LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Praktikum Mobile und Verteilte Systeme

Programming with Android

Prof. Dr. Claudia Linnhoff-Popien
Michael Beck, André Ebert
http://www.mobile.ifi.lmu.de

Wintersemester 2015/16

Überblick: Themen des Praktikums

- Android-Programmierung
- Context Awareness und Location-based Services
  - Modellierung von Kontext
  - Mobile und kontextsensitive Dienste
- Positionierung
  - Outdoor
  - Indoor
- Kommunikation
  - Kommunikationstechnologien
  - Server-Kommunikation
- Projektphase

Ablauf

- Theorie:
  - Montag, 10-12 Uhr
  - Raum 131, Oettingenstraße 67
- Praxis:
  - Montag, 13-17 Uhr **oder**
  - Dienstag, 13-17 Uhr
  - Raum G U109, Oettingenstraße 67
- Prüfung:
  - Technische und inhaltliche Präsentation
- Webseite:
  - http://www.mobile.ifi.uni-muenchen.de/studium_lehre/ws1415/msp/index.html
  - Aktuelles, Folien, Termine, Literatur
  - E-Mail: msp@mobile.ifi.lmu.de

Umfang:

- 6 ECTS (Vertiefendes Thema für Bachelor Informatik und Bachelor Medieninformatik)
- 6 ECTS (Master Informatik und Master Medieninformatik)

Projektphase

1. Einreichung der Projektideen
2. Konzeptvorstellung für Projektphase
3. Treffen der Gruppen mit Betreuer
4. Zwischenbericht der Gruppen
5. Prüfung

Programming with Android

Today:

- Android basics
- components of an Android application
- communication between components
- Google Services
- Android Studio as Android IDE
- …

What is Android?

- Android is a multi-user, **Linux-based OS** developed by Google and the Open Handset Alliance
- primarily designed for touchscreen mobile devices based on **direct manipulation** by the user
- the Android code is **open source,** released under the Apache License

- comes with some standard smartphone applications

- the **Android SDK** offers developer tools and API libraries
- allows for **simple application** (app) **development** using customized Java

http://developer.android.com/sdk/index.html

Android statistics I

- Android has become **the world's most popular smartphone platform** (82,8% market share in 2Q2015)
- is deployed on tv-sets, games consoles, digital cameras, watches, ...
- September 3, 2013: 1 billion Android devices activated

OS	2Q15 Market Share
Android	82.8%
iOS	13.9%
Microsoft	2.6%
BlackBerry	0.3%
Others	0.4%
Total	100.0%

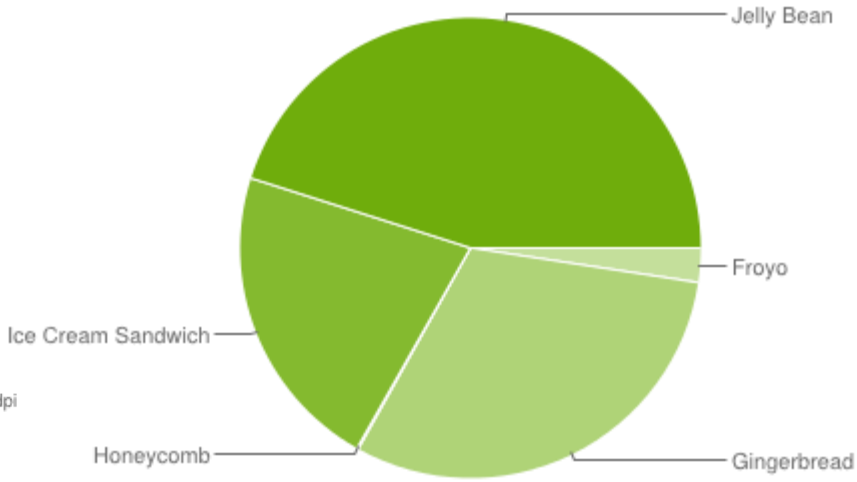
http://www.gartner.com/newsroom/id/2573415

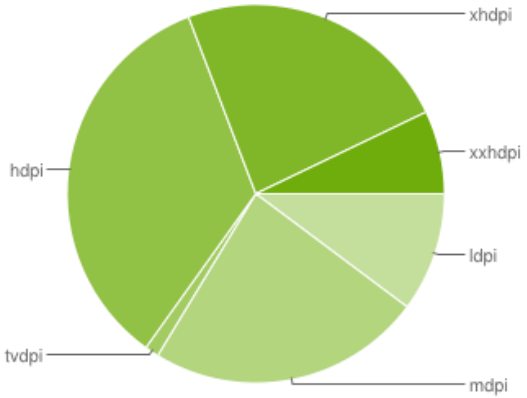
http://developer.android.com/about/index.html

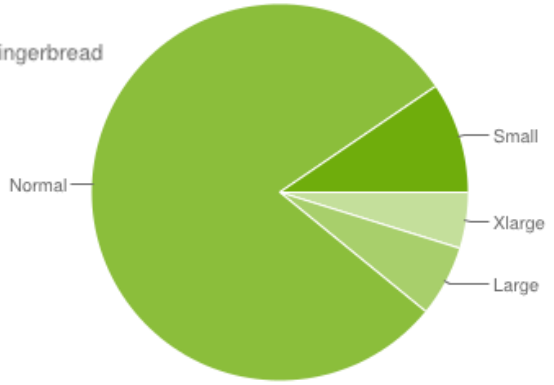
Android statistics II

- “In July 2013 there were 11,868 different models of Android devices, scores of screen sizes and eight OS versions simultaneously in use.”

http://en.wikipedia.org/wiki/Android_(operating_system)







http://developer.android.com/about/dashboards/index.html

Evolution of Android I

- Beta version **released in 2007**
- commercially released in 2008 (Android 1.0)
- from April 2009 onwards: dessert codenames, i.e., Cupcake, Donut, Eclaire, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, …

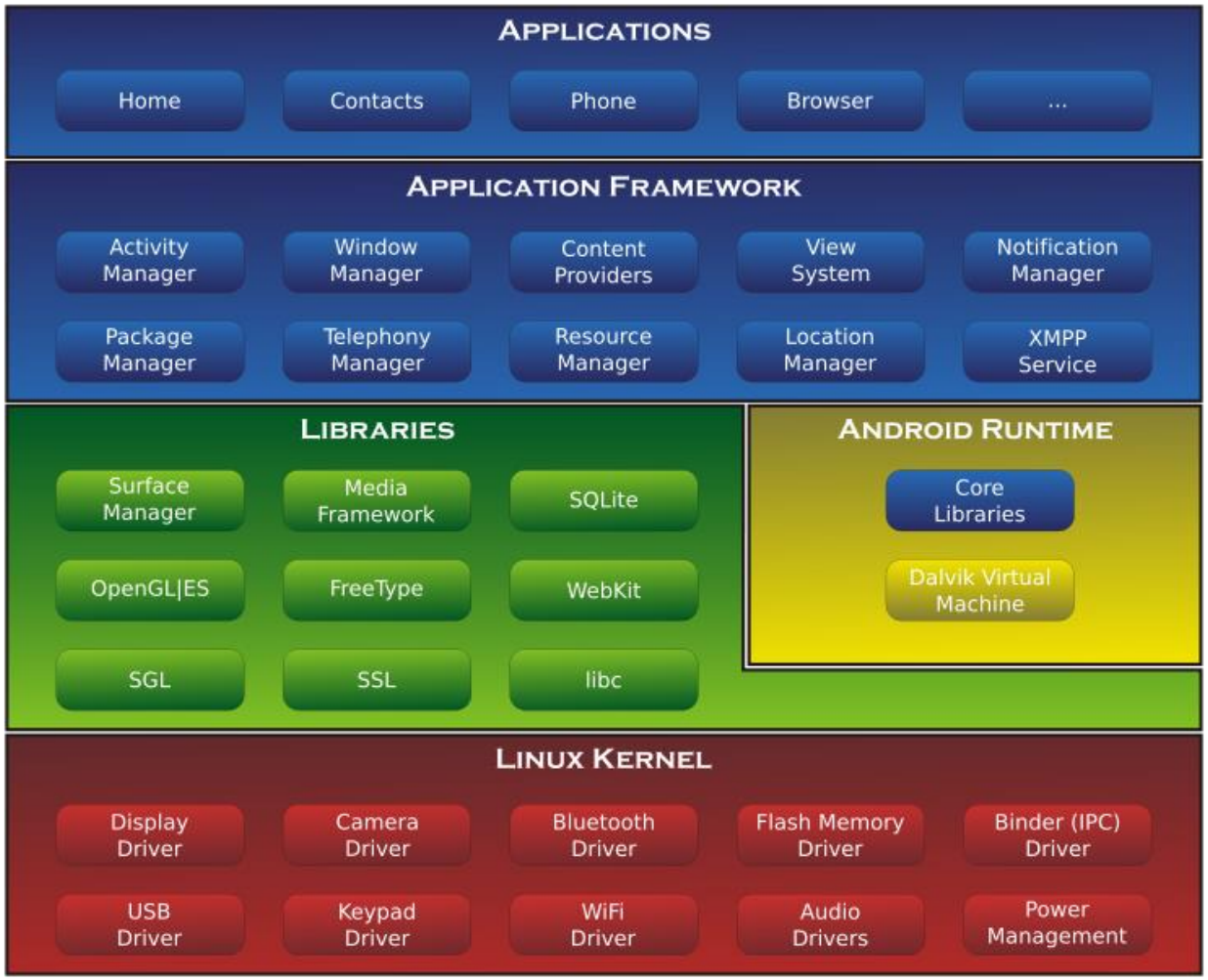
- OS **updates refer to API updates** (version codes vs. API levels)
  - offering both new functionality and restrictions for app developers

- current version: **Android 6.0 Marshmallow**
  API level 23

Evolution of Android II

- API level — New features
  - 5 — Bluetooth 2.1, support for more screen sizes, …
  - 8 — C2DM service for push notifications, …
  - 9 — UI update, NFC support, new sensors, rich multimedia, …
  - 11 — tablet-only version, new UI and animation frameworks, StrictMode for network access, …
  - 14 — unified UI framework, social API, calendar API, Android Beam, VPN API…
  - 16 — improved memory management, improved app stack navigation, new permissions, …
  - 17 — support for secondary displays, rtl-UIs, multiple users, …
  - 18 — restricted profiles, Wi-Fi scan-only mode, …
  - 19 — printing framework, new NFC reader mode, adaptive video playback, …
  - 20 — customized for smartwatches and wearables, …
  - 21 — material design, Android runtime, native 64 Bit
  - 22 — dual Sim, HD speech transmission, …
  - 23 — new permission system, USB type-c, native fingerprint scan, Android-Pay, …

Android basics – System architecture (until 5.0)

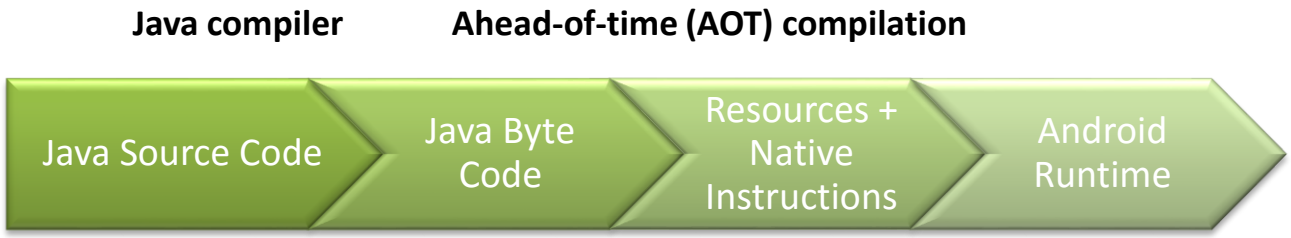


http://en.wikipedia.org/wiki/Android_(operating_system)

Android basics – Dalvik Virtual Machine

- Java code is typically compiled into **Bytecode**
- At runtime, a **Virtual Machine** translates this code into machine code
  - e.g., **Java Virtual Machine** (JVM) on Desktop PCs (stack-based)

- Android, however, uses the **Android Runtime (ART)**
  - Replaces Dalvik VM since Version 5.0 (backward compatible)
  - Transformes Bytecode directly to binary code upon installation
  - Faster execution, improved garbage collection and memory allocation
  - 64-Bit support
  - Apps are stored compiled



http://android-developers.blogspot.de/2013/06/google-play-developer-8-step-checkup.html

Android basics – Security

- Android implements the **principle of least privilege** for its apps

- Each Android app resides in its own kernel-level **security sandbox**:
  - each application is a different user
  - access permissions for all of an application's files are based on the Linux user ID
  - every application runs in its own Linux process
  - each process has its own VM (adds to stability)

- Apps can request **permission to access device data and services**, such as user's contacts, SMS messages, SD card, camera, internet, …
- All application permissions must be **requested by the developer** in the app's Manifest file and **granted by the user**

Android process and memory management

- Android employs **real application multi-tasking**, optimized for a mobile usage pattern

- Requirements:
  - apps should appear "**always running**"
  - no swap space → **hard limits on memory usage**
  - **app switching** in less than 1 second

- Implementation:
  - **LRU list** of running apps with preferences
  - when memory gets low, Android **kills the least important process**
  - `Bundle` class can be used for **saving application state**
    - developers have to take care of correctly saving an instance's state

Android application threads

- Every application is initiated with a single main thread (**UIThread**)
- If **time-consuming tasks** are performed on the main thread, **the UI blocks**
  - leads to ANR dialog after 5 seconds
  - instead, extra worker threads should be used
- the Android UI toolkit is **not thread-safe** and hence **must not be manipulated from a worker thread**

Rules:

1) Do not block the UI thread!

2) Do not access the Android UI toolkit from outside the UI thread!

- Recommendation: use the `Handler` and `AsyncTask` classes

Android application components

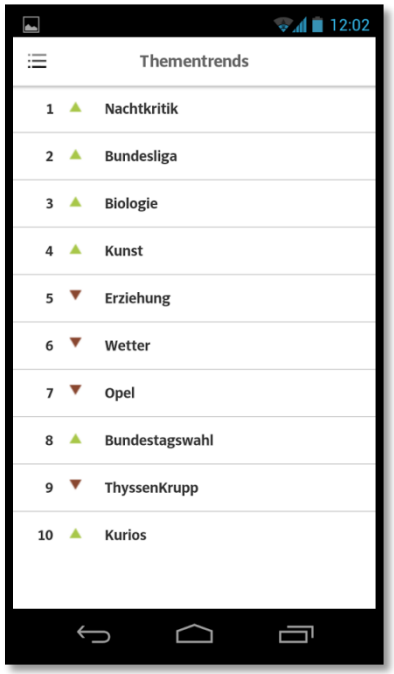
- Android apps might consist of several different building blocks
  - **Activities** (and Fragments)
  - **Services**
  - **Content Providers**
  - **Broadcast Receivers**

http://developer.android.com/guide/components/index.html

- Each component **performs different tasks**
- Each component has its own distinct **lifecycle** that you have to take care of as a developer in order to keep your app stable

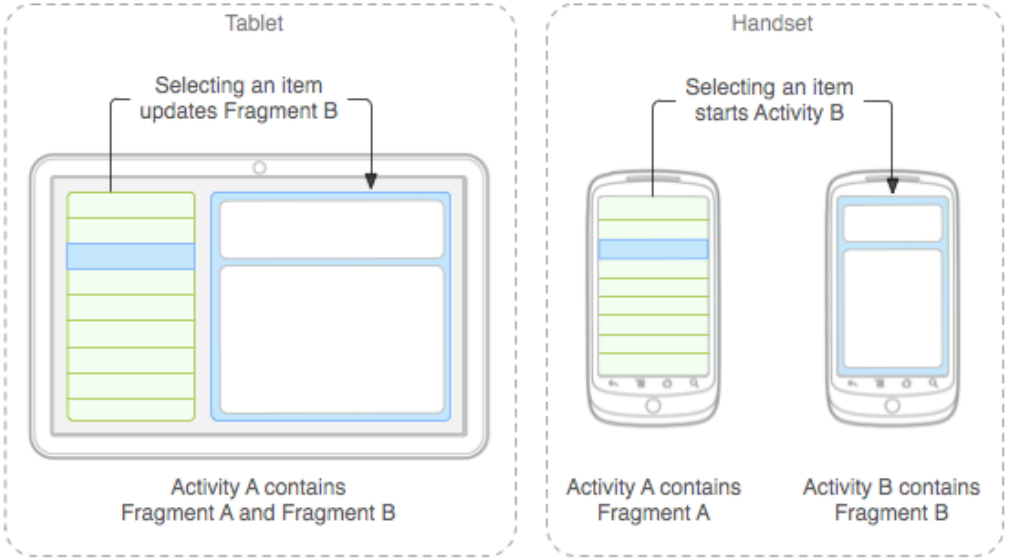
Activities

- Implemented as a subclass of `android.app.Activity`
- An activity represents **a single screen with a user interface**
  - typically defined in XML, not in code



Fragments

- represent **a UI portion of an Activity** (i.e., a “subactivity”)
- can be combined in a single activity to **build multi-pane UIs**, but cannot stand alone
- enable the **reuse of code** in multiple activities



- have their own lifecycle, too, based on the host Activity’s current state
- can be **managed in the Activity back stack**
- purpose: **different fragment combinations for different screen sizes**
  - e.g., in order to support both tablets and phones, different layout configs can be used to make optimal use of the available screen space

Services

- Java class implemented as a subclass of `android.app.Service`
- **running in the background** (without direct user interaction)
- intended for **long-running operations**, e.g. playing music, fetching network data
- can be started (and stopped) from an Activity
  - in order to interact with a Service, an **Activity can "bind"** to it

- Services can request being considered **foreground** („please dont kill me“)
  - indicated by an icon in the status bar to create user awareness

- a process running a service is ranked higher than a process with background activities (and is hence less likely to be killed)

BroadcastReceivers

- implemented as a subclass of `BroadcastReceiver`
- each broadcast is delivered as an `Intent` object

- **respond to system-wide broadcast announcements**:
  - screen turned off
  - battery status
  - picture captured
  - custom broadcasts
- do not display a user interface

- usually, a broadcast receiver is just a gateway to other app components, e.g., by starting an Activity or Service upon a certain event

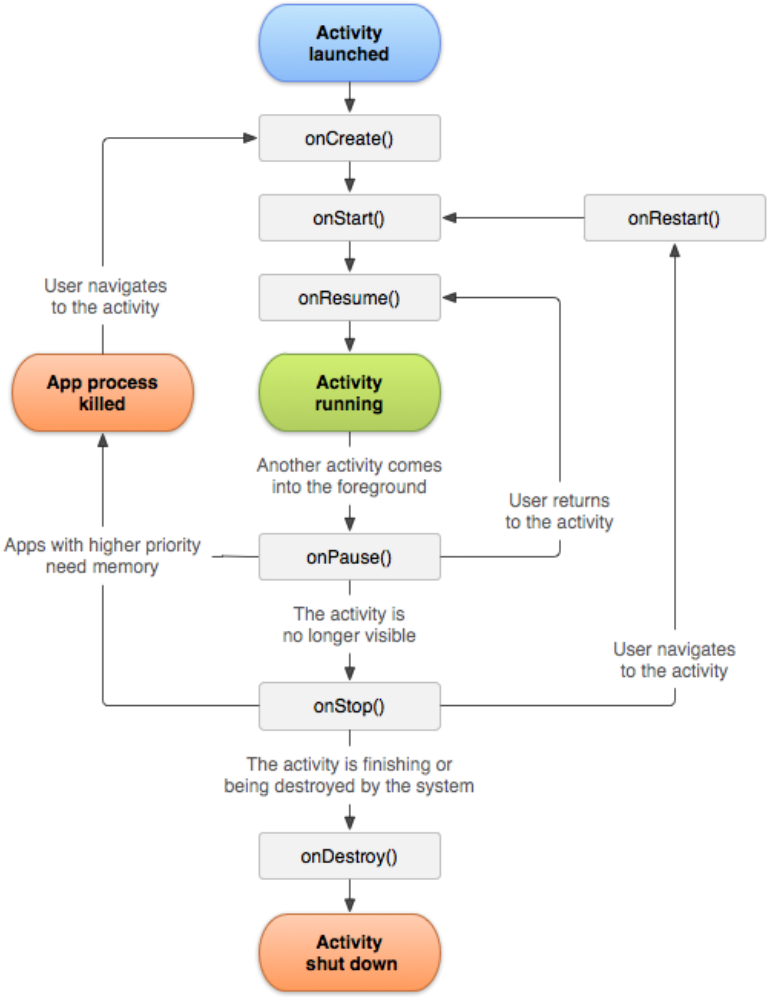
ContentProviders

- implemented as a subclass of `ContentProvider`
- must implement a standard set of APIs enabling other applications to perform transactions (CRUD operations) on the app's information

- manages shared application data, stored in files, SQLite databases, on the web, …
- can also be used internally by an app for storing/retrieving private information

- Example: **Android contact information**
  - any application (given it has the right permissions) is able to query this content provider to read or modify contact information

Activity lifecycle management



- crucial for developing strong and flexible applications
- An activity can exist in essentially three states:
  - ***Resumed***
    The activity is in the foreground of the screen and has user focus
  - ***Paused***
    Another activity is in the foreground and has focus, but this one is still visible
  - ***Stopped***
    The activity is completely obscured by another activity (i.e., in the background)

Android Manifest

```
<manifest …>
    <application …>
        <service android:name="de.lmu.ifi.…" …>
            …
        </service>
        …
    </application>
</manifest>
```

- Each application must have an **AndroidManifest.xml** file

- The manifest file **must declare**
  - an app's Java package name
  - **all of an app's components** (activities, services, …)
  - all of the app's requirements (min. Android version, hardware, …)

- and **might** also declare
  - intent filters (for implicit intents)
  - custom permissions
  - used libraries (apart from the standard Android lib)
  - **required permissions**
  - …

Android permissions

- by default, no app is allowed to perform any protected operations
- the **permission mechanism** can be used for a (moderately) fine-grained control of what features an app can access
  - internet, camera, SMS, contacts, reboot, …

- at install time, **a user has to accept the requested permissions** (do-or-die)
- since Android 4.3, there‘s a (hidden) functionality to withdraw individual permissions
- Since Android 6.0, it is possible to install Apps without granting all permissions (they are granted on-demand)

- **custom permissions** can be defined, controlling…
  - from which apps broadcasts might be received
  - who is allowed to start an activity or a service

Android resources

https://developer.android.com/guide/topics/resources/index.html

- all types of non-code resources (images, strings, layout files, etc.) should be managed externally
  - **allows for alternatives** (different strings for different languages, layouts for different screen sizes)
  - requires each resource to have a **unique resource id**

- resource types:
  - Bitmap / Drawable files (`res/drawable, res/mimap-hdpi…`)
  - XML layout files (`res/layout`)
  - string literals (`res/values`)
  - …
- alternatives are provided in separate folders: `<resource_name>-<qualifier1[-qualifier2]>`

R.java???

- when compiling your project, a class called `R.java` is generated
  - contains subclasses for each type of resources
- resources provided externally can be accessed in code using the projects `R` class and the corresponding resource‘s type and id

- **a resource id** is composed of
  - the **resource type** (e.g., string)
  - the **resource name** (filename or XML attribute “name“)

- Resources can be accessed in code: `getString(R.string.hi)` and in XML: `@string/hi`
- `(<Classcast>)findViewById(R.layout.x)`

Rules:

Never touch R.java!

Never import android.R!

Google Services

https://developer.android.com/google/index.html

- Google offers app developers a number of handy services that can be integrated into apps
- these services, however, are not part of the Android platform

  - **Google Cloud Messaging Service**
    allows developers to send push notification to their users

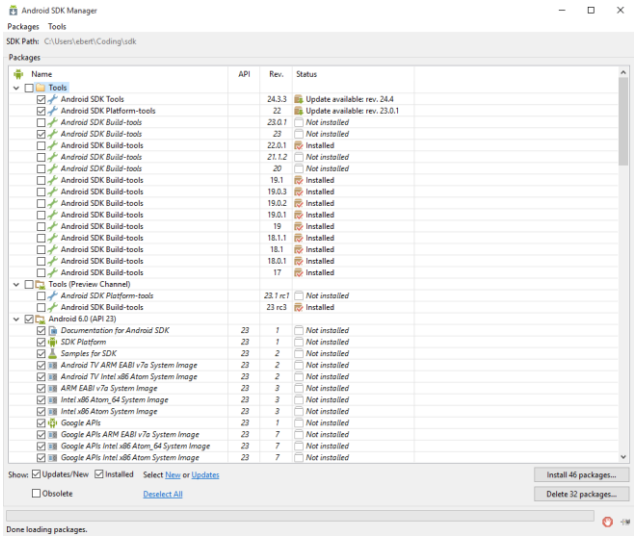
  - **Google Location Services**
    offer utilities for painlessly building location based services (LBS)

  - **Google+**
    allows authentication, social graph interactions, etc.

  - **Google Maps**, **Google Play Services**, …

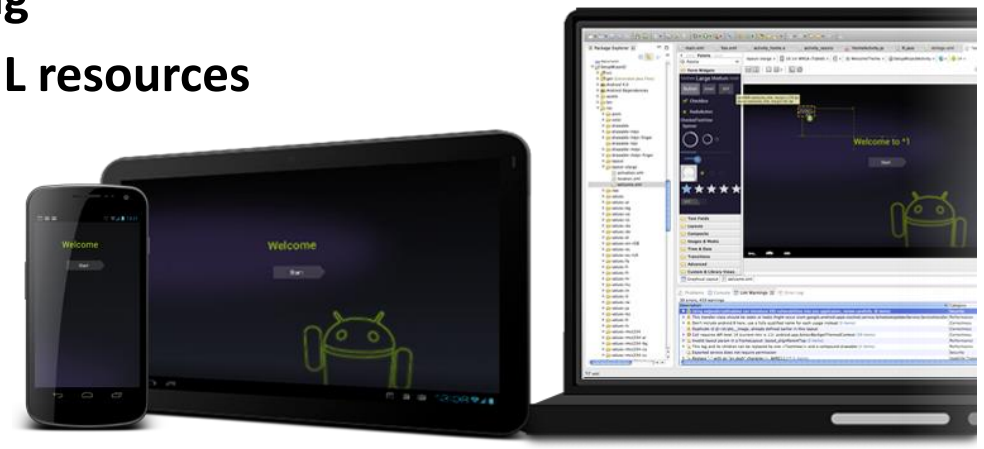
Android platform tools

- The Android Developer Tools (ADT) contain a variety of useful tools for application programming, debugging and publishing
  - **SDK Manager**
  - **ADB** (Android Debug Bridge)
    - `devices`
    - `shell`
    - `push/pull`
    - `install/uninstall`
    - `logcat`
  - **DX**
    - converts .class files into .dex format
  - **DEXDUMP**
  - **Android Device Emulator / AVD Manager**
  - **GUI Builder**
  - **DDMS**



Android IDE

- **Android Studio**
  - based on IntelliJ IDEA
  - Android-specific **refactoring**
  - **integration of Android XML resources**
  - **graphical UI editor**
  - virtual device **emulator**
  - Integrated Debugging
  - App Signing



https://developer.android.com/tools/index.html

- **Android Developer Tools** (ADT) **Eclipse plugin**
  - same Features as above
  - BUT: **Deprecated**

Where to start…

developer.android.com

Programming with Android – Practical

- IDE installation and setup
- „HelloAndroid“
- using the emulator, using adb
- …