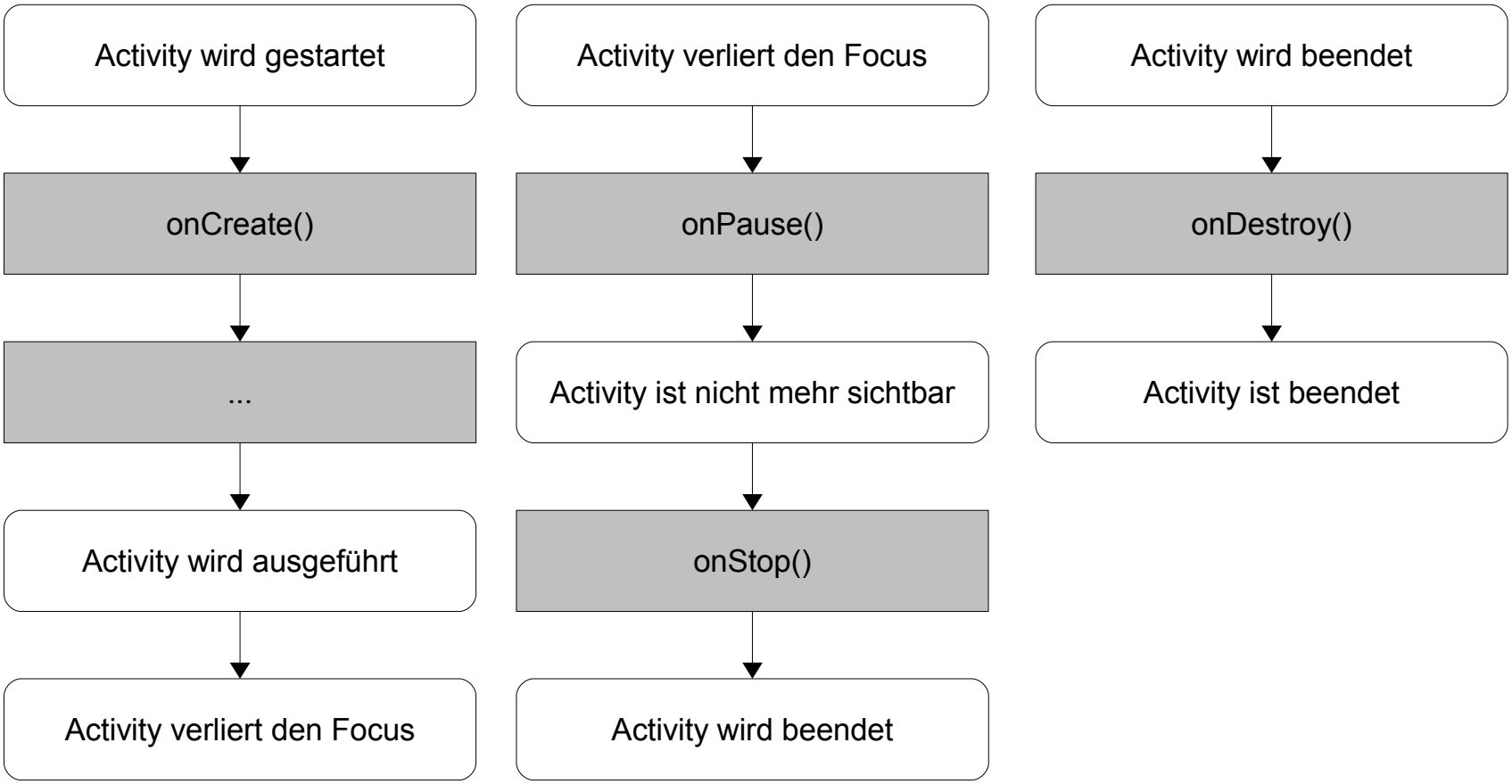
Android - Interaktion mit dem Benutzer

Activity

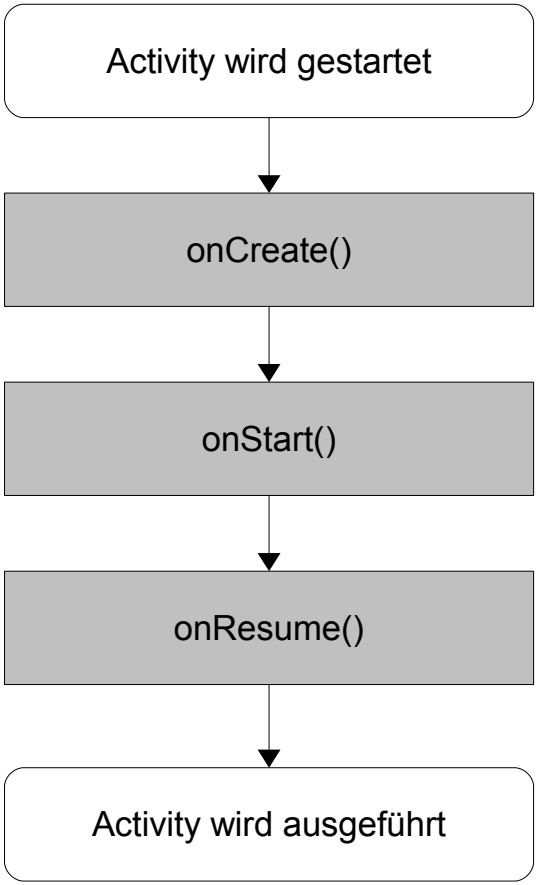
- Bildschirmseite in einer APP.
- Ablage auf einen Stack. Wenn eine Activity gestartet wird, wird diese oben auf den Stapel abgelegt. Wenn die Zurück-Schaltfläche gedrückt wird, wird die vorherige aktive Activity aufgerufen.

Leibniz Universität Hannover

Lebenszyklus einer Activity



Starten einer Activity




```
public class MainActivity extends Activity
{
   private Button btnKelvin;
   private Button btnFahrenheit;

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
   }
```

View

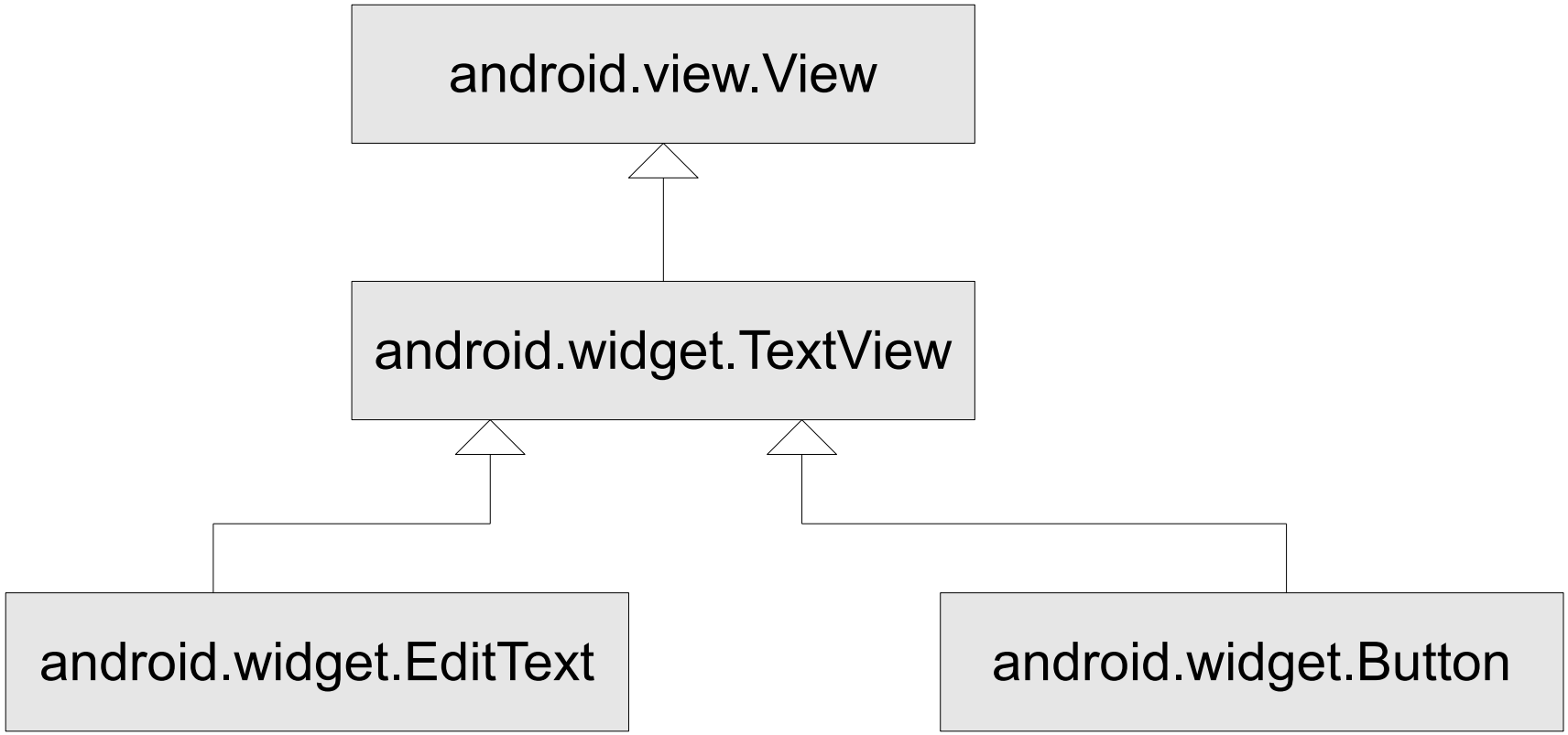
- Ansicht für den Nutzer, um mit einer Activity zu interagieren.
- Rechteckiges Element, in dem der Benutzer zeichnen kann
- Definition einer Layout-Datei.
- Siehe http://developer.android.com/reference/android/view/View.html

Widget

- Steuerelemente sind in dem Paket android.widget definiert.
- Benutzerschaltflächen, um Aktionen zu starten.
- Bezeichnungsfelder, um Informationen anzuzeigen.
- Eingabefelder, um Eingaben vom Nutzer entgegen zu nehmen und diese zu verarbeiten.
- Und so weiter.

Leibniz Universität Hannover

Objekt-Hierarchie



Import von Widgets

```
import android.widget.Button;
import android.widget.EditText;
```

- Die Klasse android.widget ist die Basisklasse für die meisten Widgets in einer App.
- Jeder Typ von Widget hat eine eigene Klasse, die vor der Nutzung importiert werden muss.
- Siehe http://www.tutorialspoint.com/android/android_user_interface_controls.htm

Bezeichnungsfelder

```
import android.widget.TextView;
```

- Anzeige von Text am Bildschirm. Der Text kann vom Nutzer nicht verändert werden.
- Rahmenloses Textfeld.
- Die Klasse android.widget.TextView stellt Methoden und Attribute für ein Bezeichnungsfeld bereit.
- Die Klasse ist Basisklasse für Schaltflächen und Eingabefelder.

Eingabefeld

```
import android.widget.EditText;
```

- Widget zur Eingabe von Text. Der angezeigte Text kann durch den Nutzer verändert werden.
- Die Klasse android.widget.EditText stellt Methoden und Attribute für ein Textfeld zur Eingabe bereit.
- Das Eingabefeld erbt von der Klasse TextView Methoden und Attribute.

Schaltfläche

```
import android.widget.Button;
```

- Mit Hilfe von Schaltflächen kann der Benutzer Aktionen starten.
- Die Klasse android.widget.button stellt Methoden und Attribute für eine Schaltfläche bereit.

Leibniz Universität Hannover

Variablen vom Typ „Widget“

```
private EditText fieldEingabe;
private Button btnFahrenheit;
```

Zugriff	Widget	name
private	Button	btnFahrenheit

Objektvariable

- Eine Objektvariable von einer bestimmten Klasse wird deklariert.
- Die Objektvariable speichert ein Verweis auf ein Objekt von der Klasse.
- Vor der Nutzung muss die Klasse importiert werden.
- Die Objektvariable ist als privat (private) deklariert.Die Variable kann nur innerhalb der dazugehörigen Klasse verändert werden. Die Variable ist vor Veränderungen von außen geschützt.

Definition von Objekten in layout / main.xml

```
<Button
        android:id="@+id/btnFahrenheit"

        android:layout_below="@id/EingabeTemperatur"
        android:layout_toRightOf="@id/btnKelvin"
        android:layout_alignParentRight="true"

        android:layout_width="@dimen/btnBreite"
        android:layout_height="wrap_content"

        android:contentDescription="@string/hinweisFahrenheit"
        android:text="@string/btnTextFahrenheit"
 />
```

Verweise auf Objekte in „R.java“

```
package de.example.AndroidExample;

public final class R {
   public static final class id {
        public static final int BeschriftungEingabefeld=0x7f050000;
        public static final int EingabeTemperatur=0x7f050001;
        public static final int btnFahrenheit=0x7f050003;
        public static final int btnKelvin=0x7f050002;
    }
}
```

Schlüsselwerte in R.java

```
public static final class id {
    public static final int BeschriftungEingabefeld=0x7f050000;
```

- In der Klasse id werden Schlüsselwerte für die verschiedenen, in der Layout-Datei definierten, Views erzeugt.
- Die Konstanten sind öffentlich (public). Auf die Konstanten kann von außen zugegriffen werden.
- Schlüsselwerte werden gleichzeitig deklariert und initialisiert. Der zugewiesen Wert kann nicht verändert werden (final).
- Die Konstanten sind für alle Objekte der Klasse id gleich (static). Schlüsselwerte werden als Klassenvariablen deklariert.

Verweis auf das Widget

```
btnFahrenheit = (Button)findViewById(R.id.btnFahrenheit);
```

- Die Konstante R.id.btnFahrenheit in der Datei *R.java* verweist auf ein Widget in der Layout-Datei der Activity.
- Die Methode findViewById() gibt ein Verweis auf eine x beliebige View zurück. Die gewünschte View wird mit Hilfe der ID ermittelt.
- Die Referenz auf eine x beliebige View wird in ein Verweis auf eine Schaltfläche (Button) konvertiert.

… und in der Objektvariablen speichern

```
btnFahrenheit = (Button)findViewById(R.id.btnFahrenheit);
```

- Die Methode findViewById() gibt ein Verweis auf eine x beliebige View zurück. Dieser Verweis wird in den gewünschten Typ konvertiert.
- Der konvertierte Verweis wird mit Hilfe des Gleichheitszeichens einer Objektvariablen vom Typ Button zugewiesen.
- Die Objektvariable und das Objekt, auf welches verwiesen wird, sollten vom gleichen Typ sein.

Event (Ereignis)

- Interaktion mit dem Benutzer.
- Aktion, die immer von einem Element der Klasse `View` oder deren Subklassen ausgelöst wird.
- „Gefilterte Systemnachrichten“.

Leibniz Universität Hannover

Beispiele

Event	Beschreibung
void onClick(View v)	Mausklick
void onLongClick(View v)	Die Maus wird gedrückt gehalten
void onFocusChange(View v, boolean f)	Fokus verlieren oder bekommen
boolean onKey(View v, int taste, KeyEvent e)	Taste gedrückt oder Taste losgelassen
boolean onTouch(View v, MotionEvent e)	Tipp-Operationen
boolean OnDrag(View v, DragEvent e)	Drag- und Drop-Operationen

Event-Handler

- Öffentliche Methoden, die auf ein Event reagiert.
- Der Name der Methode beschreibt das Event.
- Entsprechend des ausgelösten Ereignisse werden der Methode Parameter übergeben. Das Objekt, welches das Ereignis ausgelöst hat, wird an alle Methoden übergeben.
- Im Methodenrumpf wird mit Hilfe von Code auf das Ereignis reagiert.

Leibniz Universität Hannover

Beispiel: Reaktion auf Mausklick

```
public void onClick(View v)
{
    EditText fieldEingabe = (EditText)
                                findViewById(R.id.EingabeTemperatur);

    Toast AusgabeTemperatur;
    String eingabeTemperatur = fieldEingabe.getText().toString();
    String ausgabeTemperatur;
    double dblCelsius;

   if (eingabeTemperatur.length() > 0) {
       ausgabeTemperatur = Toast.makeText(v.getContext(),
                   Double.toString(kelvin), Toast.LENGTH_LONG);
        AusgabeTemperatur.show();
    }
 }
```

Event Handle onClick

```
public void onClick(View v)
{
}
```

- Als Parameter wird der Methode der Auslöser des Ereignisses übergeben. Das Ereignis kann von einer beliebigen View ausgelöst werden.

Event Listener

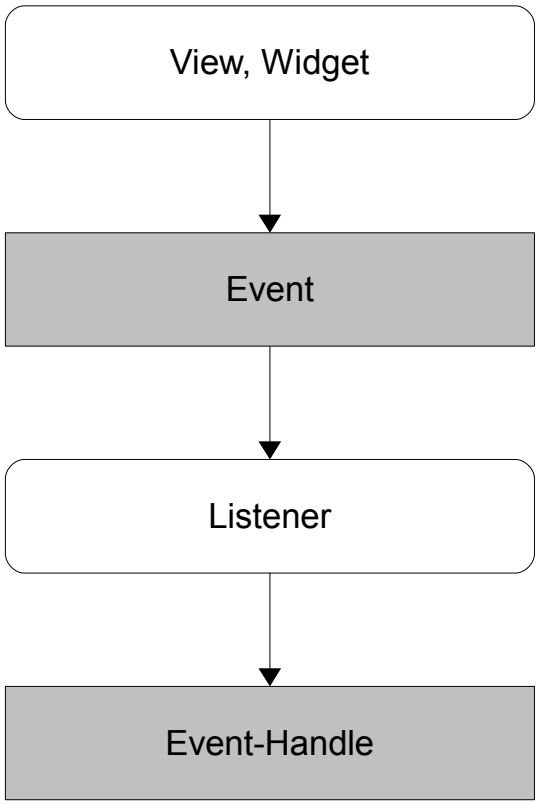
- Lauscht auf Ereignisse, die in der App ausgelöst werden.
- Schnittstelle zwischen einem Event und dem Auslöser.
- Implementierung einer Schnittstelle zwischen dem Event Handler und dem auslösenden Objekt.
- Jede Schnittstelle wartet auf einen bestimmten Typ von Event. Entsprechend werden die Methoden (Event Handler) in der Schnittstelle implementiert.

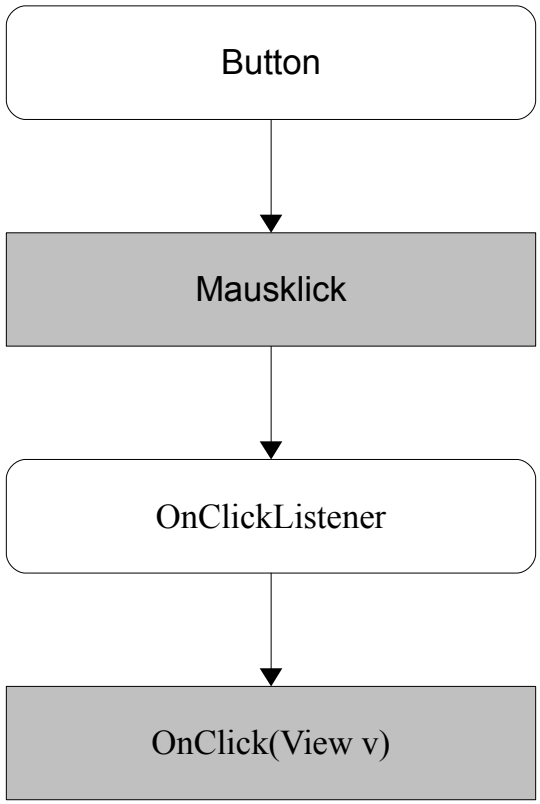
Leibniz Universität Hannover

Event und ihre Schnittstelle

Event	Schnittstelle
void onClick(View v)	OnClickListener
void onLongClick(View v)	OnLongClickListener
void onFocusChange(View v, boolean f)	OnFocusChangeListener
boolean onKey(View v, int taste, KeyEvent e)	OnKeyListener
boolean onTouch(View v, MotionEvent e)	OnTouchListener
boolean OnDrag(View v, DragEvent e)	OnDragListener

Event-Handle und -Listener





Reaktion auf ein Ereignis

```
class MyListener implements OnClickListener
{
    public void onClick(View v)
    {

    }
```

Die Methode wird in eine Klasse (Event Listener) verpackt. Die Klasse implementiert eine Schnittstelle (Interface).

Event-Handler (Ereignisprozedur) Der Code wird in eine Methode verpackt. Der Name der Methode wird durch die Schnittstelle vorgegeben.

Listener implementieren

```
import android.view.View.OnClickListener;

class MyListener implements OnClickListener { }
```

- Die Klasse MyListener verpflichtet sich alle Methoden der Klasse OnClickListener zu implementieren (implements).
- Die Klasse OnClickListener hat die Methode onClick(), die in der benutzerdefinierten Klasse implementiert werden muss.

Einbettung in die Activity-Klasse

```
import android.view.View.OnClickListener;

public class MainActivity extends Activity
{
      class MyListener implements OnClickListener { }
}
```

- Sehr übersichtliche Implementierung.
- Vorteil: Die eingebettete Klasse kann auf Fehler der äußeren Klasse zugreifen.

Registrierung des Ereignisses

```
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        btnKelvin = (Button)findViewById(R.id.btnKelvin);
        btnFahrenheit = (Button)findViewById(R.id.btnFahrenheit);

        MyClickListener btnListener = new MyClickListener();

        btnKelvin.setOnClickListener(btnListener);
        btnFahrenheit.setOnClickListener(btnListener);

    }
```

Ablauf der Registrierung

- Es muss zuerst ein Listener-Objekt vom Typ der Schnittstelle erzeugt werden.
- Dann wird ein Verweis auf eine View erzeugt.
- Die passende Listener-Methode der View wird das Listener-Objekt übergeben. Der Listener ist registriert.
- Hinweis: Es wird immer nur ein Ereignis für eine bestimmte View registriert.

Registration des Event Listener

```
MyListener btnListener = new MyListener();
btnKelvin = (Button)findViewById(R.id.btnKelvin);
btnKelvin.setOnClickListener(btnListener);
```

- Zuerst wird ein Objekt „Event Listener“ erzeugt.
- Mit Hilfe von der Methode `findViewById()` wird ein Verweis auf das gewünschte Widget erzeugt.
- Mit Hilfe der Methode `setOnClickListener()` des Widgets wird das passende Listener-Objekt registriert. Durch Auswahl der Methode werden die registrierten Events festgelegt.

Activity-Klasse als Listener-Klasse

```
import android.view.View.OnClickListener;

public class MainActivity extends Activity implements OnClickListener
{
    public void onClick(View v) { }
}
```

- Die Klasse MainActivity erbt von der Klasse Activity und implementiert die Schnittstelle OnClickListener.

Registration des Event Listener

```
btnKelvin = (Button)findViewById(R.id.btnKelvin);
 btnKelvin.setOnClickListener(this);
```

- Mit Hilfe von der Methode findViewById() wird ein Verweis auf das gewünschte Control erzeugt.
- Mit Hilfe der Methode setOnClickListener() des Widgets wird das passende Listener-Objekt registiert.
- Das Schlüsselwort this ist ein Platzhalter für das Objekt, welches die Methode aufgerufen hat. this ist in diesem Beispiel ein Platzhalter für Objekte von MainActivity.

Anonyme Listener-Klasse

```
import android.view.View.OnClickListener;

private OnClickListener myListener = new OnClickListener()
{
      public void onClick (View v) { }
};
```

Variable vom Typ ...

```
import android.view.View.OnClickListener;

private OnClickListener myListener
```

- Es wird eine Objektvariable vom Typ `OnClickListener` erzeugt.
- Die Variable ist nur in der Klasse sichtbar, in der sie definiert ist.

… erzeugen

```
import android.view.View.OnClickListener;

private OnClickListener myListener = new OnClickListener()
```

- Die Objektvariable wird gleichzeitig deklariert und intialisiert.
- Mit Hilfe von new() wird ein neues Objekt vom Typ des angegebenen Events erzeugt.

Registration des Event Listener

```
btnKelvin = (Button)findViewById(R.id.btnKelvin);
btnKelvin.setOnClickListener(myListener);
```

- Mit Hilfe der Methode findViewById() wird ein Verweis auf das gewünschte Widget erzeugt.
- Mit Hilfe der Methode setOnClickListener() des Widgets wird das passende Listener-Objekt registriert.

Reaktion auf Tastendruck

```
import android.view.View.OnKeyListener;
import android.view.KeyEvent;

private OnKeyListener myKeyListener = new OnKeyListener() {
        public boolean onKey(View v, int keyCode, KeyEvent event)
        {
            if((keyCode == KeyEvent.KEYCODE_ENTER)
              && (event.getAction() == KeyEvent.ACTION_DOWN)) {
                    return true;
            }
            return false;
        }
};
```

Wer hat die Taste gedrückt?

```
public boolean onKey(View v, int keyCode, KeyEvent event)
{
      if  (keyCode == KeyEvent.KEYCODE_ENTER)
```

- Der Parameter v enthält ein Verweis auf das Objekt, welches das Ereignis ausgelöst hat.

Welche Taste hat der Nutzer gedrückt?

```
public boolean onKey(View v, int keyCode, KeyEvent event)
{
    if  (keyCode == KeyEvent.KEYCODE_ENTER)
```

- Der Parameter keyCode wird mit Keycode-Konstanten der Klasse KeyEvent verglichen.
- Siehe http://developer.android.com/reference/android/view/KeyEvent.html.

Welches Ereignis wurde ausgelöst?

```
public boolean onKey(View v, int keyCode, KeyEvent event)
{
    if  (event.getAction() == KeyEvent.ACTION_DOWN)
```

- Jeder Tastendruck löst zwei Ereignisse aus. Mit Hilfe der Methode `event.getAction()` kann ermittelt werden, welches der zwei Ereignisse ausgelöst wurde.
- Die Taste ist gedrückt (`KeyEvent.ACTION_DOWN`).
- Die Taste wird losgelassen (`KeyEvent.ACTION_UP`).

Rückgabewert

```
public boolean onKey(View v, int keyCode, KeyEvent event)
{
      if((keyCode == KeyEvent.KEYCODE_ENTER) &&
            (event.getAction() == KeyEvent.ACTION_DOWN))
      {
            return false;
       }
        return false;
```

- True, falls der Listener das Ereignis verarbeitet hat.
- False, falls das Ereignis nicht verarbeitet wird.

Ein Event-Handler für mehrere Views

```
public class MainActivity extends Activity
{
   private Button btnKelvin;

   public void onClick (View v)
   {
        if (v == btnKelvin){
        }
```

Erläuterung

- Der Parameter `View v` verweist auf das auslösende Objekt.
- Mit Hilfe der Bedingung `v == btnKelvin` wird überprüft, ob die Objektvariable `btnKelvin` gleich dem Objekt ist, welches das Ereignis ausgelöst hat.
- Hinweis: Die Verweise auf die View müssen als Variable in der dazugehörigen Activity-Klasse gespeichert sein.

Inhalt eines Textfeldes

```
EditText fieldEingabe = (EditText)
                        findViewById(R.id.EingabeTemperatur);
String eingabeTemperatur = fieldEingabe.getText().toString();
```

- In der Variable vom Typ EditText wird ein Verweis auf ein Eingabefeld gespeichert.
- Die Methode getText() gibt den Inhalt des Eingabefeldes zurück. Die Methode liefert eine Schnittstelle zu einem editierbaren Text zurück.
- Diese Schnittstelle wird mit Hilfe von toString() in ein String konvertiert.

Textfeld setzen

```
TextView fieldCelsius = (TextView) findViewById(R.id.lblCelsius);
fieldCelsius.setText(eingabeTemperatur + " °C");
```

- In der Variable vom Typ EditText wird ein Verweis auf ein Eingabefeld gespeichert.
- Der Methode setText() wird der Text übergeben, der in dem Feld angezeigt werden soll.

Toast

```
Toast AusgabeTemperatur;
```

- Kurze Nachricht, die automatisiert am unteren Bildschirmrand eingeblendet und nach kurzer Zeit wieder ausgeblendet wird.

Nachricht erzeugen

```
AusgabeTemperatur = Toast.makeText(v.getContext(),
                                   Double.toString(kelvin),
                                   Toast.LENGTH_LONG);
```

- Die Methode makeText() erzeugt eine kurze Nachricht.

… in dem Kontext

```
public void onClick(View v)
 {
     AusgabeTemperatur = Toast.makeText(v.getContext(),
                                        Double.toString(kelvin),
                                        Toast.LENGTH_LONG);
```

- Der erste Parameter gibt den Kontext an, in dem die Nachricht ausgelöst wird.
- Die Funktion getContext() gibt den Kontext zurück, in dem die View gerade läuft.
- In diesem Beispiel: Wer hat das onClick-Ereignis ausgelöst?

Welche Nachricht wird ausgegeben?

```
AusgabeTemperatur = Toast.makeText(v.getContext(),
                                   Double.toString(kelvin),
                                   Toast.LENGTH_LONG);
```

- Der zweite Parameter definiert die Nachricht.
- In diesem Beispiel wird ein Wert von Typ double in ein String umgewandelt und als Nachricht angezeigt.

Wie lange wird die Nachricht angezeigt?

```
AusgabeTemperatur = Toast.makeText(v.getContext(),
                                   Double.toString(kelvin),
                                   Toast.LENGTH_LONG);
```

- Der dritte Parameter gibt die Dauer der Anzeige an.
- Siehe http://developer.android.com/reference/android/widget/Toast.html.

Nachricht anzeigen

```
AusgabeTemperatur.show();
```

- Die Nachricht wird kurz am unteren Rand eingeblendet.

Tastatur ausblenden

```
InputMethodManager imm =
        (InputMethodManager)
        getSystemService(Context.INPUT_METHOD_SERVICE);

imm.hideSoftInputFromWindow(fieldEingabe.getWindowToken(), 0);
```

- Hinweis: Zum Testen muss im AVD-Manager (*Tools – Android Virtual Device*) für die Emulation das Kontrollkästchen *Hardware Keyboard present* deaktiviert sein.

Veweis auf die Tastatur

```
InputMethodManager imm =
        (InputMethodManager)
        getSystemService(Context.INPUT_METHOD_SERVICE);
```

- Die Methode `getSystemService` gibt den Handle (den Referenzwert) zu einer System-Ressource zurück.
- Der Methode wird im Kontext der aktuellen Applikation das Eingabegerät übergeben.
- Das Handle wird zu einem `InputMethodManager` konvertiert.
- Siehe http://developer.android.com/reference/android/content/Context.html#getSystemService%28java.lang.String%29

InputMethodManager

- Definiert in der Bibliothek android.view.inputmethod.
- Interaktion zwischen der aktuellen Applikation und dem aktuellen Eingabegerät.
- Schnittstelle zwischen der Applikation und der Tastatur.

Tastatur ausblenden

```
imm.hideSoftInputFromWindow(fieldEingabe.getWindowToken(), 0);
```

- Mit Hilfe der Methode hideSoftInputFromWindow wird die Tastatur ausgeblendet.
- Das erste Argument enthält ein Verweis auf das Eingabegerät in Bezug auf die View, die eine Eingabe erwartet.
- Mit Hilfe des zweiten Arguments können die verschiedenen Flags für die Methode gesetzt werden. Standardmäßig wird 0 genutzt.