Java - Programmierung für Android

Bücher

- Uwe Post: Android-Apps entwickeln für Einsteiger.Eigene Apps und Spiele mit Android Studio
- Florian Franke /Johannes Ippen: Apps mit HTML5, CSS3 und JavaScript: Für iPhone, iPad und Android
- Thomas Künneth: Android 5: Apps entwickeln mit Android Studio
- John Horton: Android Programming for Beginners
- John Horton: Learning Java by Building Android Games

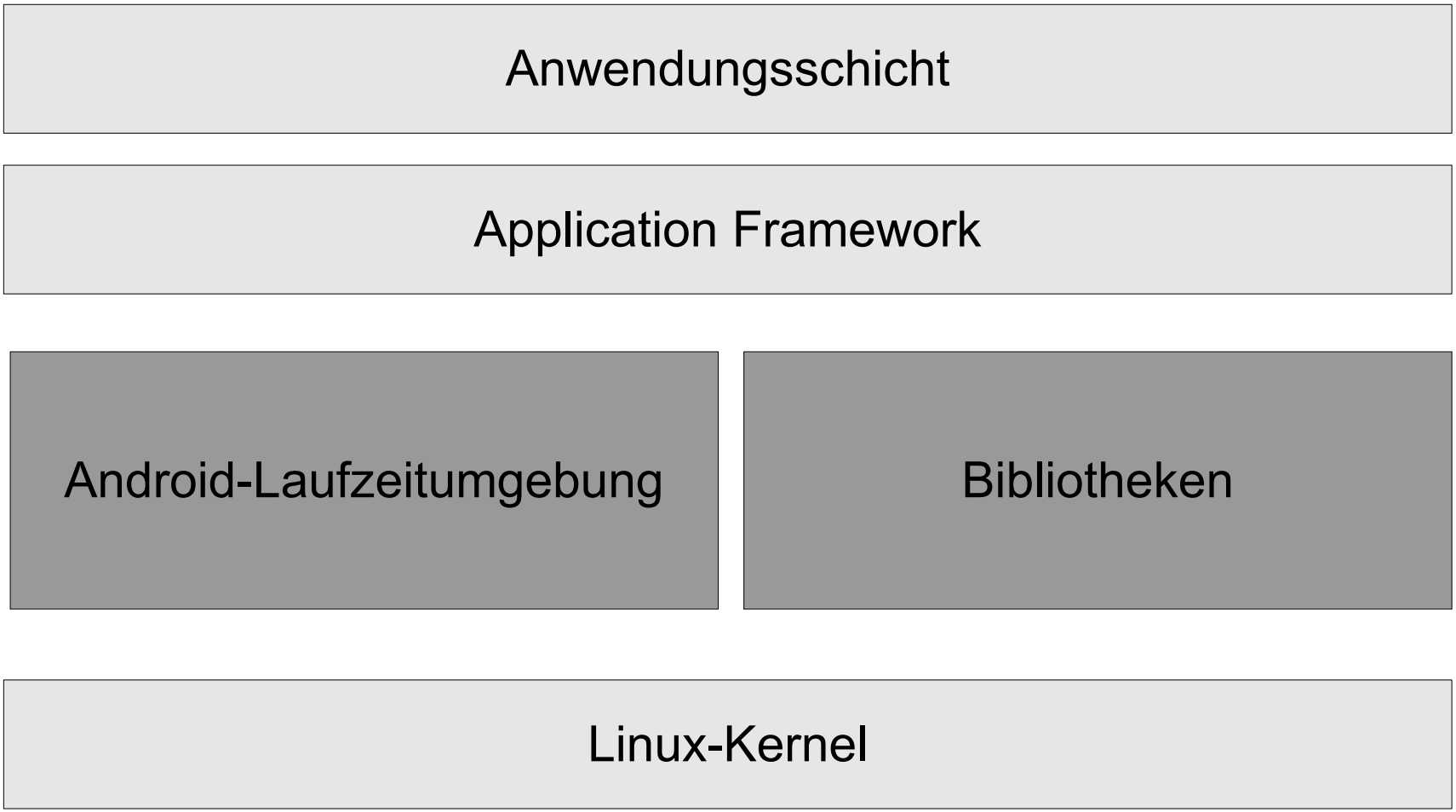
Tutorials im Web

- http://developer.android.com/training/basics/firstapp/index.html
- http://www.vogella.com/tutorials/Android/article.html
- http://www.tippscom.de/android-app-programmieren-lernen/
- https://www.androidpit.de/de/android/wiki/view/Android_Anf%C3%A4nger_Workshop
- http://www.programmierenlernenhq.de/android-app-programmieren-tutorial/
- http://individuapp.com/de/android-kurs

Android

- Betriebssystem für mobile Geräte basierend auf einer besonderen Variante von Linux.
- Software-Plattform für mobile Geräte.

Architektur

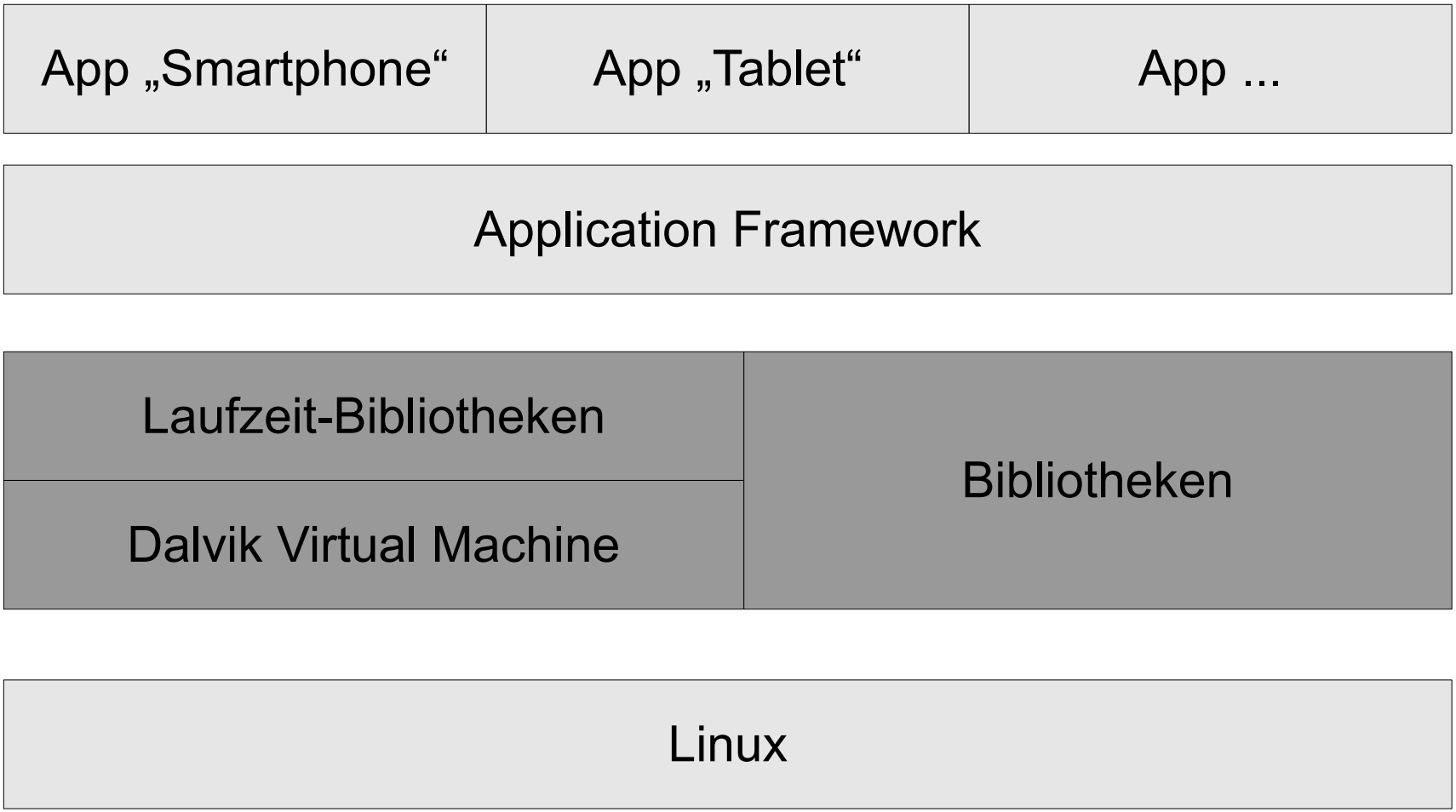


Linux-Kernel

- Embedded Linux.
- Power Management. Wenn keine Anwendung Strom verbraucht, verbraucht auch die CPU kein Strom.
- Binder. Jede Android-Anwendung ist ein Prozess. Die Prozesse tauschen per Interprozesskommunikation nur Referenzen auf Objekte aus.
- Zugriffsberechtigung.
- Virtuelle Speicherverwaltung.

Leibniz Universität Hannover

Architektur



Dalvik Virtual Machine

- Interpreter von Code.
- Jede Android-Anwendung läuft in einem eigenen Prozess und auf einer eigenen virtuellen Maschine.

Bibliotheken

- Oberflächen-Manager. Jede Aufgabe wird in einem Frame / auf einer Bildschirmseite dargestellt.
- Grafik-Bibliotheken.
- Datenbank-Bibliotheken (SQLite).
- Standard-C-Bibliothek Bionic.

Application Framework

- Felder auf der Benutzeroberfläche (Views).
- Package Manager. Informationen über die installierte Anwendung.
- Ressource Manager. Verwaltung von Ressourcen wie Bilder etc.
- Activity Manager. Lebenszyklus der App.

Anwendungsschicht

- Android-Anwendungen für mobile Geräte (Tablet, Smartphone, …).
- Drittanbieter-Anwendungen.
- Eigene Anwendungen.

Komponenten einer App

- Activity.
- Service.
- ContentProvider.
- BroadcastReceiver.
- Intents.

Content Provider

- Management von „shared data“.
- Zugriff einer App auf die Daten, die zum Beispiel in Dateien oder Datenbanken gespeichert sind.

Service

- Im Hintergrund laufende Aktivitäten. Zum Beispiel Abspielen von Musik.
- Funktionalitäten für andere Apps.
- Ein Service benötigt keine Benutzeroberfläche.

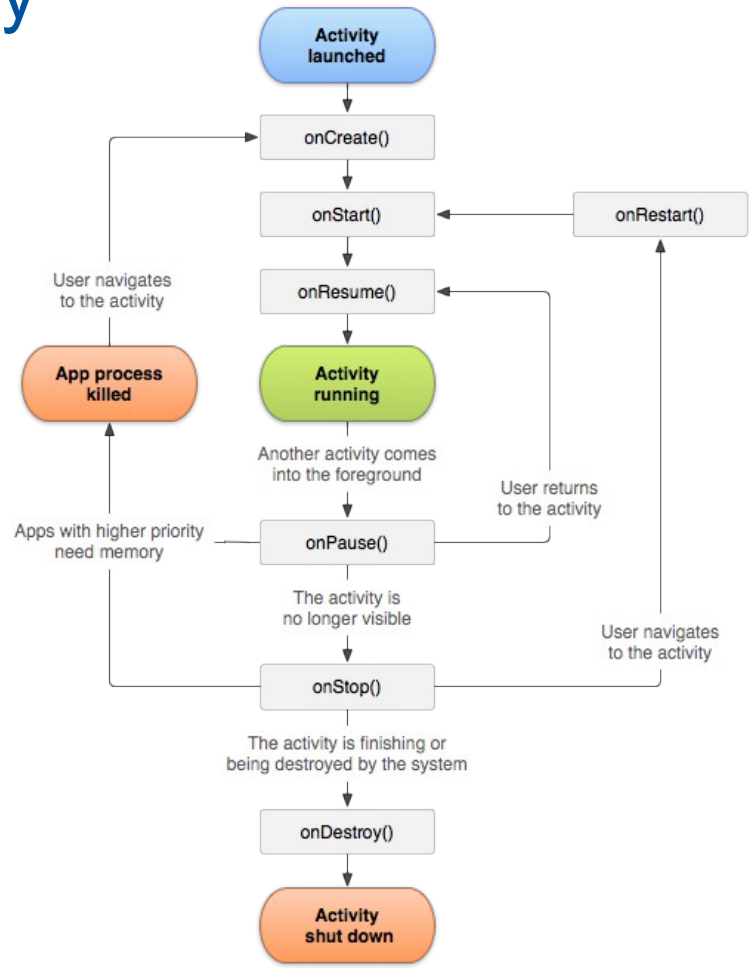
BroadcastReceiver

- Nachrichten wie zum Beispiel: „Das Gerät muss aufgeladen werden“.
- Verarbeitung von Nachrichten im Hintergrund.
- Definition im Android-Manifest.

Activity

- Eine Bildschirmseite (View) plus Code.
- Interaktion mit dem Benutzer über eine Benutzeroberfläche.
- Beschreibung einer Aufgabe.
- Hinweis: Jede Activity kann von jeder App auf einem Android-Plattform aufgerufen werden.

Lebenszyklus einer Activity



http://developer.android.com/reference/android/app/Activity.html

Leibniz Universität Hannover

„Meldepflicht“ in der Datei „Android Manifest“

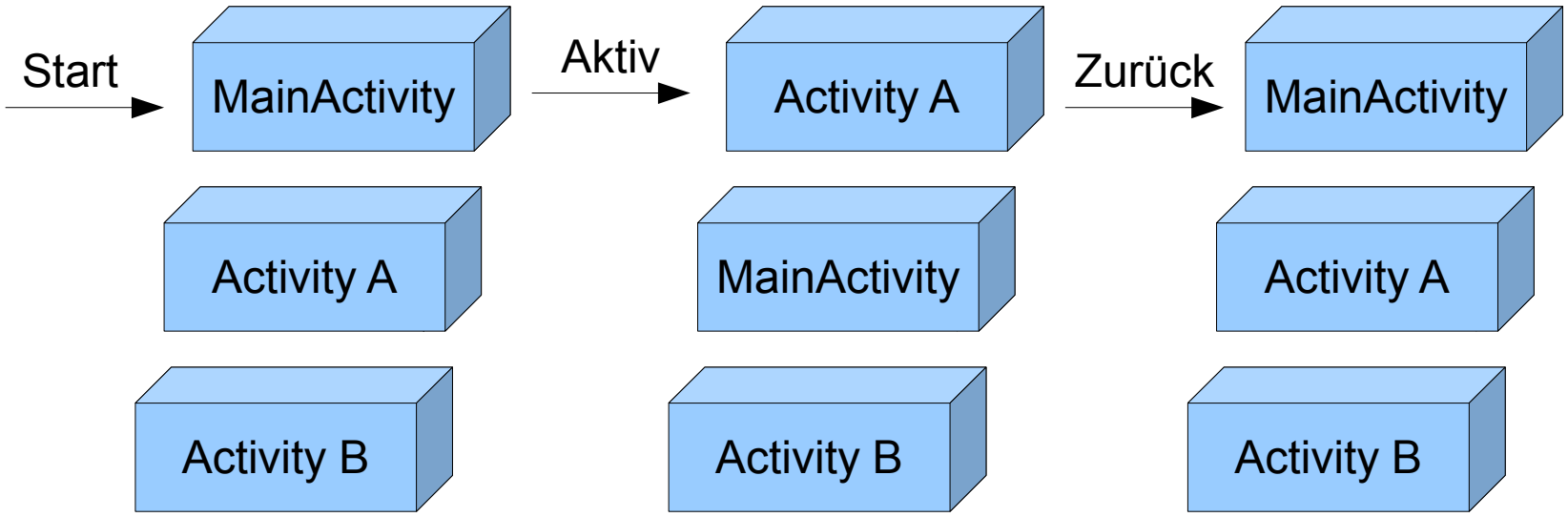
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    <application android:label="@string/app_name" >
        <activity android:name="MainActivity"
                  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="ActivityTemperatur"
                  android:label="@string/app_temperatur">

        </activity>
```

Mehrere Bildschirmseiten in einer App



Intents

- Aufruf von Activities innerhalb einer App.
- Aktivierung von Service oder BroadcastReceiver.
- Beschreibung, welche Activity aktivert werden soll und falls nötig, welche Daten dieser Activity übergeben werden.
- Versenden mit der passenden Methode.

Explizite Intents

- Nachrichten an eine bestimmte Klasse.
- Der Name der zu startenden Activity wird als Versandadresse angegeben.
- Der Sender und Empfänger befinden sich in einer App.

Implizite Intents

- Das Android-System bestimmt den Empfänger aufgrund der Beschreibung des Empfängers.
- Acitivities werden mit Hilfe von Intent-Filtern im Android Manifest.

Beispiel

Intent

Activity A

Activity B

Benötigte Software

- Java Runtime Environment (JRE)

  Download: https://www.java.com/de/download/

  Die JRE sollte von der gleichen Bit-Version wie die Entwicklungsumgebung sein.
- Android SDK.

  Download: http://developer.android.com/sdk/index.html#download.

Java Runtime Environment (JRE)

- Ausführung von Java-Programmen.
- Java API. Standard-Klassenbibliotheken.
- Java Virtual Machine (JVM). Interpreter in der Laufzeitumgebung für den Java-Bytecode.
- Java-Compiler Hotspot. Zur Laufzeit wird der Bytecode in Maschinencode umgesetzt,
- Speichermanagement des Java-Programms. Speicherfreigabe durch den Garbage Collector.

Android SDK

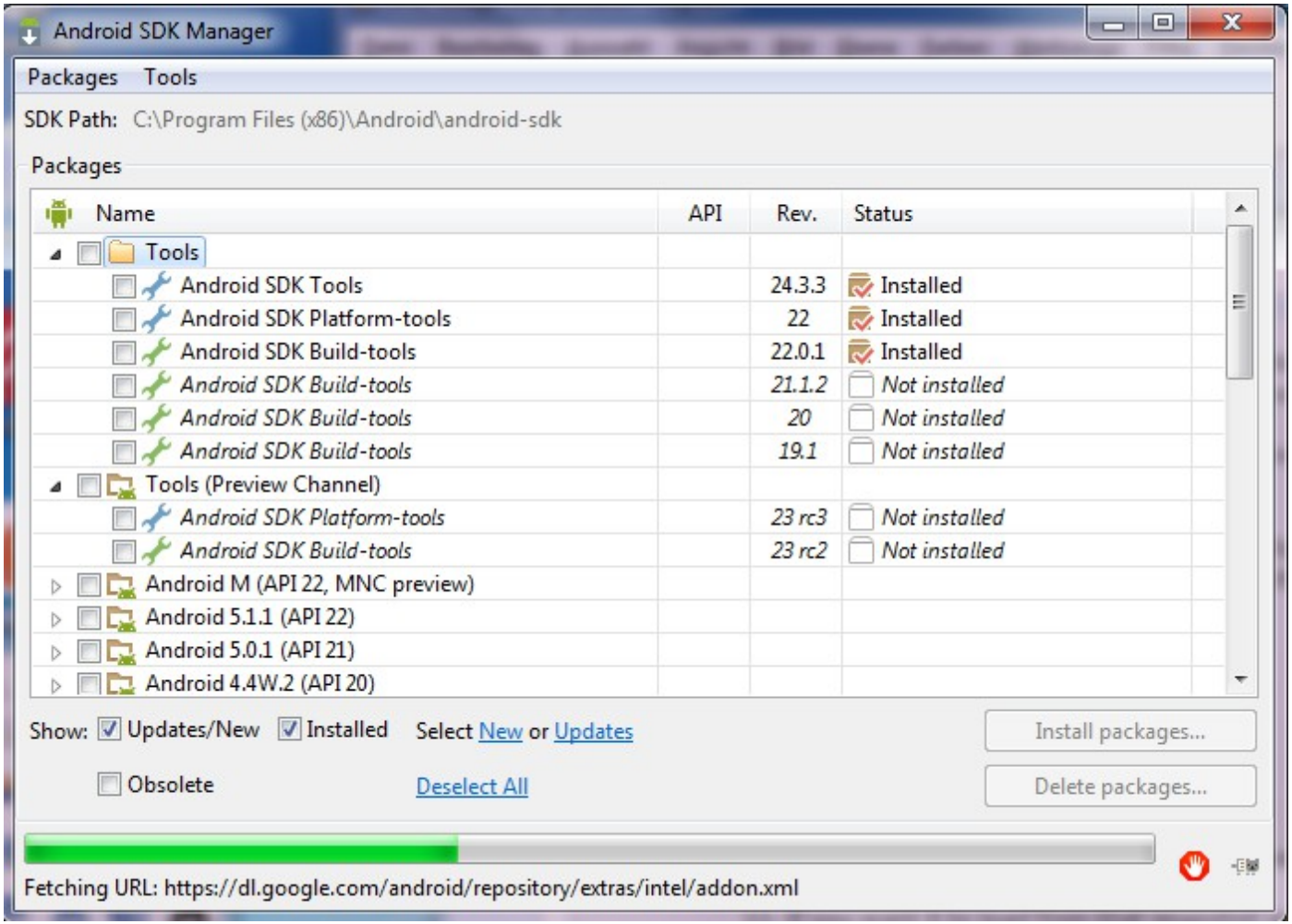
- Das Android Software Development Kit basiert auf Java.
- Zu jeder Plattform wird eine passende API (Application Programming Interface) mitgeliefert. Jede API ist abwärtskompatibel.
- SDK Manager: Installation der verschiedenen Pakete.
- AVD Manager. Emulation der verschiedenen Geräte.
- Aktuelle Version und deren Verbreitung: http://developer.android.com/about/dashboards/index.html

Beispiele für die Programmierung

- Ordner *Android – android-sdk – samples.*
- http://androidexample.com/
- http://www.java2s.com/Code/Android/CatalogAndroid.htm

Leibniz Universität Hannover

SDK - Manager (Software Development Kit)



Erläuterung

- Auswahl einer Android-Version und der entsprechenden API.
- Die installierten Versionen werden in den Ordner *Android* – *android-sdk* – *platforms* in Unterordnern abgelegt.
- Die Datei android.jar spiegelt die entsprechende Bibliothek wieder.

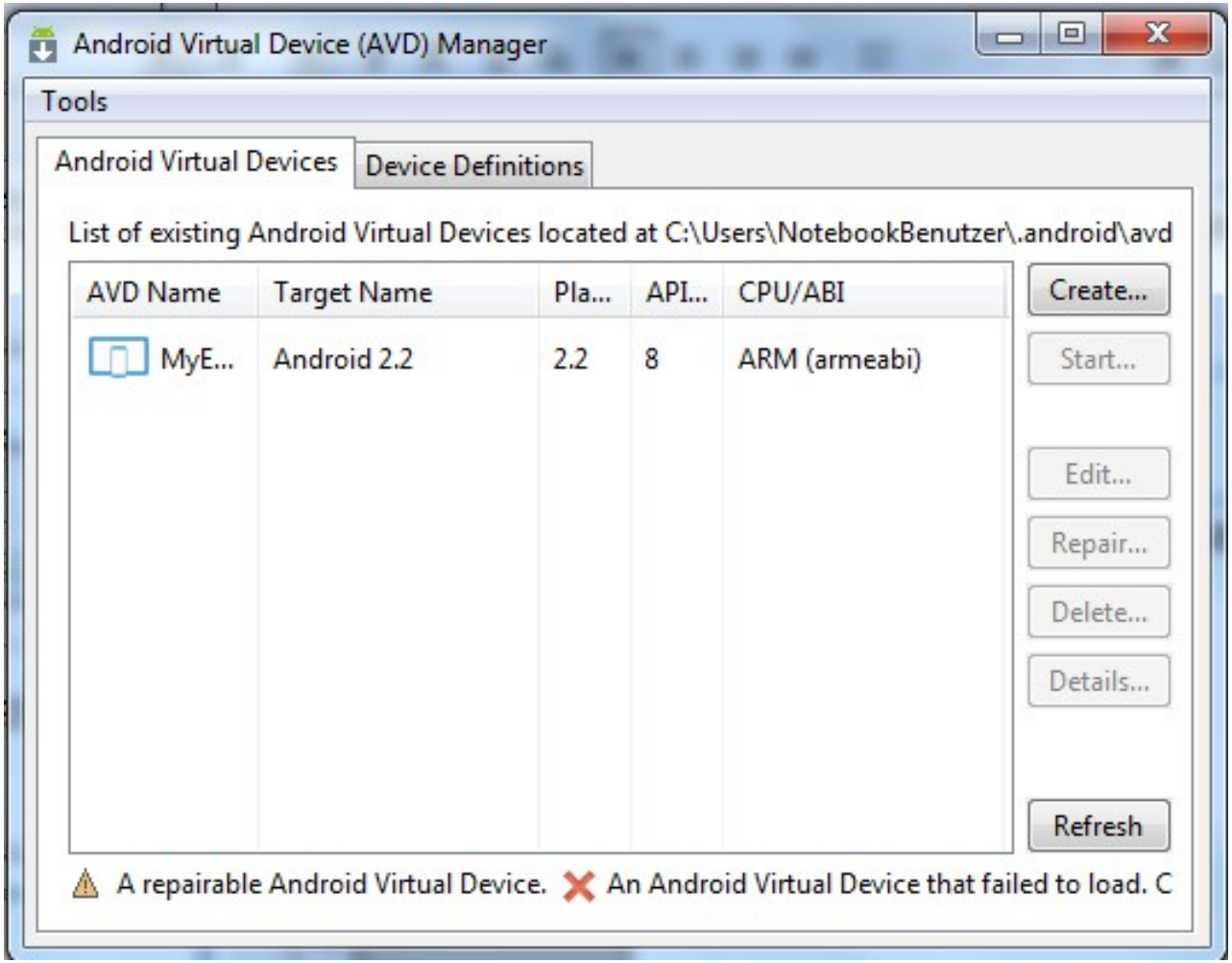
Android - Version

- Die Versionsnummer kennzeichnet die Plattform.
- Zum Beispiel die Version 4.2.2, erschienen am 11. Februar 2013. Die erste Nummer (major release) kennzeichnet größere Erweiterungen etc. Die zweite Nummer (minor release) kennzeichnet Updates. Die dritte Nummer (patch level) wird genutzt, um Fehlerbehebungen zu kennzeichnen.
- Seit der Version 1.5 wird zusätzlich eine amerikanische Süßigkeit als Codenamen genutzt.

API-Nummer

- Die Schnittstelle (Application Programming Interface) wird mit einer ganzzahligen Nummer gekennzeichnet.
- Versionierung der Bibliothek.

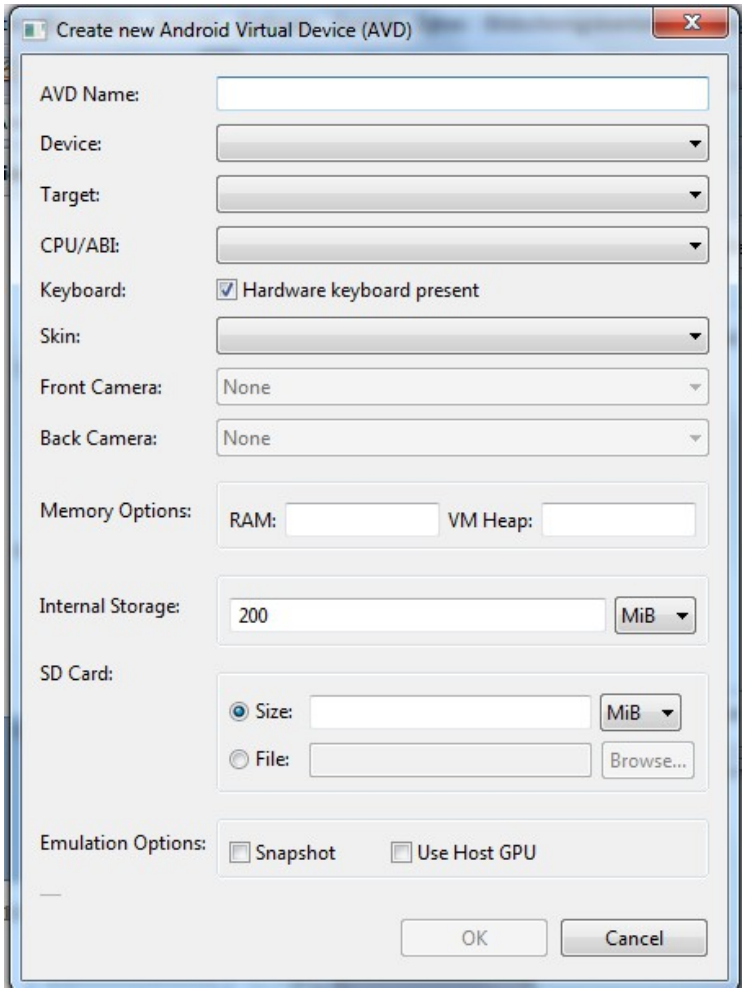
AVD Manager


Android Virtual Device (AVD) Manager
Tools
Android Virtual Devices
Device Definitions
List of existing Android Virtual Devices located at C:\Users\NotebookBenutzer\.android\avd
AVD Name
Target Name
Pla...
API...
CPU/ABI
MyE...
Android 2.2
2.2
8
ARM (armeabi)
Create...
Start...
Edit...
Repair...
Delete...
Details...
Refresh
A repairable Android Virtual Device.
An Android Virtual Device that failed to load. C

Erläuterung

- Anzeige von zu emulierenden Android-Geräten.
- Virtuelle Android-Geräte definieren (*Create*).
- Vorhandene virtuelle Geräte bearbeiten (*Edit*).
- Virtuelles Gerät starten (*Start*).

Neues virtuelles Gerät einrichten



Erläuterung

- *AVD Name*. Frei wählbare Bezeichnung für die Emulation.
- *Device*. Welches Gerät soll emuliert werden? Hier: 4“ WVGA. 480 x 800.
- *Target*. Welche Version soll emuliert werden? Hier: Android 2.2.
- *Skins*. Oberflächen für den Emulators. Hier: HVGA.
- *SD Card*. Größe der SD Card. Hier: 200 MiB.
- *Snapshot* beschleunigt das nochmalige Laden der Emulation.

Auswahl der Versionen

- *Android 2.2 (API 8)*. Minimum für ein heutiges Android-Gerät.
- *Extras - Google USB Driver* wird benötigt, um die entwickelte App direkt auf dem Smartphone oder Tablet zu testen.

Entwicklung mit Android Studio

- Download: https://developer.android.com/sdk/index.html
- Seit 2013 empfohlene Standard-IDE.
- Basiert auf IntelliJ IDEA.
- Seit der Version 0.4.0 wird ein „Gradle Offline mode“ unterstützt.

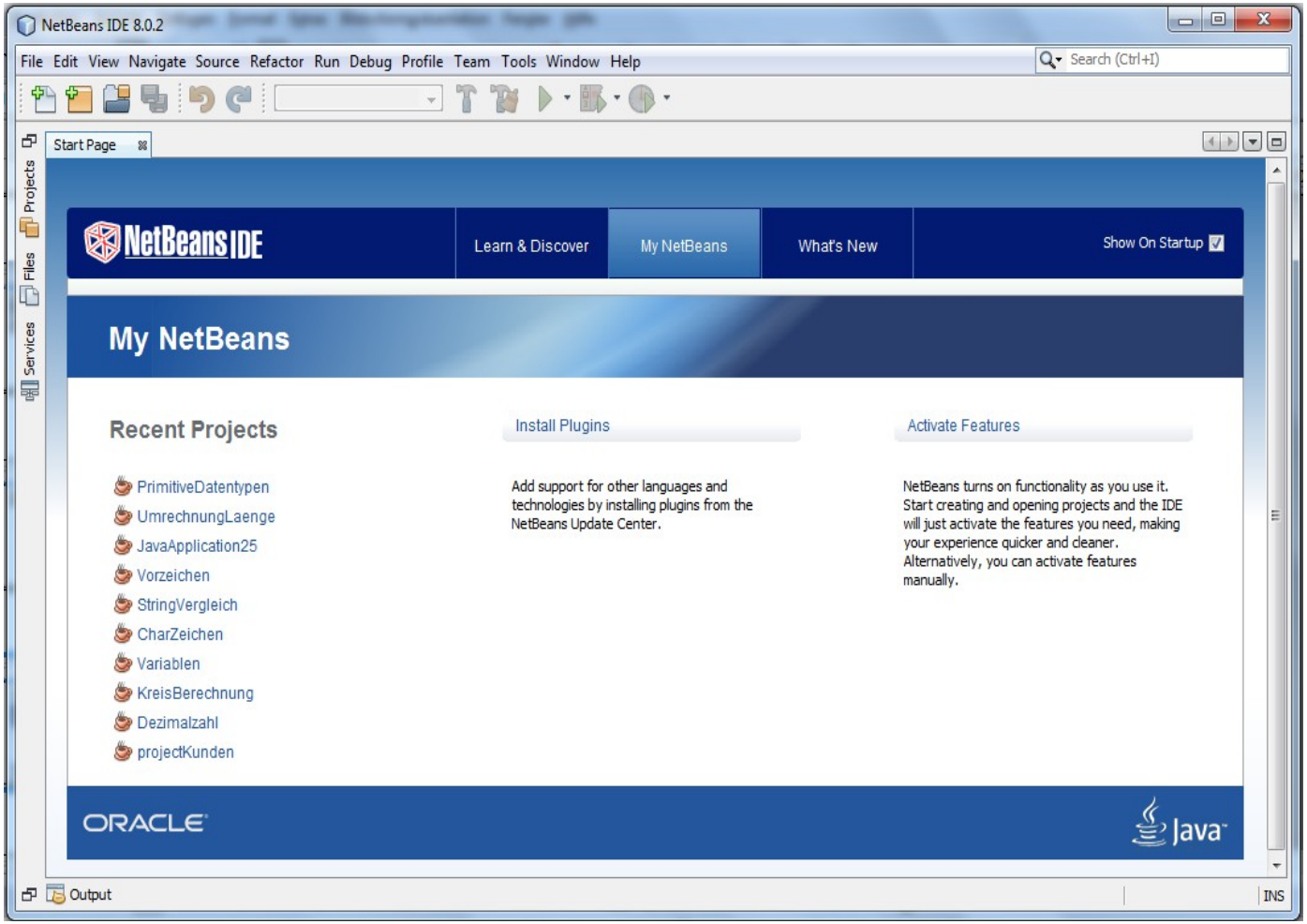
Entwicklung mit NetBeans

- NetBeans. Download unter https://netbeans.org/.
- Dokumentation der IDE: https://netbeans.org/kb/index.html
- PlugIn für NetBeans. Download unter http://www.nbandroid.org/.

Start von NetBeans

- Klick auf das Icon auf dem Desktop.
- *Start – Alle Programme – NetBeans – NetBeans IDE.*

Startseite


NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Start Page
Projects
Files
Services
NetBeans IDE
Learn & Discover
My NetBeans
What's New
Show On Startup
My NetBeans
Recent Projects
PrimitiveDatentypen
UmrechnungLaenge
JavaApplication25
Vorzeichen
StringVergleich
CharZeichen
Variablen
KreisBerechnung
Dezimalzahl
projectKunden
Install Plugins
Add support for other languages and technologies by installing plugins from the NetBeans Update Center.
Activate Features
NetBeans turns on functionality as you use it. Start creating and opening projects and the IDE will just activate the features you need, making your experience quicker and cleaner. Alternatively, you can activate features manually.
ORACLE
Java
Output
INS

Neuer Installationspfad für Plugins

- *Tools – Plugins*.
- Registerkarte *Settings*.
- *Add*. Der Name ist frei wählbar. Als URL wird http://nbandroid.org/updates/updates.xml für Android eingetragen.

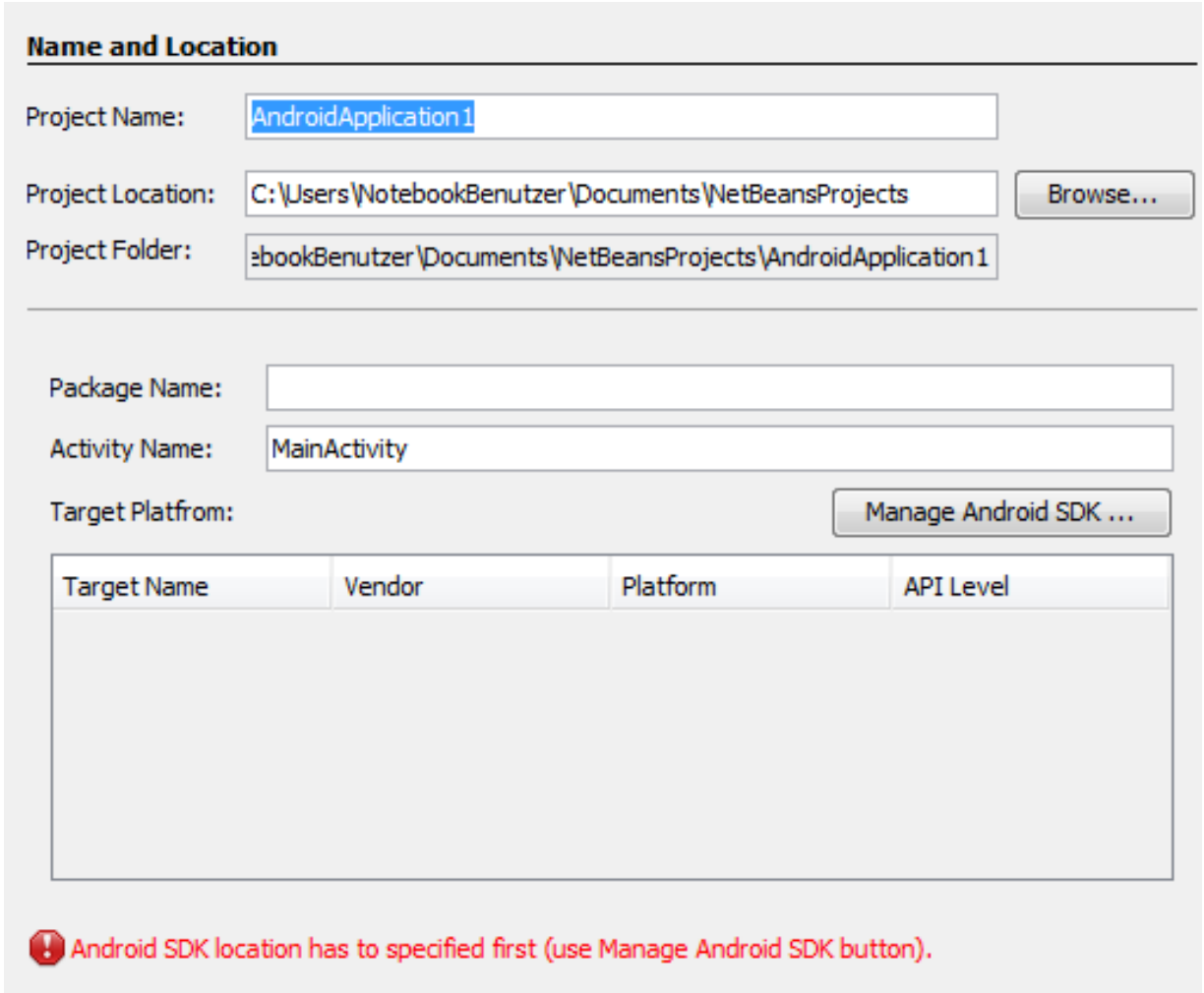
Plugin installieren

- *Tools – Plugins*.
- Registerkarte *Available Plugins*.
- Auswahl von *Android* durch einen Klick in das Kästchen links vom Namen.
- Klick auf *Install*. Falls die Version nicht mit der Version von NetBeans übereinstimmt, wird ein Fehler angezeigt.

Angabe des Speicherortes „Android SDK“

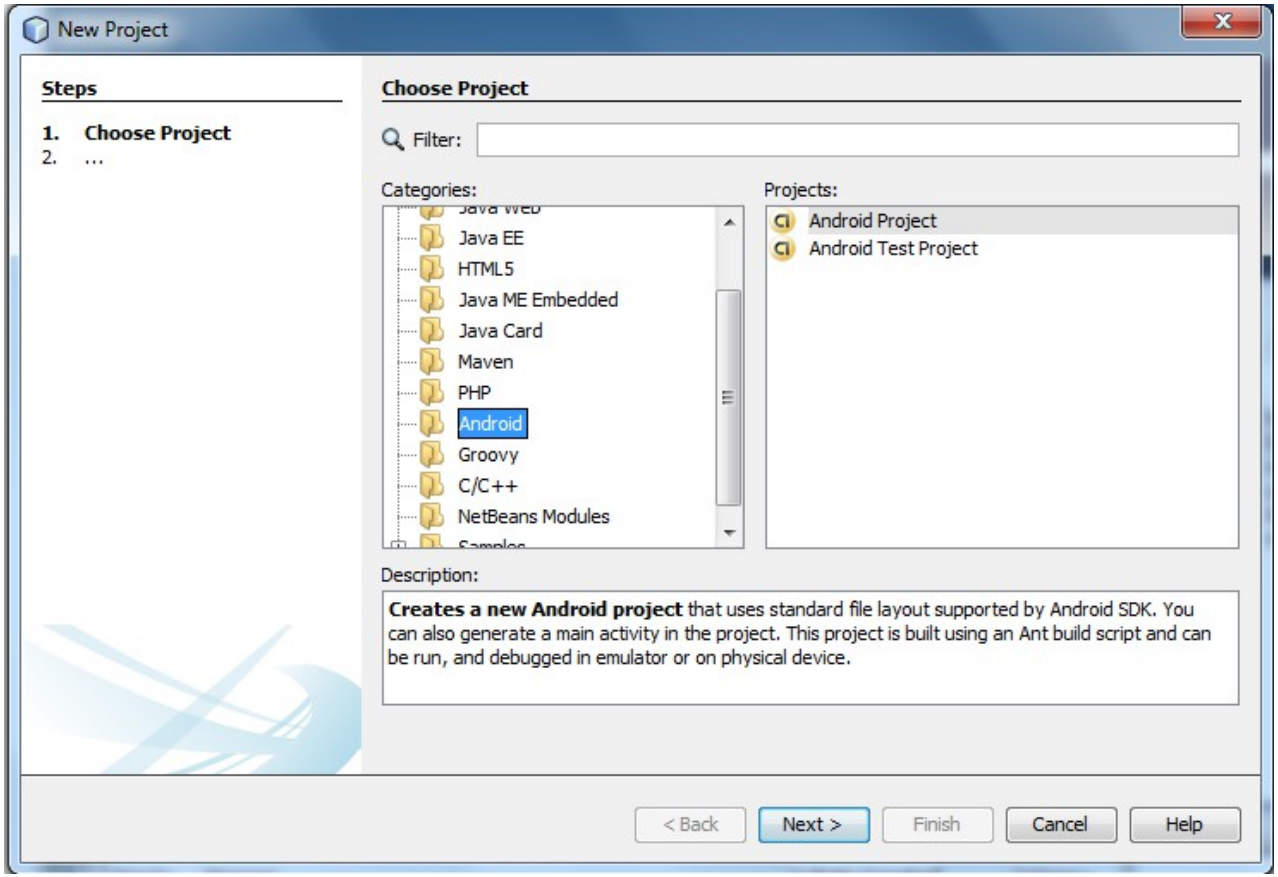
- *Tools – Options*.
- Ordner *Miscellaneous*.
- Registerkarte *Android*.

Fehlender Speicherort



Anlage eines neuen Projekts

- *File – New Project*.



Projekte

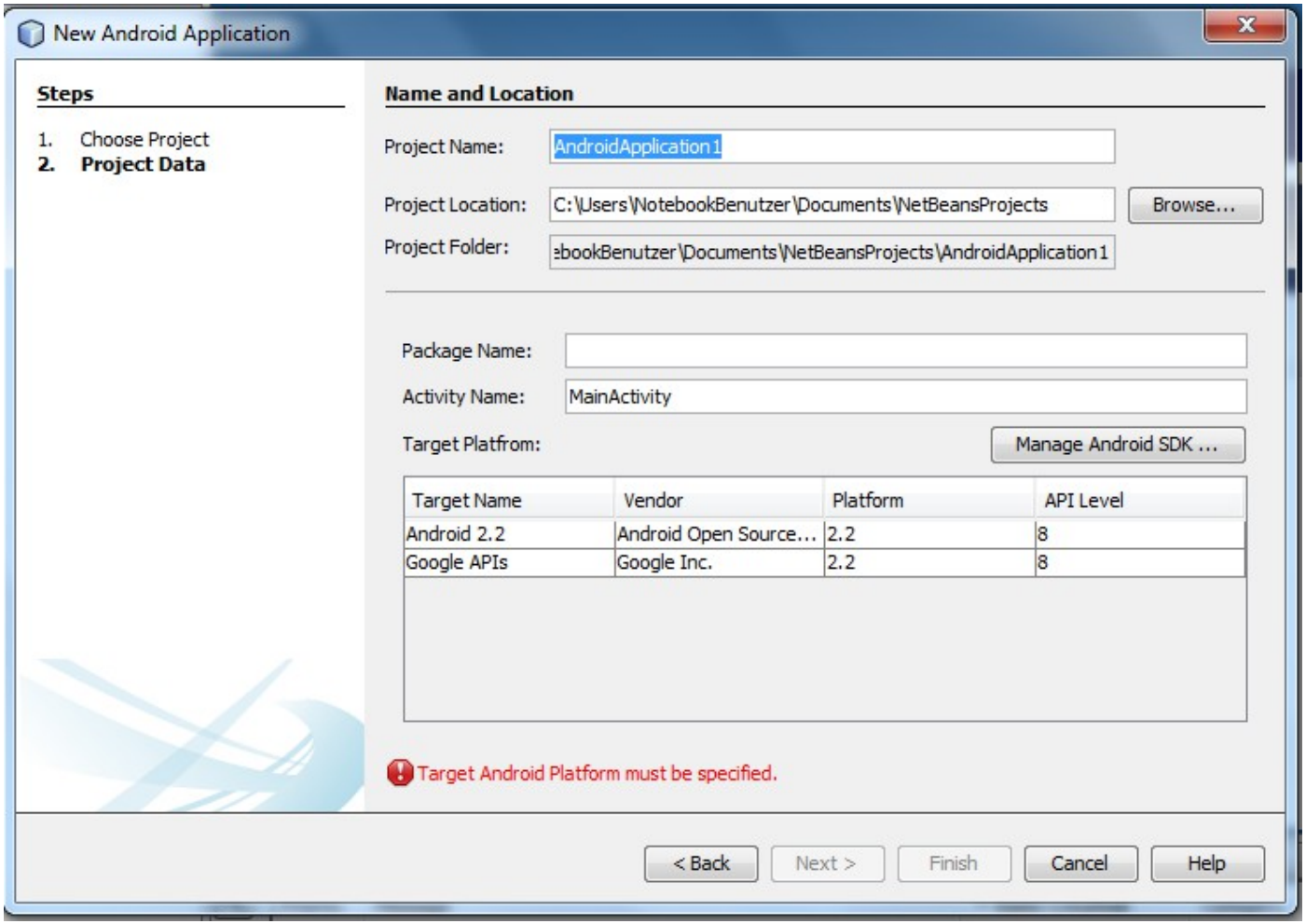
- Verwaltungseinheit in einer IDE.
- Zusammenfassung aller Dateien, die für ein lauffähiges Programm, egal auf welcher Plattform, benötigt werden.
- Sammlung von Dateien in verschiedenen Unterordnern zu einer Problemstellung.

1. Schritt: Auswahl einer Projektkategorie

- *Categories* Android. *Projects* Android Project.
- Das Grundgerüst für eine Android App wird automatisiert mit Hilfe des Assistenten erstellt.

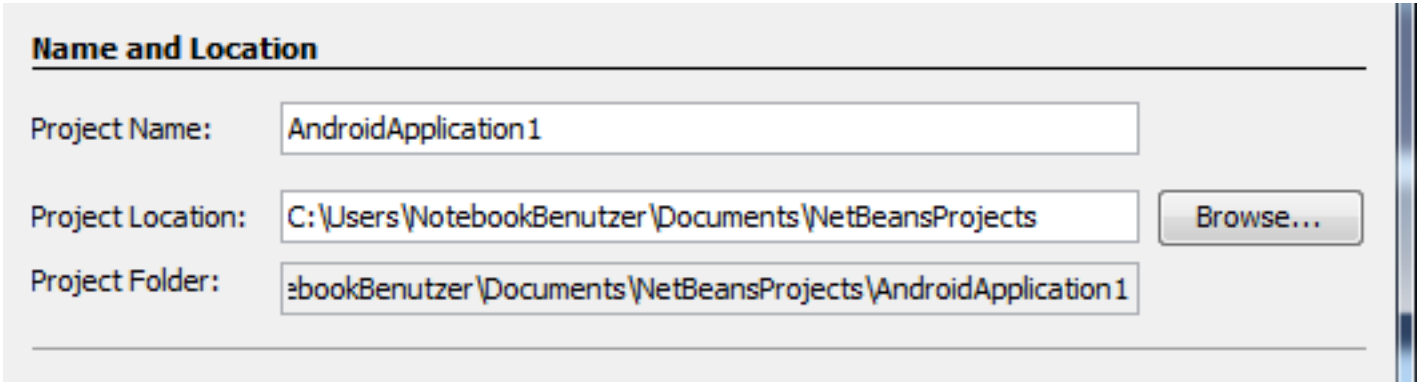
Leibniz Universität Hannover

2. Schritt: Angaben zum Projekt



Angaben zum Projekt

- Der Projektname (*Project Name*) ist frei wählbar. Die Bezeichnung spiegelt den Namen der App wieder.
- Als Speicherort (*Project Location*) wird der Standardpfad angezeigt. Mit Hilfe der Schaltfläche *Browse* kann der Pfad verändert werden.
- Der Ordner (*Project Folder*) setzt sich aus dem Speicherort und dem Namen des Projektes zusammen.

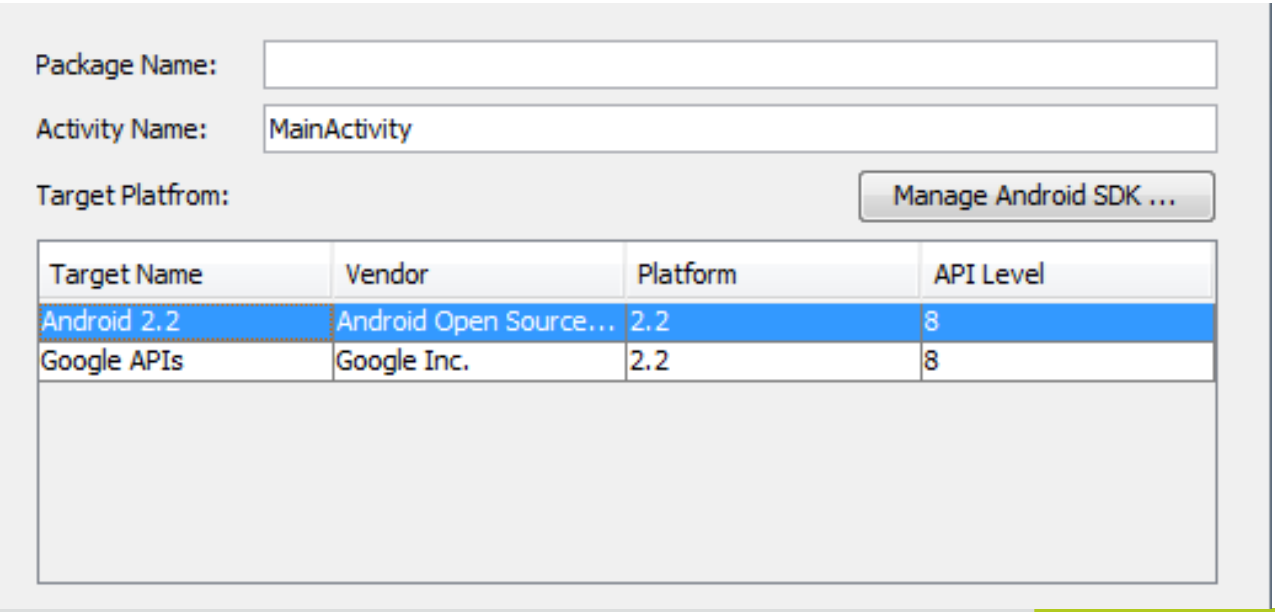


Hinweise zum Projektnamen

- In dem Projektnamen sollten keine Sonderzeichen oder Leerzeichen verwendet werden.
- Umlaute sollten vermieden werden.

Angabe zu Android

- Der Packetname (*Package Name*) ist frei wählbar.
- Im unteren Bereich wird eine Zielplattform (*Target Platform*) für die Entwicklung ausgewählt.
- Mit Hilfe von *Manage Android SDK* kann der Pfad zum Android SDK eingestellt werden.



Paketname

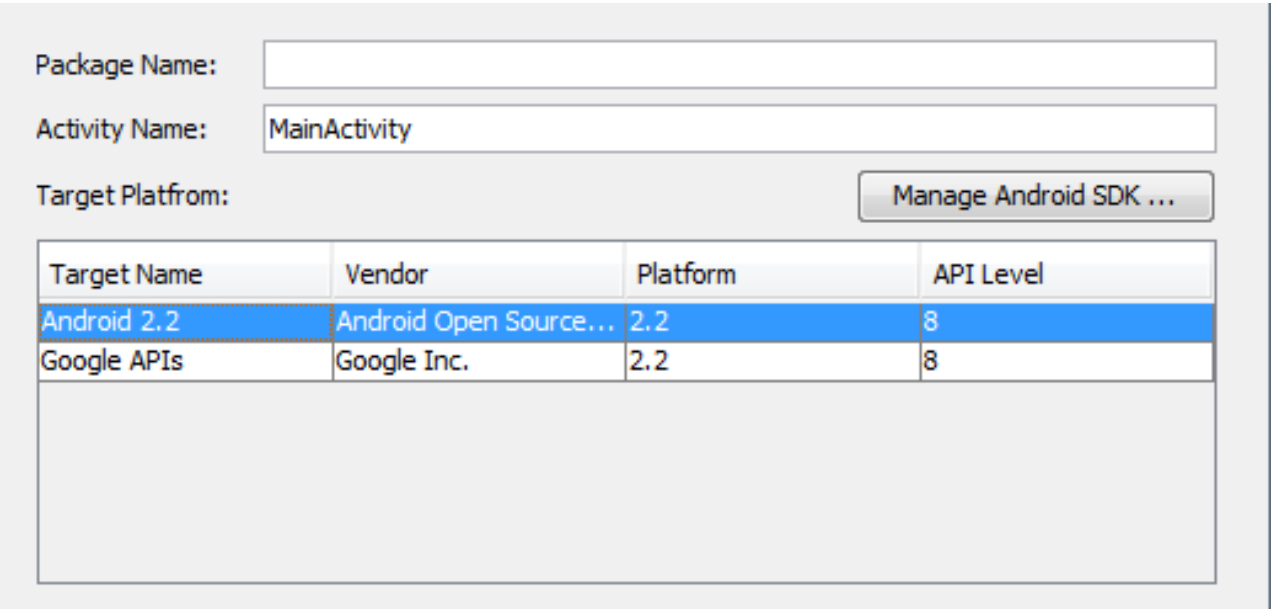
- Der Paketname besteht aus mindestens zwei Wörtern. Die Wörter werden mit einem Punkt verbunden. Jedes Wort symbolisiert einen Unterordner im Ordner *src*.
- Paketnamen werden traditionell kleingeschrieben.
- Häufig wird ein Domainname in umgekehrter Reihenfolge genutzt wie zum Beispiel de.meineWebseite.hallo.
- Der Name ist zwingend erforderlich.
- Der Name ist eindeutig.

Zielplattform

- Für welche Plattform wird die Anwendung geschrieben?
- Alle virtuellen Devices aus dem AVD Manager stehen zur Verfügung.
- Google API basiert auf Android 5 und bietet eine Schnittstelle zu Google Diensten wie zum Beispiel „Google Maps“.

Aktivität beim Start der App

- Das Feld *Activity Name* legt die Subroutine beim Starten der App fest.



Projekt erstellen

- *Run – Build Project (ProjectName)*. Das Projekt (die App) wird erstellt.
- *Run – Clean and Build Project (ProjectName)*. Das Projekt (die App) wird erneut erstellt. Vorherige Einstellungen werden entfernt.

Projekt starten (Windows 7)

- *Run – Run Project (ProjectName)*. Das Projekt (die App) wird ausgeführt.
- Der App-Emulator wird automatisch entsprechend der Einstellungen des Projektes gestartet. Eventuell muss das Smartphone durch Wischen mit der Maus freigeschaltet werden.

Projekt starten (Windows 8.1)

- Der AVD-Manager wird geöffnet. Die gewünschte Emulation wird mit Hilfe von *Start* geladen.
- Sobald die Emulation vollständig geladen ist, muss eventuell das Smartphone durch Wischen mit der Maus freigegeben werden.
- Das Menü *Run – Run Project (ProjectName)* in Netbeans startet das neu entwickelte Projekt (die App).

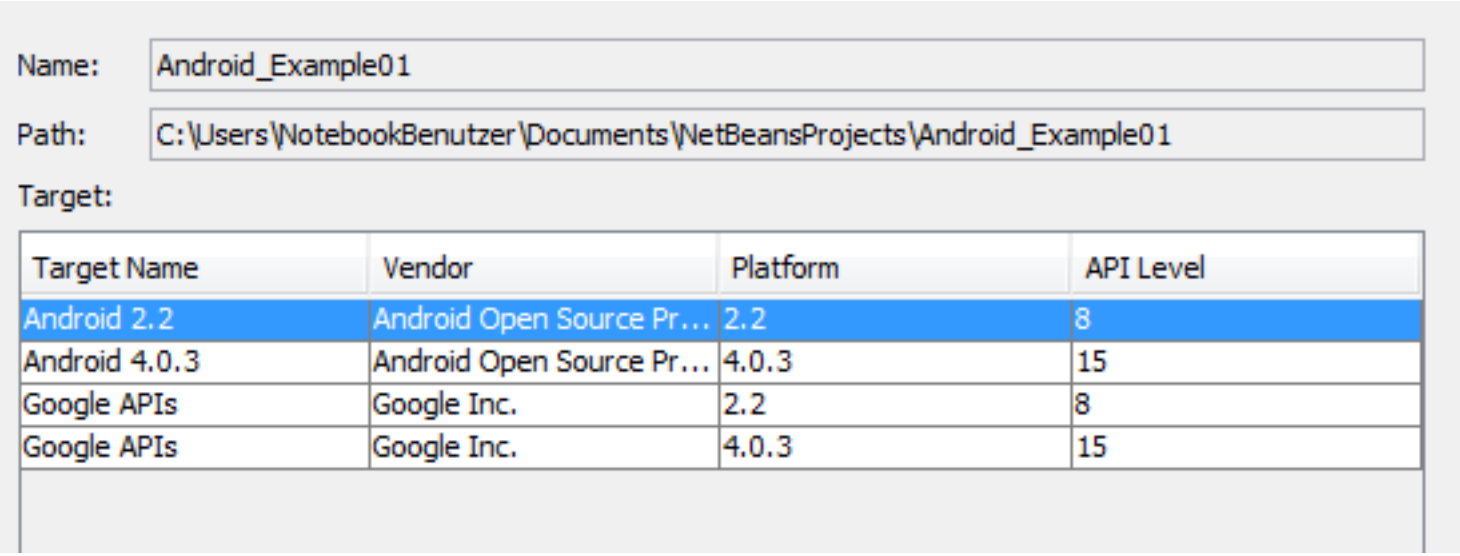
Hinweis

- Solange an dem Projekt in NetBeans gearbeitet wird, sollte der Emulator nicht geschlossen werden.

Eigenschaften eines Projekts

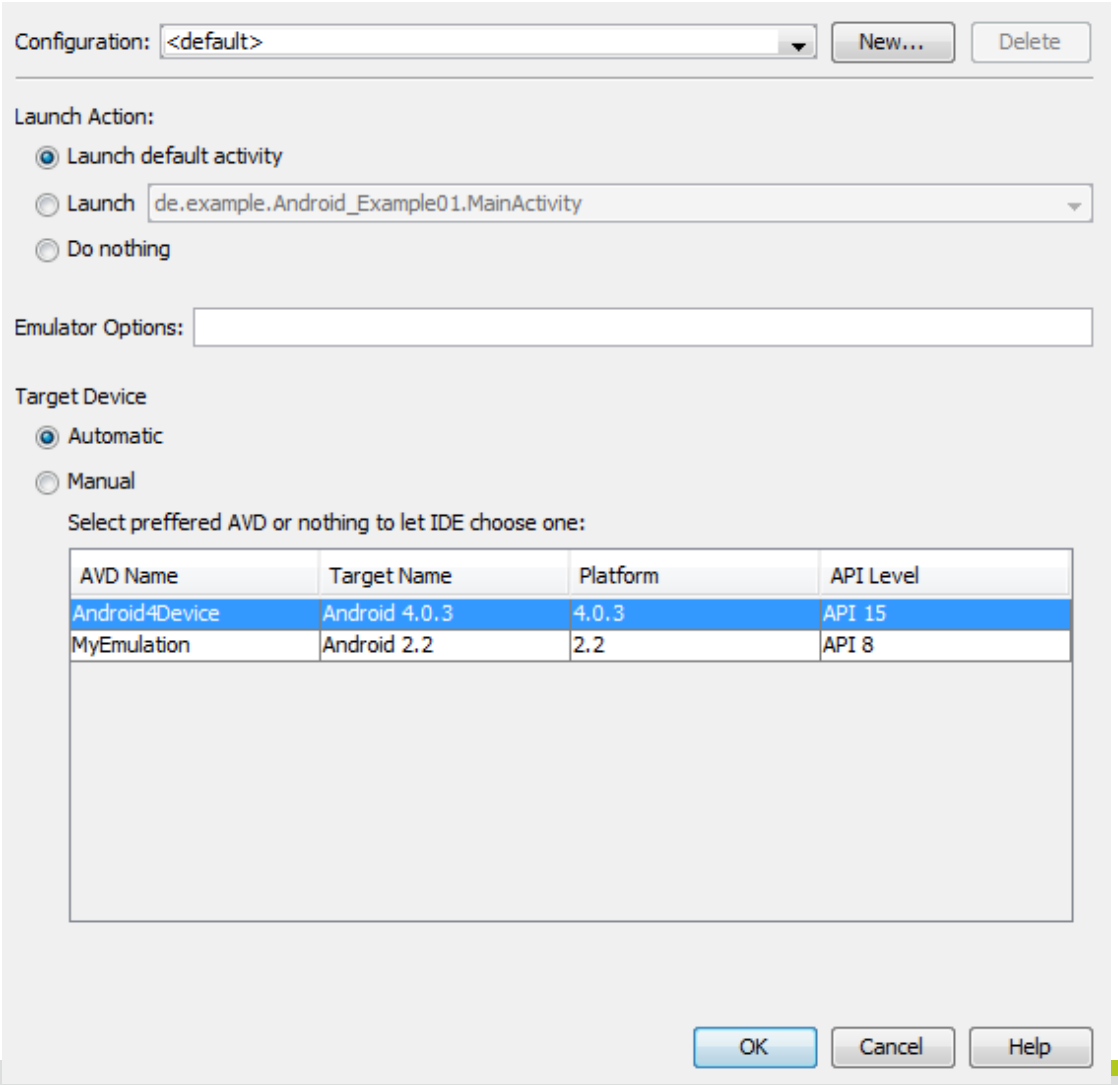
- *File – Project Properties (ProjectName)*.
- Die Registerkarte *General* enthält allgemeine Informationen zu einem Projekt.
- Die Registerkarte *Run* legt fest, für welche Zielplattform entwickelt wird und welcher Subroutine die App gestartet wird.
- Die Registerkarte *Formatting* legt Formatierungen für den Quellcode fest.

Registerkare „General“



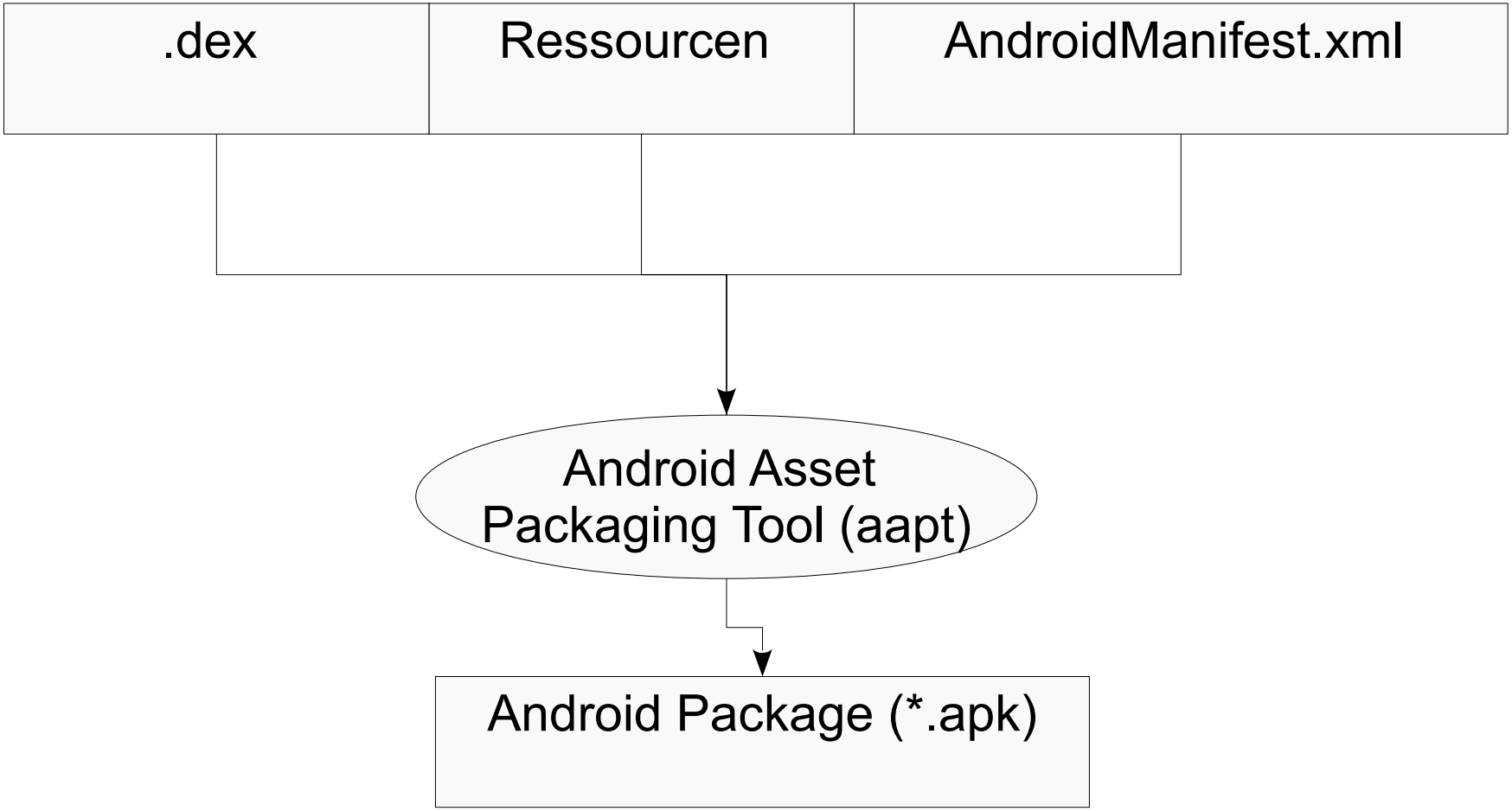
- Der Name des Projekts.
- Der Speicherpfad des Projekts.
- Alle, mit dem Android SDK Manager, installierten Zielplattformen.

Registerkarte „Run“



Leibniz Universität Hannover

Ausführbares Programm erstellen

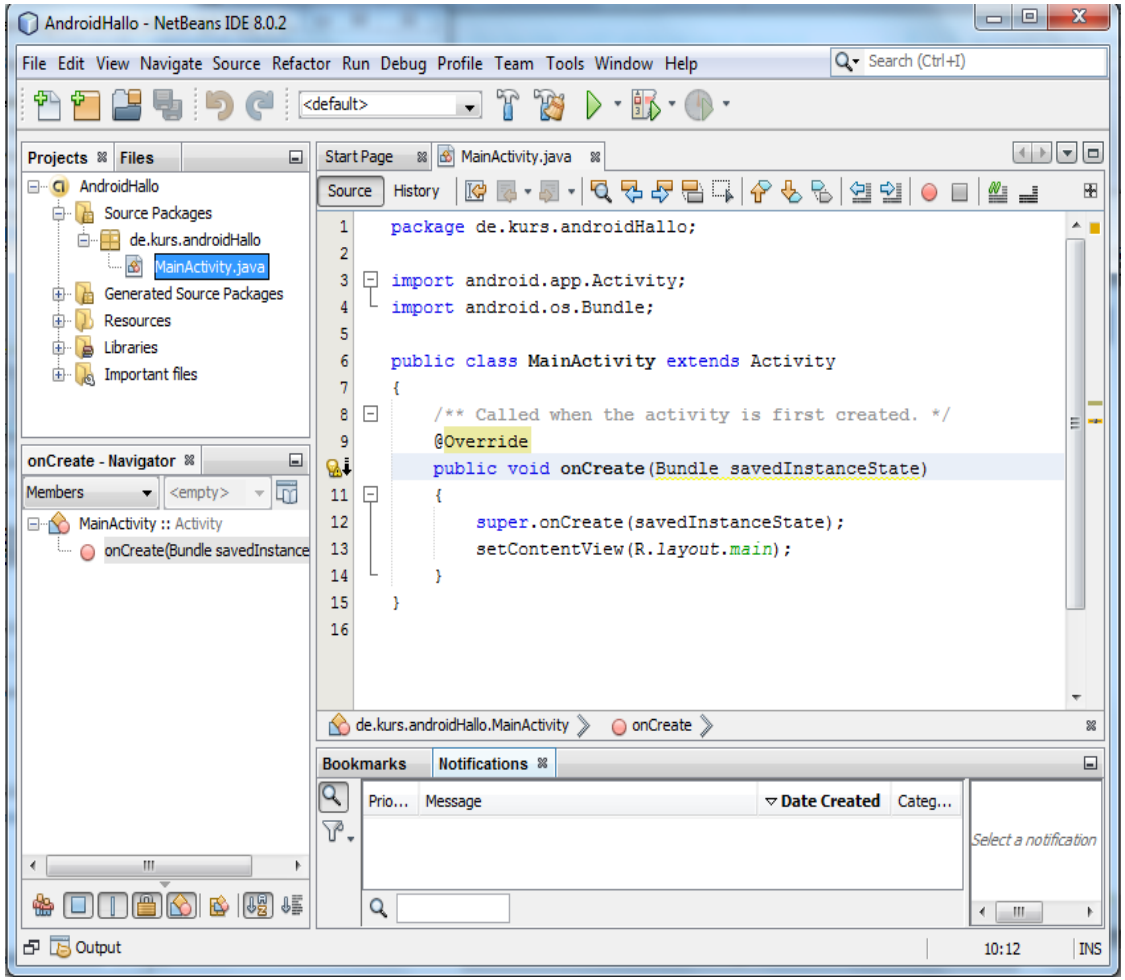


Projekt schließen

- *File – Close Project (ProjectName)*.

Leibniz Universität Hannover

Benutzeroberfläche

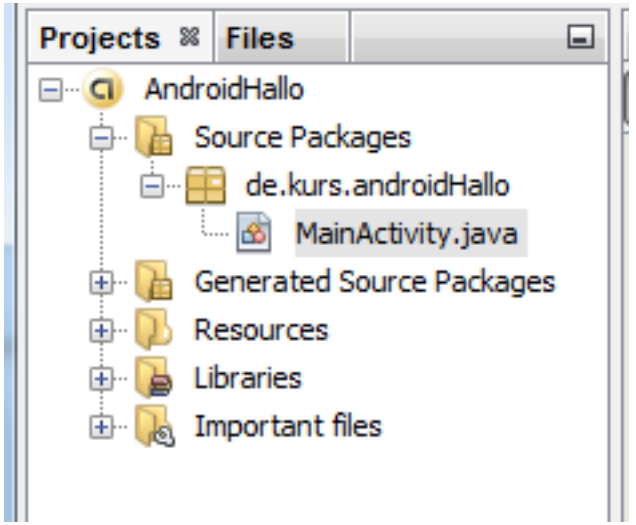


Aufbau der IDE

- In der Titelleiste wird der Name des Projekts angezeigt.
- Menüleiste.
- Symbolleiste für die wichtigsten Befehle. Mit Hilfe von *View* – *Toolbar* können Symbolleisten ein- oder ausgeblendet werden.
- Diverse Fenster für den Code und Informationen zu dem Projekt.

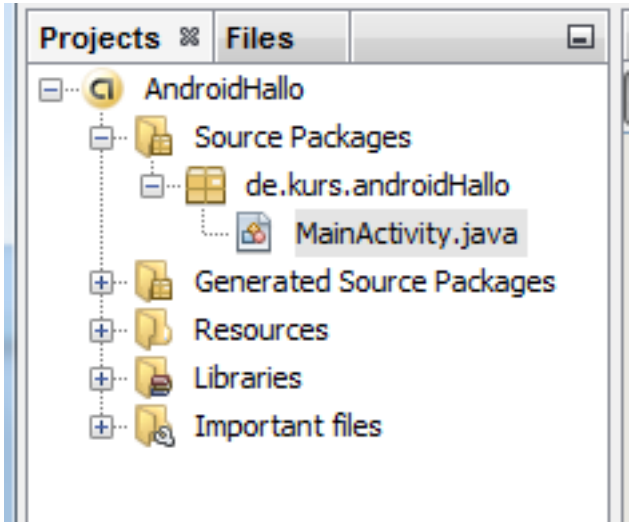
Fenster in der IDE

- Mit Hilfe von Drag & Drop kann ein Fenstern unterhalb der Symbolleiste an einer beliebigen Position abgelegt werden.
- Mit Hilfe der Schaltfläche in der rechten oberen Ecke wird ein Fenster minimiert. Die Registerkarte wird am linken Rand der IDE angezeigt.
- Fenster bündeln mit Hilfe von Registerkarten die verschiedenen Informationen. Ein Fenster besitzt mindestens eine Registerkarte.



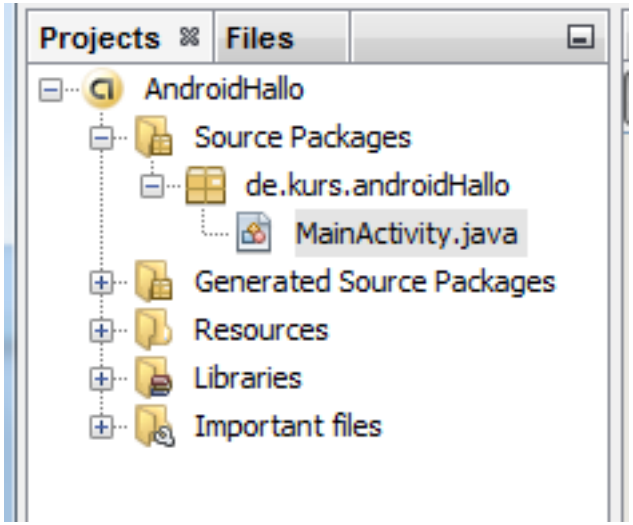
Registerkarten in einem Fenster

- Am oberen Rand der Registerkarte wird der Name und die Schließen-Schaltfläche angezeigt,
- Mit Hilfe von Drag & Drop kann eine Registerkarte von einem Fenster in ein anderes Fenster verschoben werden.
- Mit Hilfe des Menüs *Window* können die verschiedenen Registerkarten geöffnet werden.



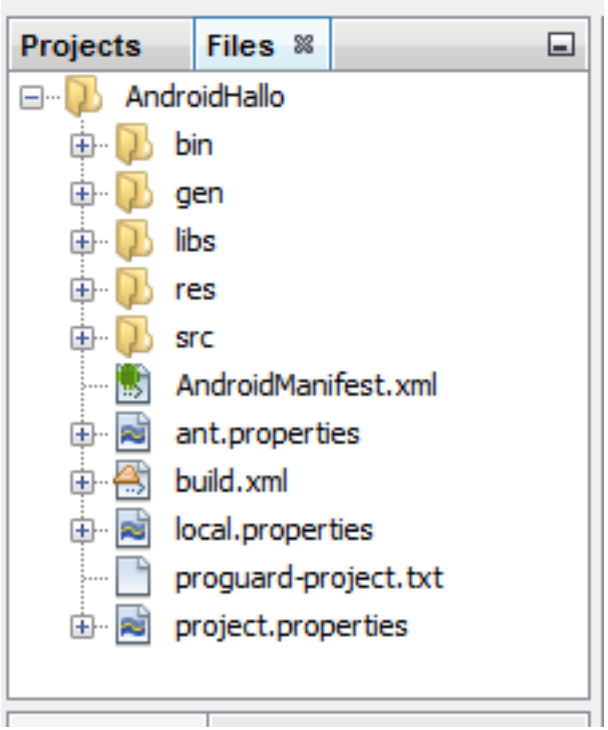
Projekt-Explorer

- Die Registerkarte *Projects* zeigt die Struktur des gewählten Projektes an.
- Im Ordner *Source Packages* befindet sich der Programmcode.
- Im Ordner *Libraries* wird immer die aktuell genutzte Standard-Bibliothek angezeigt.
- Im Ordner *Ressources* liegen Dateien im Format „xml“, die das Aussehen der App beeinflussen.



File-Explorer

- Die Registerkarte *Files* zeigt die Dateistruktur des gewählten Projektes an.
- Im Ordner *src* befindet sich der Programmcode.
- Der Ordner *gen* enthält automatisch generierte Daten.
- Der Ordner *libs* zeigt die Bibliotheken an.
- Im Ordner *res / layout* werden die, in XML definierten Layouts abgelegt.



Ordner src

- Quellcode einer App.
- Dateien mit der Endung *.java*. Pro datei darf nur eine Klasse definiert werden. Die definierte Klasse ist öffentlich (public).
- Die datei *MainActivity* wird automatisiert durch die Erstellung des Projektes angelegt. Die Datei bildet den Startpunkt der App ab.

Ordner gen

- Automatisch generierte Dateien.
- Die Dateien werden automatisiert bei der Erstellung der App verändert oder angepasst.
- Dateien im Ordner sollten nur angezeigt, aber nicht vom Entwickler verändert werden.

Manifest-Datei

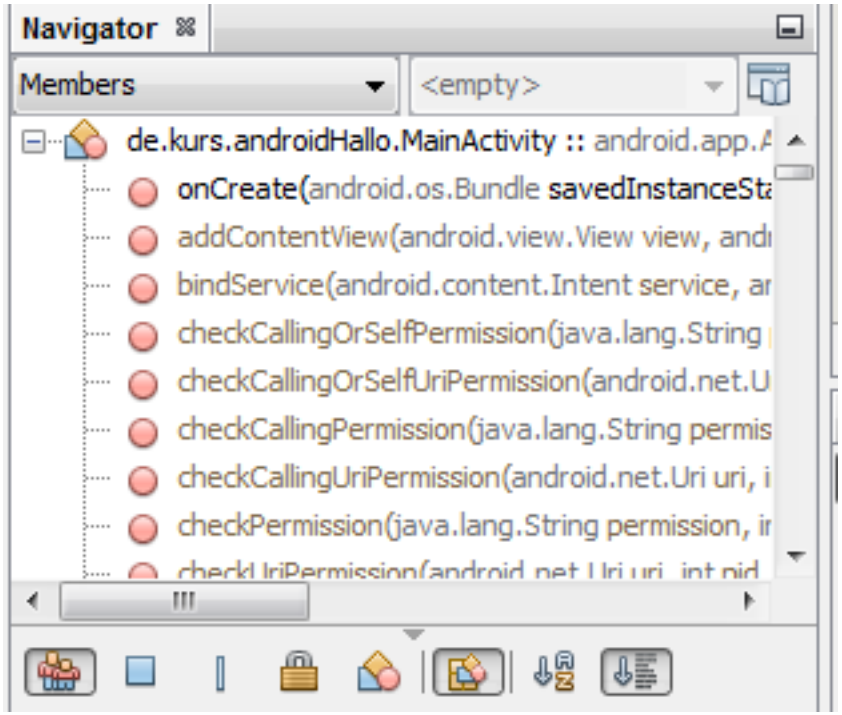
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.kurs.androidHallo"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:label="@string/app_name" >
    <activity android:name="MainActivity"
              android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Angaben in der Datei

- Welche Activities (<activity) werden ausgeführt?
- Mit welcher Activity wird die App gestartet (<action android:name="android.intent.action.MAIN" />)?
- Wie lautet der Paketnamen (Attribut package in dem Element <manifest>)?
- Für welches Gerät ist die App geschrieben (android:targetSdkVersion)?
- Auf welche Version kann die App gerade noch so ausgeführt werden (android:minSdkVersion)?

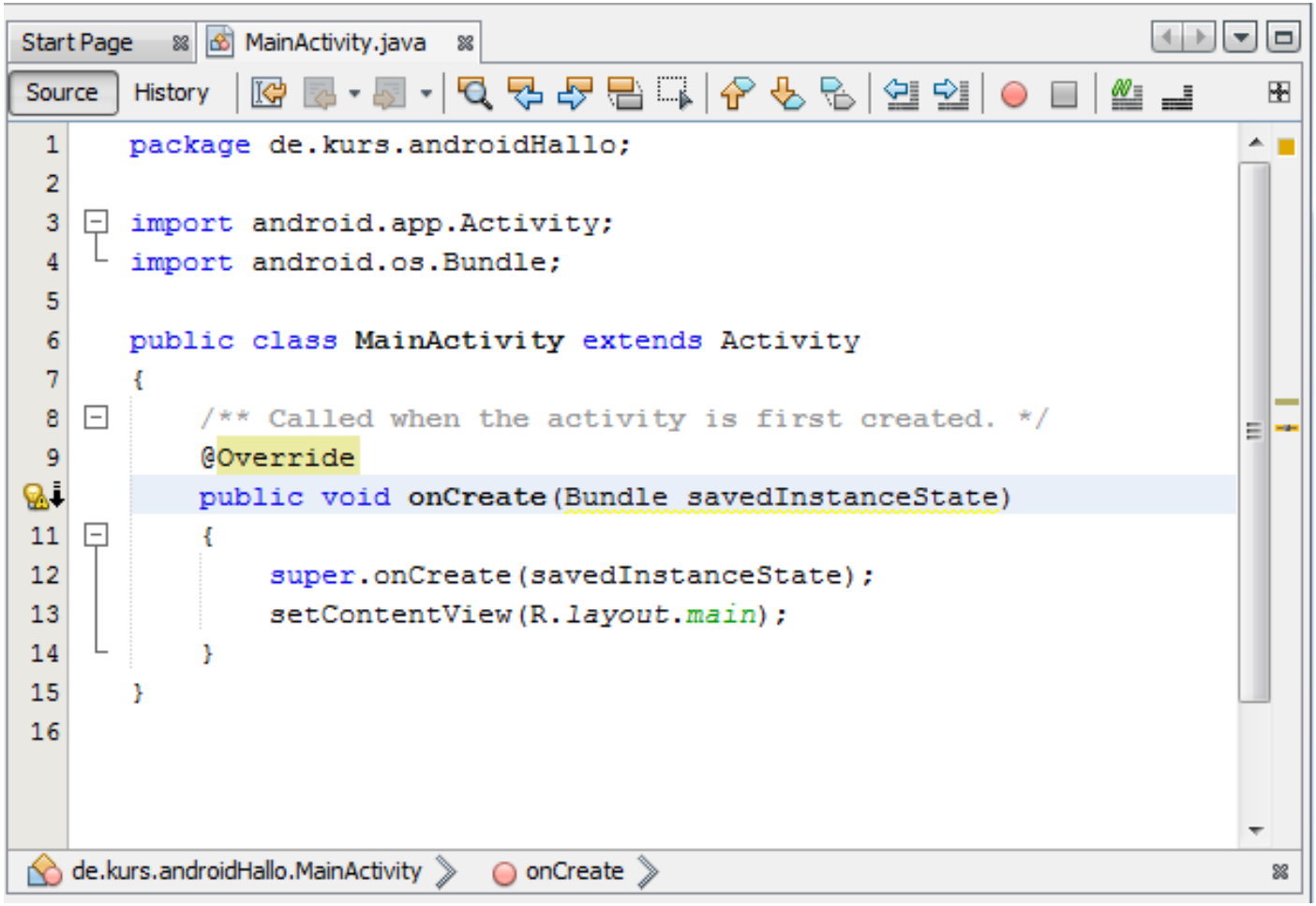
Navigator

- Strukturierte Darstellung des Codes.
- Auflistung von Methoden und Felder in dem momentan angezeigten Quellcode.



Leibniz Universität Hannover

Codefenster



Codefenster

- Inhalt einer Datei in einem Projekt.
- Mit Hilfe des Menüs *Tools* – *Options* und der Registerkarte *Editor* und *Fonts & Color* kann das Codefenster angepasst werden.

Dateien mit der Endung *.java

- Java-Code.
- Als Standardschrift wird schwarz verwendet.
- Die Schlüsselwörter der Programmiersprache Java werden in blauer Schriftfarbe angezeigt.
- Die Kommentare werden in grauer Schrift dargestellt.

Dateien mit der Endung *.xml

- Extensible Markup Language.
- Darstellung des Layouts in hierarchischer Struktur.
- Definition von String-Konstanten, Schriftgrößen etc.
- Tags werden in blauer Schrift dargestellt.
- Attribute der Tags werden in grüner Schrift dargestellt.
- Als Standardschrift wird schwarz genutzt.

Arbeitsschritte zur Erstellung einer App

- Layout festlegen. Wie wird die App auf dem Anzeigegerät dargestellt?
- Bereitstellung von Ressourcen. Welche Ressourcen werden für die Funktion der App benötigt?
- Schreiben von Java-Code. Wie verhält sich die App?
- Testen der App. Läuft die App auf den gewünschten Zielgeräten?

Grundgerüst eines Java-Programms für Android

```
package de.kurs.androidHallo;

import android.app.Activity;
import android.os.Bundle;

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

Grundgerüst eines Android-Programms

Deklaration des Paketes

Import der Bibliotheken

Klasse

Anweisungen in Java

```
package de.kurs.androidHallo;
```

- Jede Anweisung endet mit einem Semikolon.
- Das Semikolon beendet ein Schritt innerhalb eines Prozesses. Der Schritt wird in der Programmiersprache Java beschrieben.
- Pro Zeile sollte ein Arbeitsschritt beschrieben werden.

Packages

```
package de.kurs.androidHallo;
```

- Organisation von Klassen und Schnittstellen in einer Art Verzeichnisbaum.
- Vermeidung von Namenskonflikten.
- Sammlung aller Klassen einer App.
- Verbergen von Klassenelementen, die nicht `public` sind.
- Der Name wird bei der Neuanlage des Projektes mit Hilfe des Assistenten eingegeben.

Hinweise

- Der Name sollte nur aus Buchstaben von A..Z oder a..z, den Zahlen 0..9 und dem Unterstrich zusammengesetzt werden.
- Mit Hilfe des Punktes werden die Namen hierarchisch geordnet. Der Punkt wird als Pfad-Trennzeichen genutzt.
- Für Packages wird häufig die umgekehrte Domain-Namen-Schreibweise genutzt.

import-Anweisungen

```
import android.app.Activity;
import android.os.Bundle;
```

- Einbinden von Klassen aus der Android-Bibliothek.
- Import von Klassennamen zur weiteren Verwendung im Code.

Eingebundene Klassen

```
import android.app.Activity;
import android.os.Bundle;
```

- Die Klasse *Activity* ist in dem Paket *android.app definiert*. Die Klasse definiert verschiedene Methoden aus dem Lebenszyklus einer Activity.
- Die Klasse *Bundle* ist in dem Paket *android.os definiert*. Die Klasse stellt Methoden zum Datenaustausch zwischen verschiedenen Activities bereit.

Klassen in Java

```
public class MainActivity extends Activity
{
}
```

- Jede Klasse hat im Klassenkopf das Schlüsselwort class.
- Der Klassenkopf zeigt den Namen der Klasse und den Zugriff auf die Klasse an.
- Der Klassenrumpf beginnt und endet mit den geschweiften Klammern. In dem Rumpf werden Attribute und Methoden definiert.

Klassenkopf

```
public class MainActivity extends Activity
{
}
```

- Die Klasse hat den Namen MainActivity.
- Die Klasse erbt von der Klasse Activity (extends Activity)
- Die Klasse ist öffentlich (public). Die Klasse kann von außen aufgerufen werden.
- Die Klasse wird beim Starten des Programms aufgerufen (sie *AndroidManifest.xml*).

Methode onCreate()

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

- Mit Hilfe dieser Methode wird die Activity initialisiert.
- Beim Starten der Activity wird diese Methode aufgerufen.
- Hinweis: Die Methode onCreate muss für jede Activity bis zur API Level 12 implementiert werden.

Methodenkopf

```
@Override
   public void onCreate(Bundle savedInstanceState)
   {
   }
```

- Die Methode ist öffentlich (public). Auf die Methode kann von außen zugegriffen werden.
- Die Methode gibt an den Aufrufer keinen Wert zurück (void).
- Die Methode bekommt ein Bundle übergeben. Bundles speichern den Status einer Activity. Zum Beispiel ein Nutzer ändert die Orientierung von Hoch auf Quer. Die Daten in einem Formular bleiben erhalten, obwohl die Activity zerstört wurde.

Überschreiben der Methode in der Basisklasse

```
@Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);


   }
```

- super.onCreate() ruft die Methode onCreate() der Basisklasse auf. Der Platzhalter super verweist auf die Basisklasse Activity.
- In jeder überschriebenen Methode aus dem Lebenszyklus einer Activity muss die passende Methode aus der Basisklasse aufgerufen werden.

Annotationen

```
@Override
```

- Einführung mit Java 5.
- Beginn mit einem Add-Zeichen.
- Weitergabe von Metadaten.
- @override. Ist die Methode in der Basisklasse mit der gleichen Signatur vorhanden? Falls nein, wird ein Kompilierfehler ausgegeben.

Festlegung des Layouts

```
@Override
   public void onCreate(Bundle savedInstanceState)
   {

      setContentView(R.layout.main);
   }
```

- Die Methode setContentView() lädt das Layout der Benutzeroberfläche.
- Der Methode wird ein Verweis auf die dazugehörige Layout-Datei übergeben.

Verweis „R.layout.main“

```
setContentView(R.layout.main);
```

- R ist der Name einer Klasse, die automatisch erzeugt wird. Die Klasse übernimmt die Ressourcenverwaltung der App.
- Die Klasse R wird in der automatisch generierten Datei *R.java* definiert.
- In der Klasse R ist wiederum eine statische Klasse layout definiert. Diese Klasse verweist auf den Ordner *res*.
- In Klasse layout ist eine ID als Integer mit dem Namen main definiert. Diese ID verweist auf die xml-Datei *main.xml* in dem Ordner *res / layout*.

Klasse R in der Datei R.java

```
package de.tutorial.apphelloWorld;

public final class R {
    public static final class attr {
    }
    public static final class layout {
        public static final int main=0x7f020000;
    }
    public static final class string {
        public static final int app_name=0x7f030000;
    }
}
```

Fehler „R cannot resolved to a variable“

- Bebehung durch Ausführen von *Project – Build Project (ProjectName)*.
- Der Klassenamen R wird vom Compiler fehlerhaft interpretiert.

Leibniz Universität Hannover

Benutzeroberfläche in „Main.xml“

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello World, MainActivity"
    />
</LinearLayout>
```

Hinweise

- XML-Dateien zur Gestaltung des Layouts werden in den Ordner *res / layout* abgelegt.
- XML-Elemente beginnen mit den den spitzen Klammern und enden mit den spitzen Klammern.
- Für jedes XML-Element wird aus der entsprechenden Android-Klasse ein Objekt erzeugt. Das erzeugte Objekt wird entsprechend der Angaben konfiguriert.

Hierarchischer Aufbau

- Als Wurzel wird immer eine Layout-Komponente angegeben.
- Die Elemente darunter werden entsprechend der Wurzel ausgerichtet.

Layout-Komponente „LinearLayout“

- Die Elemente der App werden wie Kisten aufeinander gestapelt.
- Durch die vertikale Orientierung wird ein Element pro Zeile angezeigt, egal wie breit es ist.
- In der Beispieldatei wird die Breite und Höhe des Eltern-Containers genutzt.

View

- Element auf einer Benutzeroberfläche / Bildschirmseite.
- Zeichenflächen. Rechteckige Bereiche, in die gezeichnet werden kann.
- Widgets. Steuerelemente wie Textfelder, Schaltflächen etc. sind in dem Paket android.widget definiert.
- Container-Views, die andere Views in sich aufnehmen können.
- Layout-Views, die andere Views zusammenfassen und in einem bestimmten Raster anordnen.

Anzeige eines Textes „TextView“

- In diesem Beispiel nutzt das Textfeld die gesamte Breite der Eltern (layout_width="fill_parent").
- Die Höhe wird entsprechend des anzuzeigenden Textes gewählt (layout_height="wrap_content").
- Der Text „Hello World, MainActivity“ wird ausgegeben.

String-Variablen nutzen

```
android:text="@string/hello_world"
```

- Der Wert wird als Text an das Attribut text des Elementes TextView übergeben
- Eine String-Variable beginnt mit dem Add-Zeichen.
- Die Variable ist in der Datei *string.xml* in dem Ordner *res / value* gespeichert.
- Der Name der Variablen und der Speicherort werden durch den Schrägstrich getrennt.

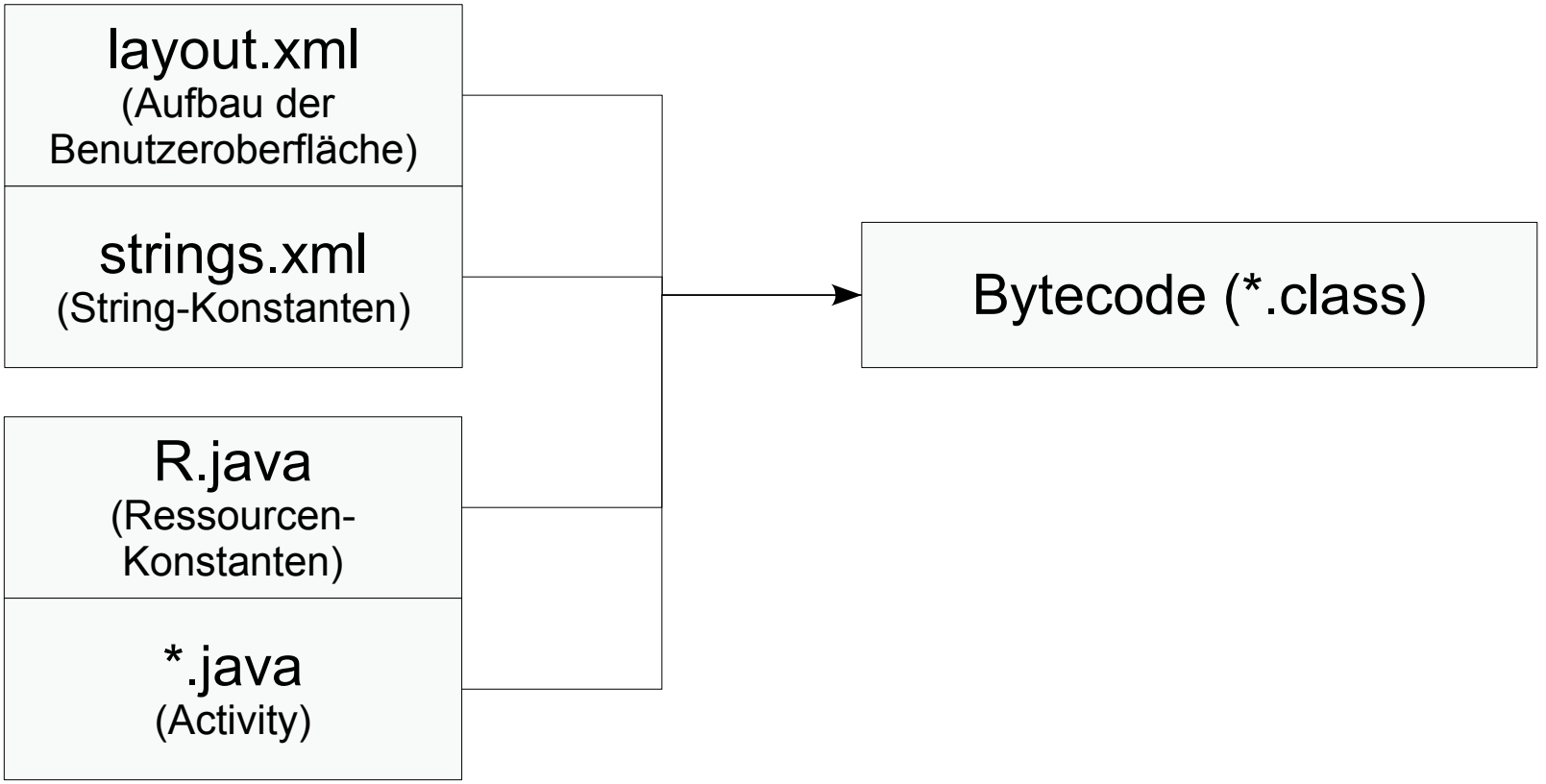
string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MainActivity</string>
    <string name="hello_world">Guten Tag</string>
</resources>
```

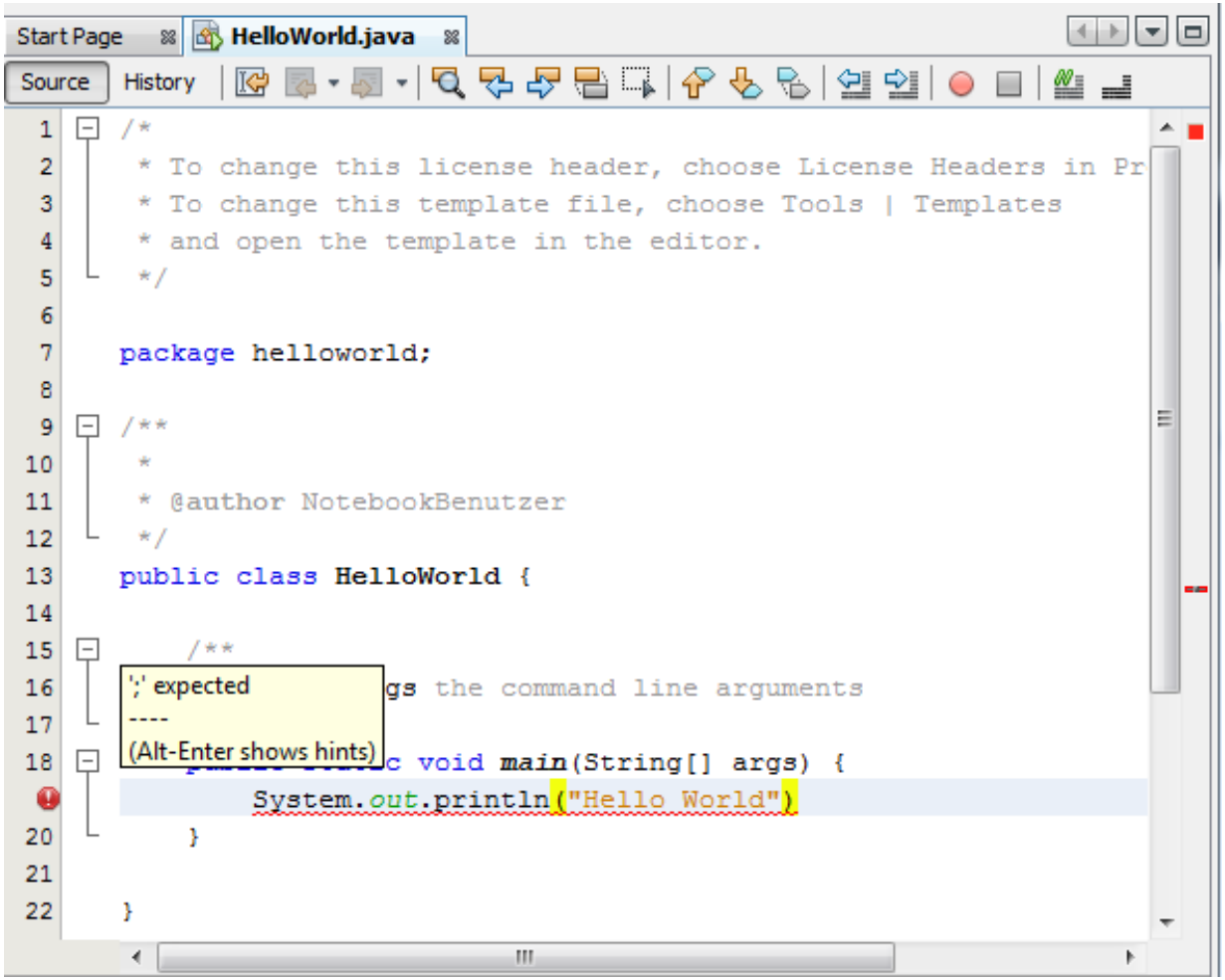
- Eine String-Variable beginnt mit dem Tag <string> und endet mit dem Tag </string>.
- Dem Attribut name wird die Bezeichnung der Variablen übergeben.
- Zwischen dem Anfangs- und Ende-Tag wird der dazugehörige Text angezeigt.

Leibniz Universität Hannover

Erstellung des Bytecodes



Syntaxfehler in NetBeans



Kompilierung