

# Android Programmierung mit Java Studiengang MI

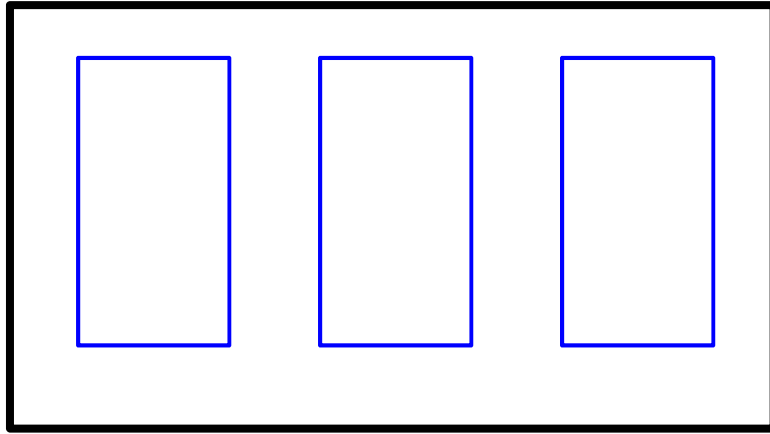
- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- <http://mwilhelm.hs-harz.de>
- Raum 2.202
- Tel. 03943 / 659 338

## Gliederung

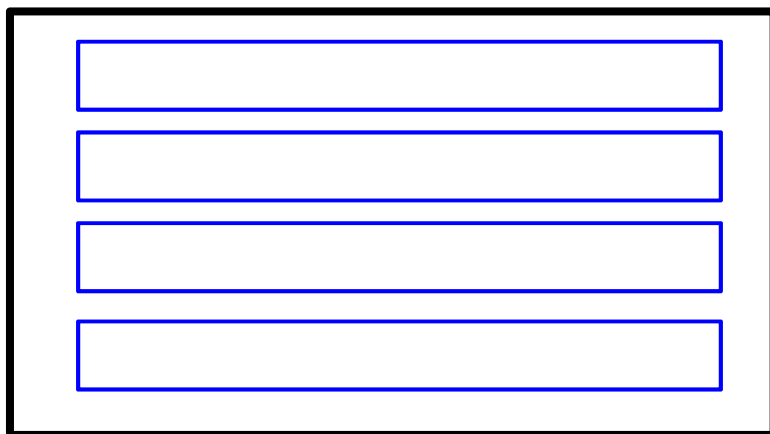
### Überblick:

- Einleitung
- **Layouts**
  - LinearLayout
  - TableLayout
  - GridLayout
  - ConstraintLayout
  - RelativeLayout
  - Fragment
- Single page
- Multi-Page
- I/O
- Datenbanken
- Sensoren
- Web
- Thread, modale Dialoge

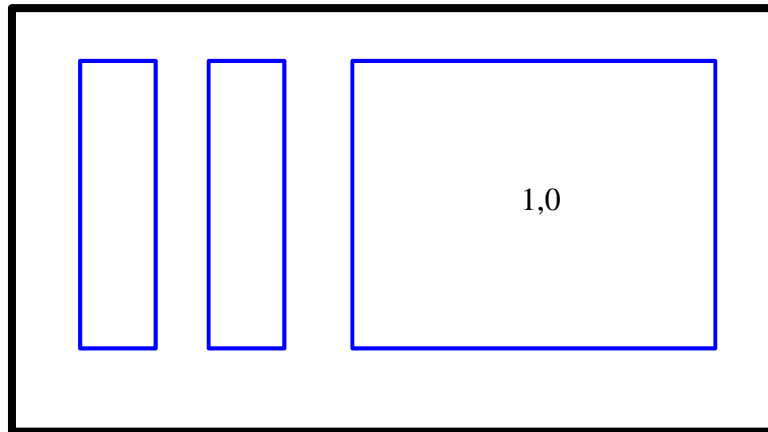
## LinearLayout



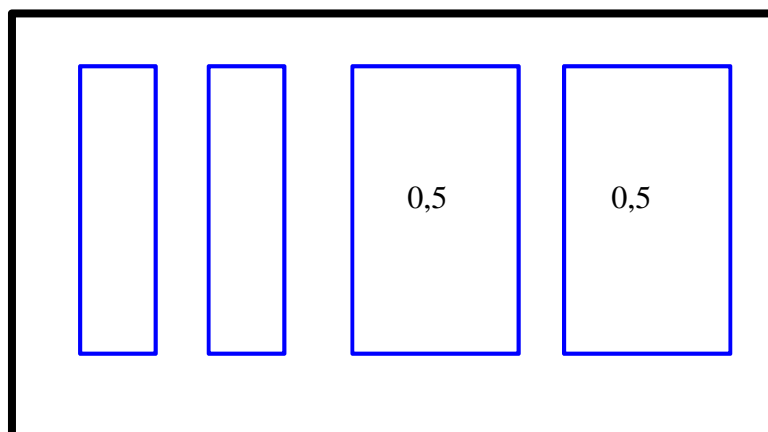
## LinearLayout



LinearLayout: weight=1.0



LinearLayout: weight=0.5, wight=0.5



## LinearLayout: Attribute

- Wie ein Regal, à la StackPanel in WPF
- Orientation
  - horizontal
  - vertikal
- **Geht auch verschachtelt**
- **android:layout\_width=**
  - "match\_parent" komplette Breite
  - "wrap\_content" nur benötigte Breite
- **android:layout\_height=**
  - "match\_parent" komplette Höhe
  - "wrap\_content" nur benötigte Höhe

## LinearLayout: Attribute

### **android:divider**

- Möglichst unsichtbarer Trennstrich.
- Funktioniert aber auch mit Margin

### **android:gravity**

- Definiert die Ausrichtungen in der Zelle.
- fill\_horizontal, fill, bottom, center, center\_horizontal, right,
- clip\_horizontal, clip\_vertical, end, fill\_vertical, left, start, top

### **android:measureWithLargestChild**

- Wenn true, dann erhalten alle gewichteten Elemente die Minimalgröße des größten Elementes.

### **android:weightSum**

- Normalerweise muss die Summe der Gewichtungen 1.0 ergeben. Hiermit kann man die Summe ändern.

## LinearLayout: Aufbau

### <LinearLayout

```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:id="@+id/activity_main"  
android:orientation="horizontal">
```

```
// Elemente
```

```
<TextView />  
<EditText />  
<Switch />
```

### </LinearLayout>

## LinearLayout: Zwei Editoren mit Gewichtung

### <LinearLayout

```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:id="@+id/activity_main"  
android:orientation="horizontal">
```

```
<TextView ... />  
<EditText  
    android:layout_weight="0.67"  
    android:inputType="textMultiLine" />  
<EditText  
    android:layout_weight="0.33"  
    android:inputType="textMultiLine"/>
```

### </LinearLayout>

## LinearLayout: Zwei Editoren mit Gewichtung

### <LinearLayout

```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:weightSum="30"  
android:orientation="horizontal">
```

```
<TextView ... />
```

```
<EditText
```

```
android:layout_weight="20"
```

```
android:inputType="textMultiLine" />
```

```
<EditText
```

```
android:layout_weight="10"
```

```
android:inputType="textMultiLine"/>
```

### </LinearLayout>

## LinearLayout: Gravity-Konstanten

Konstante	Value	Beschreibung
top	0x30	Objekt oben verankert, Größe bleibt.
bottom	0x50	Objekt unten verankert, Größe bleibt.
left	0x03	Objekt links verankert, Größe bleibt.
right	0x05	Objekt rechts verankert, Größe bleibt.
center_vertical	0x10	Objekt vertikal zentriert, Größe bleibt.
fill_vertical	0x70	Objekt wird komplett gestreckt.
center_horizontal	0x01	Objekt horizontal zentriert, Größe bleibt.
fill_horizontal	0x07	Objekt wird komplett gestreckt.
center	0x11	Objekt vertikal und horizontal zentriert, Größe bleibt.
fill	0x77	Objekt wird komplett gestreckt.
clip_vertical	0x80	Zusätzliche Option. Die Ecken verschmelzen mit dem Parent.
clip_horizontal	0x08	Zusätzliche Option. Die Ecken verschmelzen mit dem Parent.
start	0x00800003	Setzt Objekt ganz oben hin, Größe bleibt.
end	0x00800005	Setzt Objekt ganz unten hin, Größe bleibt.

## TableLayout

- Das TableLayout funktioniert wie die HTML-Tabelle
- Man definiert jeweils eine Zeile mit TableRow
- Spalten zusammenfügen
  - android:layout\_span="2,,
- Spalten direkt adressieren
  - android:layout\_column="2"

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/tableLayout1"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >

  <TableRow
    android:id="@+id/tableRow1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dip" >

    <TextView ... />
    <TextView ... />
    <TextView ... />

  </TableRow>

</TableLayout>
```

**▲ Hochschule Harz** FB Automatisierung und Informatik: Android 15

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tableLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <!-- 2 columns -->
    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dip" >

        <TextView
            android:id="@+id/textView1"
            android:text="Vorname"
            android:layout_margin="5dp"
            android:textSize="24dp"
            android:textColor="@color/colorAccent"
            android:textAppearance="?android:attr/textAppearanceLarge" />

```

**▲ Hochschule Harz** FB Automatisierung und Informatik: Android 16



```

<TextView
    android:id="@+id/textView2"
    android:text="Nachname"
    android:layout_margin="5dp"
    android:textSize="24dp"
    android:textColor="@color/colorAccent"
    android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
    android:id="@+id/textView3"
    android:text="Note"
    android:layout_margin="5dp"
    android:textSize="24dp"
    android:textColor="@color/colorAccent"
    android:textAppearance="?android:attr/textAppearanceLarge" />
</TableRow>

```

```

<TableRow
    android:id="@+id/tableRow2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="5dip" >

    <EditText
        android:id="@+id/editText1"
        android:text="Andrea" />
    <EditText
        android:id="@+id/editText2"
        android:text="Musterfrau" />
    <EditText
        android:id="@+id/editText3"
        android:text="2.3" />
</TableRow>

```

```

<TableRow
  android:id="@+id/tableRow3"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="5dip" >

  <EditText
    android:id="@+id/editText3a"
    android:text="Max" />
  <EditText
    android:id="@+id/editText3b"
    android:text="Mustermann" />
  <EditText
    android:id="@+id/editText3c"
    android:text="2.7" />
</TableRow>
<!-- just draw a red line -->
<View
  android:layout_height="2dip"
  android:background="#FF0000" />

```

```

<TableRow
  android:id="@+id/tableRow4"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="5dip" >
  <Button
    android:id="@+id/button4"
    android:layout_column="2"
    android:text="Spalte 3" />
</TableRow>
<TableRow
  android:id="@+id/tableRow5"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="5dip" >
  <Button
    android:id="@+id/button5"
    android:layout_column="1"
    android:text="Spalte 2" />
</TableRow>
</TableLayout>

```

## GridLayout

- Ähnlich wie `TabletLayout`
- **Vorab** werden aber die Spalten und Zeilen definiert
- Die Zelle jedes UI-Elementes wird über `column` und `row` definiert. Ähnlich dem `GridBagLayout`.
- Mit dem Attribut „gravity“ bestimmt man die Platzierung innerhalb der Zelle. Ähnlich dem `GridBagLayout`.
  - Dazu gibt es Konstanten
- Das Zusammenfassen von Zellen geht mit:
  - `columnSpan`
  - `rowSpan`

## GridLayout: Gravity-Konstanten

Konstante	Value	Beschreibung
<code>top</code>	<code>0x30</code>	Objekt oben verankert, Größe bleibt.
<code>bottom</code>	<code>0x50</code>	Objekt unten verankert, Größe bleibt.
<code>left</code>	<code>0x03</code>	Objekt links verankert, Größe bleibt.
<code>right</code>	<code>0x05</code>	Objekt rechts verankert, Größe bleibt.
<code>center_vertical</code>	<code>0x10</code>	Objekt vertikal zentriert, Größe bleibt.
<code>fill_vertical</code>	<code>0x70</code>	Objekt wird komplett gestreckt.
<code>center_horizontal</code>	<code>0x01</code>	Objekt horizontal zentriert, Größe bleibt..
<code>fill_horizontal</code>	<code>0x07</code>	Objekt wird komplett gestreckt.
<code>center</code>	<code>0x11</code>	Objekt vertikal und horizontal zentriert, Größe bleibt.
<code>fill</code>	<code>0x77</code>	Objekt wird komplett gestreckt.
<code>clip_vertical</code>	<code>0x80</code>	Zusätzliche Option. Die Ecken verschmelzen mit dem Parent.
<code>clip_horizontal</code>	<code>0x08</code>	Zusätzliche Option. Die Ecken verschmelzen mit dem Parent.
<code>start</code>	<code>0x00800003</code>	Setzt Objekt ganz oben hin, Größe bleibt.
<code>end</code>	<code>0x00800005</code>	Setzt Objekt ganz unten hin, Größe bleibt.

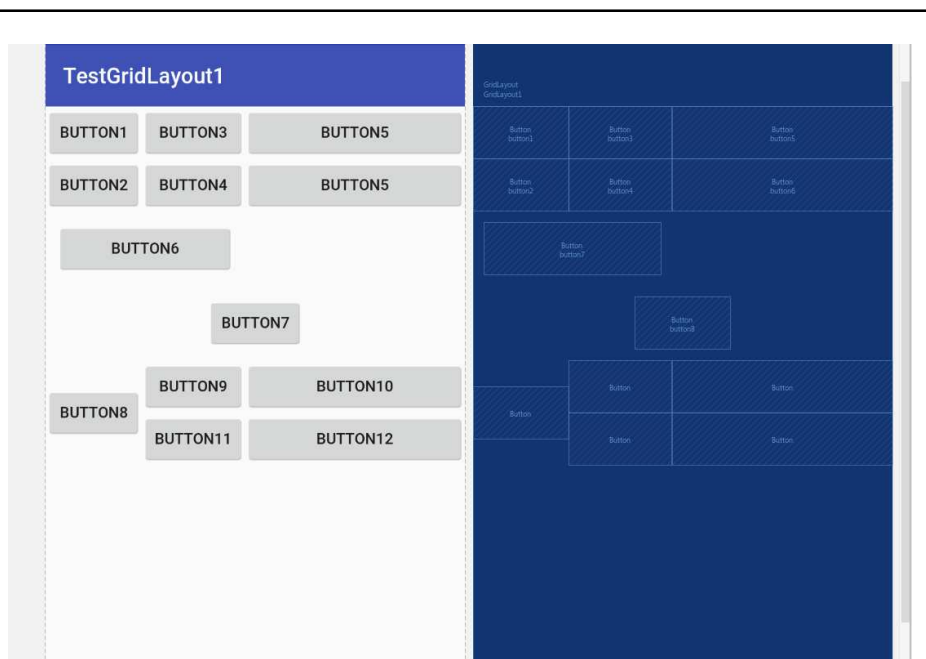
```

<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/GridLayout1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:columnCount="3"
  android:rowCount="7"
  android:orientation="vertical"
  tools:context=".GridXMLActivity" >

  <Button ...
    android:layout_column="0"
    android:layout_row="0"
    android:layout_gravity="fill_horizontal"/>

</GridLayout>

```



## GridLayout

### Bemerkungen zum Beispiel

- Der Button „Button8“ hat ein RowSpan.
- Wenn aber keine weitere Reihe danach kommt, geht dieser Schalter dann bis zum Ende des Views.
- Abhilfe:
  - Man trägt eine zusätzliche Reihe ein.
  - Diese braucht kein Element haben !
- Button8:
  - android:layout\_gravity="center\_vertical"

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/GridLayout1"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:columnCount="3"
  android:rowCount="7"
  android:orientation="vertical"
  tools:context=".GridXMLActivity" >
  <Button
    android:id="@+id/button1"
    android:layout_column="0"
    android:layout_row="0"
    android:layout_gravity="fill_horizontal"
    android:text="Button1" />
  <Button
    android:id="@+id/button2"
    android:layout_column="0"
    android:layout_row="1"
    android:layout_gravity="fill_horizontal"
    android:text="Button2" />
```

```

<Button
    android:id="@+id/button3"
    android:layout_column="1"
    android:layout_row="0"
    android:layout_gravity="fill_horizontal"
    android:text="Button3" />
<Button
    android:id="@+id/button4"
    android:layout_column="1"
    android:layout_row="1"
    android:layout_gravity="fill_horizontal"
    android:text="Button4" />

<Button
    android:id="@+id/button5"
    android:layout_column="2"
    android:layout_row="0"
    android:layout_gravity="fill_horizontal"
    android:text="Button5" />
<Button

```

```

<Button
    android:id="@+id/button7"
    android:layout_column="0"
    android:layout_row="2"
    android:layout_columnSpan="2"
    android:layout_gravity="fill_horizontal"
    android:layout_margin="10dp"
    android:layout_width="wrap_content"
    android:text="Button6" />

<Button
    android:id="@+id/button8"
    android:layout_column="0"
    android:layout_row="3"
    android:layout_columnSpan="3"
    android:layout_gravity="center_horizontal"
    android:layout_margin="10dp"
    android:layout_width="wrap_content"
    android:text="Button7" />

```

```

<Button
    android:layout_column="0"
    android:layout_row="4"
    android:layout_gravity="center_vertical"
    android:layout_rowSpan="2"
    android:text="Button8" />

<Button
    android:layout_column="1"
    android:layout_row="4"
    android:layout_gravity="fill_horizontal"
    android:text="Button9" />

<Button
    android:layout_column="2"
    android:layout_row="4"
    android:layout_gravity="fill_horizontal"
    android:text="Button10" />

```

```

<Button
    android:layout_column="1"
    android:layout_row="5"
    android:layout_gravity="fill_horizontal"
    android:text="Button11" />

<Button
    android:layout_column="2"
    android:layout_row="5"
    android:layout_gravity="fill_horizontal"
    android:text="Button12" />

```

```

</GridLayout>

```

## RelativeLayout

- Ähnlich dem Layout in iOS
- Alle UI-Elemente orientieren sich am
  - parent
  - sibling
- Die Verknüpfung geschieht durch die Id's

## RelativeLayout

- android:layout\_above
- android:layout\_alignBaseline
- android:layout\_alignBottom
- android:layout\_alignEnd
- **android:layout\_alignLeft**
- android:layout\_alignParentBottom
- android:layout\_alignParentEnd
- **android:layout\_alignParentLeft**
- android:layout\_alignParentRight
- android:layout\_alignParentStart
- android:layout\_alignParentTop
- android:layout\_alignRight
- android:layout\_alignStart
- android:layout\_alignTop
- android:layout\_alignWithParentIfMissing
- android:layout\_below
- android:layout\_centerHorizontal
- android:layout\_centerInParent
- android:layout\_centerVertical
- android:layout\_toEndOf
- android:layout\_toLeftOf
- android:layout\_toRightOf
- android:layout\_toStartOf

## 1. Aufgabe



```
<TextView
    android:id="@+id/textview1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Eingabe" />
<EditText
    android:id="@+id/bn2"
    android:layout_width="..."
    android:layout_height="..."
    android:text="In diesem Textfield kann man bestimmt etwas
eingeben" />
```

## 2. Aufgabe



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Klick mich"
/>
```



### 3. Aufgabe



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Klick mich"
/>
```

### 4. Aufgabe



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Abbrechen"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ok"
/>
```

## 5. Aufgabe



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Abbrechen"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ok"
/>
```

## 6. Aufgabe



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Abbrechen"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ok"
/>
```

## 7. Aufgabe

The screenshot shows a dialog box with a blue header bar containing the text "TestRelativLayout2". Below the header, the text "Name Andrea Musterfrau" and "Matrikelnummer 25678" is displayed. At the bottom of the dialog, there are two buttons: "OK" and "ABBRECHEN".

## 8. Aufgabe

The screenshot shows the same dialog box as in the previous task, but with a status bar at the top. The status bar contains a Wi-Fi icon, a battery icon, and the time "6:00". The dialog box title "TestRelativLayout2" is in a blue header bar. Below it, the text "Name Andrea Musterfrau" and "Matrikelnummer 25678" is displayed. At the bottom, there are two buttons: "OK" and "ABBRECHEN".

```

<TextView
    android:id="@+id/textview1"
    />
<EditText
    android:id="@+id/edittextName"
    android:layout_toRightOf="@id/textview1"
    android:layout_alignBaseline="@id/textview1"
    android:layout_alignLeft="@id/edittextMatrnr"
    android:lines="1"
    android:text="Andrea Musterfrau"
    />
<TextView
    android:id="@+id/textview2"
    />
<EditText
    android:id="@+id/edittextMatrnr"
    />

```

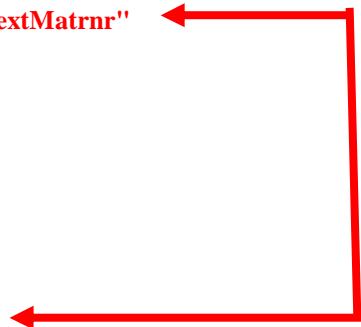
## RelativeLayout: Problem

- Wenn man auf eine id verweisen will, so muss diese **VORER** bekannt sein. Also oberhalb definiert sein.
- Manchmal kann diese aber zu Problemen führen.
- **Regel:**
  - Wenn man vorher eine „id“ als Referenz benötigt, so kann man diese oberhalb „an der Verweisstelle“ definieren (mit Pluszeichen).
    - **@+id/edittextMatrnr**
  - Am Originalelement benötigt man dann nur eine Referenz (ohne Pluszeichen)
    - **@id/edittextMatrnr**

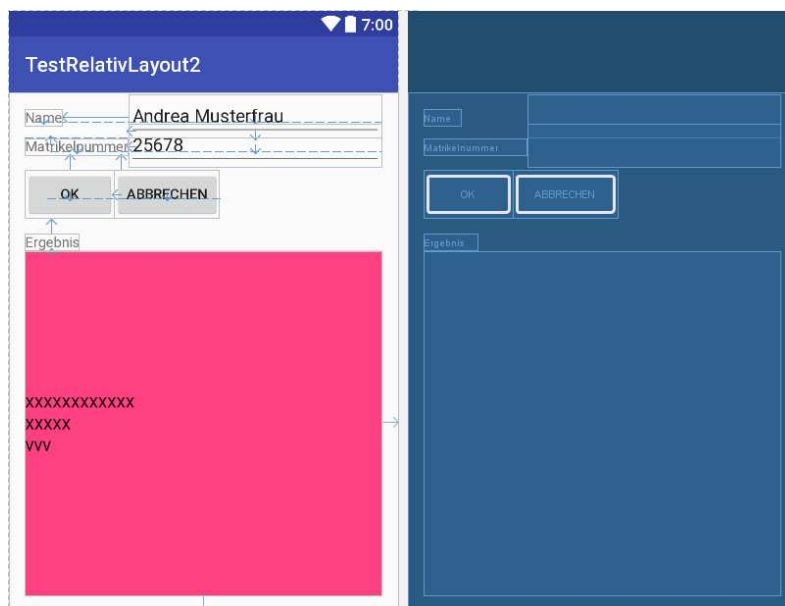
```

<TextView
    android:id="@+id/textview1"
/>
<EditText
    android:id="@+id/edittextName"
    android:layout_toRightOf="@id/textview1"
    android:layout_alignBaseline="@id/textview1"
    android:layout_alignLeft="@+id/edittextMatrnr"
    android:lines="1"
    android:text="Andrea Musterfrau"
/>
<TextView
    android:id="@+id/textview2"
/>
<EditText
    android:id="@id/edittextMatrnr"
/>

```



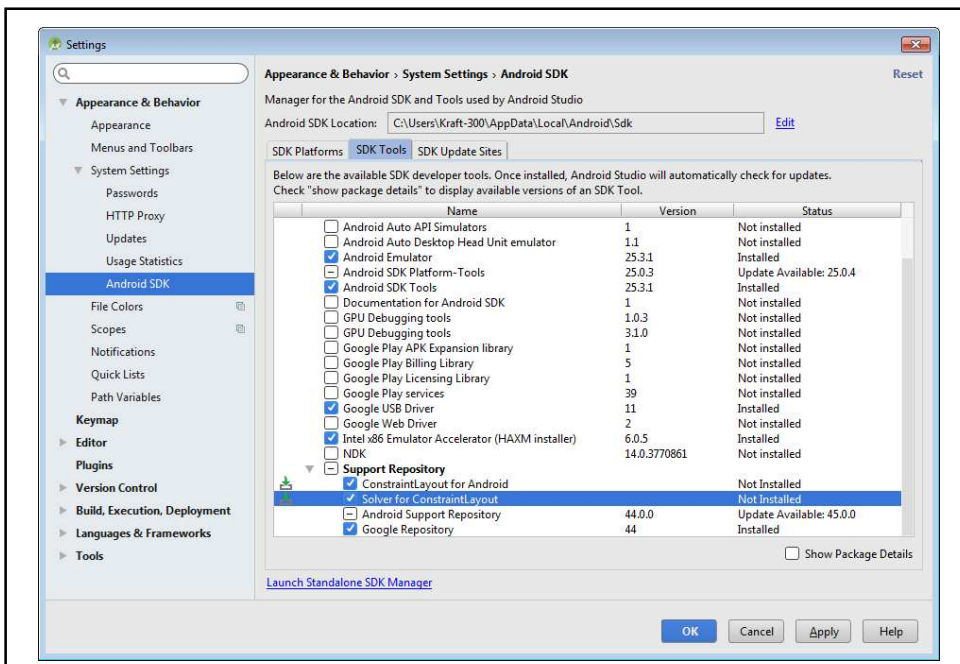
### 9. Aufgabe: mit Multiline-Editor

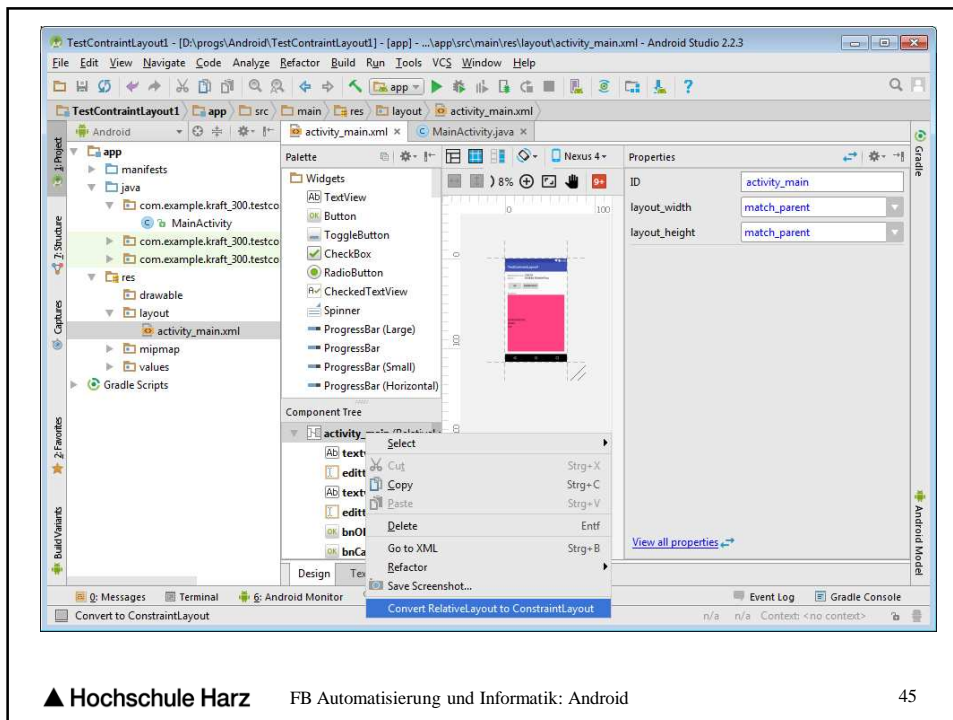


```

"
• />
• <EditText
• android:id="@+id/ergebnis"
•
android:layout_width="match_parent"
"
•
android:layout_height="wrap_content"
"
•
android:layout_below="@id/textviewergebnis"
"
•
android:layout_alignParentBottom
="true"

```





## ConstraintLayout

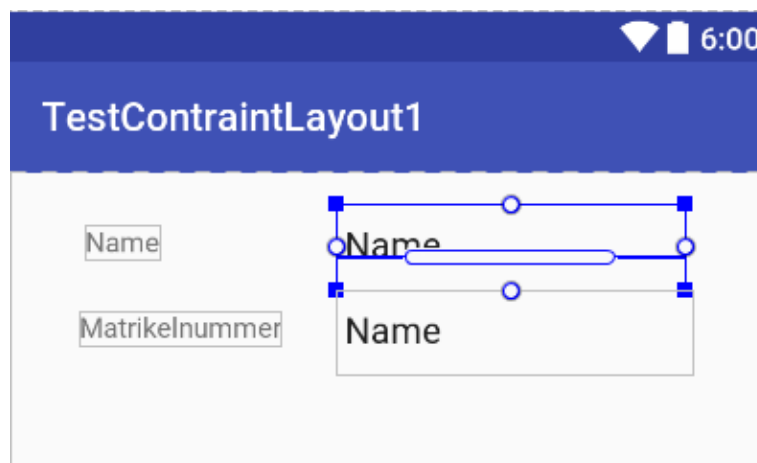
- Neues Layout
- Stamm von ViewGroup
- Ist sehr flexibel
- Erlaubt das Erstellen und Ändern mittels Drag & Drop & Maus
- <https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html>

## ConstraintLayout

### Arten der Constraints:

- Relative positioning
- Margins
- Centering positioning
- Visibility behavior
- Dimension constraints
- Chains
- Virtual Helpers objects

## ConstraintLayout



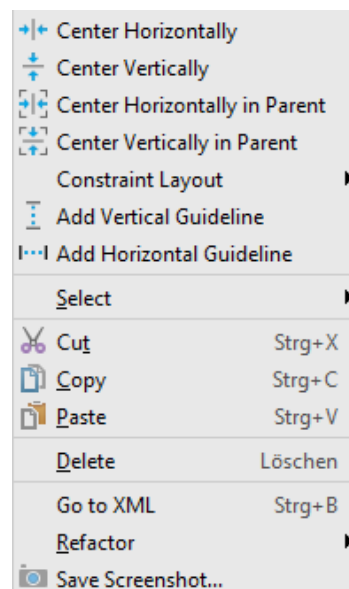
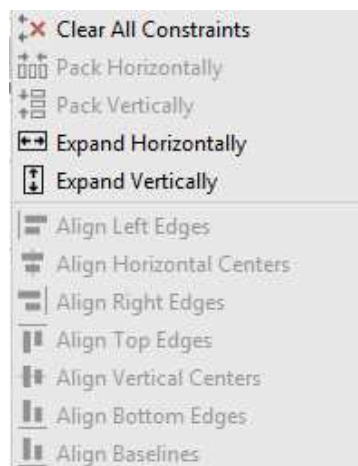


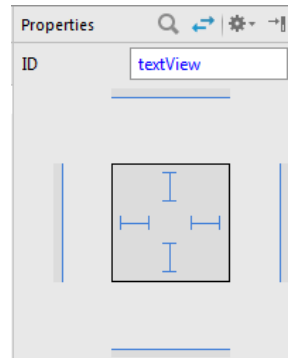
## ConstraintLayout: Grundstruktur




```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

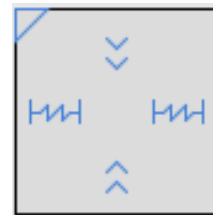
</android.support.constraint.ConstraintLayout>
```

## ConstraintLayout: Funktionen



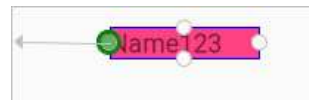


-  **Fixed:** Feste Breite /Höhe
-  **AnySize:** Gewichtung 1, Skalierung
-  **Wrap Content:** This option only expands to fill the widget with the contained element such as text or a drawable.

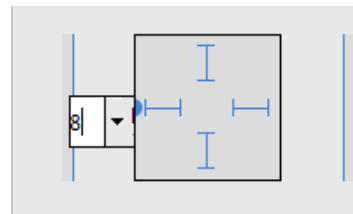


## ConstraintLayout: Linker Rand

**Drag & Drop zum linken Rand**



**Ändern der Breite**



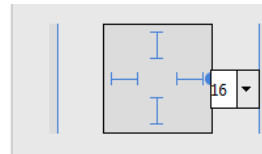
`android:layout_marginLeft="16dp"`

## ConstraintLayout: Rechter Rand

Drag & Drop zum rechten Rand



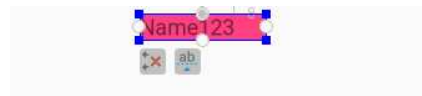
Ändern der Breite



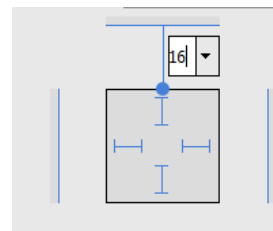
```
android:layout_marginRight="16dp"
```

## ConstraintLayout: Oberer Rand

Drag & Drop zum oberen Rand



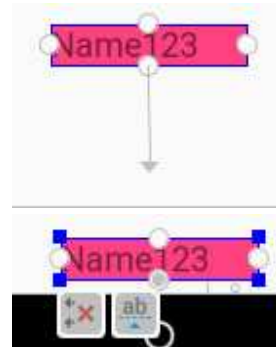
Ändern des Abstands



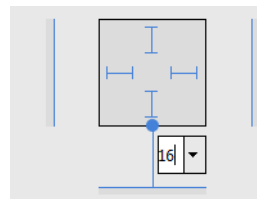
```
android:layout_marginTop="16dp"
```

## ConstraintLayout: Unterer Rand

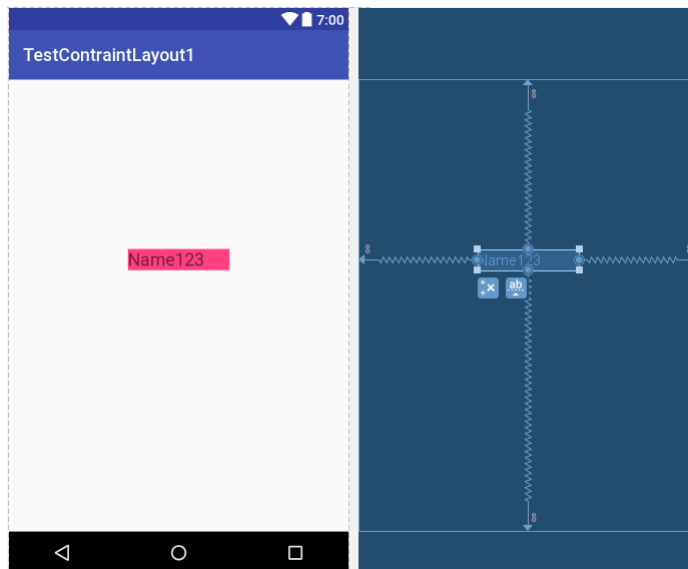
Drag & Drop zum unteren Rand



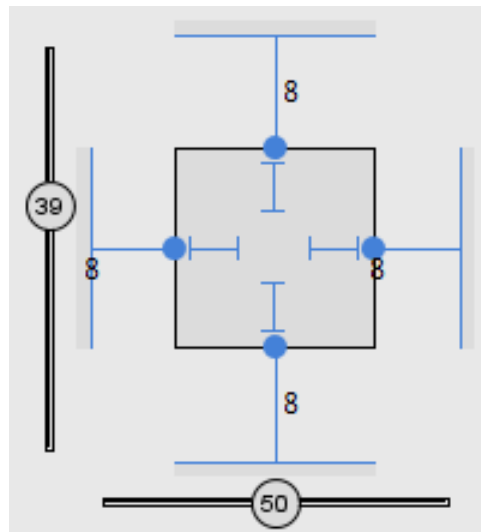
Ändern des Abstands



## ConstraintLayout: Ausrichtung / Ränder

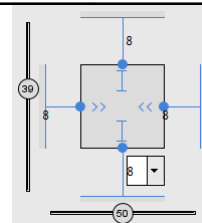


## ConstraintLayout: Ausrichtung / Ränder



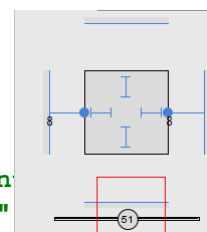
## Größe des Inhalts

```
app:layout_constraintRight_toRightOf="parent"
android:layout_marginLeft="8dp"
app:layout_constraintLeft_toLeftOf="parent"
```



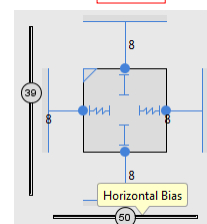
## Größe wird definiert

```
android:layout_marginLeft="8dp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintHorizontal_bias="0.51"
```



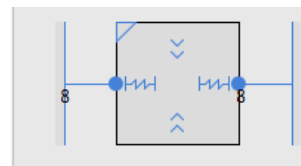
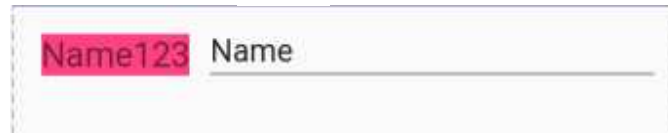
## Komplette Breite: erst beide Seiten <- ->

```
android:layout_marginLeft="8dp"
android:layout_marginRight="8dp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```



## TextView mit einem TextField

- Positionieren des Textfeldes neben dem TextView
- Anker auf beiden Seiten
- Type „anySize“

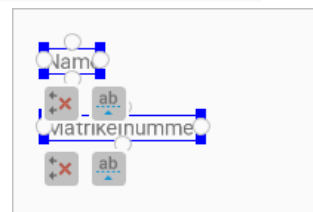
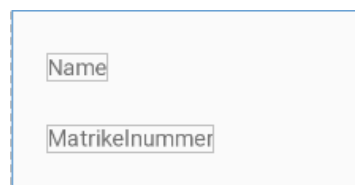
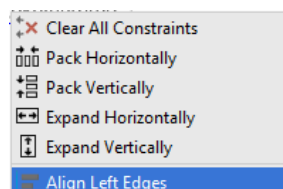


## ConstraintLayout-Beispiel

### linksbündig

### Ablauf

- Beide UI-Elemente markieren
- UI-Element „Name“ nach oben schieben
- Align Left Edges



Beide Elemente werden linksbündig in der Mitte dargestellt

```
app:layout_constraintLeft_toLeftOf="@+id/textView"
```

## ConstraintLayout-Beispiel



•Schalter „ab“

Baseline

## Absolute Layout

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
```

```
<TextView
    android:layout_x="10px"
    android:layout_y="110px"
    android:text="@string/username"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

```
</AbsoluteLayout>
```

## FrameLayout: Übereinanderliegende UI-Elemente

### <FrameLayout

```
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android">
```

```
<ImageView
  android:id="@+id/image1"
  android:src="@mipmap/ic_launcher"
  android:scaleType="fitCenter"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:gravity="center"
  android:onClick="image_click"
/>
```

```
<TextView
  android:id="@+id/textview1"
  android:text="Dies ist ein Bild"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:textSize="30sp"
  android:gravity="center"
  android:onClick="textview_click"
/>
```

```
</FrameLayout>
```

```
import android.media.Image;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;

public class MainActivity extends AppCompatActivity {

    private TextView textview1 = null;
    private ImageView image1 = null;
```



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    textview1 = (TextView) findViewById(R.id.textview1);
    image1 = (ImageView) findViewById(R.id.image1);
} // onCreate
```

```
public void image_click(View view) {
    textview1.setVisibility(View.VISIBLE);
    image1.setVisibility(View.GONE);
}

public void textview_click(View view) {
    textview1.setVisibility(View.GONE);
    image1.setVisibility(View.VISIBLE);
}
}
```

- <https://codelabs.developers.google.com/codelabs/constraint-layout/index.html#4>

- <https://codelabs.developers.google.com/codelabs/constraint-layout/index.html#7>