Apps

Programmierung von Android-Smartphones

Android-Apps

Gliederung:

- Warum? / Warum Android?
- Grundlagen
- Beispiel (sehr kurz)
- weitere Möglichkeiten
- Einsatz im Unterricht
- Diskussion / Fragen

Smartphone-Programmierung – Warum?

Contra:

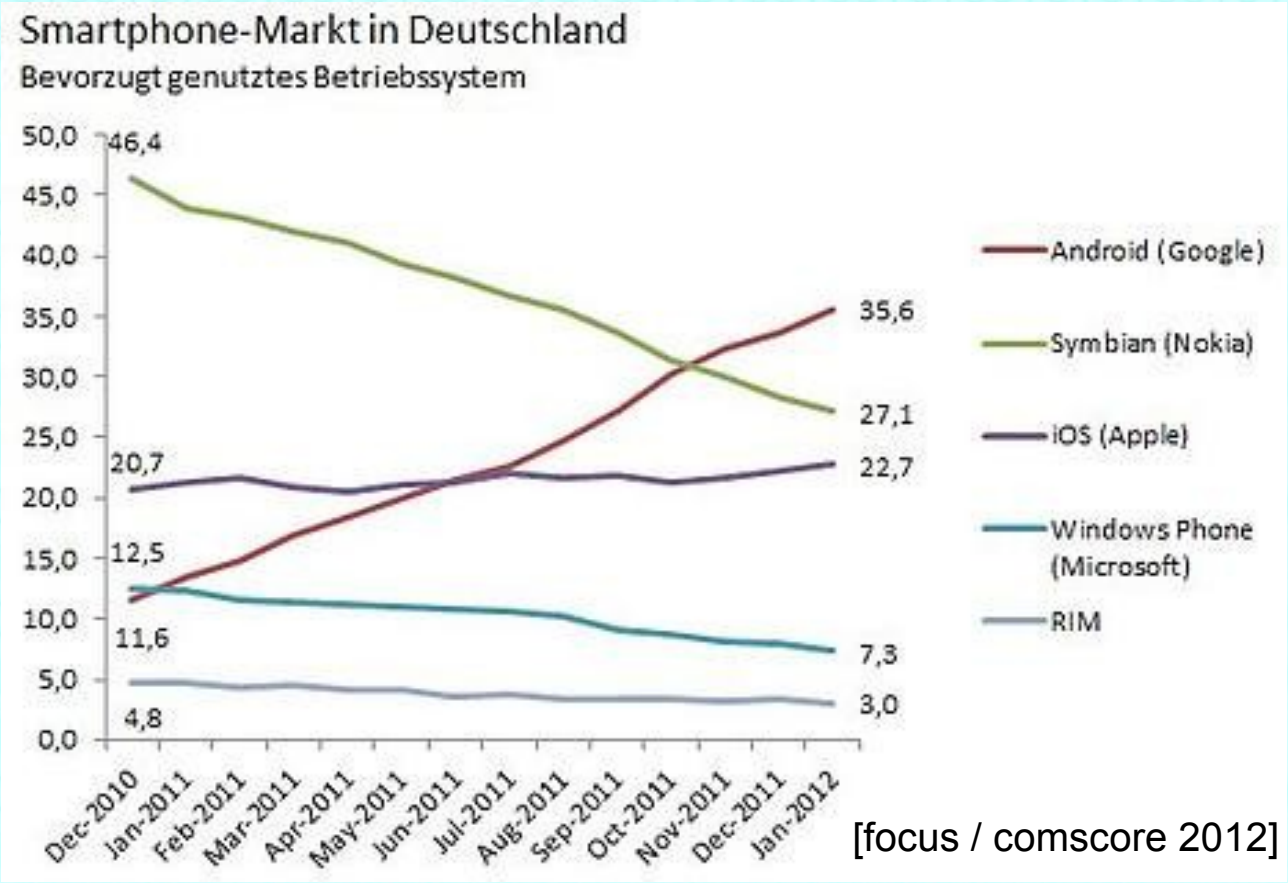
- klein, limitierte Möglichkeiten
- keine *in situ* Entwicklung möglich (bis auf Script-Sprachen)
- noch kein Standardgerät

Pro:

- 30% der Handynutzer benutzen bereits ein Smartphone, 90% der über 16 jährigen besitzen ein Handy [nielsen 12/2011]
- seit dem letzten Jahr werden mehr Smartphones als PC verkauft
- das Smartphone ist ein Computer, der immer dabei ist.
- neue Möglichkeiten (GPS, SMS, …)
- Apps, aber …
  - Sicherheitsprobleme
  - häufig nicht passgenau

Warum Android?

- Marktführer
- Relativ günstige Geräte
- offenes System / keine Zensur (*IPhone*)
- Sandbox-Prinzip
- Kostenlose Entwicklungsplattform
- *Java* / *XML* Symbiose



[focus / comscore 2012]

Android: Grundlagen

- Basis ist ein *Linux*-Kernel
- (ursprünglich) konzipiert für *ARM*-Prozessoren
- optimiert in Bezug auf Energieverbrauch und Speichermanagement
- (in kürze) verfügbar für Netbooks, Tablett-PC, Festnetztelefone, Spielekonsolen, Auto-Infotainment-Systeme, Set-Top-Boxen, Kühlschränke, …
- Android Anwendungen sind offen. Eigene Komponenten können von anderen Anwendungen benutzt werden und eigenen Anwendungen können andere Komponenten benutzen (z.B. können die Adressdaten aus der Kontakte-DB ausgelesen werden oder die SMS-Funktion benutzt werden)
- Programmiert wird in *Java*, die Ein- und Ausgabe wird mit *XML* beschrieben
- Das *Android SDK* (Software Development Kit) kann in die *Java* Entwicklungsumgebung *Eclipse* eingebunden werden
- Durch das Sandbox-Prinzip kann der Anwender absolute Sicherheit erreichen
- Im Market vertriebene Apps sind eindeutig zertifiziert, ansonsten sind eigene Zertifikate möglich.

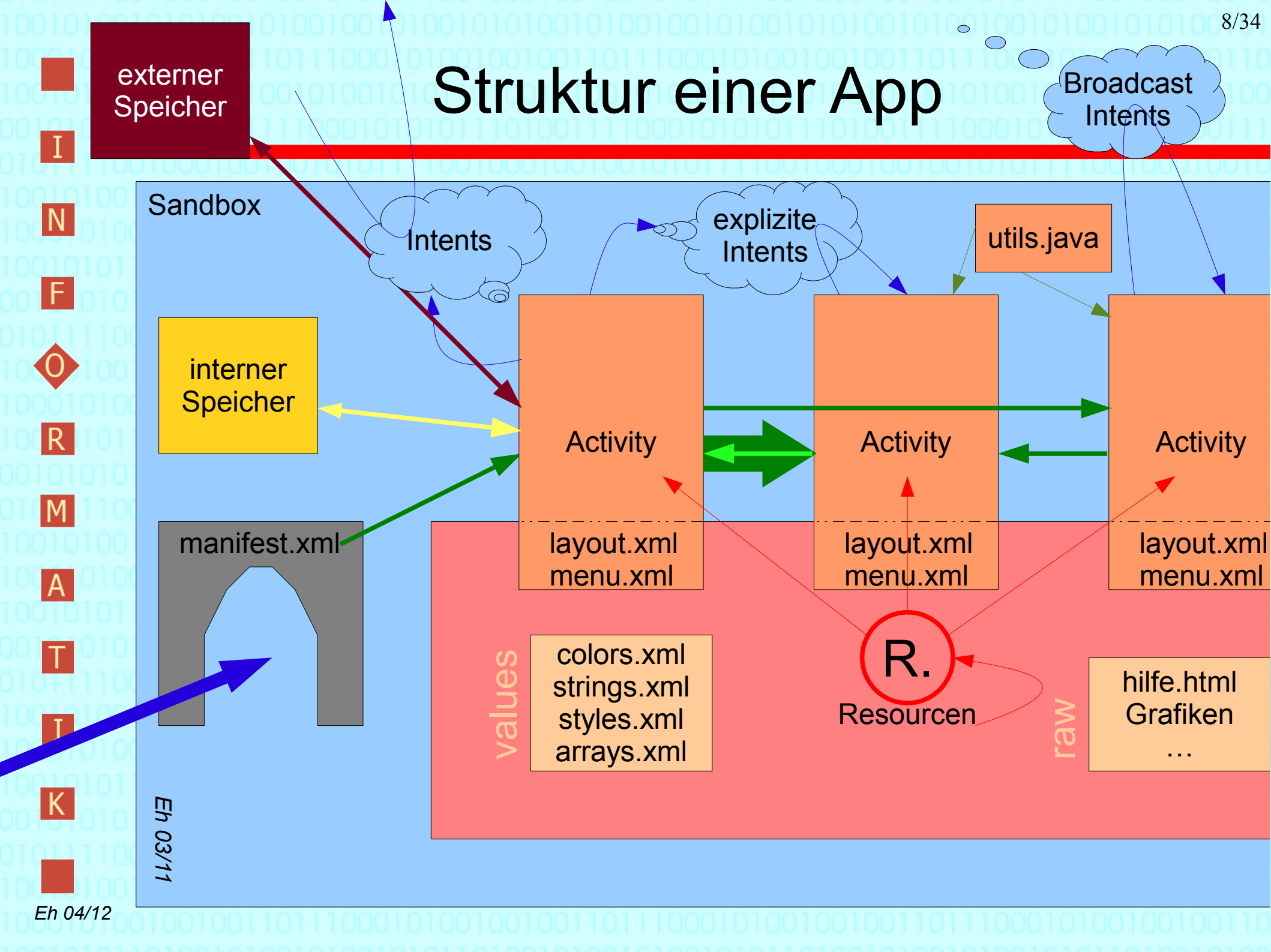
Dalvik Virtual Machine (*DVM*)

- *DVM* ist nicht *JVM*
- *Android* lässt sich aber komplett in *Java* programmieren
- Per Cross-Compiling wird aus *Java*-Bytecode *Dalvik* Bytecode
- Dieser wird zur Laufzeit von der *DVM* ausgeführt
- Warum *DVM*?
  - *JVM*s nutzten moderne Prozessorarchitektur (*ARM*) nicht aus (z.B. Zwischenspeicher direkt im Mikroprozessor)
  - Mit *DVM* können auf kleinen Computern mehrere Instanzen parallel laufen (→ Sandbox)
  - statische Grafik wird bereits bei der Entwicklung compiliert
  - Lizenzrechte (*JVM*-Rechte bei *Oracle*, Programmiersprache *Java* nicht)

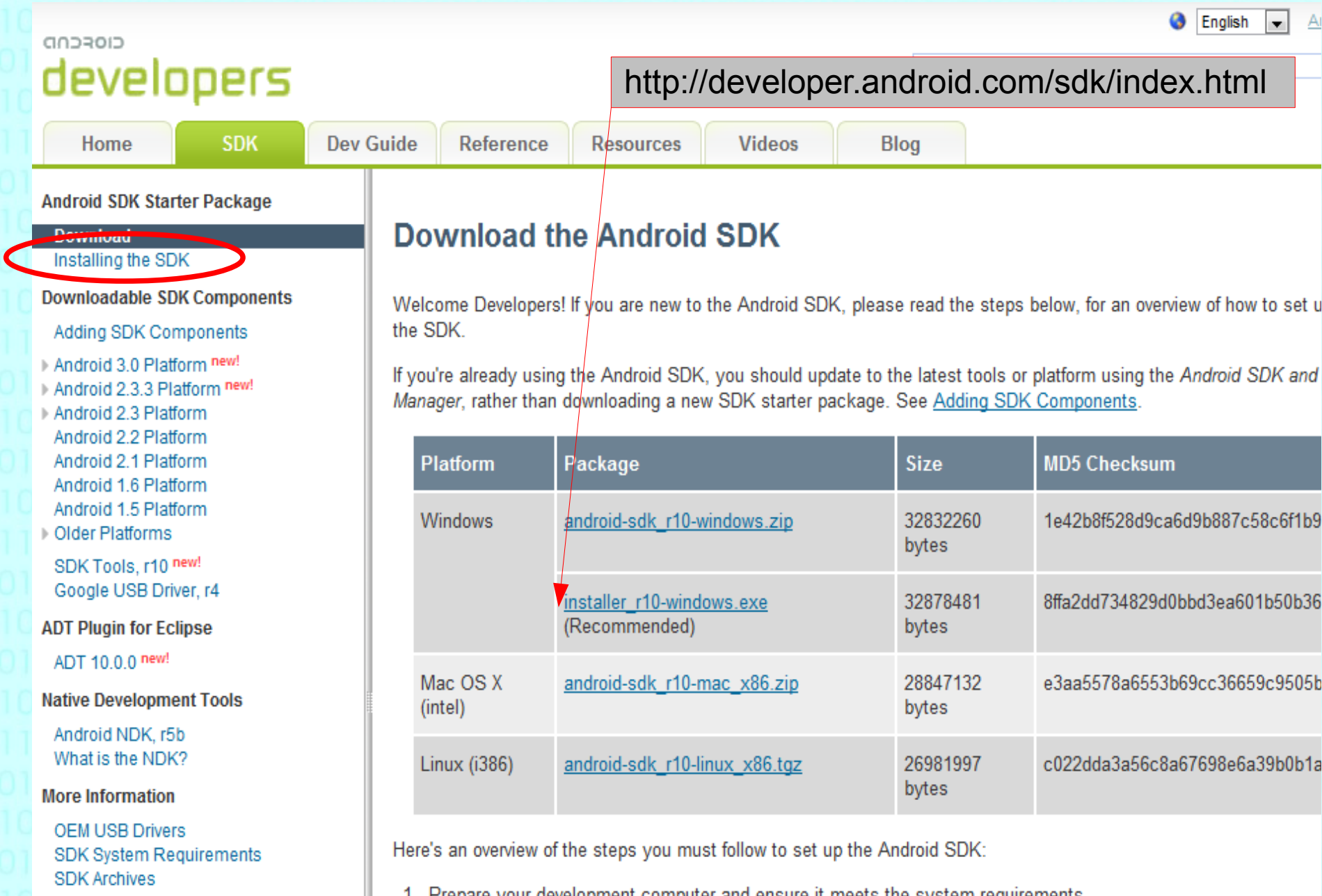
Sandbox

- Die *DVM* ermöglicht es, auch auf kleinen Computern mehrere Instanzen parallel laufen zu lassen.
- Sandbox-Prinzip: Jede App im eigenen „Sandkasten“
  - eigener Prozess
  - eigener Betriebssystem-User
  - eigene *DVM*
  - eigener Bereich im Hauptspeicher
  - eigener Bereich im Dateisystem
- Der Anwender muss vor der Installation einer App alle Zugriffe aus der Sandbox genehmigen. (Oder ggf. auf die Installation verzichten)
- Genehmigungspflichtig sind z.B.:
  - Internetzugriff
  - GPS-Daten-Zugriff
  - Empfang/Versand von SMS
  - Zugriff auf die Kontakte

Struktur einer App



Android SDK


http://developer.android.com/sdk/index.html
English
android
developers
Home
SDK
Dev Guide
Reference
Resources
Videos
Blog
Android SDK Starter Package
Installing the SDK
Downloadable SDK Components
Adding SDK Components
Android 3.0 Platform new!
Android 2.3.3 Platform new!
Android 2.3 Platform
Android 2.2 Platform
Android 2.1 Platform
Android 1.6 Platform
Android 1.5 Platform
Older Platforms
SDK Tools, r10 new!
Google USB Driver, r4
ADT Plugin for Eclipse
ADT 10.0.0 new!
Native Development Tools
Android NDK, r5b
What is the NDK?
More Information
OEM USB Drivers
SDK System Requirements
SDK Archives
Download the Android SDK
Welcome Developers! If you are new to the Android SDK, please read the steps below, for an overview of how to set u
the SDK.
If you're already using the Android SDK, you should update to the latest tools or platform using the Android SDK and
Manager, rather than downloading a new SDK starter package. See Adding SDK Components.
Platform
Package
Size
MD5 Checksum
Windows
android-sdk_r10-windows.zip
32832260 bytes
1e42b8f528d9ca6d9b887c58c6f1b9
installer_r10-windows.exe (Recommended)
32878481 bytes
8ffa2dd734829d0bbd3ea601b50b36
Mac OS X (intel)
android-sdk_r10-mac_x86.zip
28847132 bytes
e3aa5578a6553b69cc36659c9505b
Linux (i386)
android-sdk_r10-linux_x86.tgz
26981997 bytes
c022dda3a56c8a67698e6a39b0b1a
Here's an overview of the steps you must follow to set up the Android SDK:

Android Virtual Device

Literatur / Hilfen - deutsch

- Arno Becker und Marcus Pant:
  Android 2: Grundlagen und Programmierung,
  Dpunkt Verlag, ISBN-13: 978-3898646772, 39,90€

  gute Java und Eclipse-Kenntnisse erforderlich,
  anspruchsvoll

- Thomas Künneth: Android 3 – Apps entwickeln mit dem Android SDK, Galileo Computing, ISBN-13: 978-3-8362-1697-5, 34,90€

  viele Bsp-Apps, Java-Kenntnisse erforderlich,
  an sich für professionelle App-Entwickler

- http://www.android-hilfe.de

  Das beste deutschsprachige Hilfeforum befindet sich im Unter-Unterpunkt Android Developer | Android App Entwicklung

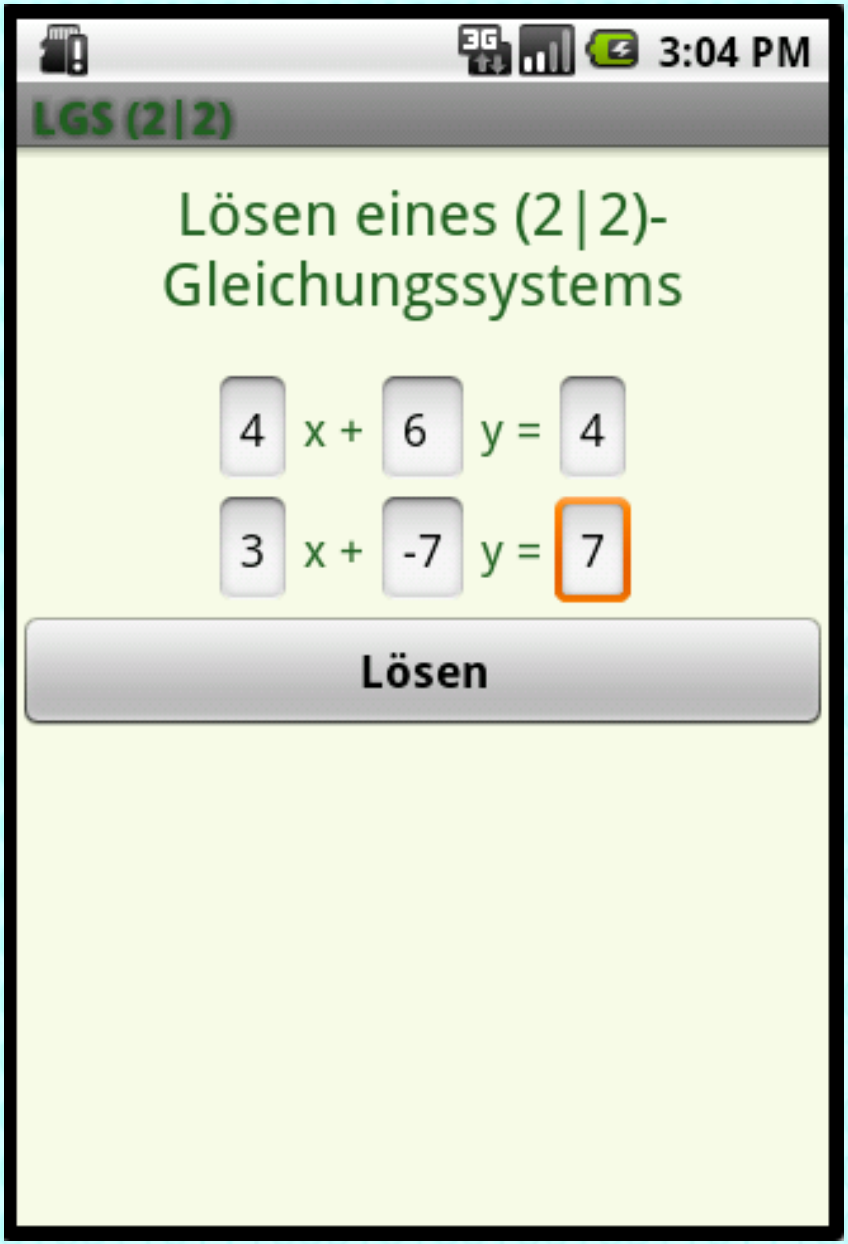
- http://www.gailer-net.de/tutorials/java5/index.html

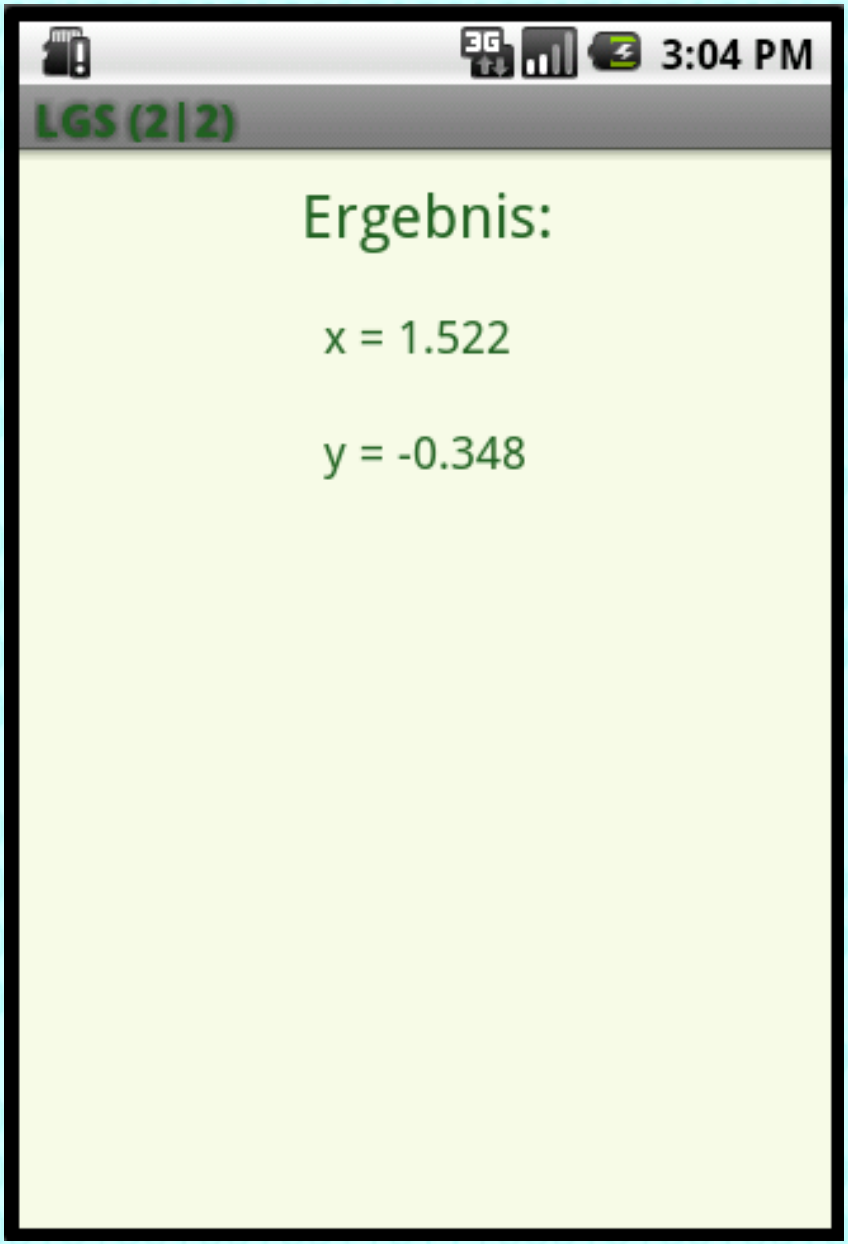
  interaktives Java-Tutorial

Literatur / Hilfen - englisch

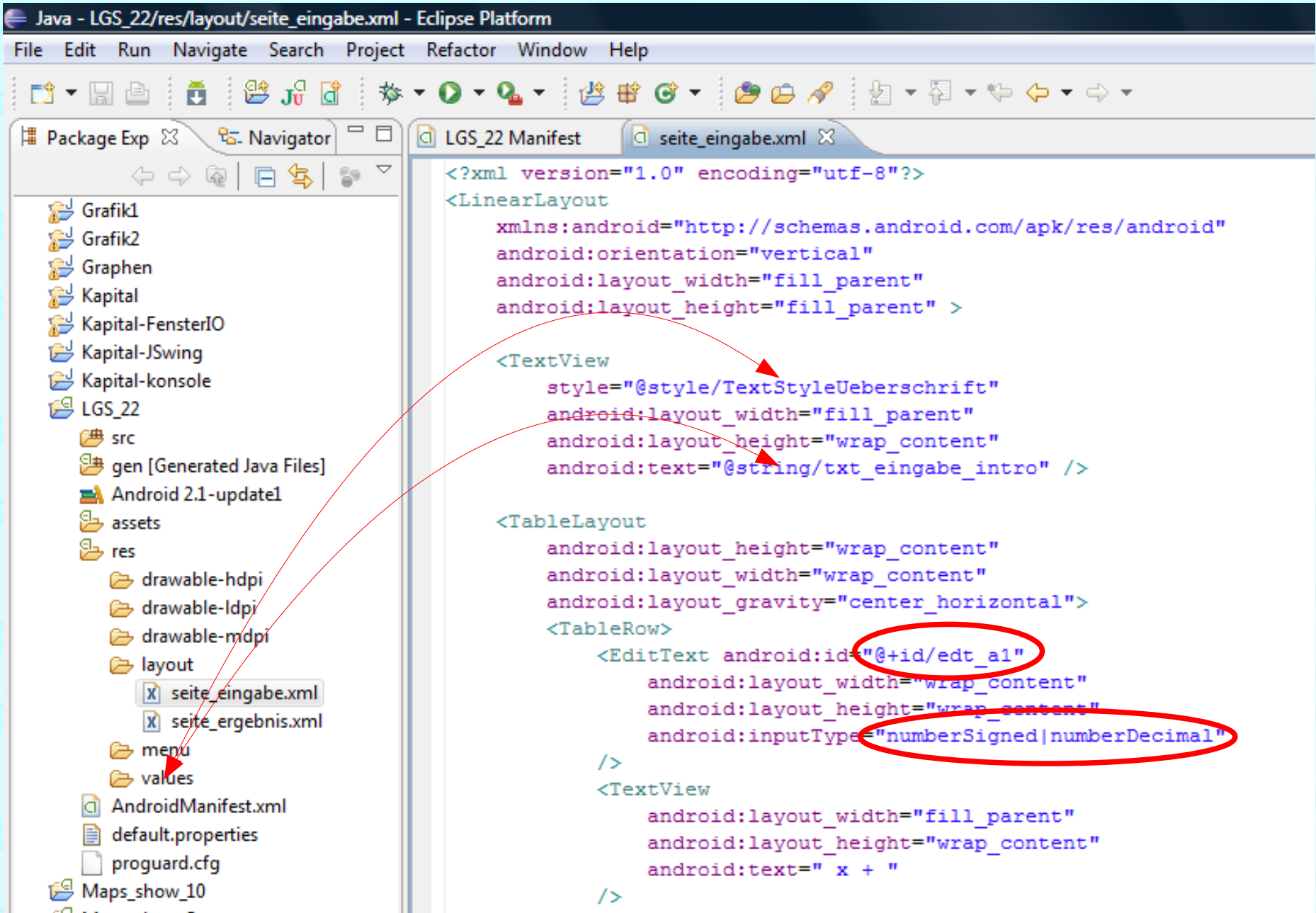
- http://developer.android.com/guide/index.html
  The Developer's Guide von Google
- http://developer.android.com/resources/browser.html?tag=article
  Technical Resources mit Code-Beispielen und längeren Artikeln
- http://developer.android.com/reference/packages.html
  Reference Guide für alle Packages und Klassen
- http://www.vogella.de/android.html
  Android Tutorials von Lars Vogel
- http://www.tutorialforandroid.com
  teilweise nützlich
- http://stackoverflow.com/
  recht umfangreiches Forum *auch* zu Android

App-Struktur an einem Beispiel

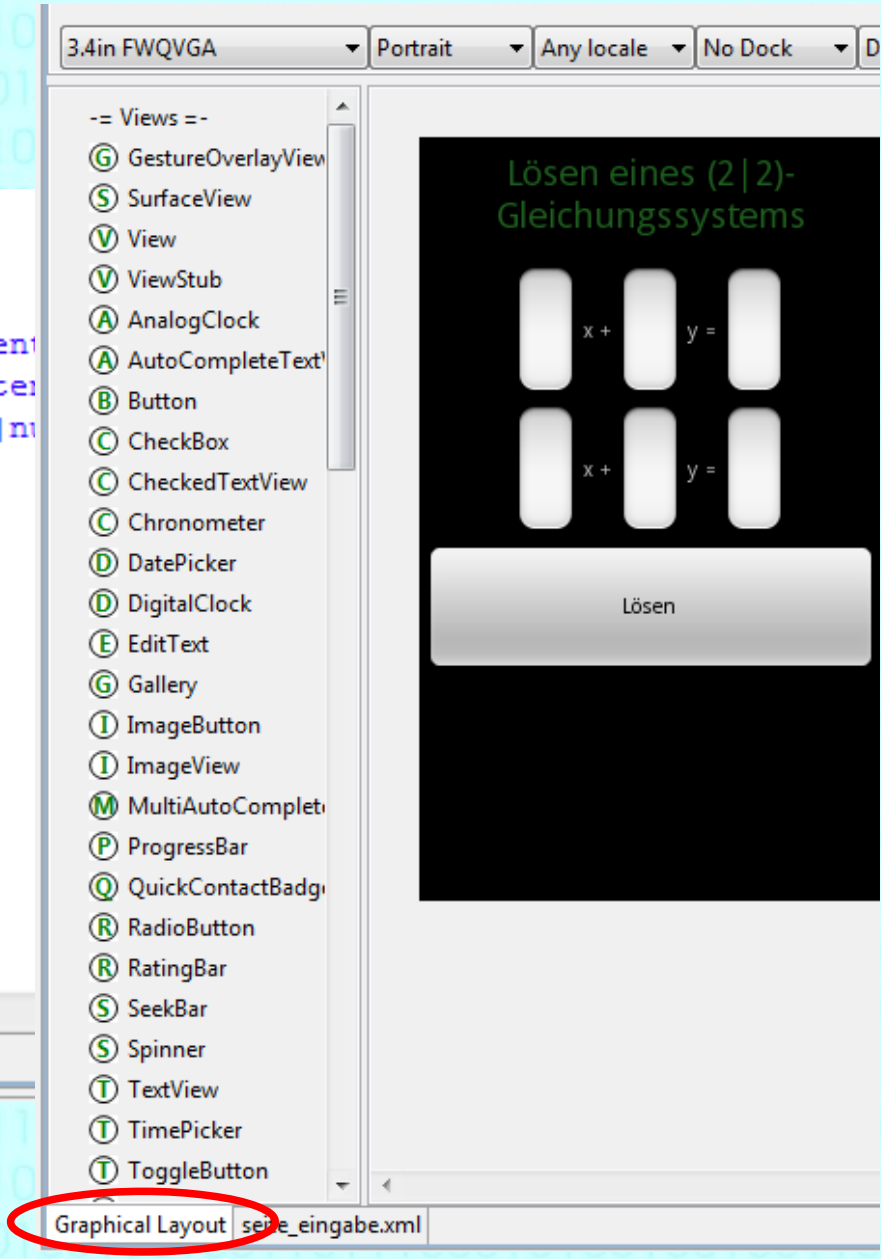




App-Struktur an einem Beispiel



App-Struktur an einem Beispiel



App-Struktur an einem Beispiel

LGS_22 Manifest | seite_eingabe.xml | seite_ergebnis.xml | styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MathTheme">
        <item name="android:windowBackground">@color/hintergrund</item>
        <item n
        <item n
    </style>

    <style name
        <item n
        <item n
    </style>

    <style name
           pare
        <item n
        <item n
        <item n
        <item n
    </style>

    <style name="TextStyleWar
           parent="@style/Tex
        <item name="android:t
        <item name="android:t
        <item name="android:g
    </style>

    <style name="Schaltflaech
           parent="android:Wi
        <item name="android:t
        <item name="android:t
    </style>

    <style name="Schaltflaeche"
           parent="android:Widget.Button">
        <item name="android:textAppearance">@style/SchaltflaechenText</item>
```

LGS_22 Manifest | seite_eingabe.xml | seite_ergebnis.xml | styles.xml | strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">LGS (2|2)</string>

    <string name="txt_eingabe_intro">Lösen eines (2|2)-Gleichungssystems</string
    <string name="txt_btn_loesen">Lösen</string>
    <string name="men_loeschen">Eingaben löschen</string>
    <string name="men_beenden">Prg beenden</string>
    <string name="men_zurueck">zurück</string>

    <string name="txt_ergebnis_intro">Ergebnis:</string>
</resources>
```

LGS_22 Manifest | colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="schwarz">#000000</color>
    <color name="hintergrund">#FFF1FBE6</color>
    <color name="textfarbe">#206020</color>
    <color name="textfarbe_warnung">#FF0000</color>
</resources>
```

App-Struktur an einem Beispiel

LGS_22 Manifest | colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="eh.beispiele.lgs_22"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name"
                 android:theme="@style/MathTheme">
        <activity android:name=".eingabe"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ergebnis" />

    </application>
    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

Styles

Start-Activity

weitere Activities

App-Struktur an einem Beispiel

eingabe.java

```
package eh.beispiele.lgs_22;

// mit Strg Shift "O" werden die benötigten Klassen automatisch eingebunden
import android.app.Activity;

public class eingabe extends Activity {

    /* Definition der Bezeichnungen für die Variablen,
     * die per Intent an die Ergebnis-Activity übergeben werden. */
    static final String A1 = "a1";
    static final String A2 = "a2";
    static final String B1 = "b1";
    static final String B2 = "b2";
    static final String D1 = "d1";
    static final String D2 = "d2";

    /* Wird direkt nach der Erzeugung der Activity aufgerufen. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* zeigt das bereits compilierte Layout der Seite an
         * die XML-Seite dazu heisst res/layout/seite_eingabe.xml */
        setContentView(R.layout.seite_eingabe);
    }
```

App-Struktur an einem Beispiel

```
/* Einlesen der Daten und Übergabe an die Ergebnis-Activity */
public void onClickLoesen(final View sfNormal) {
    /* Die Eingabefelder werden wieder über ihre id gefunden und der Inhalt
     * der EditText den Fließkommazahl-Variablen zugewiesen. */
    final EditText txt_a1 = (EditText) findViewById(R.id.edt_a1);
    final float a1 = secure(txt_a1.getText().toString());
    final EditText txt_a2 = (EditText) findViewById(R.id.edt_a2);
    final float a2 = secure(txt_a2.getText().toString());
    final EditText txt_b1 = (EditText) findViewById(R.id.edt_b1);
    final float b1 = secure(txt_b1.getText().toString());
    final EditText txt_b2 = (EditText) findViewById(R.id.edt_b2);
    final float b2 = secure(txt_b2.getText().toString());
    final EditText txt_d1 = (EditText) findViewById(R.id.edt_d1);
    final float d1 = secure(txt_d1.getText().toString());
    final EditText txt_d2 = (EditText) findViewById(R.id.edt_d2);
    final float d2 = secure(txt_d2.getText().toString());

    /* Erzeugen des Intents für die Datenübergabe */
    final Intent i = new Intent(this, ergebnis.class);

    /* die sechs Parameter werden an den Intent übergeben */
    i.putExtra(A1, a1);
    i.putExtra(A2, a2);
    i.putExtra(B1, b1);
    i.putExtra(B2, b2);
    i.putExtra(D1, d1);
    i.putExtra(D2, d2);

    /* starten der Ergebnis-Activity */
    startActivity(i);
}
```

App-Struktur an einem Beispiel

ergebnis.java

```
package eh.beispiele.lgs_22;

//mit Strg Shift "O" werden die benötigten Klassen automatisch eingebunden
import android.app.Activity;

public class ergebnis extends Activity{

    /* Definition der Bezeichnungen für die Variablen,
     * die per Intent von der Eingabe-Activity übergeben werden.
     * Muss identisch sein!*/
    static final String A1 = "a1";
    static final String A2 = "a2";
    static final String B1 = "b1";
    static final String B2 = "b2";
    static final String D1 = "d1";
    static final String D2 = "d2";
```

App-Struktur an einem Beispiel

```
/* Wird direkt nach der Erzeugung der Activity aufgerufen. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /* zeigt das bereits compilierte Layout der Seite an
     * die XML-Seite dazu heisst res/layout/seite_ergebnis.xml */
    setContentView(R.layout.seite_ergebnis);

    /* Der komplette Inhalt des Intents wird eingelesen */
    final Bundle extras = getIntent().getExtras();
    if (extras != null) {
        /* Instanz der Klasse berechnung (extra Datei) */
        final berechnung erg = new berechnung();

        /* Zuweisen der sechs Parameter */
        erg.a1 = extras.getFloat(A1);
        erg.a2 = extras.getFloat(A2);
        erg.b1 = extras.getFloat(B1);
        erg.b2 = extras.getFloat(B2);
        erg.d1 = extras.getFloat(D1);
        erg.d2 = extras.getFloat(D2);
        /* Ergebnis berechnne und anzeigen (s.u.) */
        zeigeErgebnis(erg);
    }
}

 /* Erzeugt das OptionsMenü
  * die XML-Seite dazu heisst res/menu/ergebnis_menu.xml */
 public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.ergebnis_menu, menu);
    return super.onCreateOptionsMenu(menu);
```

App-Struktur an einem Beispiel

```
/* Ergebnis berechnen und Anzeigen */
private void zeigeErgebnis(berechnung erg) {
    /* Titel der Activity festlegen */
    setTitle("LGS (2|2)");

    /* Ergebnis wird berechnet, in msg wird ggf. Fehlermeldung geliefert */
    String msg = erg.berechne();

    if (msg == "") {
        /* falls kein Fehler gemeldet wird, wird TextView per id gesucht und
         * das Ergebnis formatiert geschrieben. */
        final TextView txt_ergx = (TextView) findViewById(R.id.txt_x);
        txt_ergx.setText(String.format("%,.3f%n", erg.x));
        final TextView txt_ergy = (TextView) findViewById(R.id.txt_y);
        txt_ergy.setText(String.format("%,.3f%n", erg.y));
    }
    else {
        /* falls Fehler, Ausgabe der Fehlermeldung */
        final TextView txt_ergx = (TextView) findViewById(R.id.txt_info);
        txt_ergx.setText(msg);
    }
}
```

Debuggen mit der LogCat

```
Log.d("log_tag", "Meldung: "+line);
if (line.compareTo("nix") == 0) {
    out = "keine Räume belegt";
}
else {
    out = "Datum      Raum     Std.\n";
```

Problems | Search | Console | File Explorer | Progress | LogCat

Log | DebugMeldungen | log_tag

Time		pid	tag	Message
01-18 11:53...	W	303	KeyCharacterMap	Using default keymap: /system/usr/keychars/qwerty.kcm.bin
01-18 11:53...	I	38	ActivityManager	Starting activity: Intent { cmp=eh.beispiele.db/.alle (has extras) }
01-18 11:53...	I	38	ActivityManager	Displayed activity eh.beispiele.db/.alle: 557 ms (total 557 ms)
01-18 11:54...	D	38	SntpClient	request time failed: java.net.SocketException: Address family not sup...

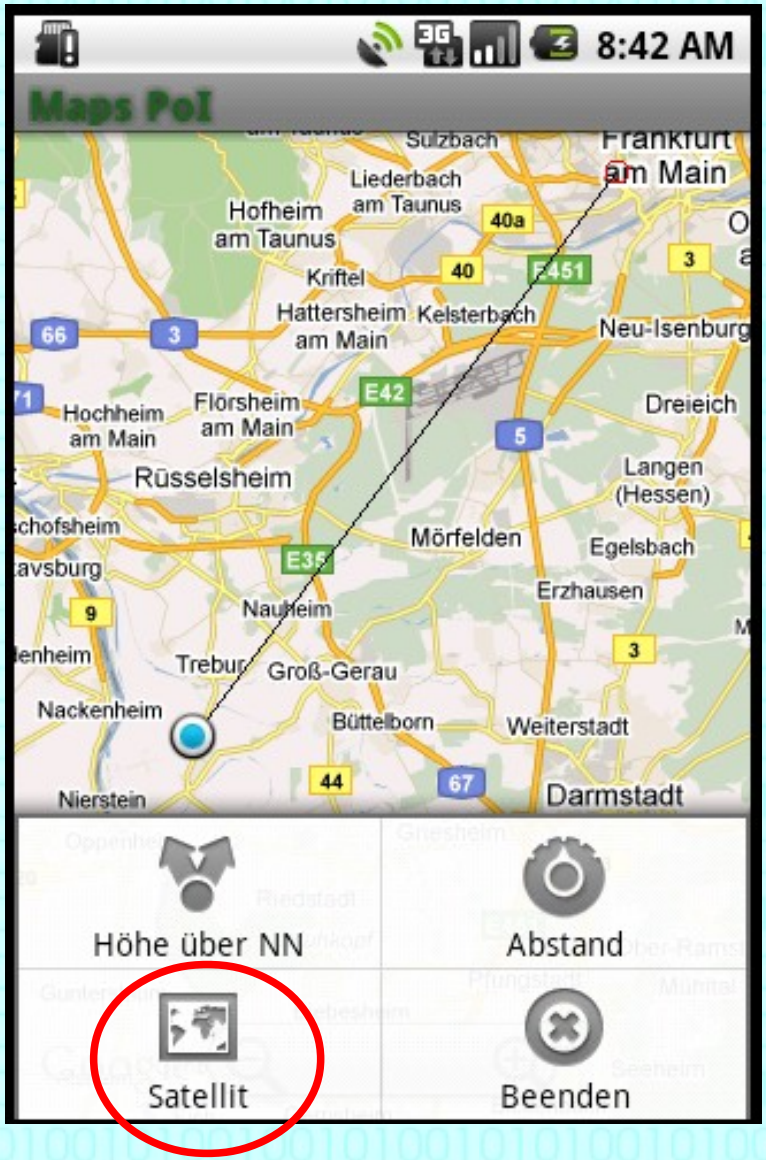
Filter:

Problems | Search | Console | File Explorer | Progress | LogCat

Log (4) | DebugMeldungen | log_tag

Time		pid	tag	Message
01-18 11:53...	D	303	log_tag	Meldung: nix

Weitere Möglichkeiten: Google Maps



Weitere Möglichkeiten: html und Internet

Weitere Möglichkeiten: Datenbanken

- Android verfügt auch über eine eigene einfache Datenbank (SQLite)
- Android (java.sql.*) kann auf externe DB direkt zugreifen
- Besser und sicherer ist der Zugriff über ein Webinterface (php-Scripte auf einem Webserver)
  - Keine Zugriffsdaten zur DB außerhalb des Webservers
  - Bei vielen Änderungen/Optimierungen an der DB muss keine Änderung der App erfolgen, sondern nur die php-Scripte angepasst werden.
- In der Regel sind XAMPP-Kenntnisse vorhanden, so dass auf diesem Wege die Neuerungen minimiert werden.

Weitere Möglichkeiten: Datenbanken

```
ArrayList<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
try{
    nameValuePairs.add(new BasicNameValuePair("pid",pid));

    HttpClient httpclient = new DefaultHttpClient();
    HttpPost httppost = new HttpPost("http://www.beispiel.webwserver.de/demo/app.php");
    httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
    HttpResponse response = httpclient.execute(httppost);
    HttpEntity entity = response.getEntity();
    InputStream ins = entity.getContent();
    BufferedReader reader = new BufferedReader(new InputStreamReader(ins,"iso-8859-1"),8);
    String line = null;
    line = reader.readLine();  // nix oder Tripel

    if (line.compareTo("nix") == 0) {
        out = "keine Räume belegt";
    }
    else {
        out = "Datum       Raum      Std.\n";
        out += getdatum(line) + ":   ";
        line = reader.readLine();
        // ... usw. ...
    }

    ins.close();
}
catch(Exception e){
    Log.e("log_tag", "Fehler: "+e.toString());
    out = e.toString();
}
```

Anfrage

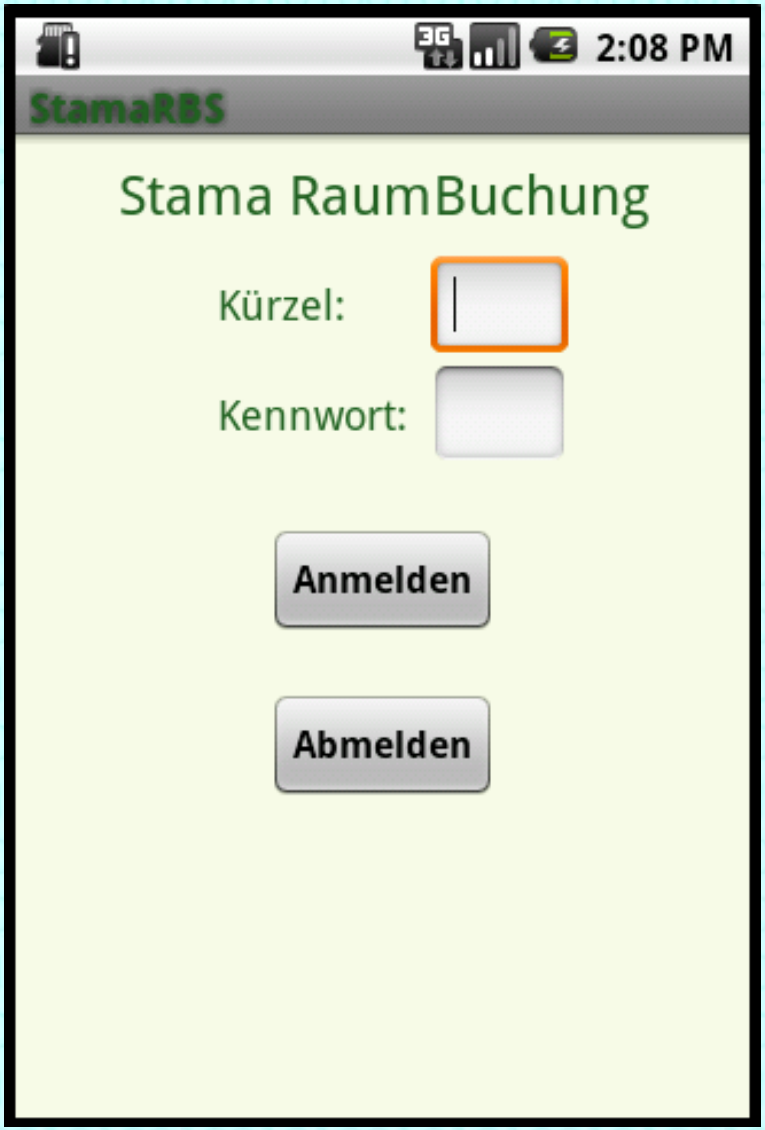
Antwort

Adresse des php-Scriptes

Übergabeparameter

INFORMATIK

Weitere Möglichkeiten: Datenbanken



Weitere Möglichkeiten

- SMS versenden und empfangen
- Beschleunigungssensor abfragen
- Telefonbuch und Kalender verwenden
- Fotografieren, Bilder bearbeiten und versenden
- Audio aufnehmen und abspielen
- …

Apps vertriebsfertig machen

- Im Debug-Modus kann die gerade entwickelte App auf ein angeschlossenes Smartphone übertragen werden.
- Bevor eine App weitergegeben oder im Market vertrieben werden kann,
  - muss ein Zertifikat erstellt werden,
  - muss die App signiert werden und
  - sollte der Code optimiert werden
- Für die Benutzung von Google-Maps benötigt man einen speziellen API-Key
- Wird im „Vertriebsmodus“ gearbeitet, kann nicht mehr direkt mit dem angeschlossenen Smartphone gearbeitet werden.

Einsatz im Unterricht

- nur in der MSS
- Android ist keine Einstiegssprache
- XML Grundkenntnisse und
- solide Programmierkenntnisse notwendig, …
- … aber nicht unbedingt Java, gute Delphi-Kenntnisse sind völlig ausreichend
- Einsatz z.B. als weitere Programmiersprache
- oder in Projekten
- Weiterer Vorteil: sinnvolle Anwendung von XML

technische Voraussetzungen

- Einsatz nicht möglich unter MNS+ !!!
  (Paketierung nicht möglich)
- Für die Simulierung von Sensordaten (GPS, Bewegung) sind Adminrechte notwendig (command-Shell)
- Voraussetzung sind also genügend Schüler-Laptops (Quote 50%)

MSS-12 Projekttage: Zeitplan

31.März 2011 5./6. Std.	• Warum? / Warum *Android*? • Struktur / Grundlagen • Installation / Literatur / Hilfen • Ein einfaches Beispiel
15.April 3./4. Std.	• Ein komplexeres Beispiel (Listen, WebView, GPS, *GoogleMaps*, Rückmeldungen von Activities, Akku, …) • Zertifizieren von Apps
Vorbereitungszeit	• Installation von *Eclipse* / *Android SDK* / PlugIns. • Einarbeiten in die Entwicklungsumgebung. • Gruppenbildung (ca. 3 Schüler) / Ideensammlung
3. Mai früh	Gruppen stellen ihre Idee vor
3.-5. Mai	Entwicklung der Apps
5. Mai nachmittags	Vorstellen der Apps und des Codes.

Ergebnisse: http://stamaonline.de/index.php?id=205

Apps

Programmierung von Android-Smartphones

Fragen?

PL-Fortbildung in Speyer: 22.-24.10.2012
Apps- Programmierung unter Android
Veranstaltungs-Nr.: **211330901**