Android - In einer View zeichnen

Was wird benötigt?

- Ein Atelier (onDraw()).
- Eine Leinwand (android.graphics.Canvas) .
- Pinsel (android.graphics.Paint) und Farbe.

Leinwand importieren

```
import android.graphics.Canvas;
```

- Die Klasse android.graphics ist die Basisklasse für die meisten grafischen Elemente in einer View.
- Die Basisklasse android.view.View verfügt über eine Leinwand. Aber auch die Klasse android..widget.ImageView verfügt über eine Malfläche.

Beispiele für ein Atelier

```
import android.view.View;
import android.widget.ImageView;
```

- View wird in einer Layout-Datei definiert und gestaltet den Bildschirm.
- ImageView ist ein Basis-Widget zur Anzeige von Bildern, Grafiken etc.
- Von diesen Beispiel-Atelier muss ein eigenes Atelier abgeleitet werden, um den Event-Handler OnDraw() zu überschreiben.

Klasse „Eigenes Atelier“

```
public class DrawView extends ImageView
{


}
```

- Das Schlüsselwort class kennzeichnet eine Klasse in Java.
- Die Klasse hat einen eindeutigen Namen (DrawView).
- Die Klasse ist öffentlich (public). Die Klasse kann von außen aufgerufen werden.
- Die Klasse erbt von der Klasse ImageView Attribute und Methoden.

Bauanleitung für das eigene Atelier

```
public class DrawView extends ImageView
{

    public DrawView(Context context)
    {
        super(context);
    }
```

Konstruktoren (Bauanleitung)

- Methoden zur Erzeugung von Objekten einer Klasse. In diesem Beispiel wird eine `DrawView` konstruiert.
- Konstruktoren und die dazugehörige Klasse haben den gleichen Namen. Durch das Schlüsselwort `new()` wird automatisch der Konstruktor der Klasse aufgerufen.
- Übergabe von Startwerten für die Attribute von Objekten. In diesem Beispiel wird der Parameter `Context context` übergeben. Der Parameter gibt darüber Auskunft, wie in welcher Umgebung das Atelier gebaut werden soll.

Konstruktoren der Basisklasse

- super ist ein Platzhalter für die Basisklasse. In diesem Beispiel erbt die Klasse mit Hilfe von extends von der Klasse ImageView.
- Mit Hilfe des Schlüsselwortes super(arg01, …, argN) wird der passende Konstruktor in der dazugehörigen Basisklasse aufgerufen. In diesem Beispiel wird der Konstruktor der Basisklasse ImageView aufgerufen.
- Der Konstruktor der Basisklasse muss immer in der ersten Zeile eines Konstruktors aufgerufen werden.

Speicherung in einer eigenen Datei

```
public class DrawView extends ImageView
{


}
```

- In NetBeans: *File – New File*. File Type: *Java Class*. Eingabe eines Klassennamens.

Leibniz Universität Hannover

Atelier bauen

```
public class MainActivity extends Activity
{
   DrawView layoutZeichnen;

   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);

      layoutZeichnen = new DrawView(this);
      setContentView(layoutZeichnen);
   }
```

Objektvariable

```
DrawView layoutZeichnen;
```

- Klasse varName;
- Eine Objektvariable von einer bestimmten Klasse wird deklariert. In diesem Beispiel wird eine Variable von der Klasse DrawView deklariert.
- Klassen, die nicht im aktuellen Projekt vorhanden sind, werden importiert.

OnCreate() der Activity

```
super.onCreate(savedInstanceState);
```

- Die Activity wird erzeugt.
- super ist ein Platzhalter für die Basisklasse Activity.
- Im ersten Schritt wird die Methode onCreate der Basisklasse aufgerufen.
- Der Methode wird der letzte aktuelle Status der Activity übergeben.

Instanz einer Zeichenfläche erzeugen

```
DrawView layoutZeichnen;
layoutZeichnen = new DrawView(this);
```

- new Klasse(arg01, … argN) erzeugt ein Objekt einer bestimmten Klasse. Die Anzahl der Argumente sind abhängig vom gewählten Konstruktor der Klasse.
- In diesem Beispiel wird ein Objekt von der Klasse DrawView erzeugt. Das Atelier wird gebaut
- Der Parameter this ist ein Platzhalter für die aktuelle Instanz einer Klasse (in diesem Beispiel MainActivity). Die aktuelle Instanz wird in Bezug zu der Zeichenfläche gesetzt.

Hintergrundfarbe der Zeichenfläche

```
layoutZeichnen.setBackgroundColor(Color.BLACK);
```

- Der Methode setBackgroundColor() wird die Hintergrundfarbe übergeben.
- Die Klasse Color hat vordefinierte Farbkonstanten.
- Mit Hilfe von Color.rgb(red, green, blue) kann eine eigene Farbe gemischt werden.

Farben in Android

- Das RGB-Farbsystem addiert (mischt) Licht in den Farben Rot, Grün und Blau.
- Jede der drei Farbe wird in 256 Helligkeitsstufen unterteilt. Um so eine Farbe sich dem Wert „Weiß“ nähert, um so heller wird diese.
- Für die Farbanteile Rot, Grün und Blau kann jeweils ein hexadezimaler Wert von 00 (= 0) bis FF (= 255) eingegeben werden.

„Zeichenfläche“ laden

```
layoutZeichnen = new DrawView(this);
setContentView(layoutZeichnen);
```

- Mit Hilfe der Methode setContentView() wird die neu erstellte Zeichenfläche geladen.
- Der Methode wird die Ressourcen-ID der Zeichenfläche übergeben.

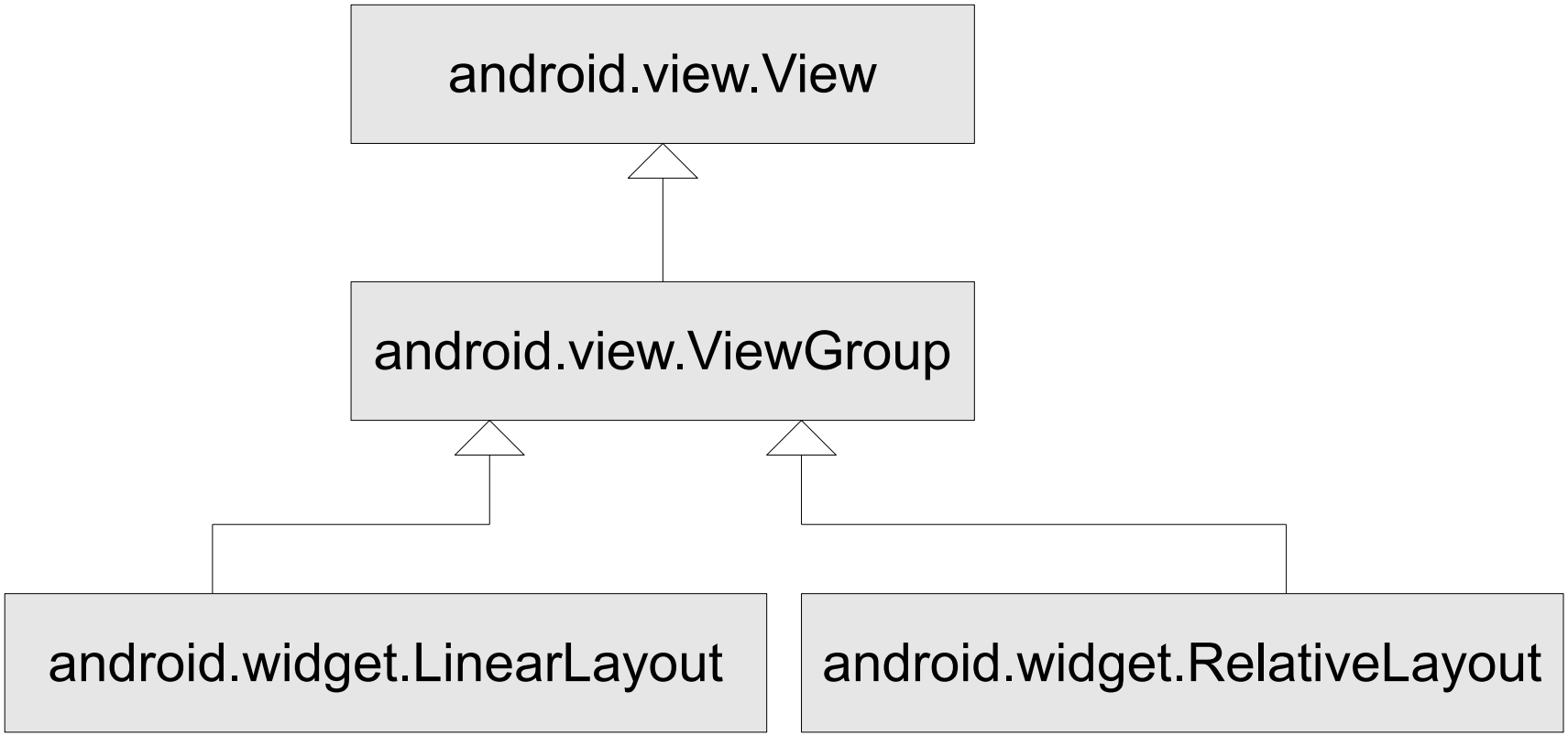
… in ein bestehendes Layout einfügen

```
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
public class MainActivity extends Activity{
   DrawView layoutZeichnen;

   public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      layoutZeichnen = new DrawView(this);

      LayoutParams myLayoutParams = new LayoutParams(LayoutParams.MATCH_PARENT,
                                                     250);
      layoutZeichnen.setLayoutParams(myLayoutParams);
      LinearLayout layout = (LinearLayout) findViewById(R.id.myLayout);
      layout.addView(layoutZeichnen);
   }
```

ViewGroups

- Container für andere Views.
- Anordnung von Views nach bestimmten Regeln.
- Positionierung von Views mit der Hilfe von bestimmten Layout-Parametern.

Objekthierarchie



Layout-Parameter einer ViewGroup

```
import android.view.ViewGroup.LayoutParams;

LayoutParams myLayoutParams = new
      LayoutParams(LayoutParams.MATCH_PARENT, 250);
```

- new LayoutParams(int breite, int hoehe) erzeugt ein neues Objekt vom Typ LayoutParams.
- Die Layout-Parameter für eine ViewGroup werden definiert.
- Die Angaben zur Breite und Höhe einer View sind die einzigen Layout-Parameter einer ViewGroup.

Setzen von Layout-Parametern einer View

```
LayoutParams myLayoutParams = new
        LayoutParams(LayoutParams.MATCH_PARENT, 250);

layoutZeichnen = new DrawView(this);
layoutZeichnen.setLayoutParams(myLayoutParams);
```

- Das Objekt layoutZeichnen ruft die Methode setLayoutParams() auf. Aufrufer und Methode werden durch ein Punkt verbunden.
- Methoden, die mit set beginnen, setzen häufig Attribute. In diesem Beispiel werden die Layout-Parameter der Zeichenfläche gesetzt.
- Als Parameter wird der Methode eine Objektvariable von der Klasse LayoutParams übergeben.

Zeichenfläche in ein lineares Layout einhängen

```
import android.widget.LinearLayout;
LinearLayout layout = (LinearLayout) findViewById(R.id.myLayout);
layoutZeichnen = new DrawView(this);
layout.addView(layoutZeichnen);
```

- Die Ressourcen-ID des aktuellen Layouts (*main.xml*) wird mit Hilfe der Methode findViewById() in einer Variablen vom Typ „Layout“ gespeichert.
- Mit Hilfe der Methode addView() wird der definierten Ressource eine View hinzugefügt. In diesem Beispiel wird der Benutzeroberfläche eine Zeichenfläche hinzugefügt.

Layout-Parameter für ein relatives Layout setzen

```
import android.widget.RelativeLayout.LayoutParams;

LayoutParams myLayoutParams = new
                LayoutParams(LayoutParams.MATCH_PARENT, 250);

myLayoutParams.addRule(
                        RelativeLayout.ALIGN_PARENT_BOTTOM);

layoutZeichnen.setLayoutParams(myLayoutParams);
```

… definieren

```
import android.widget.RelativeLayout.LayoutParams;

LayoutParams myLayoutParams = new
                    LayoutParams(LayoutParams.MATCH_PARENT, 250);
```

- new LayoutParams(int breite, int hoehe) erzeugt ein neues Objekt vom Typ LayoutParams.
- In diesem Beispiel wird die Breite und Höhe gesetzt. Die Breite wird in Bezug auf die Eltern gesetzt.

View positionieren

```
import android.widget.RelativeLayout.LayoutParams;

myLayoutParams.addRule(
                        RelativeLayout.ALIGN_PARENT_BOTTOM);
```

- Wie wird die Zeichenfläche in einem relativen Layout positioniert?
- In diesem Beispiel wird die Zeichenfläche am unteren Rand der Eltern positioniert.
- Weitere Möglichkeiten: http://developer.android.com/reference/android/widget/RelativeLayout.html

Hinweise

- Für jede Positionsbeschreibung / Verankerung wird eine eigene Regel geschrieben.
- Möglichkeiten zur Positionierung in einem relativen Layout siehe http://developer.android.com/reference/android/widget/RelativeLayout.html

Setzen von Layout-Parametern einer View

```
layoutZeichnen = new DrawView(this);
layoutZeichnen.setLayoutParams(myLayoutParams);
```

- Das Objekt layoutZeichnen ruft die Methode setLayoutParams() auf. Aufrufer und Methode werden durch ein Punkt verbunden.
- Methoden, die mit set beginnen, setzen häufig Attribute. In diesem Beispiel werden die Layout-Parameter der Zeichenfläche gesetzt,.
- Als Parameter wird der Methode eine Objektvariable von der Klasse LayoutParams übergeben.

… und einhängen

```
import android.widget.RelativeLayout;
RelativeLayout layout = (RelativeLayout)
                                findViewById(R.id.LayoutKreisBerechnung);
layout.addView(layoutZeichnen);
```

- Die Ressourcen-ID des aktuellen Layouts (*main.xml*) wird mit Hilfe der Methode findViewById() in einer Variablen vom Typ „Layout“ gespeichert.
- Mit Hilfe der Methode addView() wird der definierten Ressource eine View hinzugefügt. In diesem Beispiel wird der Benutzeroberfläche eine Zeichenfläche hinzugefügt.

Zeichnen

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

public class DrawView extends ImageView {
        Paint kreis = new Paint();
        int radius = 15;

       @Override
        public void onDraw(Canvas leinwand){
            super.onDraw(leinwand);

            kreis.setARGB(255, 255, 255, 255);
            leinwand.drawCircle(100, 100, radius, kreis);
        }
```

onDraw()

```
@Override
public void onDraw(Canvas leinwand){
        super.onDraw(leinwand);
}
```

- Die Methode überschreibt die, in der Basisklasse definierte, Methode.
- Als Parameter wird der Methode die Zeichenfläche übergeben, auf der gemalt werden soll.

Aufruf der Methode onDraw() der Basisklasse

```
@Override
public void onDraw(Canvas leinwand){
        super.onDraw(leinwand);
}
```

- In der Methode der Subklasse wird die passende Methode der Basisklasse in der ersten Zeile aufgerufen.
- super ist ein Platzhalter für die Basisklasse.
- Der Platzhalter ruft die verdeckte Methode OnDraw() der Basisklasse auf. Der Methode wird eine Zeichenfläche übergeben.

Pinsel

```
import android.graphics.Paint;
Paint kreis = new Paint();
```

- Die Stärke und die Farbe des Pinsels wird festgelegt.
- Linien- und Farbinformationen für geometrische Körper, Bilder und Schrift können mit Hilfe von Methoden für den Pinsel gesetzt werden.

Farbe

```
kreis.setARGB(255, r, g, b);
kreis.setColor(Color.rgb(r, g, b));
kreis.setColor(Color.RED);
```

- Eine Farbe besteht aus den drei Farbanteilen rot, grün und blau.
- Jeder dieser drei Farben wird durch ein Integer-Wert von 0 bis 255 definiert.
- Standardmäßig deckt die Farbe die Zeichenfläche vollständig ab. Eine Farbe kann aber auch nur einen bestimmten Anteil Deckkraft haben.

Möglichkeiten

- Der Methode setARGB(alpha, red, green, blue) wird als Parameter die Deckkraft eines Pixels (alpha), der Rotanteil, der Grünanteil und der Blauanteil übergeben. Die Deckkraft wird durch einen Integer-Wert von 0 bis 255 angegeben.
- Der Methode setColor() kann direkt eine, in der Klasse Color, definierte Konstante übergeben werden. Mit Hilfe der Methode Color.rgb() kann auch eine Farbe mit Hilfe von Rot, Grün und Blau gemischt werden.

Koordinaten

```
leinwand.drawPoint(x1, y1, punkt);
leinwand.drawCircle(x1, y1, radius, kreis);
```

- Koordinaten können als integer- oder float-Werte angegeben werden.
- Das Ereignis onTouch liefert float-Werte zurück.
- Die linke obere Ecke der Zeichenfläche hat die Koordinaten 0,0.
- Die x-Werte werden nach rechts größer. Die y-Werte werden nach unten hin größer.

Pinselstärke

```
punkt.setStrokeWidth(2);
punkt.setStyle(Paint.Style.FILL);
```

- Mit Hilfe der Methode setStrokeWidth() wird die Pinselstärke festgelegt.
- Standardmäßig ist ein Punkt ausgefüllt (Paint.Style.FILL).

Kreis auf die Leinwand zeichnen

```
kreis.setARGB(255, r, g, b);
leinwand.drawCircle(x1, y1, radius, kreis);
```

- Mit Hilfe der Methode drawCircle() wird ein Kreis in einem bestimmten Radius um einen Mittelpunkt gezeichnet.
- Als letzter Parameter wird der Pinsel angegeben, mit dem der Kreis auf die Zeichenfläche gezeichnet wird.

Canvas-Methoden zum Zeichnen

Methode	Erläuterung
drawCircle(float x, float y, float radius, Paint pinsel)	Kreis
drawLinie(float startX, float startY, float endX, float endY , Paint pinsel)	Linie von start... bis end...
drawRect(float links, float rechts, float oben, float unten, Paint pinsel)	Rechteck

- Siehe http://developer.android.com/reference/android/graphics/Canvas.html.

… auf Mausklick verschieben

```
public class DrawView extends ImageView  implements OnTouchListener {
        public boolean onTouch(View v,MotionEvent event){
                        int aktion = event.getAction();

                        if(aktion == MotionEvent.ACTION_DOWN){
                                        x1 = (int)event.getX()-(radius/2);
                                        y1 = (int)event.getY()-(radius/2);

                                        invalidate();
                                        return true;
                        }
                        return false;
        }
```

Klassen importieren

```
import android.view.View;
import android.view.View.OnTouchListener;
import android.view.MotionEvent;
```

- Die Klasse OnTouchListener reagiert auf Tipp- und Wischbewegungen.
- In der Emulation werden diese Bewegungen mit Hilfe der linken Maustaste simuliert.
- Die Klasse MotionEvent muss für die entsprechenden Ereignisse importiert werden.

Tipp- und Wischereignisse abfangen

```
public class DrawView extends ImageView
                              implements OnTouchListener {
    public boolean onTouch(View v,MotionEvent event){
    }
}
```

- Die Klasse DrawView verpflichtet sich alle Methoden der Klasse OnTouchListener zu implementieren (implements).
- Die Klasse OnTouchListener hat die Methode onTouch(), die in der benutzerdefinierten Klasse implementiert werden muss.

Implementierung des Listeners

```
public class DrawView extends ImageView
                              implements OnTouchListener {
```

- Die Klasse DrawView verpflichtet sich alle Methoden der Klasse OnTouchListener zu implementieren (implements).
- Die Klasse OnTouchListener hat die Methode onTouch(). Diese Methode muss in der Klasse DrawView implementiert werden.

Methode onTouch()

```
public boolean onTouch(View v,MotionEvent event)
{
        int aktion = event.getAction();

        if(aktion == MotionEvent.ACTION_DOWN){
          x1 = (int)event.getX()-(radius/2);
          y1 = (int)event.getY()-(radius/2);

          invalidate();
          return true;
        }
        return false;
}
```

Parameter des Event-Handlers

```
public boolean onTouch(View v,MotionEvent event)
{


}
```

- Dem Event-Handler wird der Auslöser des Ereignisses als Parameter übergeben. Das Ereignis wird durch eine x beliebige View ausgelöst.
- Mit Hilfe des zweiten Parameters wird das auslösende Ereignis beschrieben.

Rückgabewert des Event-Handlers

```
public boolean onTouch(View v,MotionEvent event)
{
    return true;
}
```

- Mit Hilfe des Rückgabewertes true wird dem Aufrufer mitgeteilt, dass das Ereignis behandelt wurde.
- Eine Weiterreichung von der übergeordneten Layout-View zu dem Ziel-View wird unterbunden.

Welches Ereignis wurde ausgelöst?

```
int aktion = event.getAction();
```

- Wenn der Benutzer den Touchscreen berührt, wird `MotionEvent.ACTION_DOWN` ausgelöst.
- Wenn der Benutzer den Finger hebt, wird `MotionEvent.ACTION_UP` ausgelöst.
- Mit Hilfe der Methode `event.getAction()` wird ermittelt, welches Ereignis den Event-Handler ausgelöst hat.

Aktuelle Koordinaten

```
x1 = (int)event.getX()-(radius/2);
y1 = (int)event.getY()-(radius/2);
```

- event.getX() gibt die aktuelle x-Koordinate vom Typ float zurück.
- event.getY() gibt die aktuelle y-Koordinate vom Typ float zurück.

Neu zeichnen

```
invalidate();
```

- Die Funktion invalidate() löst das Ereignis onDraw() aus.