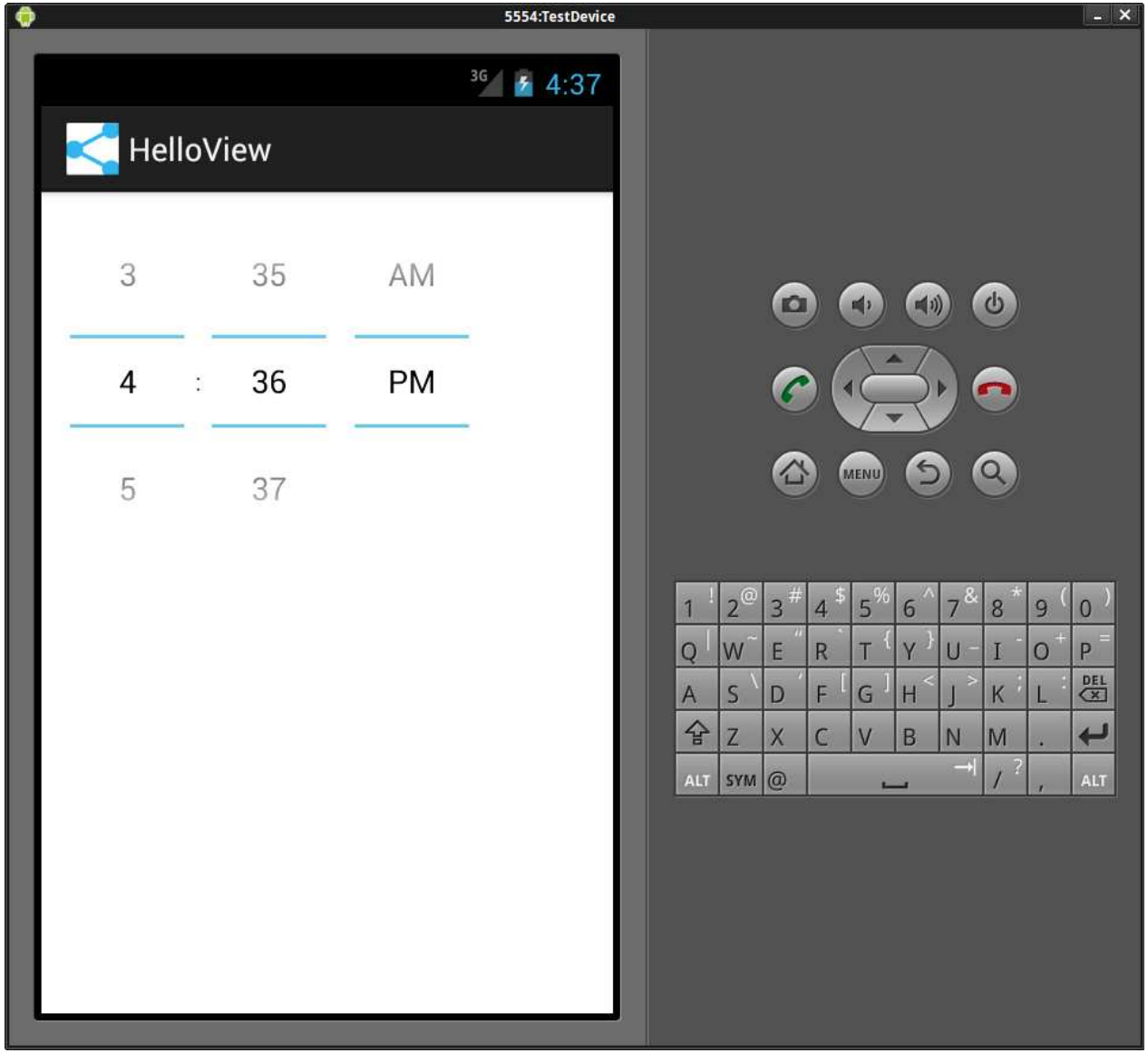
Allgemeines

- *TimePicker* ist abgeleitet von Klasse *FrameLayout*
- Oberfläche zur Auswahl einer Uhrzeit im AM/PM-Format

```
<TimePicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Listing : TimePicker

Screenshot



Allgemeines

- *DatePicker* ist abgeleitet von Klasse *FrameLayout*
- Oberfläche zur Auswahl eines Datums
- Kann als mehrerer Spinner oder Kalender (*CalendarView*) angezeigt werden

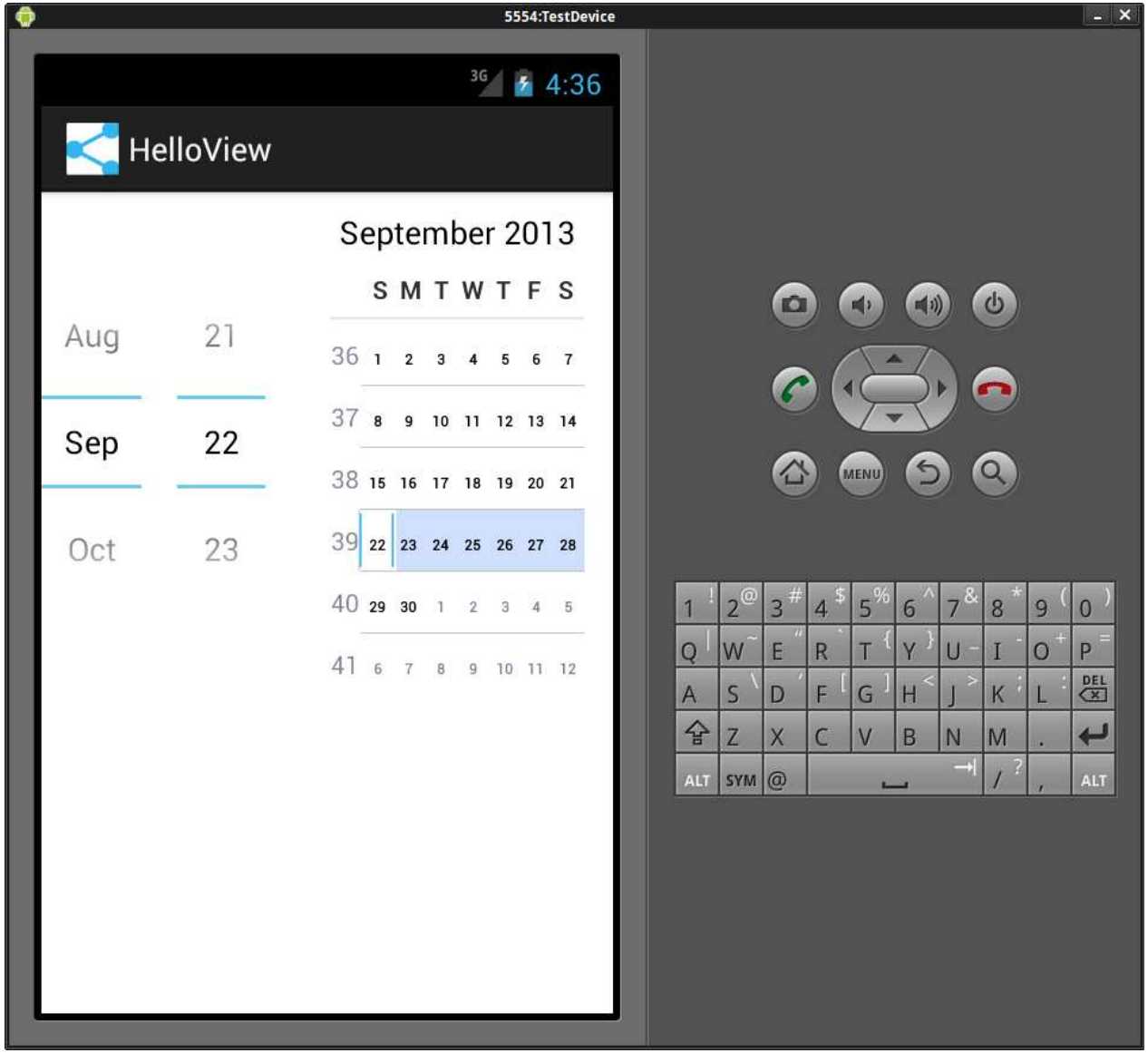
Attribut	Beschreibung
android:startYear	Das erste Jahr, das als Eingabe möglich sein soll [int]
android:endYear	Das letzte Jahr, das als Eingabe möglich sein soll [int]
android:maxDate	Das minimal angezeigte Datum im CalendarView (Format: mm/dd/yyyy) [string]
android:minDate	Das maximal angezeigte Datum im CalendarView (Format: mm/dd/yyyy) [string]
android:spinnersShown	Gibt an ob Spinner angezeigt werden sollen [boolean]
android:calendarViewShown	Gibt an ob ein CalendarView angezeigt werden soll [boolean]

Deklaration im Layout

```
<DatePicker
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:endYear="2030"
    android:minDate="11/01/2012"
    android:calendarViewShown="true" />
```

Listing : DatePicker

Screenshot


5554:TestDevice
3G
4:36
HelloView
September 2013
S M T W T F S
Aug 21
Sep 22
Oct 23
36 1 2 3 4 5 6 7
37 8 9 10 11 12 13 14
38 15 16 17 18 19 20 21
39 22 23 24 25 26 27 28
40 29 30 1 2 3 4 5
41 6 7 8 9 10 11 12
MENU

ViewGroup-Klassen

Contents

23 LinearLayout

24 RelativeLayout

25 FrameLayout

26 TableLayout

27 ScrollView

Allgemeines

- Ermöglicht eine horizontale oder vertikale Anordnung seiner Kindelemente
- Reihenfolge in der Definition der Kindelemente wird eingehalten
- Kann als mehrerer Spinner oder Kalender (*CalendarView*) angezeigt werden
- Unterstützt Gewichtungen von Kindelementen (*android:layout_weight*)

Attribut	Beschreibung
android:divider	Definiert ein Drawable, das als Trenner zwischen vertikal angeordneten Elementen angezeigt wird [drawable]
android:gravity	Definiert, wie der Inhalt des Layouts angeordnet werden soll [int]
android:measureWithLargestChild	Falls dieses Flag aktiviert wird, werden alle Kindelemente mit gewichtung dazu gezwungen ihre minimale Größe an die des größten Elements anzupassen[boolean]
android:orientation	Legt fest, ob die Kindelemente horizontal oder vertikal angeordnet werden [int]
android:weightSum	Legt die maximale Summe der Gewichtungen fest

Gewichtung

Layouts gewichten

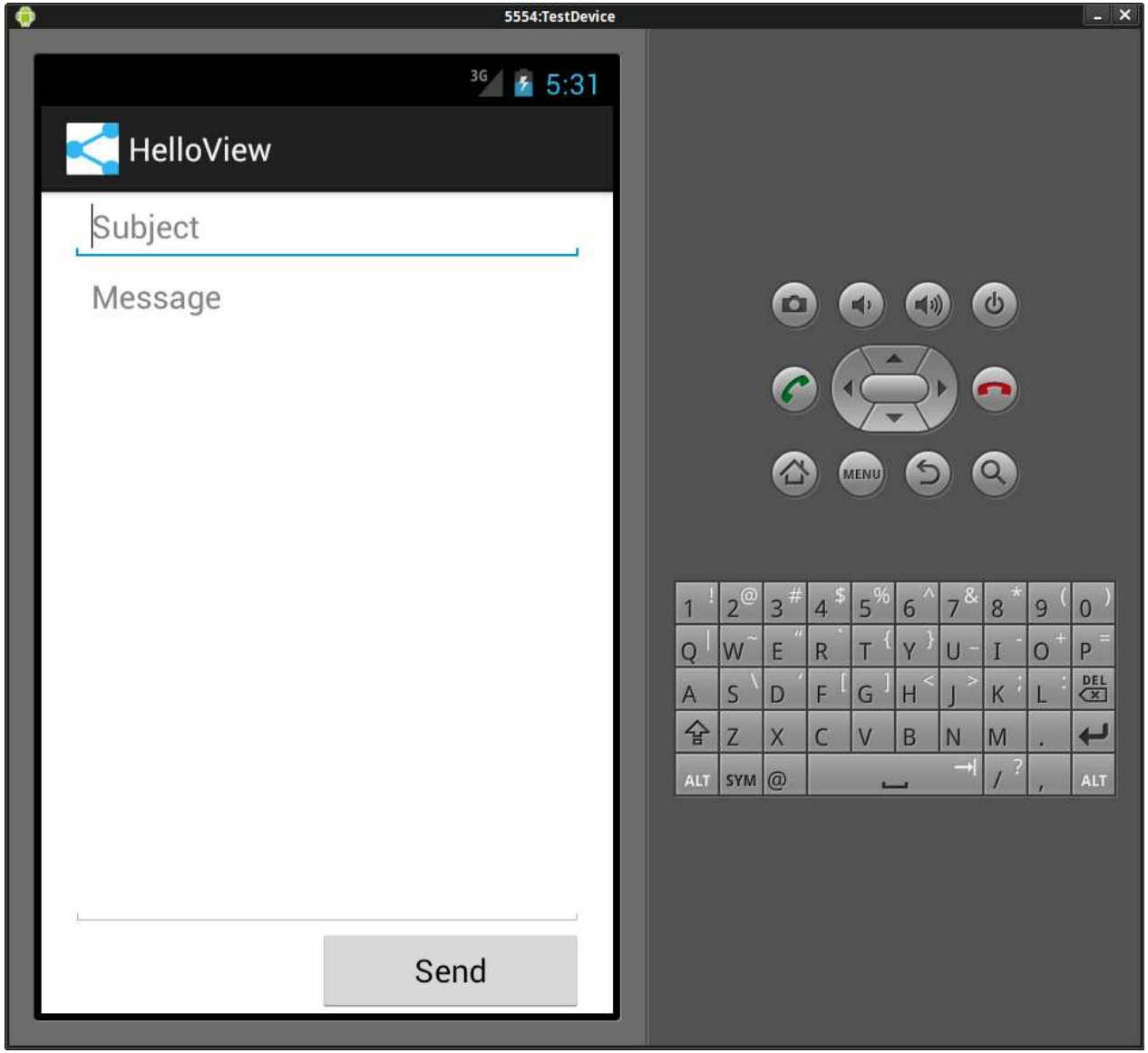
Um allen Kindelementen eines LinearLayouts den gleichen Raum zu geben kann die Breite des Kindelements im horizontalen Layout auf *0dp*, im vertikalen Layout die Höhe auf *0dp* gesetzt werden. Damit dies jedoch funktioniert müssen alle Kindelement mit 1 gewichtet werden. Dazu setzt man das Attribut *android:layout_weight* jedes Kindelemnts auf 1.

Layout

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <EditText
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:hint="Subject" />
   <EditText
      android:layout_width="fill_parent"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:gravity="top"
      android:hint="Message" />
   <Button
      android:layout_width="150dp"
      android:layout_height="wrap_content"
      android:text="Send" />
</LinearLayout>
```

Listing : LinearLayout

Screenshot


5554:TestDevice
3G
5:31
HelloView
Subject
Message
Send
MENU

Allgemeines

- Ordnet seine Kindelemente relativ zueinander an
- View kann seine Position relativ zu anderen Views auf der selben Ebene oder zum Vater-Element angeben kann

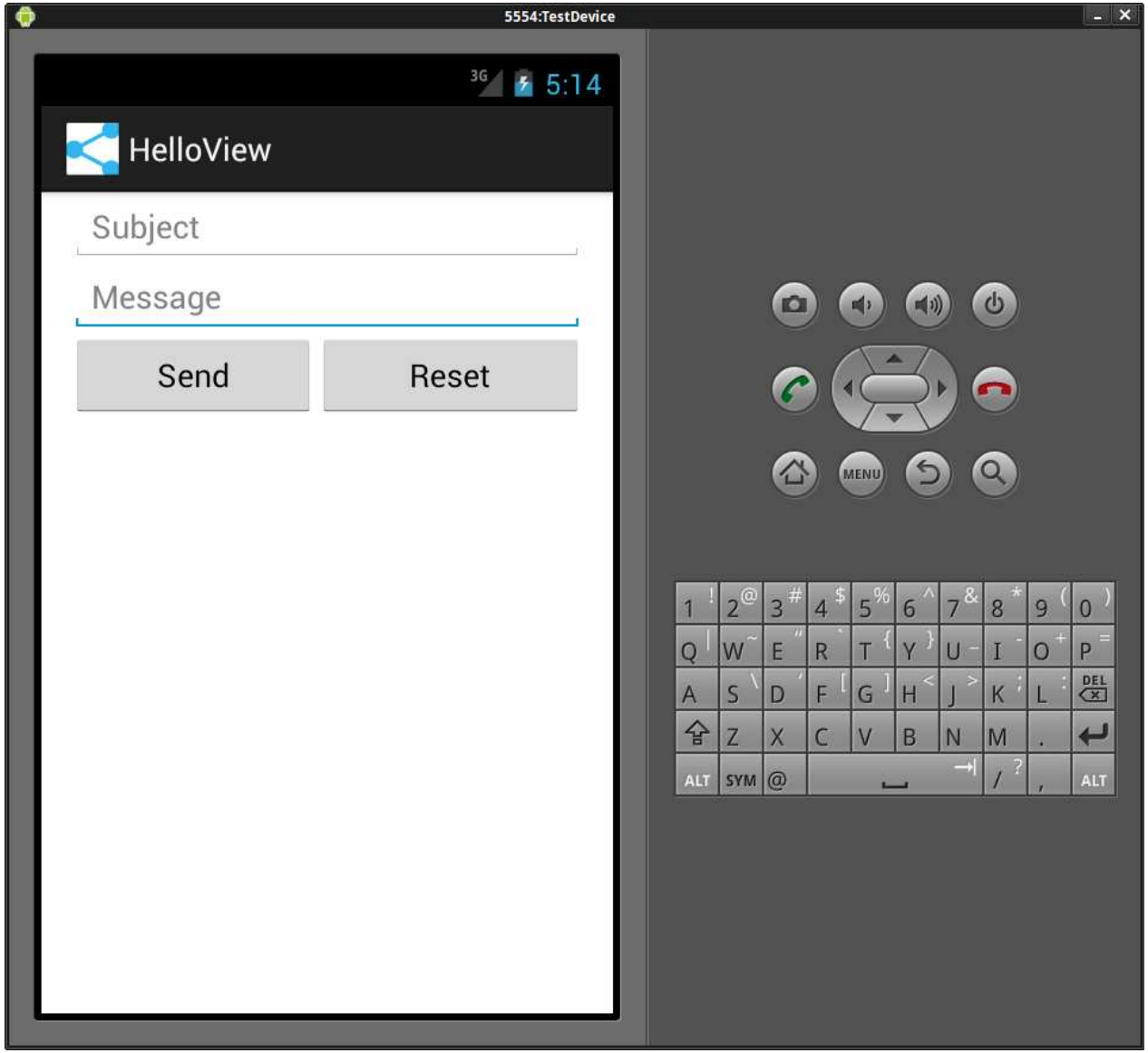
Attribut	Beschreibung
android:gravity	Legt fest, wie der Inhalt innerhalb des Views positioniert werden soll [int]
android:ignoreGravity	Legt fest welches View das *android:gravity* Attribut ignoriert [int]
android:layout_alignParentTop	Legt die Oberkante des Kindelements an die Oberkante des Vaters [boolean]
android:layout_centerInParent	Zentriert das Kindelement vertikal und horizontal in seinem Vater[boolean]
android:layout_centerHorizontal	Zentriert das Kindelement horizontal in seinem Vater[boolean]
android:layout_centerVertical	Zentriert das Kindelement vertikal in seinem Vater[boolean]
android:layout_above	Positioniert das Element über dem View dessen ID angegeben wurde [resource]
android:layout_below	Positioniert das Element unter dem View dessen ID

Deklaration im Layout

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <Button
        android:id="@+id/reset"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/message"
        android:text="Reset" />
    <EditText
        android:id="@id/message"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_below="@+id/subject"
        android:hint="Message"
        android:gravity="top" />
    <Button
        android:id="@+id/send"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/message"
        android:layout_toLeftOf="@id/reset"
        android:text="Send" />
    <EditText
        android:id="@id/subject"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:hint="Subject" />
</RelativeLayout>
```

Listing: RelativeLayout

Screenshot


5554:TestDevice
3G
5:14
HelloView
Subject
Message
Send
Reset
MENU

Allgemeines

- Hält einen gewissen Bereich auf dem Bildschirm für ein einzelnes View frei
- Mehrere Kindelemente in einem FrameLayout können sich überlappen
- Größe des FrameLayouts wird durch größtes Kindelement festgelegt (zzgl. Padding)
- Egal ob Kinder sichtbar oder nicht

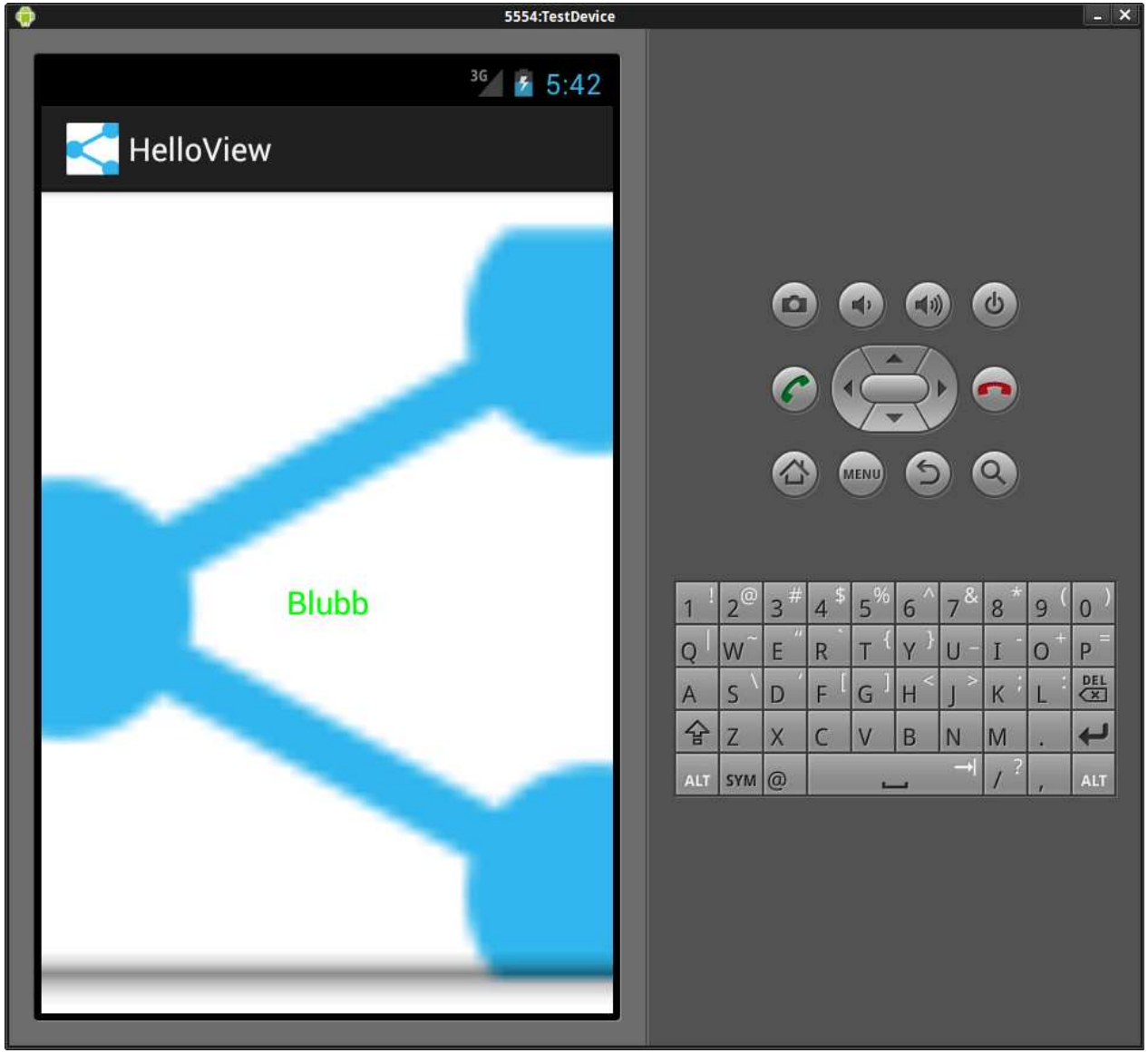
Attribut	Beschreibung
android:foreground	Legt ein Bild fest, dass über dem Inhalt gezeichnet werden soll [drawable]
android:foregroundGravity	Legt die Ausrichtung des Bildes fest [int]
android:measureAllChildren	Legt fest ob die Bemaßung aller Kindelemente angepasst werden soll [boolean]

Deklaration im Layout

```
<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <ImageView
        android:src="@drawable/ic_launcher"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:scaleType="centerCrop" />
    <TextView
        android:text="Blubb"
        android:textColor="#00ff00"
        android:textSize="18dp"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:gravity="center" />
</FrameLayout>
```

Listing : FrameLayout

Screenshot



Allgemeines

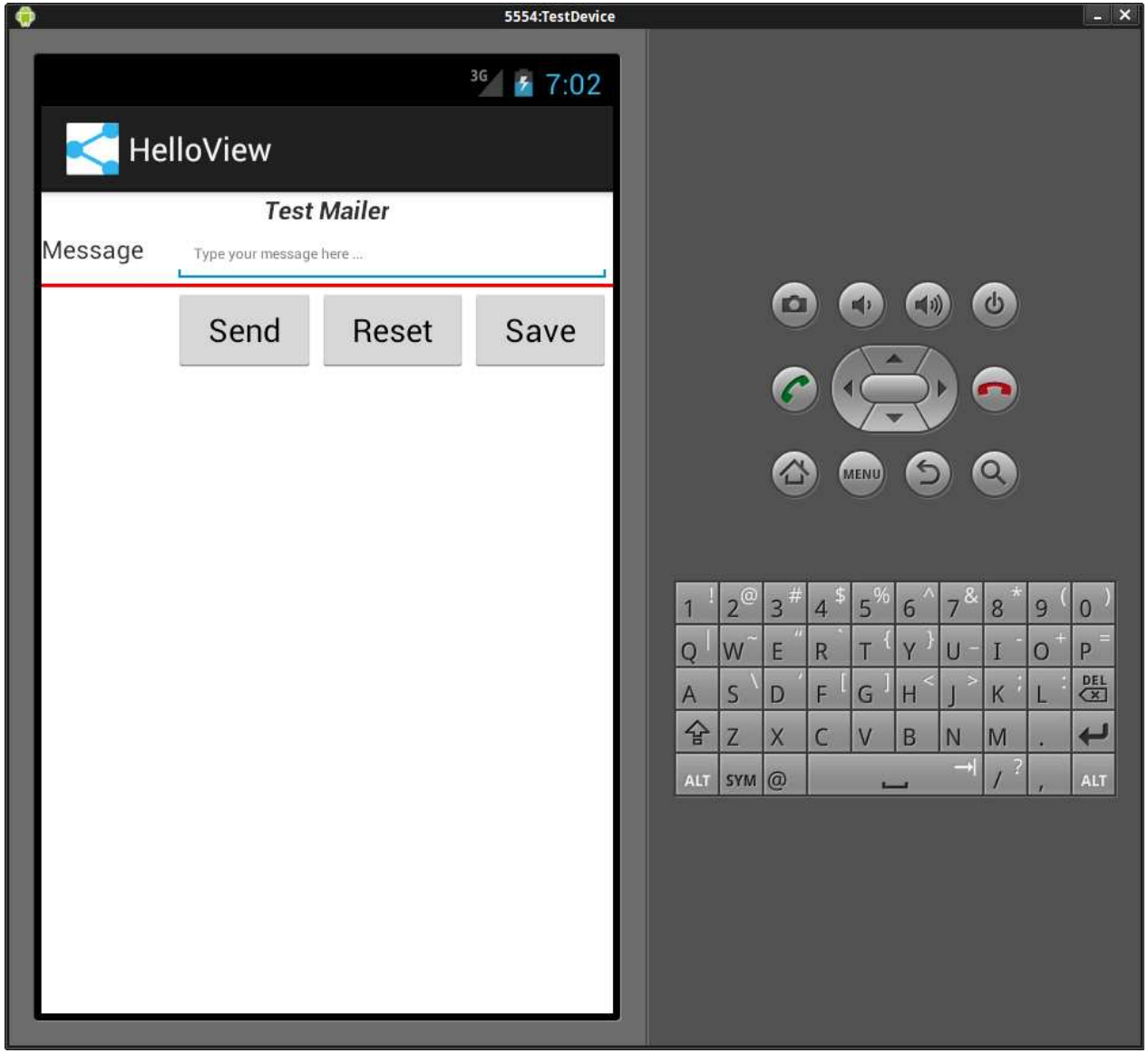
- Platziert Inhalte in einer Tabellenstruktur
- Zeilen werden explizit angegeben, Spalten implizit angenommen werden
- Größe des FrameLayouts wird durch größtes Kindelement festgelegt (zzgl. Padding)
- Egal ob Kinder sichtbar oder nicht

Attribut	Beschreibung
android:collapseColumns	Index der Spalten die zusammengefügt werden sollen (beginnend bei Null) [int,boolean]
android:shrinkColumns	Spalten werden automatisch verkleinert [boolean]
android:stretchColumns	Spalten werden automatisch vergrößert [boolean]
android:layout_column	Spalte in der ein View eingefügt werden soll [int]
android:layout_span	Anzahl an Spalten die ein View einnehmen soll [int]

Deklaration im Layout

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="*"
     android:stretchColumns="*" >
    <TableRow
         android:layout_width="match_parent"
         android:layout_height="wrap_content">
       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Test Mailer"
          android:layout_gravity="center_vertical|center_horizontal"
          android:layout_span="4"
          android:textSize="14dp"
          android:textStyle="bold|italic" />
    </TableRow>
    <TableRow
         android:layout_width="match_parent"
         android:layout_height="wrap_content">
       <TextView
          android:text="Message"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />
       <EditText
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:hint="Type your message here ..."
          android:layout_span="3"
          android:textSize="8dp" />
    </TableRow>

     <!-- just draw a red line -->
     <View
          android:layout_height="2dip"
```

Screenshot


5554:TestDevice
3G
7:02
HelloView
Test Mailer
Message
Type your message here ...
Send
Reset
Save
MENU

Allgemeines

- ScrollView von FrameLayout abgeleitet
- Erlaubt, dass das Layout größer ist als das eigentliche Display erlaubt
- Kann nur ein Kind enthalten
- Egal ob Kinder sichtbar oder nicht

ListViews

Auch wenn dies möglich ist, sollte man ein ScrollView niemals mit einem ListView verwenden, da sich dies bereits selbst um das scrollen seiner Elemente kümmert. Sollte man dies dennoch tun, zwingt man das ListView dazu sich auf die benötigte länge auszudehnen um alle Elemente anzeigen zu können. Dies würde dafür sorgen, dass die in ListView implementierten Optimierungen zur Anzeige großer Listen umgangen werden.

Attribute

Attribut	Beschreibung
android:fillViewport	Ermöglicht es den Inhalt so auszudehnen, dass er den verfügbaren Display-Bereich nutzt [boolean]

Horizontales Scrollen

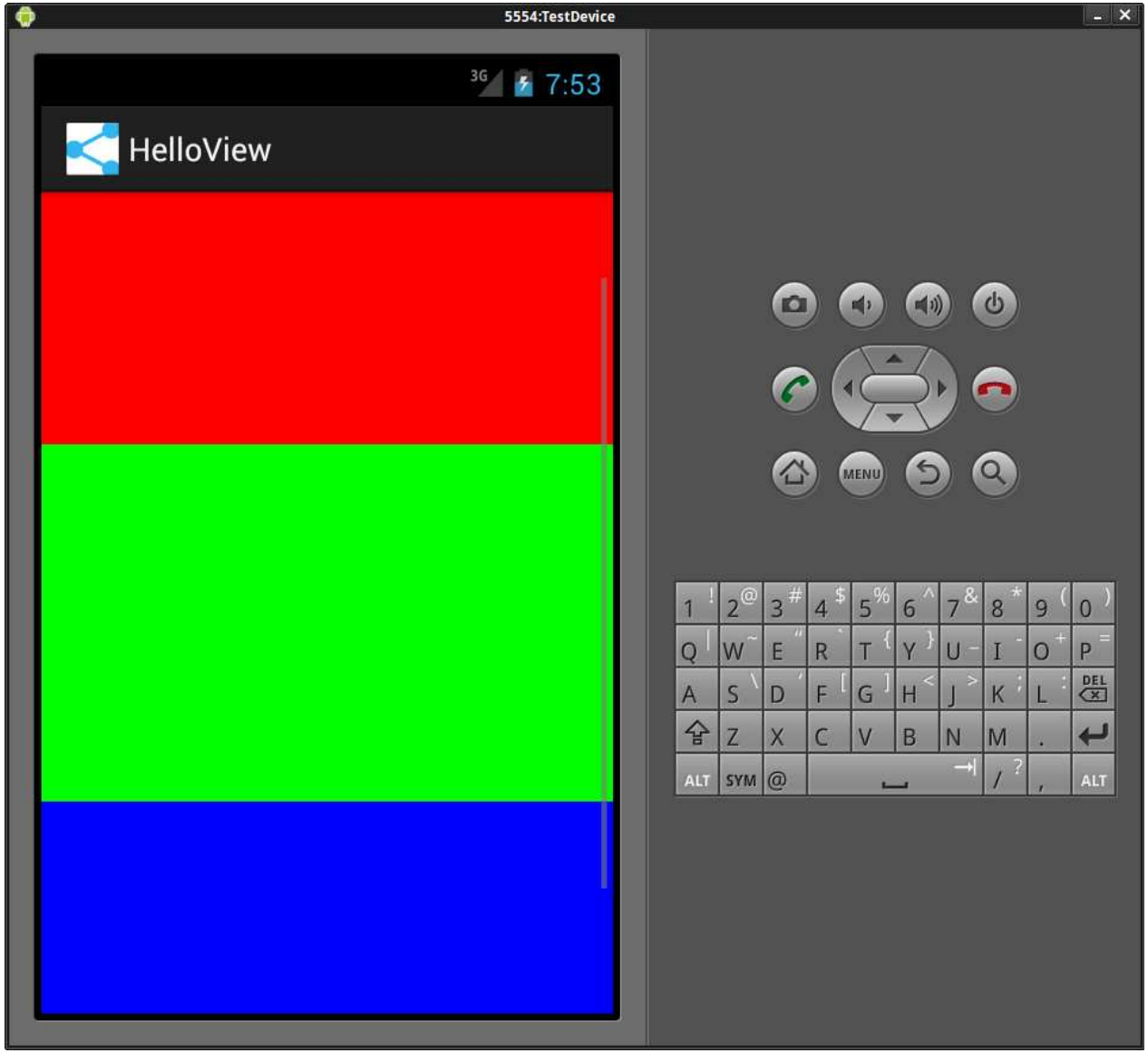
Ein ScrollView kümmert sich nur um das vertikale Scrollen seines Inhalts. Sollte es einmal nötig sein den Inhalt horizontal zu scrollen, so sollte man auf die Klasse HorizontalScrollView zurückgreifen.

Deklaration im Layout

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <TextView
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:background="#ff0000" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:background="#00ff00" />
        <TextView
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:background="#0000ff" />
    </LinearLayout>
</ScrollView>
```

Listing : ScrollView

Screenshot



AdapterView-Klassen

Contents

28 ListView

29 GridView

30 ExpandableListView

ListView

- Scrollbare Liste von Einträgen
- Nutzt spezielle Aktivität ListActivity
- ListActivity nutzt automatisch ein Standard-Layout
- Eigenes Layout kann mit *setContentView()* gesetzt werden
- Liste muss ID *@android:id/list* tragen
- View für leere Listen muss ID *@android:id/empty* tragen

ListViews & ScrollView

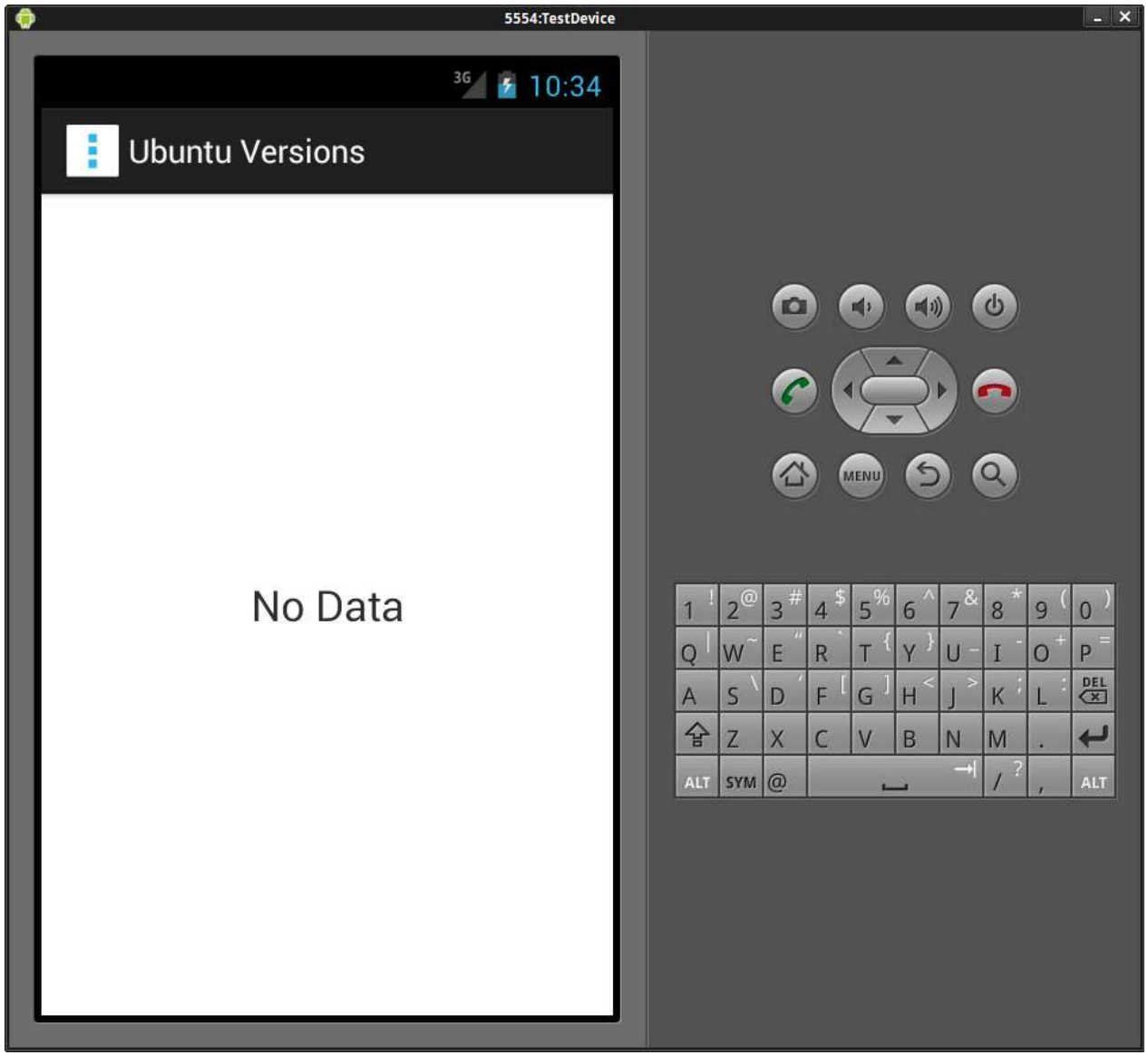
Auch wenn dies möglich ist, sollte man ein ScrollView niemals mit einem ListView verwenden, da sich dies bereits selbst um das scrollen seiner Elemente kümmert. Sollte man dies dennoch tun, zwingt man das ListView dazu sich auf die benötigte länge auszudehnen um alle Elemente anzeigen zu können. Dies würde dafür sorgen, dass die in ListView implementierten Optimierungen zur Anzeige großer Listen umgangen werden.

Deklaration im Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="27dp" />
    <TextView
        android:id="@android:id/empty"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center_vertical|center_horizontal"
        android:textSize="24dp" />
</RelativeLayout>
```

Listing : ListView

Screenshot



ListActivity

```
public class HelloList extends ListActivity {
   private ArrayAdapter<CharSequence> adapter;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.hello_list);

      // Setup the adapter ...
      adapter = ArrayAdapter.createFromResource(this, R.array.ubuntu_array,
            android.R.layout.simple_list_item_1);

      // ... and assign it to the listview.
      this.setListAdapter(adapter);
   }

   @Override
   protected void onListItemClick(ListView l, View v, int position, long id) {
      // Load the selected item ...
      String ubuntu = (String) this.adapter.getItem(position);

      // ... and show a short Toast.
      Toast.makeText(this, "Selected " + ubuntu + ".",
            Toast.LENGTH_LONG).show();
   }
}
```

Listing : Die Klasse HelloList

Resourcen

Array-Resourcen

Wie Zeichenketten, Bemaßungen und Styles können auch Arrays als Resourcen hinterlegt werden.

Arrays werden wie gewohnt in XML deklariert. Als Werte des Arrays können praktisch beliebige Resource-Typen, wie Strings, Integer oder Drawables dienen. Dabei können auch die Typen von verschiedenen Resourcen vermischt werden. Die entstehende Datei kann unter *res/values/arrays.xml* abgelegt werden.

```
<resources>
    <array name="game_icons">
        <item>@drawable/wow</item>
        <item>@drawable/diablo</item>
        <item>@drawable/cs</item>
    </array>
    <array name="ubuntu_array">
        <item>Ubuntu 4.10 (Warty Warthog)</item>
        <item>Ubuntu 5.04 (Hoary Hedgehog)</item>
        <item>Ubuntu 5.10 (Breezy Badger)</item>
        ...
    </array>
</resources>
```

Listing : Array Resourcen

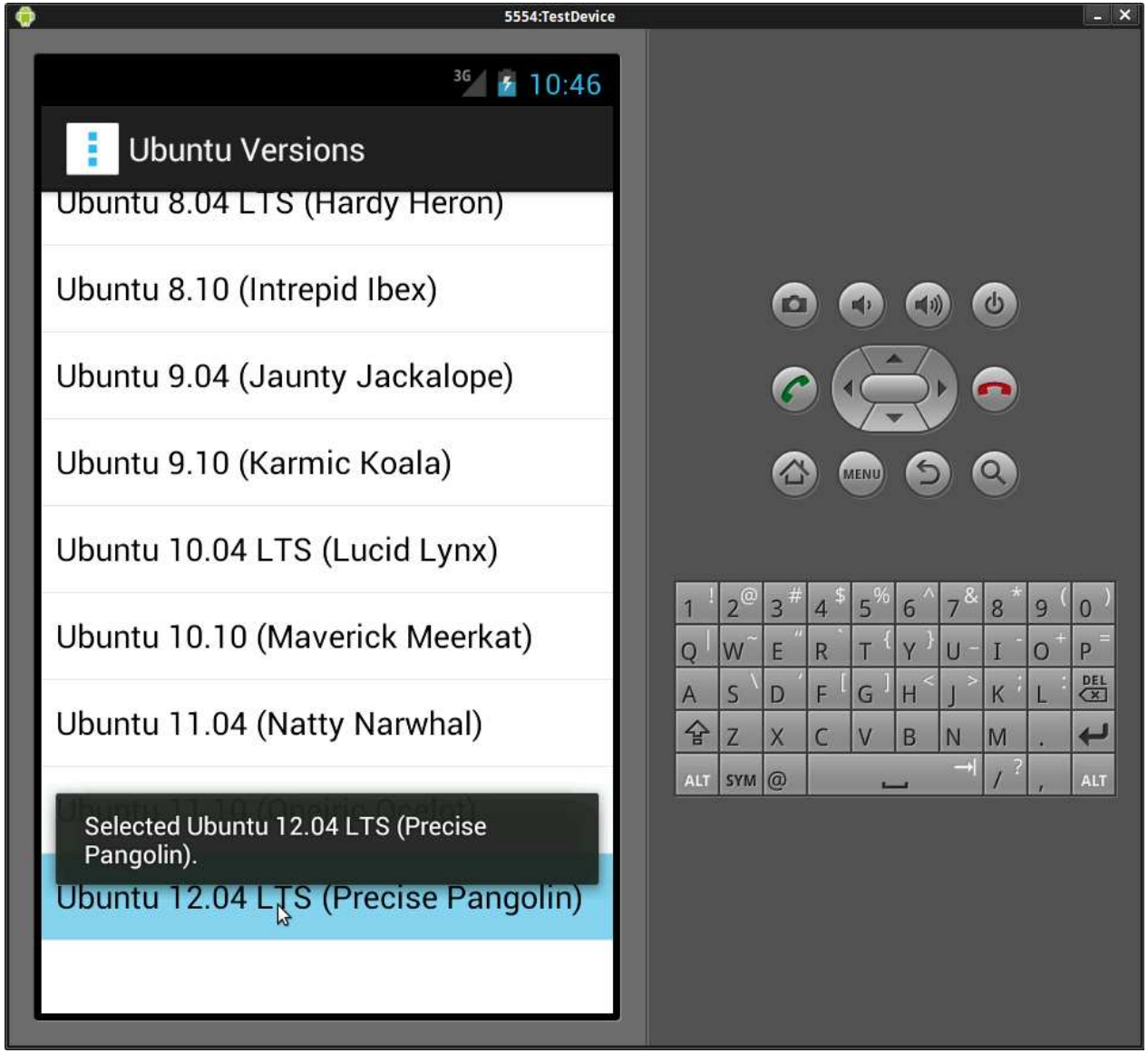
Resourcen

Android-Layouts

Beim betrachten der Klasse HelloList sollten auffallen, dass ein Layout referenziert wird (*simple_list_item_1*), das bisher nicht erstellt wurde.

Es handelt sich um ein von Android bereitgestelltes Standard-Layout, das mit dem Android-SDK mitgeliefert wird. Es gibt weitere Layouts, wie *alert_dialog*, *date_picker*, *search_bar* und auch *simple_list_item_1*. Die Layouts findet man unter *<path-to-sdk>/platforms/<android-platform>/data/res/layout*.

Screenshot II



Allgemeines

- GridView ist eine von ListView abgeleitete ViewGroup
- Zweidimensionales Gitter bereitstellt, das auch scrollbar

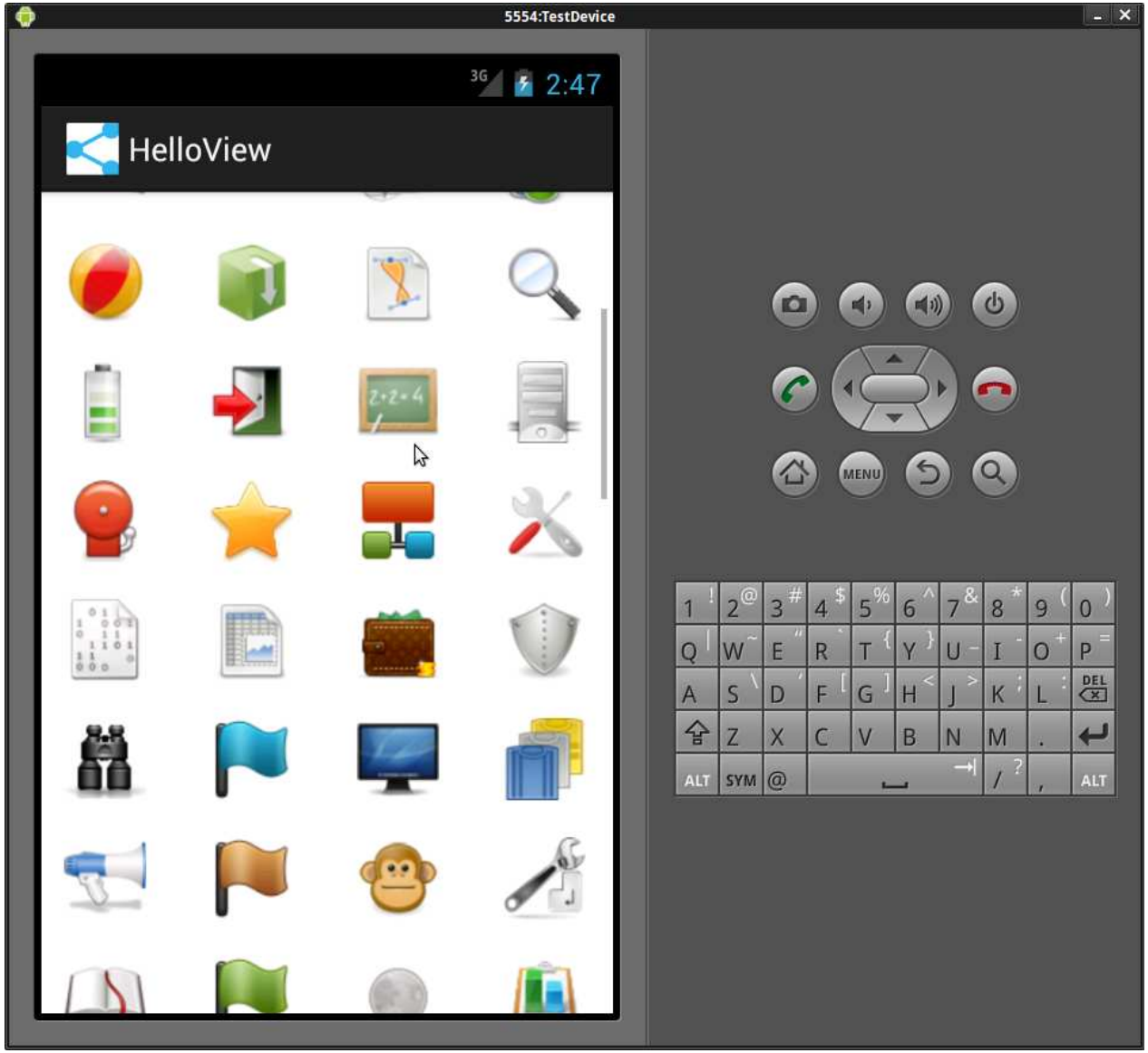
Attribut	Beschreibung
android:columnWidth	Breite einer Spalte [dimension]
android:gravity	Positionierung des Inhalts in einer Zelle [int]
android:numColumns	Anzahl der anzuzeigenden Spalten [int]
android:stretchMode	Legt fest, wie der Inhalt einer Zelle ausgedehnt werden soll, um die Zelle komplett auszufüllen [int]
android:horizontalSpacing	Horizontaler Abstand zwischen den Zellen [dimension]
android:verticalSpacing	Vertikaler Abstand zwischen den Zellen [dimension]

Deklaration im Layout

```
<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/icongrid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="70dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center" />
```

Listing : GridView

Screenshot



ExpandableListView

- Scrollbare Liste von Einträgen (wie ListView)
- Einzelne Einträge können aufgeklappt werden

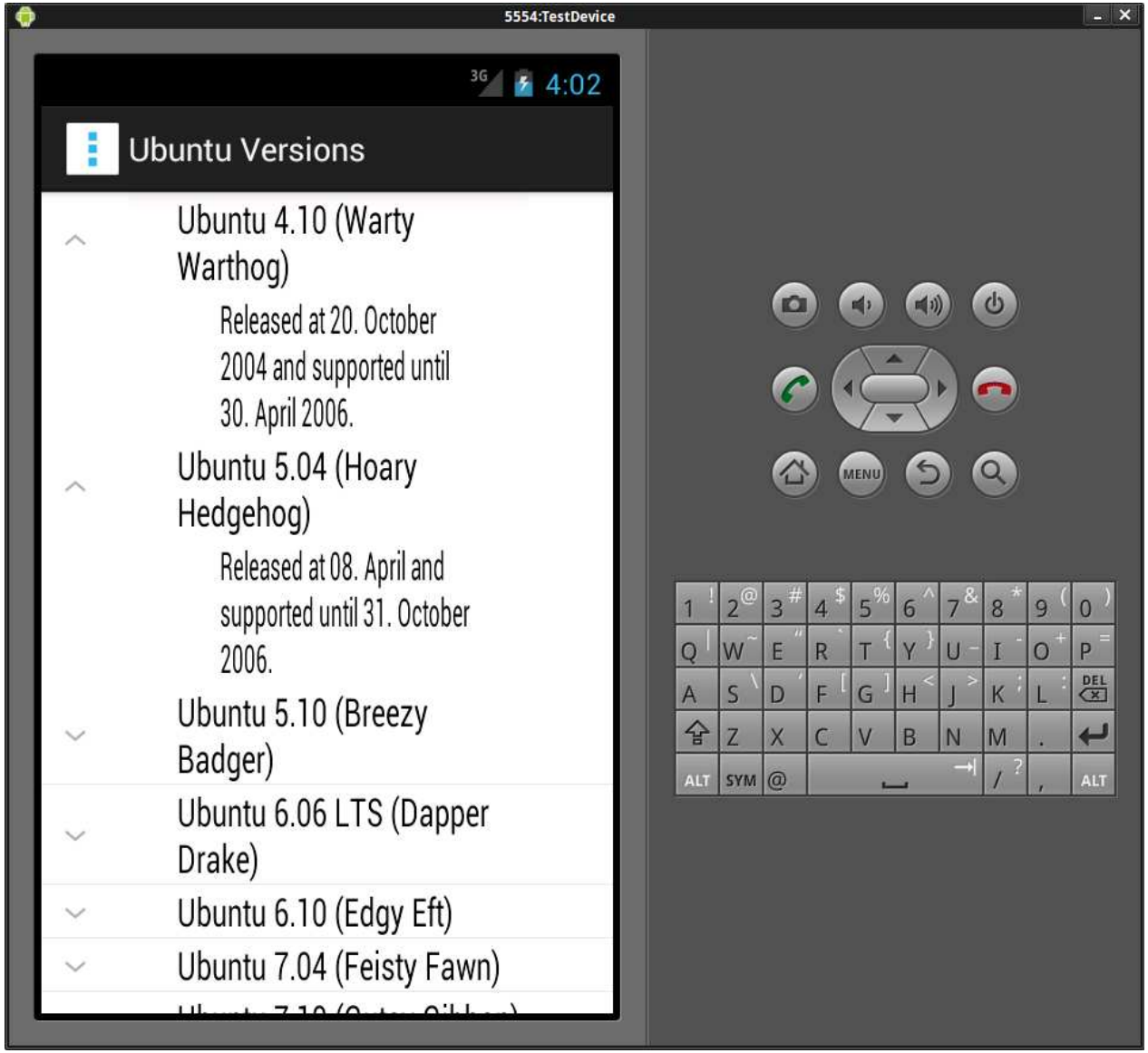
Attribut	Beschreibung
android:childDivider	Drawable oder eine Farbe zur Trennung der Kinder [resource]
android:childIndicator	Bild das neben dem Kindelement angezeigt wird [drawable]
android:childIndicatorLeft	Linke Begrenzung des Bildes für das Kindelement [int]
android:childIndicatorRight	Rechte Begrenzung des Bildes für das Kindelement [int]
android:groupIndicator	Bild das neben dem Gruppenelement angezeigt wird [drawable]
android:indicatorLeft	Linke Begrenzung des Bildes für das Gruppenelement [int]
android:indicatorRight	Linke Begrenzung des Bildes für das Gruppenelement [int]

Deklaration im Layout

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ExpandableListView
        android:id="@id/android:list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <TextView
        android:id="@id/android:empty"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="No Data"/>
</LinearLayout>
```

Listing : ExpandableListView

Screenshot


5554:TestDevice
3G
4:02
Ubuntu Versions
Ubuntu 4.10 (Warty Warthog)
Released at 20. October 2004 and supported until 30. April 2006.
Ubuntu 5.04 (Hoary Hedgehog)
Released at 08. April and supported until 31. October 2006.
Ubuntu 5.10 (Breezy Badger)
Ubuntu 6.06 LTS (Dapper Drake)
Ubuntu 6.10 (Edgy Eft)
Ubuntu 7.04 (Feisty Fawn)
MENU
ALT
SYM
ALT

Adapter

Contents

Allgemeines

- Von AdapterView abgeleitete Views müssen einen Adapter verwenden
- Adapter agiert als Brücke zwischen dem View und den anzuzeigenden Daten
- Standard-Adapter, wie ArrayAdapter, CursorAdapter oder SimpleCursorAdapter
- Adapter-Interfaces, wie ExpandableListAdapter
- Zuweisung eines Adapters mit *setAdapter*
- Änderung an den Daten können über *notifyDataSetChanged()* mitgeteilt werden
- Allgemeine Klasse BaseAdapter bietet gute Grundlage

Methode	Beschreibung
int getCount()	Anzahl der Elemente
Object getItem(int position)	Zugriff auf die Elemente anhand ihrer Position
long getItemId(int position)	Zugriff auf die ID der Elemente anhand ihrer Position
View getView(int position, View convertView, ViewGroup parent)	Bereitstellung des Layouts für ein Element anhand der Position

Deklaration eines Layouts

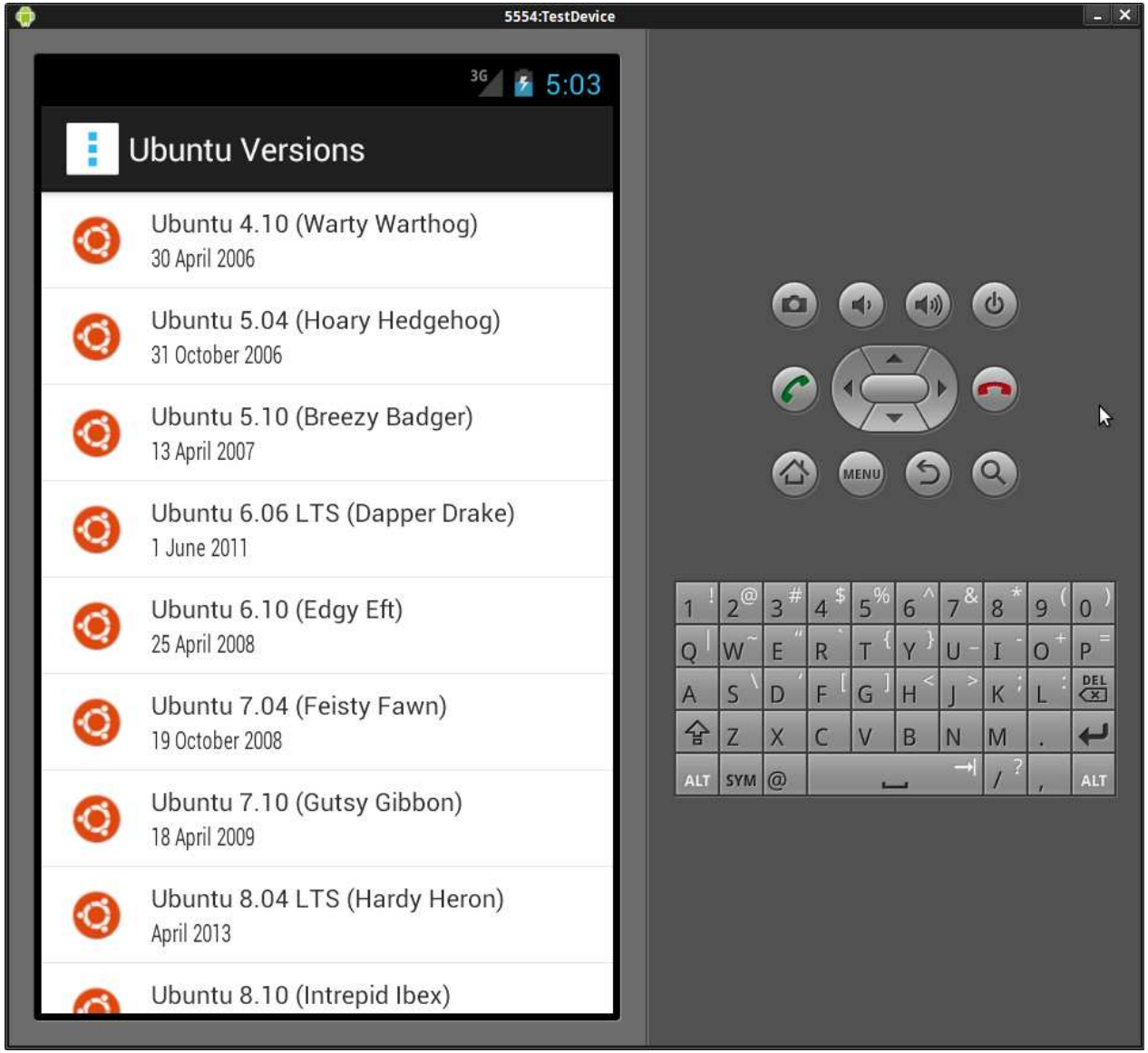
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="7dip">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:src="@drawable/ubuntu"
        android:padding="3dip" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/versionName"
            android:gravity="left|center_vertical"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:padding="7dip"
            android:singleLine="true" />
        <TextView
            android:id="@+id/supportDate"
            android:gravity="left|center_vertical"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:padding="7dip"
            android:singleLine="true"
            android:textScaleX="0.7" />
    </LinearLayout>
</LinearLayout>
```

Listing : Das Layout eines Eintrags

Implementierung

```
public class UbuntuAdapter extends BaseAdapter {
   private String[] ubuntuVersions;
   private String[] ubuntuSupport;
   private LayoutInflater inflater;
   private Resources res;

   public UbuntuAdapter(Context c) {
      this.res = c.getResources();
      this.inflater = (LayoutInflater) c.getSystemService(LAYOUT_INFLATER_SERVICE);
      ubuntuVersions = this.res.getStringArray(R.array.ubuntu_array);
      ubuntuSupport = this.res.getStringArray(R.array.ubuntu_support_array);
   }

   public int getCount() {
      return this.ubuntuVersions.length;
   }

   public String getItem(int position) {
      return this.ubuntuVersions[position];
   }

   public long getItemId(int position) {
      return position;
   }

   public View getView(int position, View convertView, ViewGroup parent) {
      View row = this.inflater.inflate(R.layout.row_layout, null);

      TextView versionName = (TextView) row.findViewById(R.id.versionName);
      versionName.setText(this.ubuntuVersions[position]);
      TextView supportDate = (TextView) row.findViewById(R.id.supportDate);
      supportDate.setText(this.ubuntuSupport[position]);

      return row;
   }
}
```

Screenshot


5554:TestDevice
3G
5:03
Ubuntu Versions
Ubuntu 4.10 (Warty Warthog)
30 April 2006
Ubuntu 5.04 (Hoary Hedgehog)
31 October 2006
Ubuntu 5.10 (Breezy Badger)
13 April 2007
Ubuntu 6.06 LTS (Dapper Drake)
1 June 2011
Ubuntu 6.10 (Edgy Eft)
25 April 2008
Ubuntu 7.04 (Feisty Fawn)
19 October 2008
Ubuntu 7.10 (Gutsy Gibbon)
18 April 2009
Ubuntu 8.04 LTS (Hardy Heron)
April 2013
Ubuntu 8.10 (Intrepid Ibex)
MENU
ALT
SYM
ALT