

# Mobile Applications

## Android App Event Handlers

Prof. Dr.-Ing. Volodymyr Brovko

# Zielstellung

- Erstellen wir ein einfaches Projekt
  - nur eine einzige Activity
    - Button,
    - TextView,
    - TextEdit,
    - ...
  - Business Logic:
    - die Umrechnung der Temperatur von Celsius ins Kelvin und umgekehrt

# Zielstellung 2

- Button Click Handler Varianten:
  - internal Instance of Class OnClickListener
  - Event-Verarbeitung von mehreren Knöpfe mit einem gemeinsamen Handler
  - Interface OnClickListener
  - Handler Definierung in der Datei layout???.xml

# Projektbestandteile

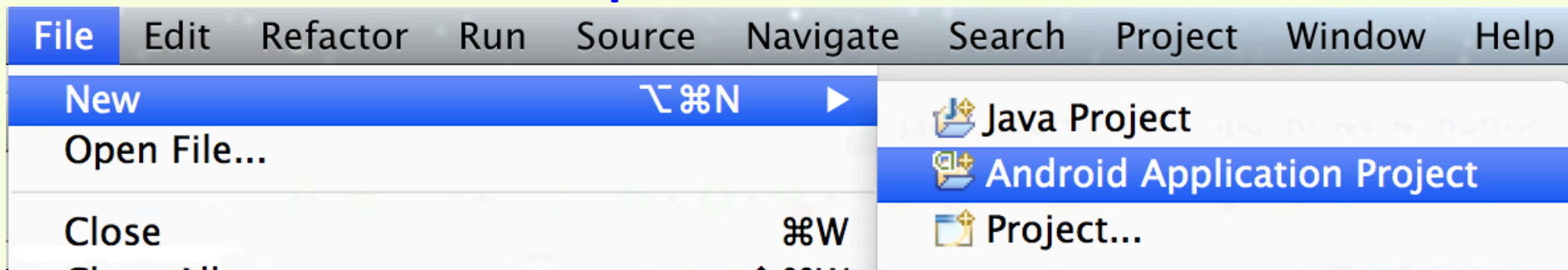
- AndroidManifest
- res
  - layout
    - activity\_main.xml
  - values
    - strings.xml
    - color.xml
- src
  - MainActivity.java

# Die Projekt-Erstellung

- in ECLIPSE IDE:
  - Projekt Name **Calc1**
  - Paket Name **ua.opu.brovkov.calc1**
  - Ein eigenes Icon (besser zum Starten am reellen Gerät)
  - Alle andere Parameter - default
  - Darunter auch:
    - MainActivity.java
    - activity\_main.xml

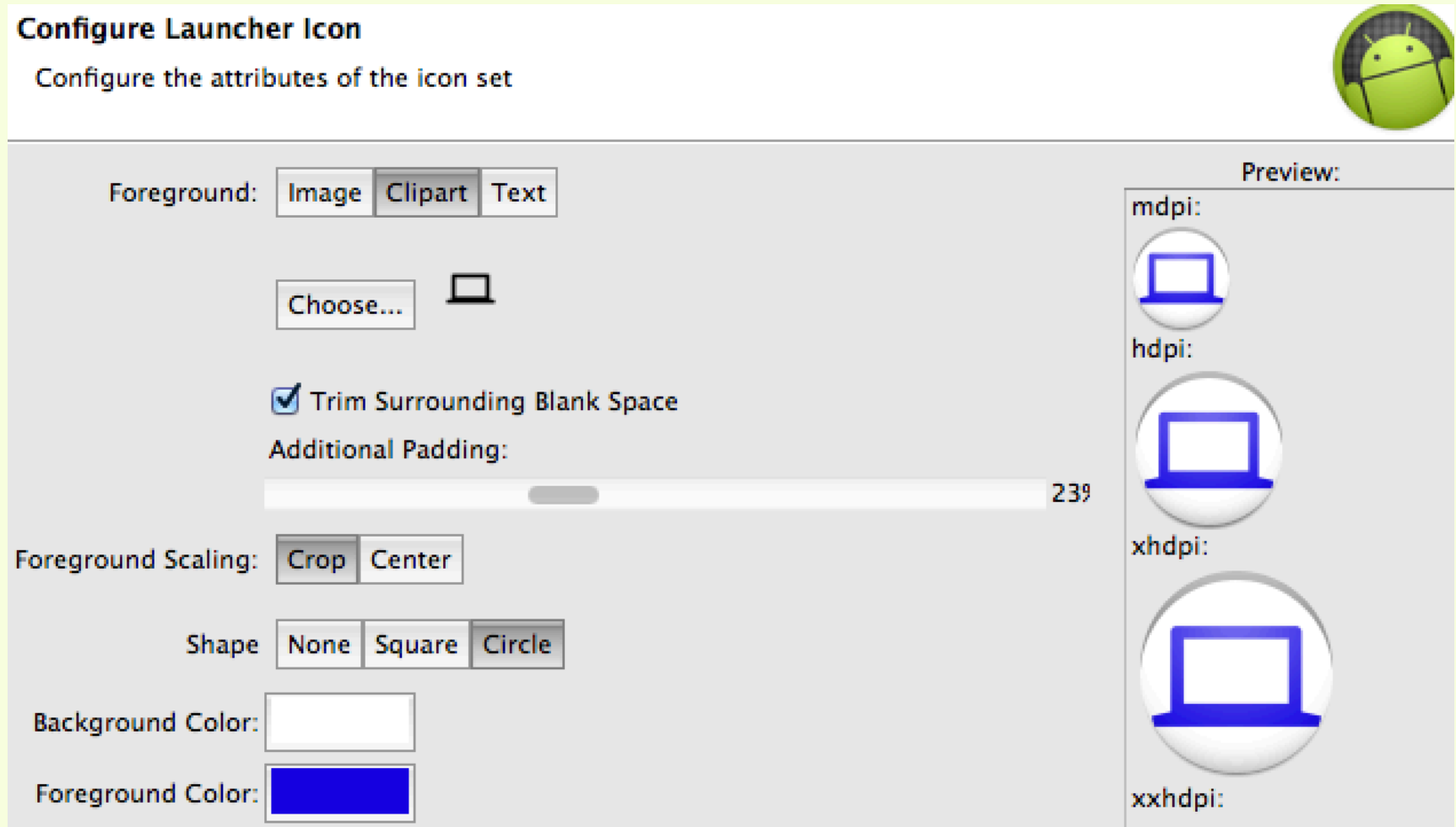
# Die Projekt-Erstellung 2

- in ECLIPSE IDE:
  - Projekt Name **Calc1**
  - Paket Name **ua.opu.brovkov.calc1**

A screenshot of the 'New Android Application Project' wizard in Eclipse. The wizard has several input fields and dropdown menus. The 'Application Name' field contains 'Calc1'. The 'Project Name' field also contains 'Calc1'. The 'Package Name' field contains 'ua.opu.brovkov.calc1' and is highlighted with a blue border. Below these are three dropdown menus for SDK levels: 'Minimum Required SDK' is set to 'API 8: Android 2.2 (Froyo)', 'Target SDK' is set to 'API 17: Android 4.2 (Jelly Bean)', and 'Compile With' is set to 'API 17: Android 4.2 (Jelly Bean)'. At the bottom, there is a 'Theme' dropdown menu set to 'Holo Light with Dark Action Bar'. Each field has an information icon (i) to its left.

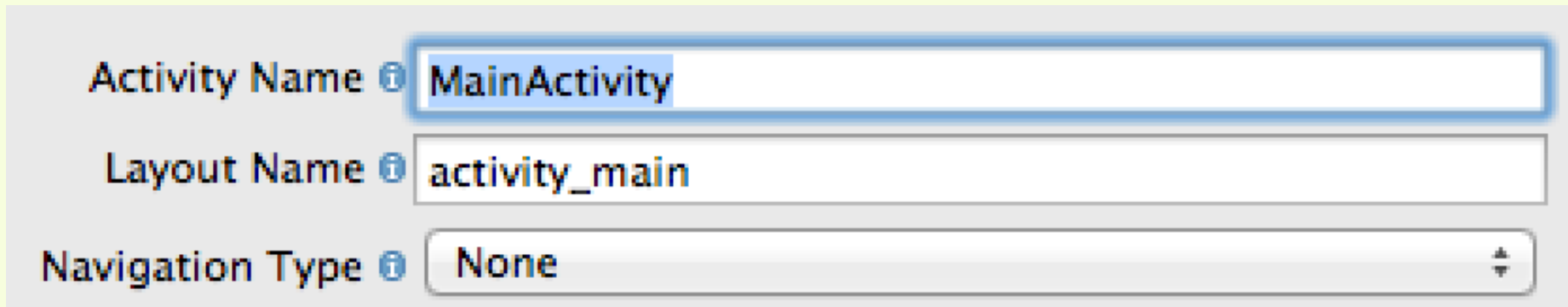
# Die Projekt-Erstellung 3

- Ein eigenes Icon (besser zum Starten am reellen Gerät)



# Die Projekt-Erstellung 4

- Alle andere Parameter - default
- Darunter auch:
  - MainActivity.java
  - activity\_main.xml



Activity Name ⓘ MainActivity

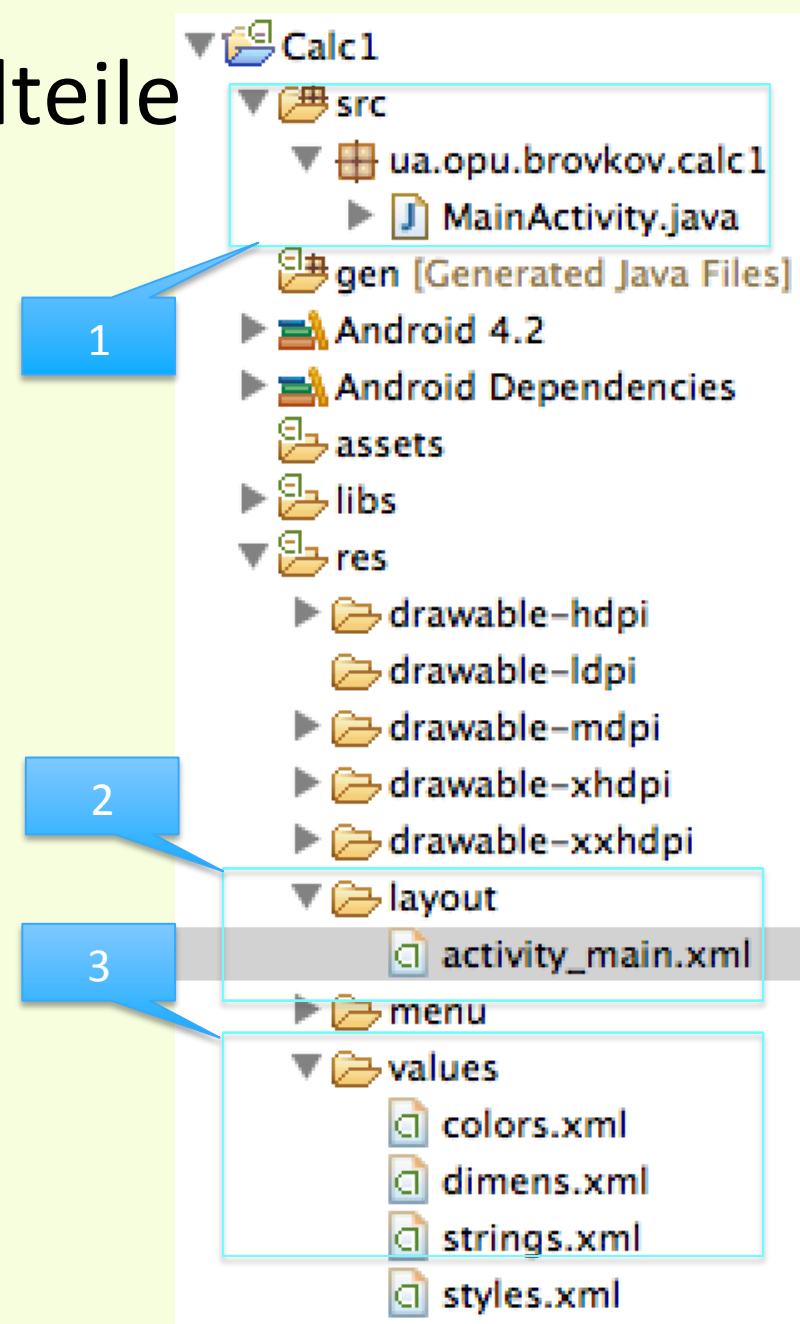
Layout Name ⓘ activity\_main

Navigation Type ⓘ None



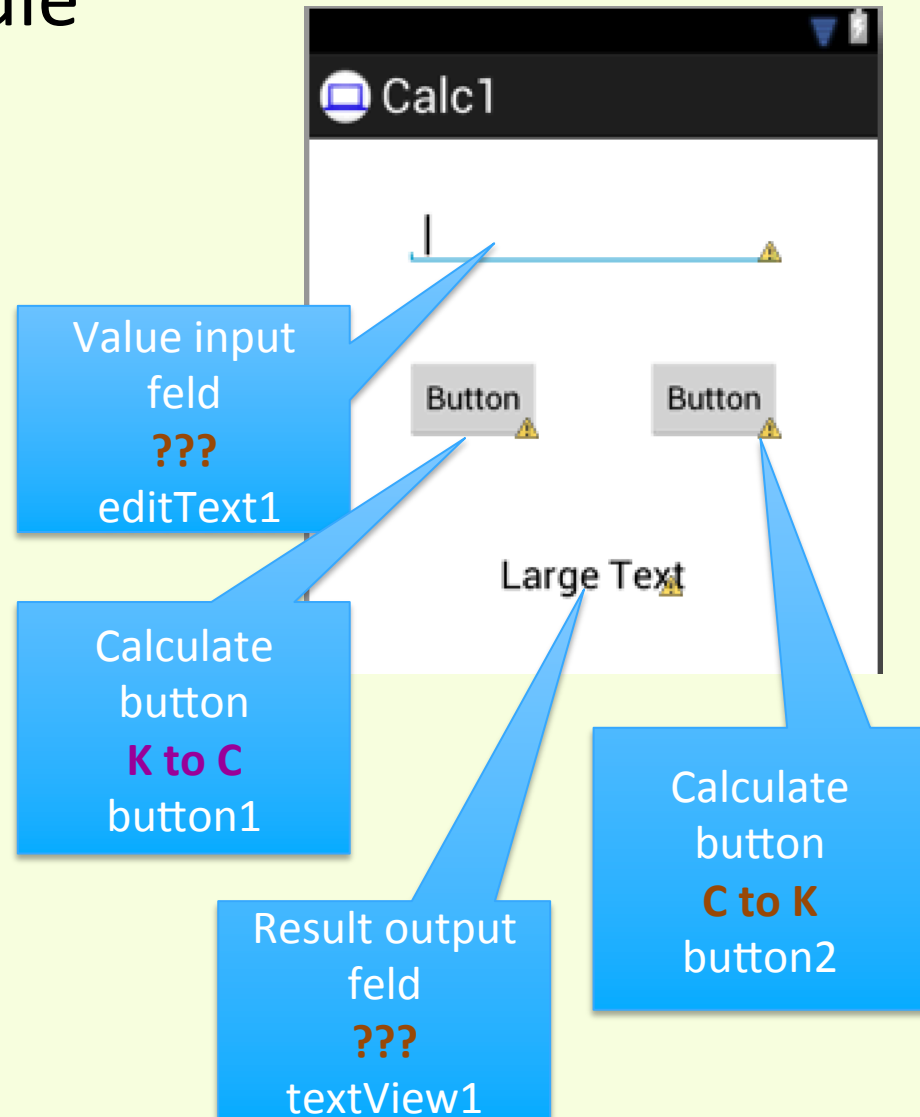
# Projekt-Bestandteile

- Die wichtigste Projekt-Bestandteile:
  - Ressourcen
    - layout
      - **activity\_main.xml (2)**
    - values
      - **strings.xml (3)**
      - ...
  - Source Code
    - **MainActivity.java (1)**
  - Die Manifest-Datei
    - AndroidManifest.xml**



# User Interface 1

- Am Bildschirm müssen die folgende Komponenten vorhanden sein:
- Die Beschriftung von Komponenten ist **rot** dargestellt.



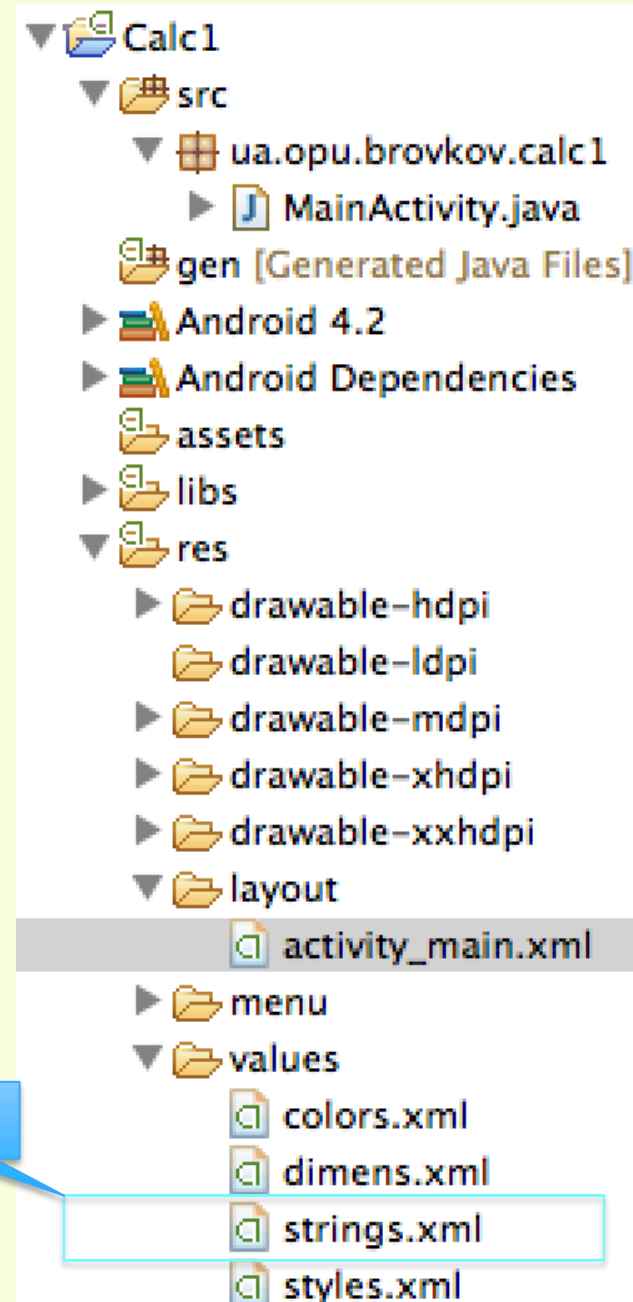
# User Interface 2

- Alle Beschriftungen tragen wir in die ressourcen Datei **strings.xml (3)** ein.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

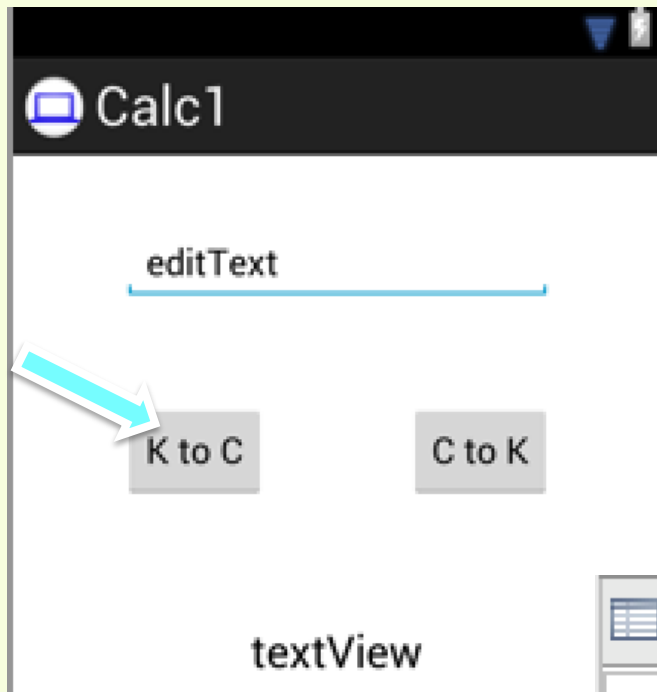
    <string name="app_name">Calc1</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="editText1">???</string>
    <string name="button1">K to C</string>
    <string name="button2">C to K</string>
    <string name="textView">???</string>

</resources>
```



# User Interface 3




- Verbinden wir die Komponenten-Beschriftungen mit den entsprechenden Elementen der Datei **strings.xml**:



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

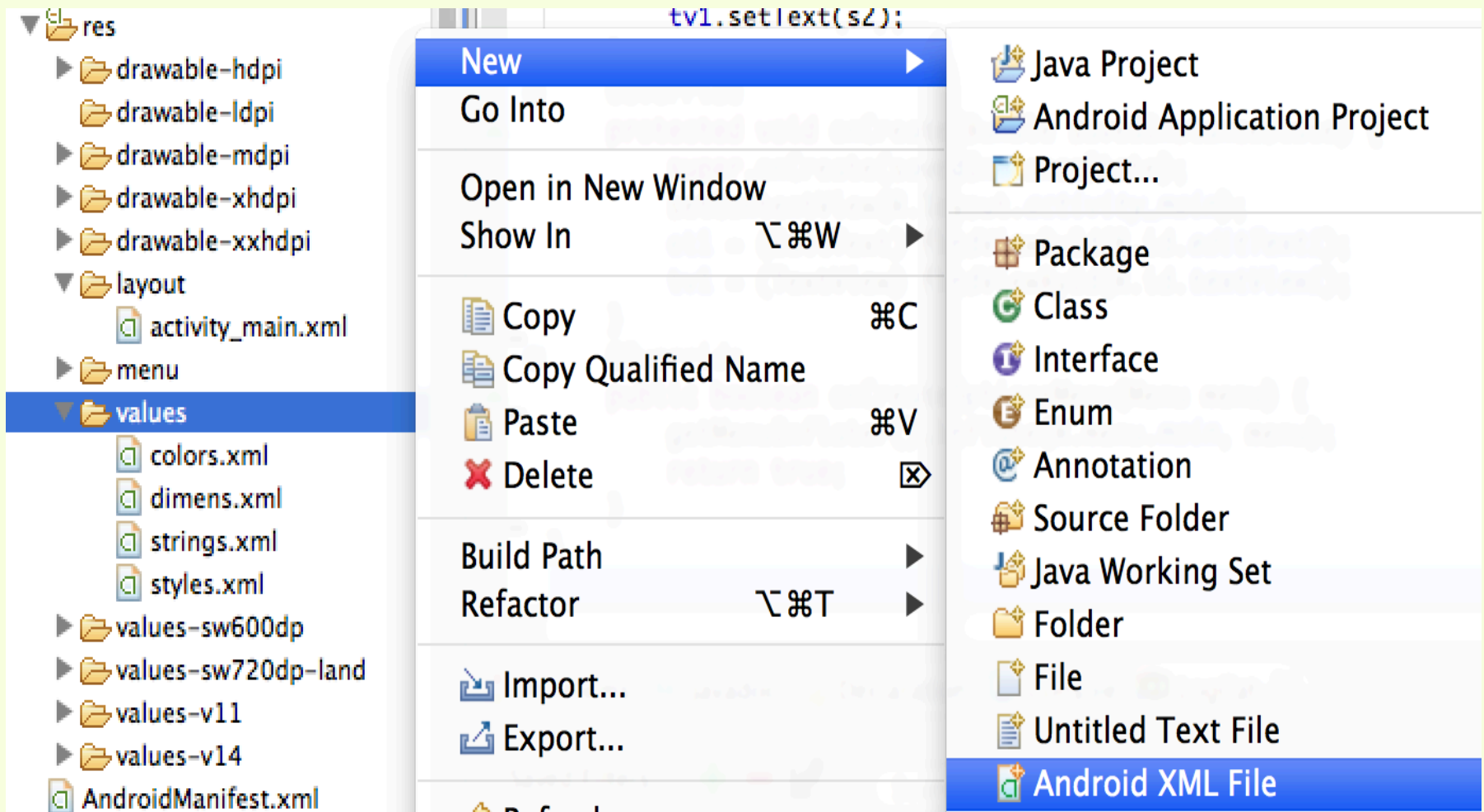
    <string name="app_name">Calc1</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="button1">K to C</string>
    <string name="button2">C to K</string>
    <string name="textView">textView</string>
    <string name="editText">editText</string>

</resources>
```

| Properties  |                          |  |  |
|---|--------------------------|---|---|
| Id  | @+id/button1             |   |   |
|  Layout Parameters | []                       |   |   |
| Style   | android:buttonStyle      |   |   |
| Text  | @string/button1 (K to C) |   |   |

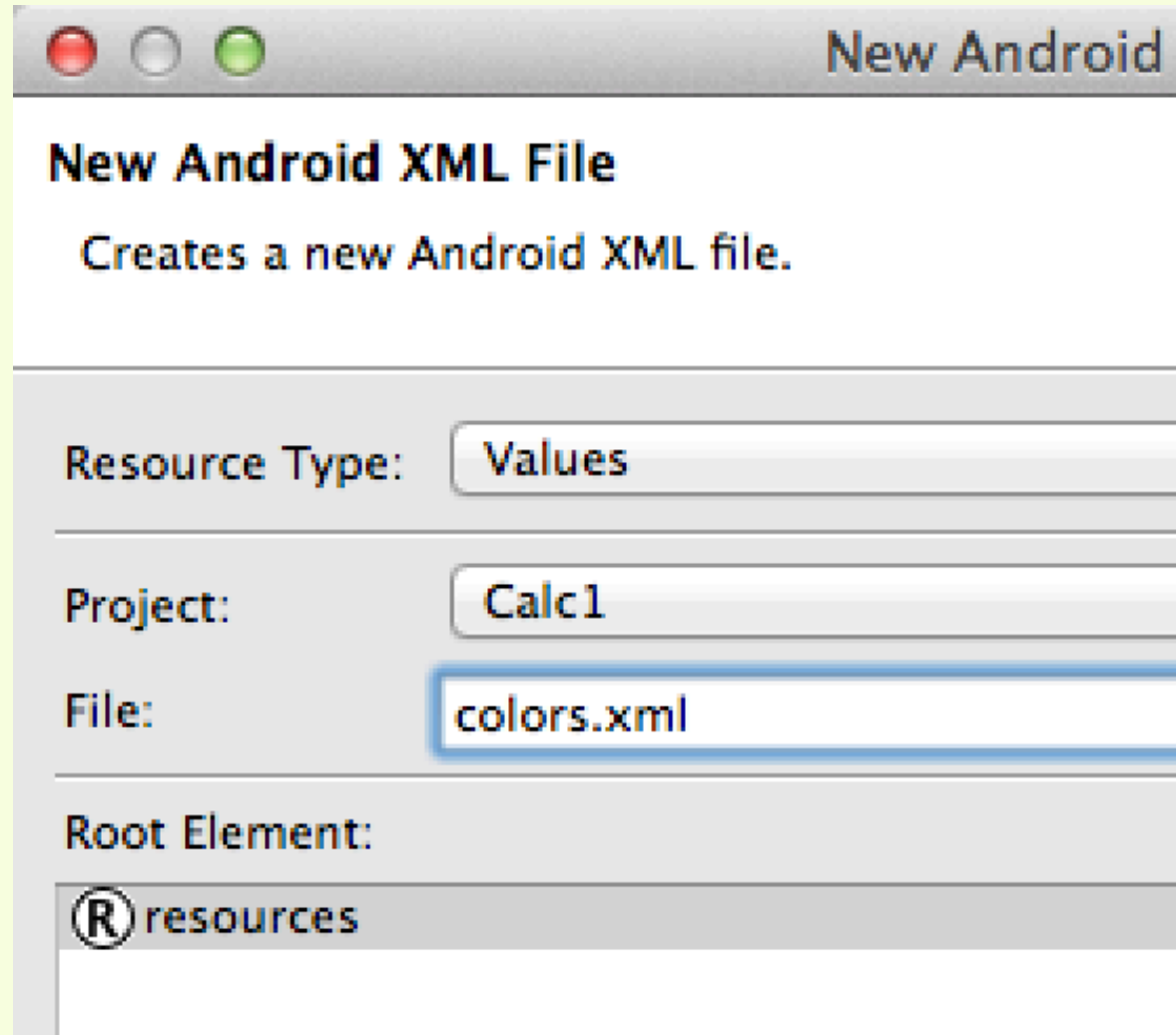
# User Interface 4

- Erstellen wir die Datei colors.xml Mit einem rechten Mausklick auf die Zeile **values** :



# User Interface 5

- Definieren wir die Name der Datei als **colors.xml**



# User Interface 6

- Definieren wir zwei Farben. Das kann mit Hilfe der XML-Datei oder mit Resources Editor gemacht werden:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="color1">#DFFFDf</color>
    <color name="color2">#8888ff</color>
</resources>
```

The screenshot shows the 'Android Resources (default)' editor. On the left, the 'Resources Elements' list contains 'color1 (Color)' and 'color2 (Color)'. To the right of this list are buttons for 'Add...', 'Remove...', 'Up', and 'Down'. On the right side of the editor, the 'Attributes for color1 (Color)' panel is visible. It contains a description of the 'color' attribute, followed by a 'Name' field with the value 'color1' and a 'Value\*' field with the value '#DFFFDf'.

**Resources Elements** (S) (C) (D) (D) (S) (I) (S) (I) (P) Az

- color1 (Color)
- color2 (Color)

**Attributes for color1 (Color)**

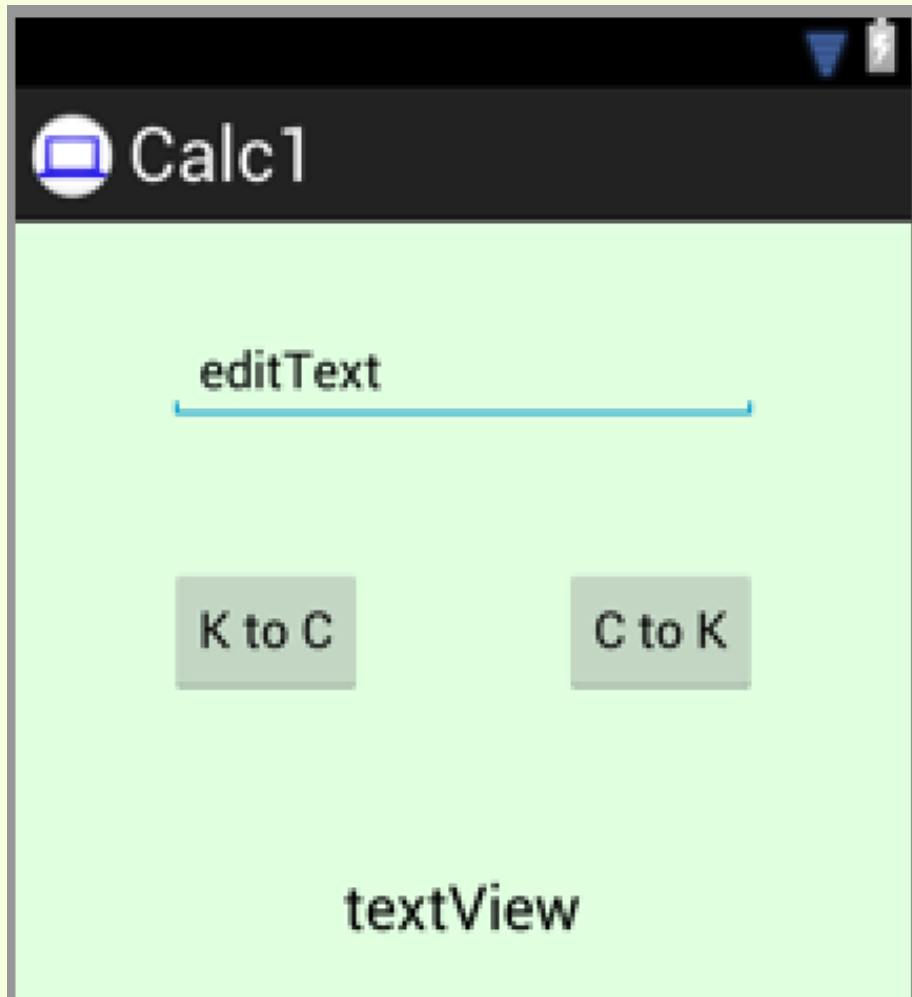
(C) A [color](#) value specifies an alpha channel, which can be placed in places such as specifying a Drawable or the color to use. It begins with a # character and the alpha-red-green-blue values in the following formats: #RRGGBB, #AARRGGBB.


Name: color1

Value\*: #DFFFDf

# User Interface 7

- Definieren wir das Backcolor der Activity

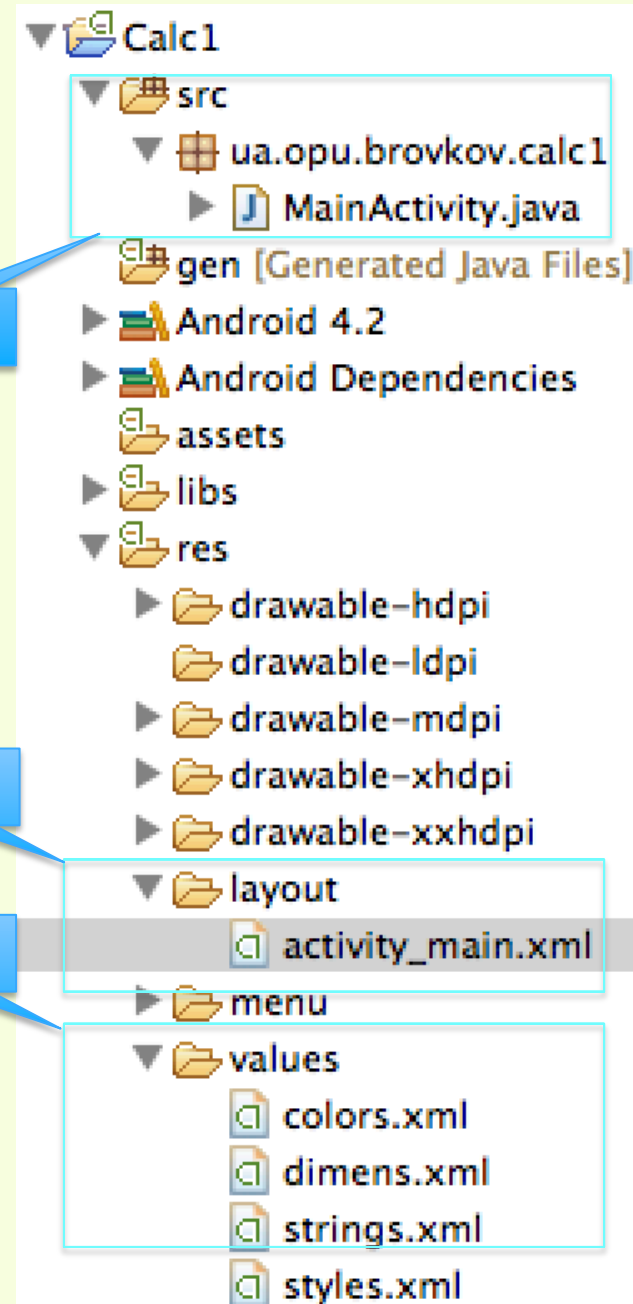


| Properties          |   |
|---------------------|---|
| Id                  |   |
| + Layout Parameters | []  |
| Background          |  @color/color1 |



# Java Code 1

- Wir haben erstellt:
  - layout
    - **activity\_main.xml (2)**
  - values
    - **strings.xml (3)**
    - **colors.xml**
- Wir brauchen jetzt java-Code  
**MainActivity.java (1)**  
womit
  - die Input-Daten berechnet werden
  - die Ergebnisse dargestellt werden



# Java Code 2.

## Symbolische Namen mit Views verbinden

*Eine Empfehlung: Es kann CTRL+SHIFT+O (oder CMD+SHIFT+O on MAC) SEHR helfen!*

Symbolische  
Namen

Symbolische  
Namen mit  
Komponenten  
verbinden

Type Casting

```
public class MainActivity extends Activity {  
    Button bt1, bt2;  
    EditText et1;  
    TextView tv1;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        bt1 = (Button) findViewById(R.id.button1);  
        bt2 = (Button) findViewById(R.id.button2);  
        et1 = (EditText) findViewById(R.id.editText1);  
        tv1 = (TextView) findViewById(R.id.textView1);  
    }  
}
```

# Java Code 3. Ein genannte Listener

```
public class MainActivity extends Activity {  
    Button bt1, bt2;  
    EditText et1;  
    TextView tv1;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        bt1 = (Button) findViewById(R.id.button1);  
        bt2 = (Button) findViewById(R.id.button2);  
        et1 = (EditText) findViewById(R.id.editText1);  
        tv1 = (TextView) findViewById(R.id.textView1);  
        OnClickListener ocl_bt1 = new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                // TODO Auto-generated method stub  
            }  
        };  
        bt1.setOnClickListener(ocl_bt1);  
    }  
}
```

Internal Klass  
mit Name.

Komponent-  
Handler  
Verbindung

Event-  
Bearbeitung

## Java Code 4. Data Processing (Temperatur Berechnung und Darstellung)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    bt1 = (Button) findViewById(R.id.button1);
    bt2 = (Button) findViewById(R.id.button2);
    et1 = (EditText) findViewById(R.id.editText1);
    tv1 = (TextView) findViewById(R.id.textView1);
    OnClickListener ocl_bt1 = new OnClickListener() {
        @Override
        public void onClick(View v) {
            String s1 = et1.getText().toString();
            double d1 = Double.parseDouble(s1);
            double d2 = d1 + 273.0;
            String s2 = Double.toString(d2);
            tv1.setText(s2);
        }
    };
    bt1.setOnClickListener(ocl_bt1);
}
```

# Java Code 5. Exceptions

Fehler Schutz

Alle mögliche Fehler  
müssen gesichert werden,  
sonst kann die App  
abstürzen. Hier kann das  
Text keine Zahl beinhalten.

```
bt1 = (Button) findViewById(R.id.button1);
bt2 = (Button) findViewById(R.id.button2);
et1 = (EditText) findViewById(R.id.editText1);
tv1 = (TextView) findViewById(R.id.textView1);
OnClickListener ocl_bt1 = new OnClickListener() {
    @Override
    public void onClick(View v) {
        String s1 = et1.getText().toString();
        String s2;
        try{
            double d1 = Double.parseDouble(s1);
            double d2;
            if (v.getId() == R.id.button1){
                d2 = d1 + 273.0;
            } else {
                d2 = d1 - 273.0;
            }
            s2 = Double.toString(d2);
        } catch (Exception e){
            s2 = "Error";
        }
        tv1.setText(s2);
    }
};
bt1.setOnClickListener(ocl_bt1);
bt2.setOnClickListener(ocl_bt1);
}
```

# Java Code 6. Anonymes Event Handler

```
public class MainActivity extends Activity {
    Button bt1, bt2;
    EditText et1;
    TextView tv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bt1 = (Button) findViewById(R.id.button1);
        bt2 = (Button) findViewById(R.id.button2);
        et1 = (EditText) findViewById(R.id.editText1);
        tv1 = (TextView) findViewById(R.id.textView1);
        bt1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
            }
        });
    }
}
```

# Java Code 7. Zwei Knöpfe mit einem Listener

```
OnClickListener ocl_bt1 = new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        String s1 = et1.getText().toString();  
        double d1 = Double.parseDouble(s1);  
        double d2;  
        if (v.getId() == bt1){  
            d2 = d1 + 273.0;  
        } else {  
            d2 = d1 - 273.0;  
        }  
        String s2 = Double.toString(d2);  
        tv1.setText(s2);  
    }  
};  
bt1.setOnClickListener(ocl_bt1);  
bt2.setOnClickListener(ocl_bt1);
```

Ein genannte  
Event Handler

Event Source  
finden

Das Objekt  
**View v** hat die  
Information  
über Event  
Quelle

Beide Knöpfe  
benutzen den  
gleichen Event  
Handler!



# Java Code 8. Interface

```
public class MainActivity extends Activity implements OnClickListener{
    Button bt1, bt2;
    EditText et1;
    TextView tv1;
    public void onClick(View v) {
        //TODO Button Click Processing
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bt1 = (Button) findViewById(R.id.button1);
        bt2 = (Button) findViewById(R.id.button2);
        bt1.setOnClickListener(this);
        bt2.setOnClickListener(this);
        et1 = (EditText) findViewById(R.id.editText1);
        tv1 = (TextView) findViewById(R.id.textView1);
    }
}
```

Interface  
Implementierung

onClick() Method  
Realisierung. Das  
Method muss die  
Signal Quelle  
definieren können!

Event Handler  
Definierung für  
beide Knöpfe



# Java Code 9. Event Listener Definierung in der xml-Datei

In der XML-Datei, die einen Activity Layout definiert, kann auch die Name von Method **click()**, definiert werden, um das Event **onClick()** zu verarbeiten.

Die Method-Name in der java-Datei muss dem genannten in der xml-Datei entsprechen. Die Methd Signature soll dem **onClick()** entsprechen!!!

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="50dp"
    android:onClick="click"
    android:text="@string/button1" />
```

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/textView1"
    android:layout_alignRight="@+id/editText1"
    android:onClick="click"
    android:text="@string/button2" />
```

# Java Code 10. Event Listener Definierung in der xml-Datei

Die Method-Name in der java-Datei muss dem genannten in der xml-Datei entsprechen. Die Method Signature soll dem onClick() entsprechen!!!

In der xml-Datei sind zwei Objekte zum Method click() zugeschaltet. Das Method muss die Signal-Quelle definieren können.

```
public void click(View v) {  
    String s1 = et1.getText().toString();  
    String s2;  
    try{  
        double d1 = Double.parseDouble(s1);  
        double d2;  
        if (v.getId() == R.id.button1){  
            d2 = d1 + 273.0;  
        } else {  
            d2 = d1 - 273.0;  
        }  
        s2 = Double.toString(d2);  
    } catch (Exception e){  
        s2 = "Error";  
    }  
    tv1.setText(s2);  
}
```

# Java Code 11. Event Listener Definierung in der xml-Datei

```
public class MainActivity extends Activity{
    Button bt1, bt2;
    EditText et1;
    TextView tv1;
    public void Click(View v) {
        String s1 = et1.getText().toString();
        String s2;
        try{
            double d1 = Double.parseDouble(s1);
            double d2;
            if (v.getId() == bt2){
                d2 = d1 + 273.0;
            } else {
                d2 = d1 - 273.0;
            }
            s2 = Double.toString(d2);
        }catch(Exception e){
            s2 = "Error";
        }
        tv1.setText(s2);
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bt1 = (Button) findViewById(R.id.button1);
        bt2 = (Button) findViewById(R.id.button2);
        et1 = (EditText) findViewById(R.id.editText1);
        tv1 = (TextView) findViewById(R.id.textView1);
    }
}
```

Symbolische Namen  
definieren

Ein Event-Handler, der mit  
den Objekten in der xml-  
Datei verbunden ist.

Die Verbindung von  
symbolischen Namen mit  
den Objekten.

**Die Frage:**  
**Sind die Namen bt1, bt2**  
**unbedingt erforderlich?**

# Java Code 12. Event Listener Definierung in der xml-Datei

```
public class MainActivity extends Activity {  
    EditText et1;  
    TextView tv1;  
    public void click(View v) {  
        String s1 = et1.getText().toString();  
        String s2;  
        try{  
            double d1 = Double.parseDouble(s1);  
            double d2;  
            if (v.getId() == R.id.button1){ d2 = d1 + 273.0; }  
            else { d2 = d1 - 273.0; }  
            s2 = Double.toString(d2);  
        }catch(Exception e){ s2 = "Error"; }  
        tv1.setText(s2);  
    }  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        et1 = (EditText) findViewById(R.id.editText1);  
        tv1 = (TextView) findViewById(R.id.textView1);  
    }  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }  
}
```

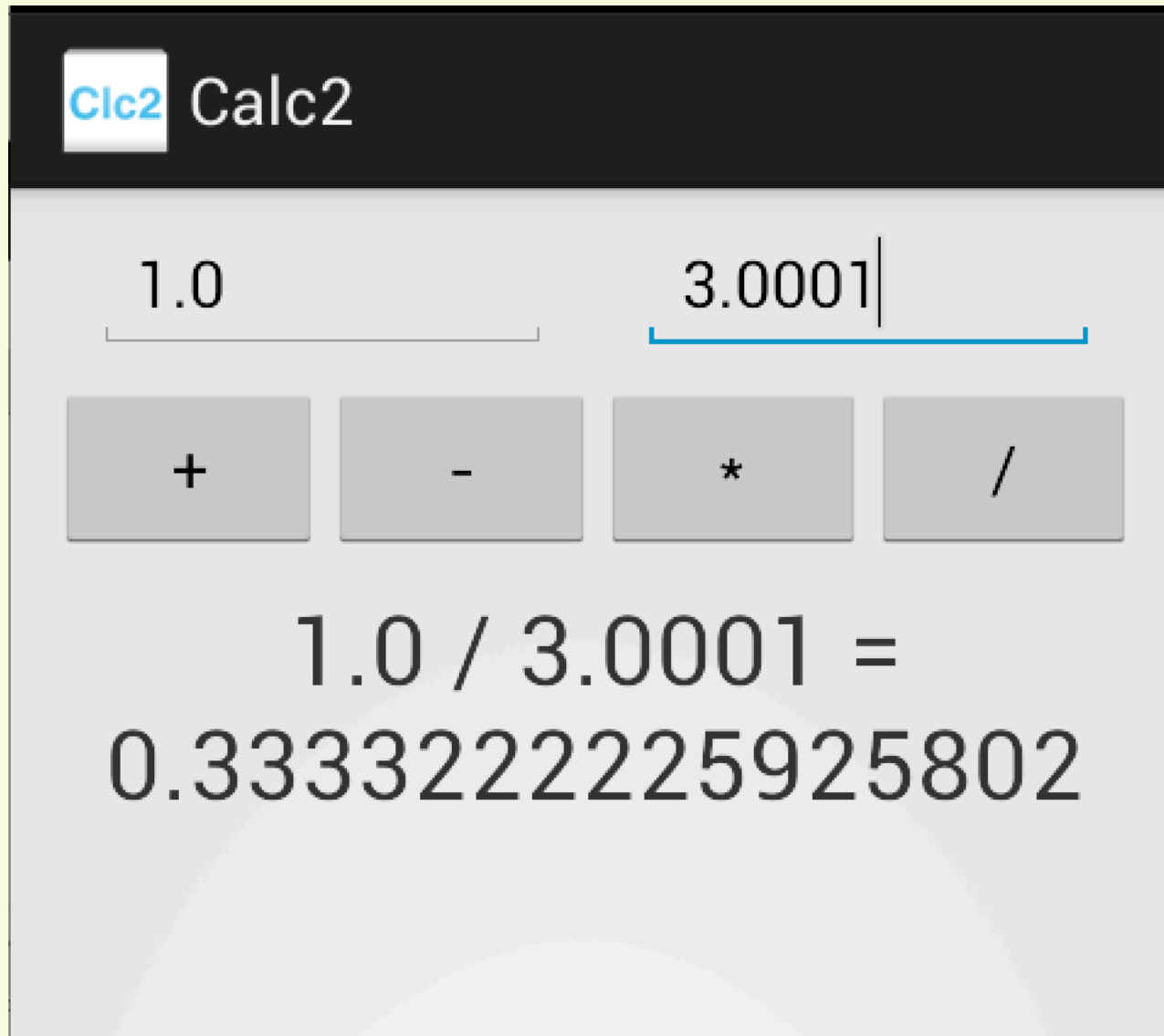
Symbolische Namen  
definieren

Ein Event-Handler, der mit  
den Objekten in der xml-  
Datei verbunden ist.

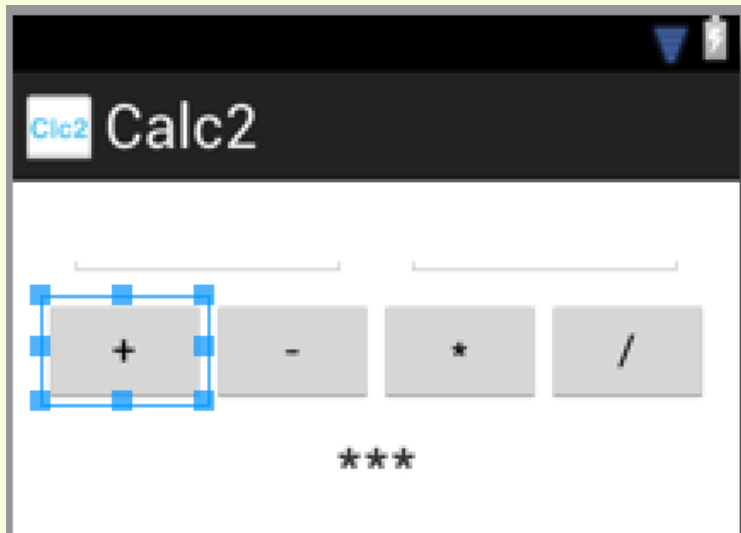
Es funktioniert! Es sind  
nur die Namen im Einsatz,  
die in der XML-Datei  
definiert sind.  
Die Namen bt1, bt2 sind  
nicht erforderlich.

# Projekt-Beispiel. Calculator

<http://startandroid.ru/ru/uroki/vse-uroki-spiskom/54-urok-19-pishem-prostoj-kalkuljator.html>



# Projekt-Beispiel 2. Ressourcen.



| Properties          |                     |
|---------------------|---------------------|
| Id                  | @+id/btnAdd         |
| Layout Parameters   | []                  |
| Style               | android:buttonStyle |
| Text                | @string/plus (+)    |
| Hint                |                     |
| Content Description |                     |
| TextView            | []                  |
| Text                | @string/plus (+)    |
| Hint                |                     |

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">Calc2</string>
5     <string name="action_settings">Settings</string>
6     <string name="hello_world">Hello world!</string>
7     <string name="plus">+</string>
8     <string name="minus">-</string>
9     <string name="mult">*</string>
10    <string name="div">/</string>
11    <string name="empty"></string>
12    <string name="stern">***</string>
13
14 </resources>
```

## Projekt-Beispiel 3. Importe und Deklarationen.

```
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.text.TextUtils;
6 import android.view.Menu;
7 import android.view.MenuItem;
8 import android.view.View;
9 import android.view.View.OnClickListener;
10 import android.widget.Button;
11 import android.widget.EditText;
12 import android.widget.TextView;
13 import android.widget.Toast;
14 public class MainActivity extends Activity implements OnClickListener {
15     final int MENU_RESET_ID = 1;
16     final int MENU_QUIT_ID = 2;
17     EditText etNum1;
18     EditText etNum2;
19     Button btnAdd;
20     Button btnSub;
21     Button btnMult;
22     Button btnDiv;
23     TextView tvResult;
24     String oper = "";
```

## Projekt-Beispiel 4. onCreate().

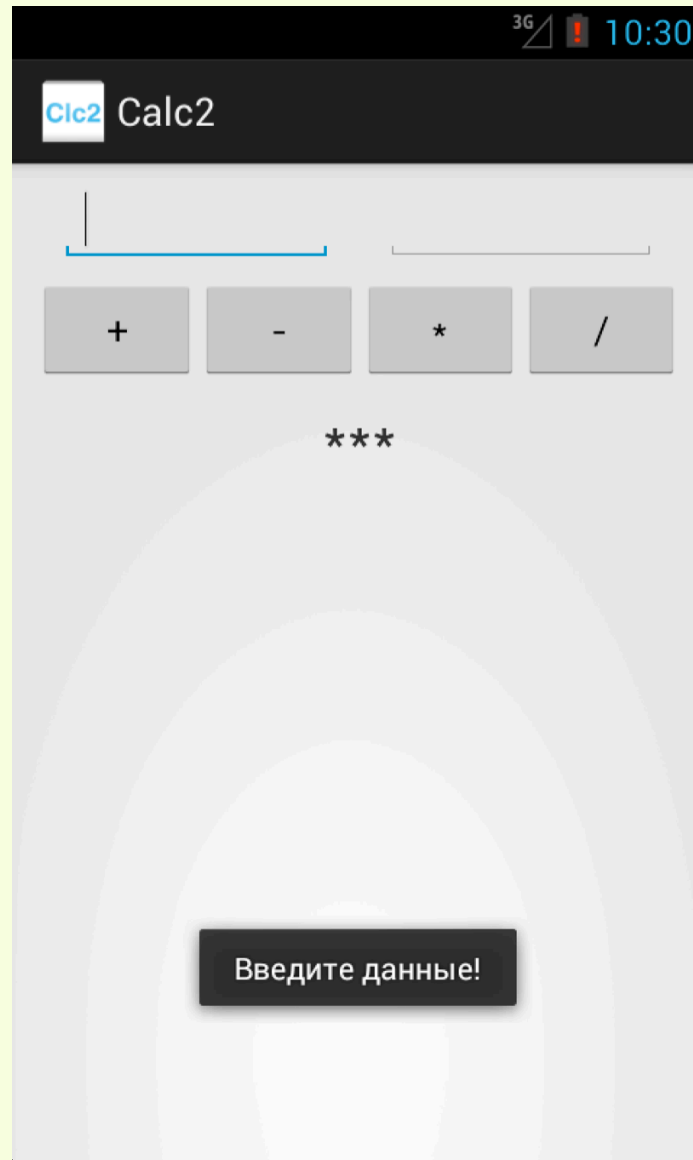
```
25  /** Called when the activity is first created. */
26  @Override
27  public void onCreate(Bundle savedInstanceState) {
28      super.onCreate(savedInstanceState);
29      setContentView(R.layout.main);
30      // find Views
31      etNum1 = (EditText) findViewById(R.id.etNum1);
32      etNum2 = (EditText) findViewById(R.id.etNum2);
33      btnAdd = (Button) findViewById(R.id.btnAdd);
34      btnSub = (Button) findViewById(R.id.btnSub);
35      btnMult = (Button) findViewById(R.id.btnMult);
36      btnDiv = (Button) findViewById(R.id.btnDiv);
37      tvResult = (TextView) findViewById(R.id.tvResult);
38      // define a Listener
39      btnAdd.setOnClickListener(this);
40      btnSub.setOnClickListener(this);
41      btnMult.setOnClickListener(this);
42      btnDiv.setOnClickListener(this);
43  }
```



## Projekt-Beispiel 5. onClick()

```
44 @Override
45 public void onClick(View v) {
46     // TODO Auto-generated method stub
47     float num1 = 0;
48     float num2 = 0;
49     float result = 0;
50     // Check data available
51     if (TextUtils.isEmpty(etNum1.getText().toString())
52         || TextUtils.isEmpty(etNum2.getText().toString())) {
53         Toast.makeText(this, "Enter operands!", Toast.LENGTH_LONG).show();
54         return;
55     }
56     // Get Operands
57     num1 = Float.parseFloat(etNum1.getText().toString());
58     num2 = Float.parseFloat(etNum2.getText().toString());
59     // Find a signal source and define an operation
60     switch (v.getId()) {
61     case R.id.btnAdd:
62         oper = "+";
63         result = num1 + num2;
64         break;
```

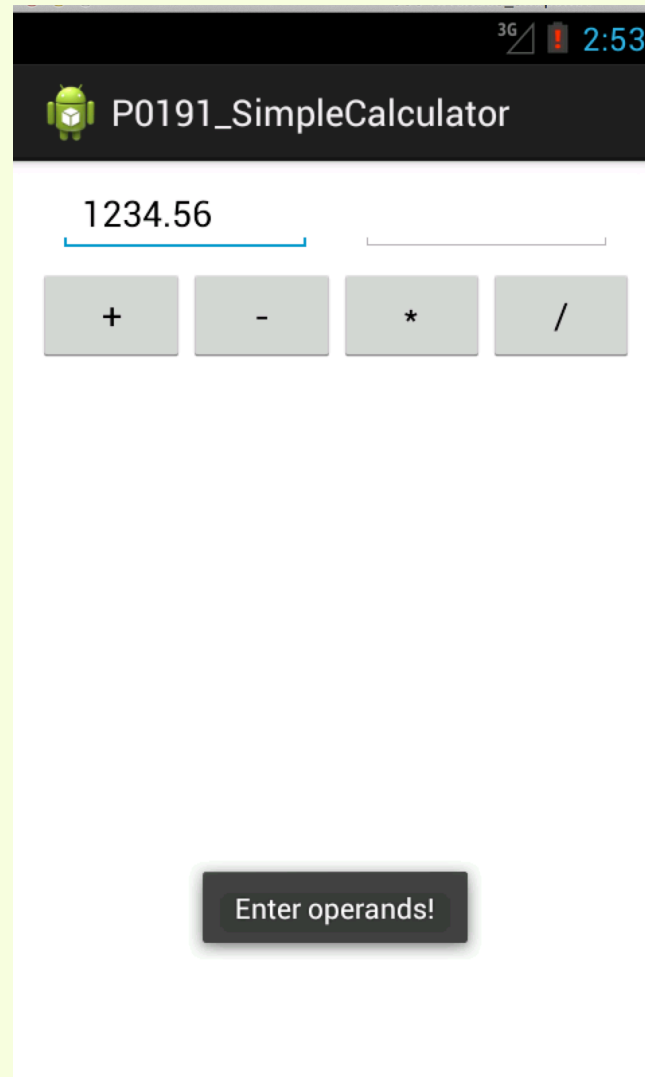
# Projekt-Beispiel 6. Objekt Toast.



## Projekt-Beispiel 7. Rezult Darstellung.

```
65     case R.id.btnSub:
66         oper = "-";
67         result = num1 - num2;
68         break;
69     case R.id.btnMult:
70         oper = "*";
71         result = num1 * num2;
72         break;
73     case R.id.btnDiv:
74         oper = "/";
75         result = num1 / num2;
76         break;
77     default:
78         break;
79 }
80 // create an output string
81 tvResult.setText(num1 + " " + oper + " " + num2 + " = " + result);
82 }
```

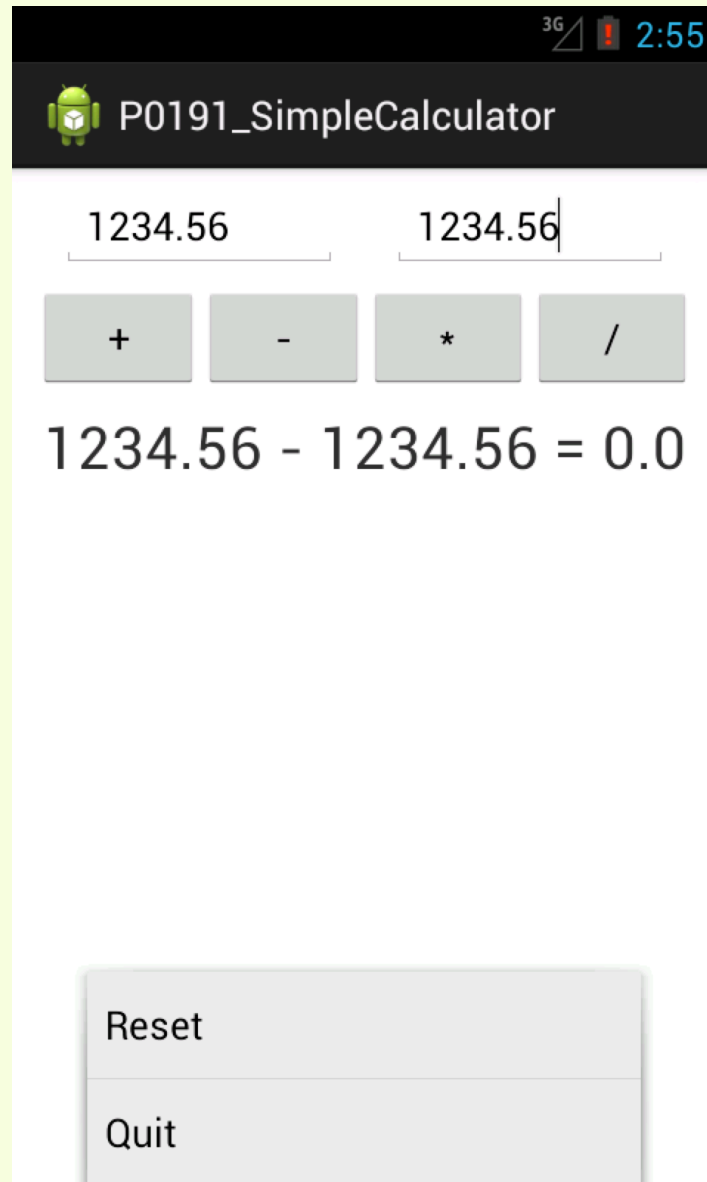
# Projekt-Beispiel 8. Menu.



## Projekt-Beispiel 9. Menu Erstellung.

```
83 @Override
84 public boolean onCreateOptionsMenu(Menu menu) {
85     // TODO Auto-generated method stub
86     menu.add(0, MENU_RESET_ID, 0, "Reset");
87     menu.add(0, MENU_QUIT_ID, 0, "Quit");
88     return super.onCreateOptionsMenu(menu);
89 }
90 @Override
91 public boolean onOptionsItemSelected(MenuItem item) {
92     // TODO Auto-generated method stub
93     switch (item.getItemId()) {
94     case MENU_RESET_ID:
95         // clear input & output fields
96         etNum1.setText("");
97         etNum2.setText("");
98         tvResult.setText("");
99         break;
100    case MENU_QUIT_ID:
101        // exit
102        finish();
103        break;
104    }
105    return super.onOptionsItemSelected(item);
106 }
107 }
```

# Projekt-Beispiel 10



# Ein Android Projekt

- Untersuchen wir ein Android Projekt
  - Calc1 und Calc3
  - Calc2

# ANDROID App Event Handlers

**Fragen?**