

App-Entwicklung mit Android Studio

- Installation und Einrichtung von Android Studio -

Autor: Jochen Pogrzeba, StR
Max-Weber-Schule, Freiburg

Inhaltsverzeichnis:

1	Vorbemerkungen	1
2	Einrichtung der Entwicklungsumgebung	2
2.1	Einrichtung Android Studio	2
2.2	Aktualisierung des Android SDK	4
2.3	Einrichtung des Emulators über den AVD-Manager	5
2.4	Einrichtung des Tablets	8
3	Erste Schritte mit Android Studio	10
3.1	Die Bedienoberfläche von Android Studio	10
3.2	Grundgerüst eines Android-Projektes	14
3.3	Laden eines bestehenden Projektes	15
4	Testen einer App	16
4.1	Testen einer App im Emulator (Android Virtual Device)	16
4.2	Testen einer App direkt auf dem Tablet	17
4.3	Testen einer App mittels Deployment	18
5	Weitere Vorgehensweise	19
6	Linkliste	20

1 Vorbemerkungen

Diese Handreichung soll durch die Installation und Einrichtung der Entwicklungsumgebung *Android Studio* führen. Sie dient somit als Voraussetzung für die Unterrichtseinheiten, die die konkrete Umsetzung einer App behandeln (siehe *UE BMI-App* oder *UE Taschenrechner-App*)

Als Programmieroberfläche wurde für dieses Unterrichtsbeispiel *Android Studio* benutzt, da diese Software von Google als offizielle IDE zur App-Entwicklung für Android unterstützt wird. Eclipse mit seinen Plugins wird von Google nicht mehr unterstützt.

Android Studio basiert auf der IDE *IntelliJ IDEA*, welche sich ähnlich wie Eclipse bedienen lässt, der Einarbeitungsaufwand für Lehrer und Schüler ist daher sehr überschaubar.

In Kapitel 2 wird die Installation und Einrichtung des Android Studios und seiner Peripherie besprochen. Da dies an manchen Stellen komplex sein kann, richtet sich dieses Kapitel vor allem Administratoren. Dann werden erste Schritte im Umgang mit Android Studio (Kapitel 3) besprochen. In Kapitel 4 wird abschließend das Testen einer selbst erstellten App behandelt.

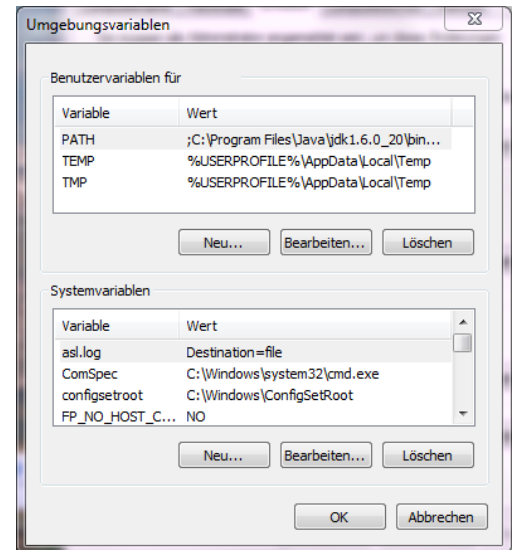
2 Einrichtung der Entwicklungsumgebung

2.1 Einrichtung Android Studio

Die Einrichtung des Android Studios nimmt einige Zeit in Anspruch, daher gilt als erste Regel: Bringen Sie sich Zeit und Muße mit, wenn Sie Android Studio und seine Peripherie installieren und einrichten.

Zuerst muss ein aktuelles *Java Developer Kit (JDK)* Version 7 oder höher installiert werden. Dies findet man zum Beispiel auf der Internetseite von Oracle (Linkliste). Diese Java-Installation muss später von Android Studio erkannt werden.

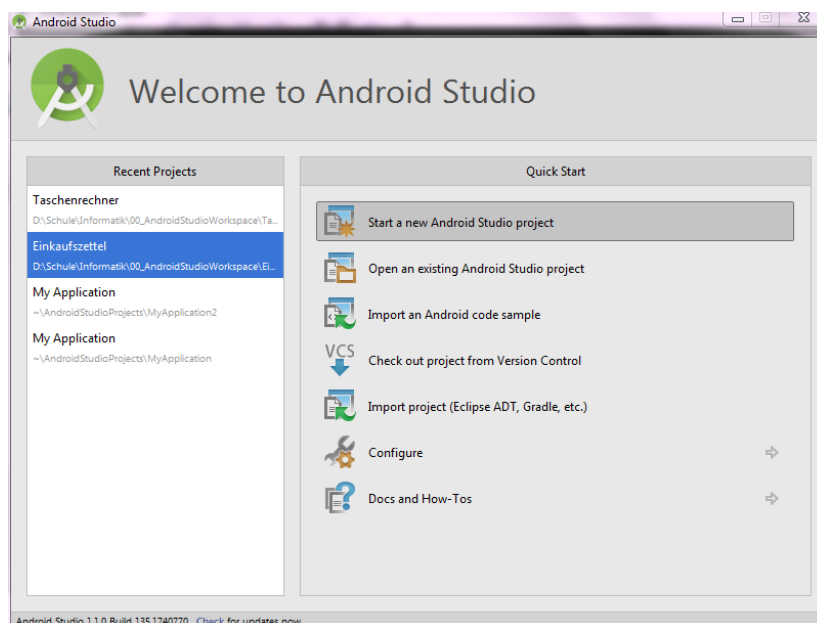
Sollte dies nicht der Fall sein, muss eine Umgebungsvariable mit dem Pfad zur Java-Installation angelegt werden. Gehen Sie dazu zu den erweiterten Systemeinstellungen (über *Systemsteuerung* → *System und Sicherheit* → *System* → *Erweiterte Systemeinstellungen* → *Umgebungsvariablen*). Dort legt man eine Variable mit dem Namen *JDK_HOME* an, welche auf das Verzeichnis zeigt, in dem das JDK im vorherigen Schritt installiert wurde.



Danach wird Android Studio installiert. Diese Software unterliegt der Open Source-Lizenz und ist daher frei verfügbar. Es kann beispielsweise von der Android Entwickler Seite heruntergeladen werden. (Linkliste)

Android Studio ist kompatibel ab Windows 2003, ab Mac OS X 10.8.5, sowie Gnome, KDE oder Unity unter Ubuntu oder Fedora. Es werden mindestens 2 GB RAM benötigt, 4 GB werden empfohlen. Der verfügbare Festplattenspeicherplatz sollte mindestens 1,5 GB betragen.

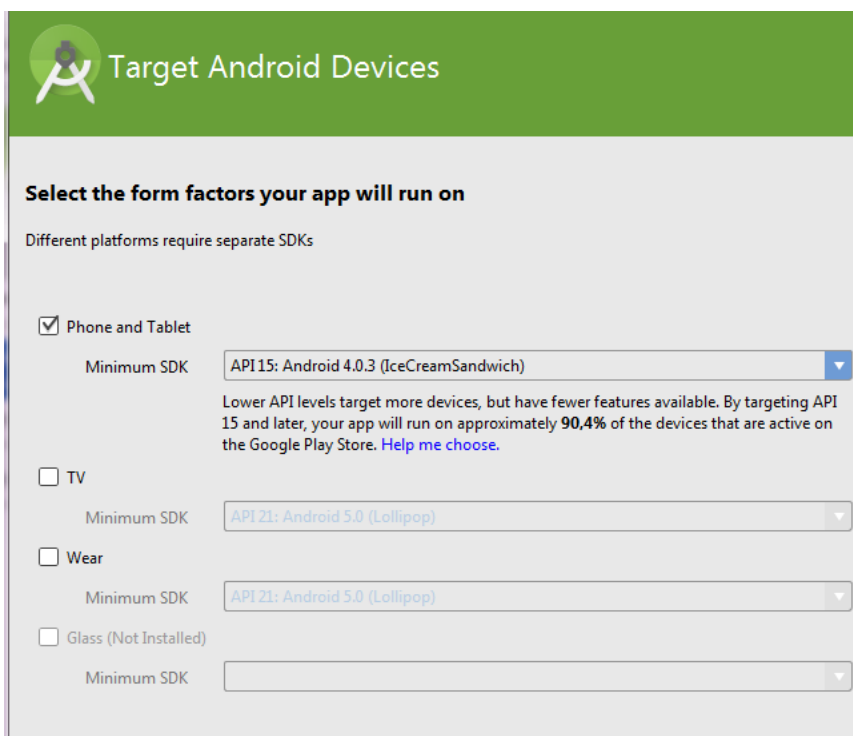
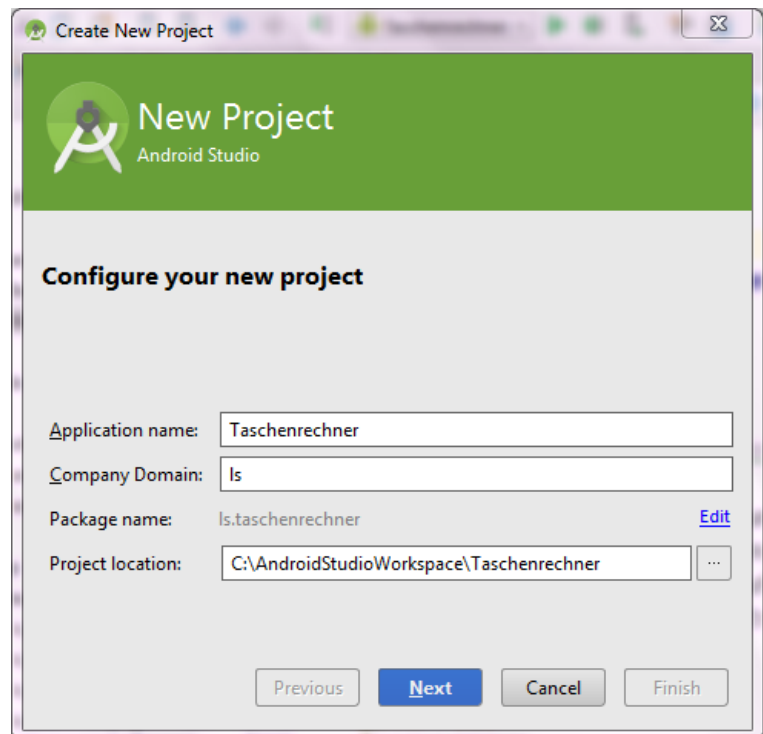
Nach dem Start des Android Studios öffnet sich das Willkommen-Fenster:



Anschließend wird das zu erstellende Projekt konfiguriert.

Wählen Sie einen Namen für die Applikation und die Domain aus (z.B. der Schulname). Diese beiden Informationen bestimmen darüber, wie später der Paketname der erstellten App lauten wird.

Wählen Sie außerdem hier das Arbeitsverzeichnis für das Projekt aus.



Im folgenden Fenster *Form Factors* werden die Zielgeräte der App und die minimalste unterstützte Android-Version ausgewählt.

Wählen Sie *Phone und Tablet* als Gerät und z.B. die Version 4.0 (Ice-CreamSandwich) als Minimum SDK aus.

Im nächsten Fenster *Add Activity to an Mobile* kann man vorgefertigte Activities auswählen. Eine Activity ist eine Art Template für ein Benutzerinterface, auf das die Elemente der graphischen Benutzeroberfläche gesetzt werden. Um Freiheit im Design zu haben, bietet es sich für die Anwendung im Unterricht an, eine *Blank Activity* zu wählen.

Im Fenster *Customize the Activity* werden nun die Namen der wichtigsten Dateien des Projektes festgelegt:

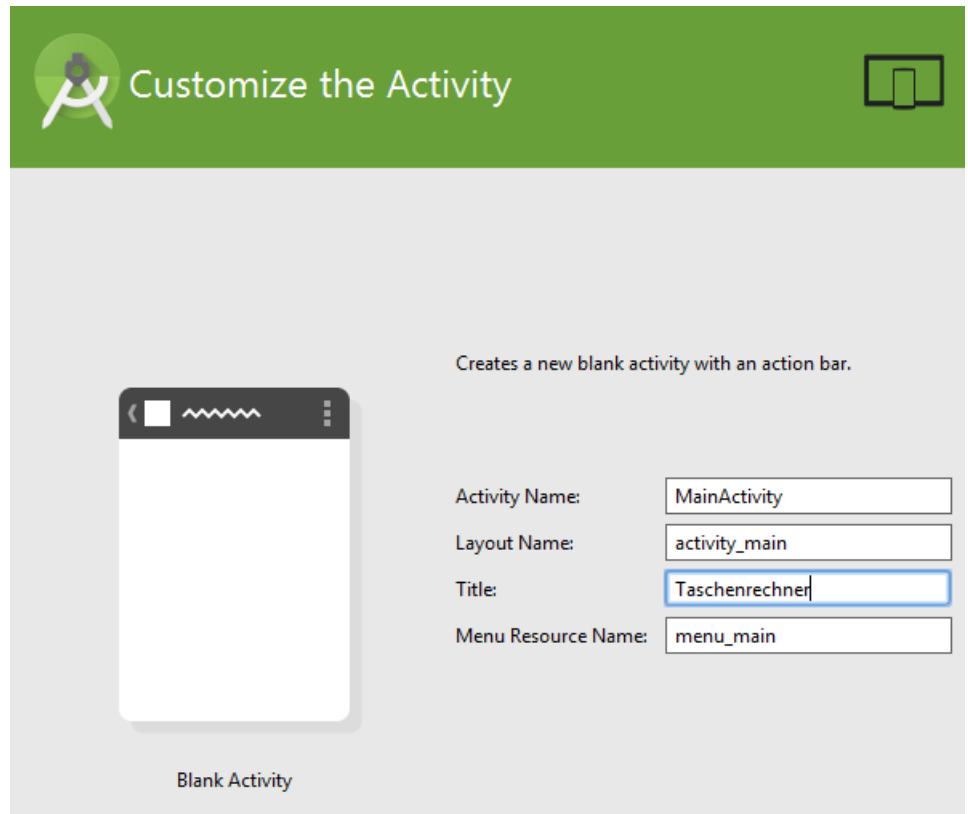
Activity Name:

Name der Java-Klasse welche die Ereignisse der GUI-Elemente steuert.

Layout Name:

Name der XML-Datei, die die GUI beschreibt.

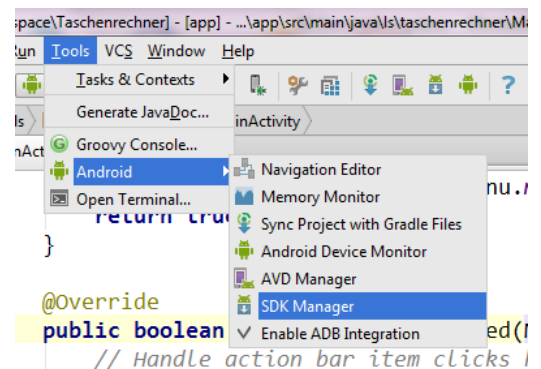
Für den Unterricht empfiehlt es sich, diese beiden Bezeichnungen zu belassen, damit im laufenden Unterricht alle Dateien der Schülerprojekte die gleichen Namen haben und jeder nachvollziehen welche Datei gemeint ist.



Der *Title* kann an den Projekttitel angepasst werden. Mit ihm wird die App dann auf der Android-Oberfläche angezeigt.

2.2 Aktualisierung des Android SDK

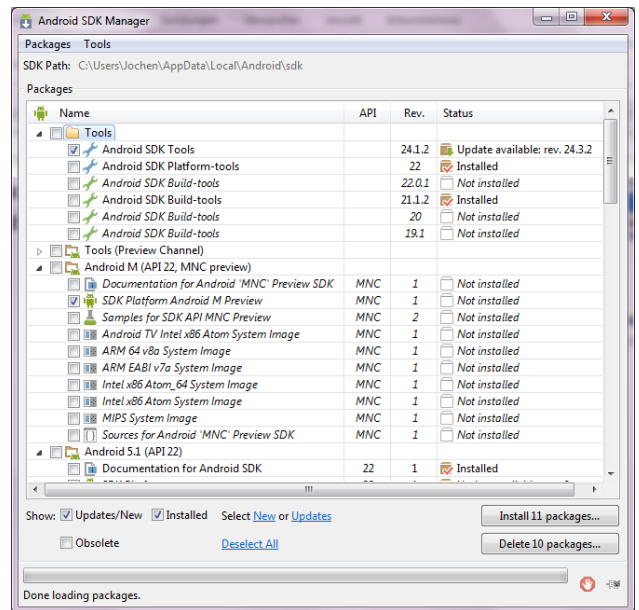
Das *Android Software Developer Kit* wird mit dem Android Studio mitinstalliert, man sollte aber unbedingt vor dem Arbeitsbeginn das Android SDK aktualisieren. Dazu starten Sie den SDK Manager über die Symbolleiste oder im Menü über *Tools* → *Android* → *SDK Manager*.



Der SDK Manager zeigt die Pakete an die aktuell installiert sind. Es empfiehlt sich vor der ersten Verwendung, die fehlenden / nicht aktuellen Pakete über den Button *install xx packages...* nachzuinstallieren. Dieser Vorgang kann bis zu einer halben Stunde dauern!

Folgende Pakete werden in mindestens einer Versionsnummer benötigt:

- *Android SDK Tools*
- *Android SDK Platform Tools*
- *Android SDK Build-Tools* (unterstützen bei der Erstellung der Apps)
- *Google API* (bieten nützliche Klassen an)
- *Android Support Library* (bieten nützliche Klassen an)
- *Google USB-Driver* (wird benötigt, um ein über USB angebundenes Tablet von Android Studio aus anzusteuern)



Falls nach der Installation noch Pakete zum Installieren übrig sind (d.h. der Button bietet noch Installationen an), dann starten Sie den Installationsprozess noch einmal. Dies liegt daran, dass manche Pakete andere Lizenzbedingungen benötigen und daher separat voneinander installiert werden müssen.

Danach muss noch der Emulator und/oder das Tablet eingerichtet werden, um die zu erstellenden Apps testen zu können. Es bieten sich drei Möglichkeiten an:

- Die Apps werden in einen Software-Emulator gestartet (siehe Kapitel 2.3 und 4.1).
- Die Apps werden auf dem angeschlossenen Tablet im Entwicklermodus getestet (siehe Kapitel 2.4 und 4.2).
- Die Apps werden als apk (Android Package) gespeichert und auf dem Tablet installiert. Dies benötigt keine weiteren Vorbereitungen (siehe Kapitel 4.3)

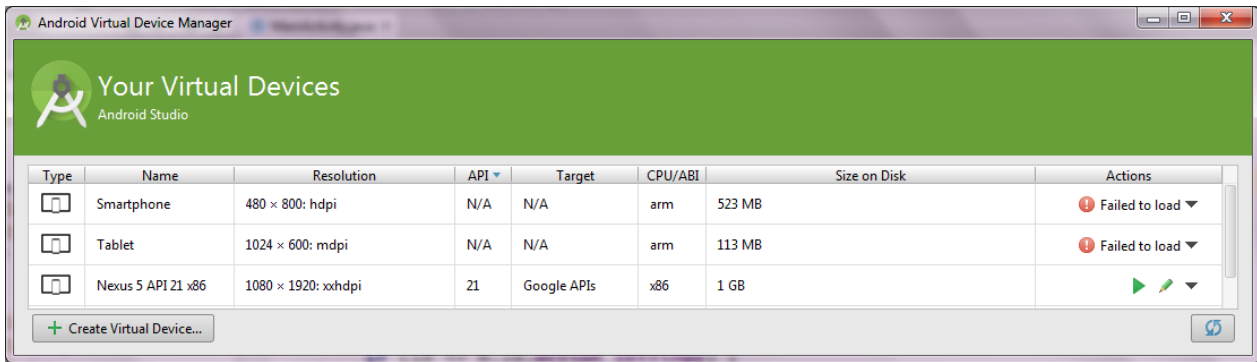
Der Vorteil des Emulators besteht darin, dass man kein Tablet dabei haben muss um Apps zu testen. Der Nachteil ist seine Trägheit (Das Starten kann mehrere Minuten dauern). Vorteil der zweiten Variante ist, dass die App beim Ausprobieren gleich durch das Android Studio auf dem Tablet installiert wird. Die dritte Variante ist zwar die umständlichste, benötigt dafür aber keine Vorbereitungen auf dem Tablet oder dem Arbeits-PC.

Es bietet sich an, beide Möglichkeiten einzurichten, damit man je nach Bedarf auf beide Varianten zurückgreifen kann.

2.3 Einrichtung des Emulators über den AVD-Manager

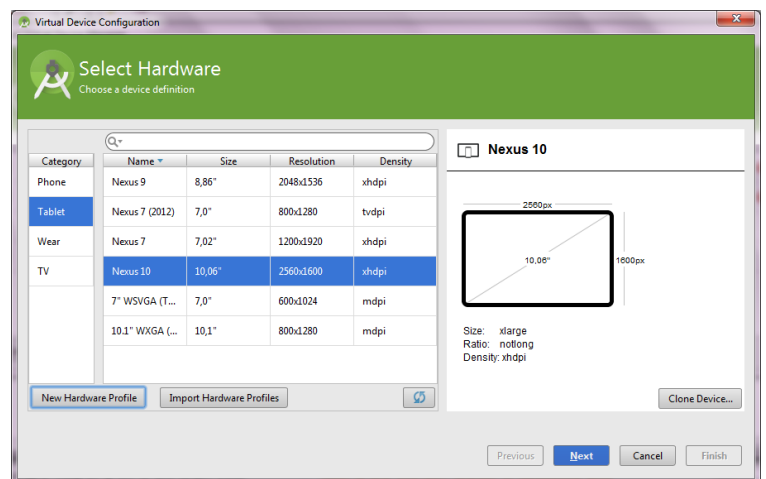


Ein Software-Emulator ist ein Programm in dem ein sogenanntes AVD (Android Virtual Device) gestartet wird. Dies sind virtuelle Geräte, die in Android Studio über den AVD-Manager verwaltet werden. Er wird über *Tools* → *Android* → *AVD Manager* gestartet.



Hier lässt sich ein bereits voreingestelltes AVD auswählen oder ein neues erstellen (Button *Create Virtual Device*). Für erste Versuche kann man das vorgefertigte AVD *Tablet* verwenden. Es empfiehlt sich aber ein eigenes AVD zu erstellen:

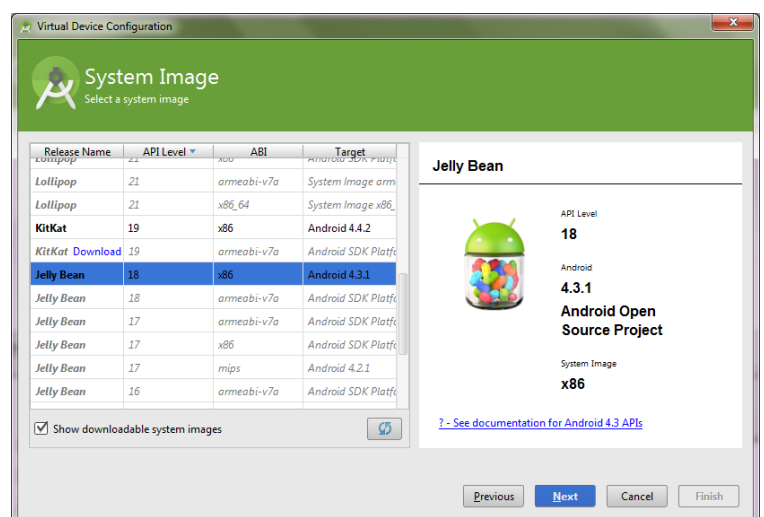
Wählen Sie eine Hardware-Emulation aus, die zur Größe des Tablets passt. Bei Bedarf lässt sich über *New Hardware Profile* ein neues Profil erstellen.



Im nächsten Fenster wählen Sie die Version des Android-Betriebssystems des AVD aus. Bleiben Sie bei der Version für die Sie im Kapitel 2.2 die Pakete installiert haben.

Im Punkt *ABI (Application Binary Interface)* wird die emulierte Prozessorumgebung des gewählten AVD angezeigt. Es werden drei Architekturen unterstützt:

- ARM
- Intel x86
- mips

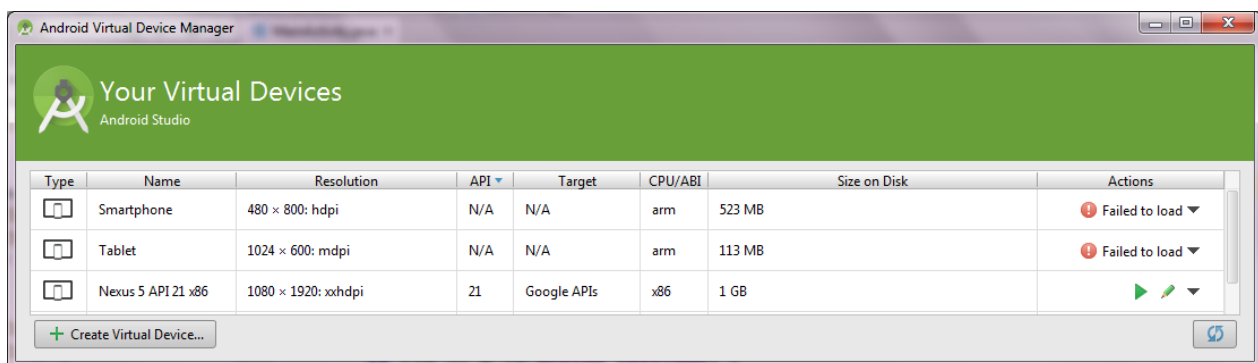
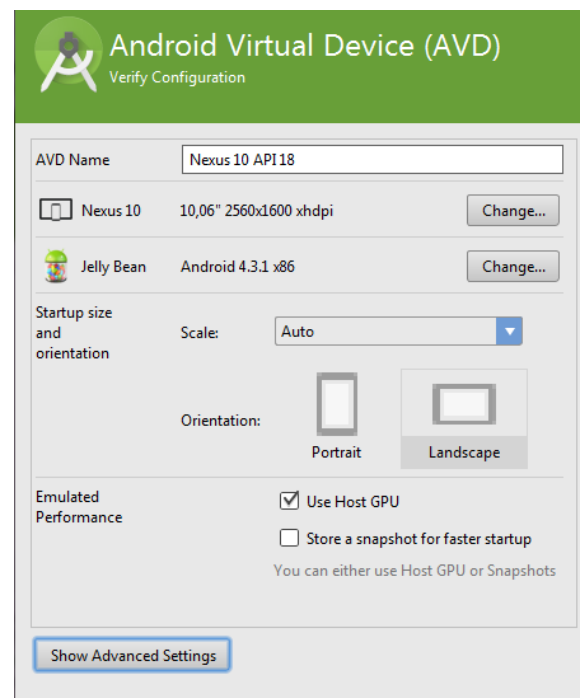


Sehr viele Mobilgeräte besitzen Prozessoren der ARM-Architekturfamilie, daher ist dies die empfohlene Einstellung.

Sollten Sie die Einstellung x86 wählen, so kann es sein, dass man für das reibungslose Funktionieren des AVD noch einen Hardwarebeschleuniger installieren muss. Dieser kann im SDK Manager nachinstalliert werden. (*Intel x86 Emulator Accelerator (HAXM)*)

Sollte die gewünschte Version nicht dabei oder ausgegraut sein, verlassen Sie den AVD Manager, gehen Sie in den SDK Manager und installieren diese nach. (z.B. *Intel x86 Atom System Image* oder *ARM EABI v7a System Image*).

Im abschließenden Fenster bestätigen Sie die Konfiguration. Über den Button *Show Advanced Settings* lassen sich Einstellungen zum Speicher des AVD tätigen.



Anschließend erscheint das neue virtuelle Gerät in der Liste und kann über das Bleistift-Symbol bearbeitet und über das Pfeil-Symbol gestartet werden. Der Start des AVD kann mehrere Minuten dauern.

Falls Sie ein AVD mit x86-Prozessorarchitektur und Hardwarebeschleuniger HAXM gewählt haben, und beim Starten folgende Fehlermeldung erhalten "*HAX is not working and emulator runs in emulation mode*", dann müssen Sie in den o.g. *Advanced Settings* den RAM-Speicher des Gerätes heruntersetzen (512 MB sind meist ausreichend).

Im Emulator läuft nun das gewählte AVD. Er lässt sich nun wie ein Mobilgerät bedienen. Der Mauszeiger ersetzt dabei den Finger. Zum Start schiebt man das Schloss-Symbol aus dem Kreis. Strg + F11 ändert die Ausrichtung von Landscape zu Portrait und zurück.



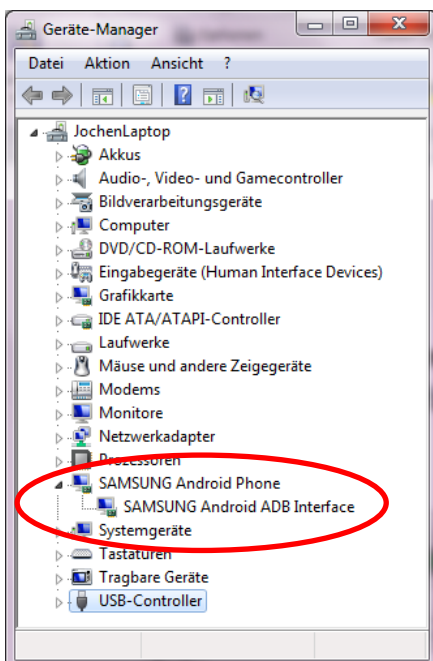
2.4 Einrichtung des Tablets

Im Originalzustand kann ein Android-Tablet nur Apps aus dem App-Store starten. Um das Tablet als Endgerät für die Ausführung der zu testenden App vorzubereiten, müssen dort die Entwickleroptionen aktiviert werden. Diese sind standardmäßig ausgeschaltet und müssen manuell aktiviert werden:

Wählen Sie in den Einstellungen des Tablets die Option *Allgemein* → *Geräteinformationen* (je nach Android-Version). Die Entwickleroptionen bekommt man zu sehen in dem man sieben Mal schnell hintereinander auf die Buildnummer klickt. Im Menü erscheinen nun die Entwickleroptionen, die man nun aktivieren kann. Tätigen Sie dann folgende Einstellungen:

- Einschalten des *USB-Debugging*. Diese Einstellung sorgt dafür, dass eine Anwendung (hier die zu erstellende App) auf Funktionen des Betriebssystems zugreifen kann. Dies ist unbedingte Voraussetzung dafür, dass man das Tablet als Testgerät für Apps benutzen kann. Dieser Vorgang kann unter Umständen sicherheitskritisch sein, so dass der USB-Debugging-Modus nach Ende dieser Unterrichtseinheit unbedingt wieder ausgeschaltet werden sollte.
- *Wach bleiben* sollte aktiviert sein. Dies verhindert, dass das Tablet beim Aufspielen der App in den Standby-Modus geht.

Für die weiteren Schritte muss das Tablet nun über ein USB-Kabel am PC angeschlossen sein.



Die Kommunikation mit angeschlossenen Tablets wird von der *Android Debug Bridge (ADB)* erledigt. Über die ADB können wir unsere selbst erstellten Android Apps auf ein Android Gerät übertragen und installieren.

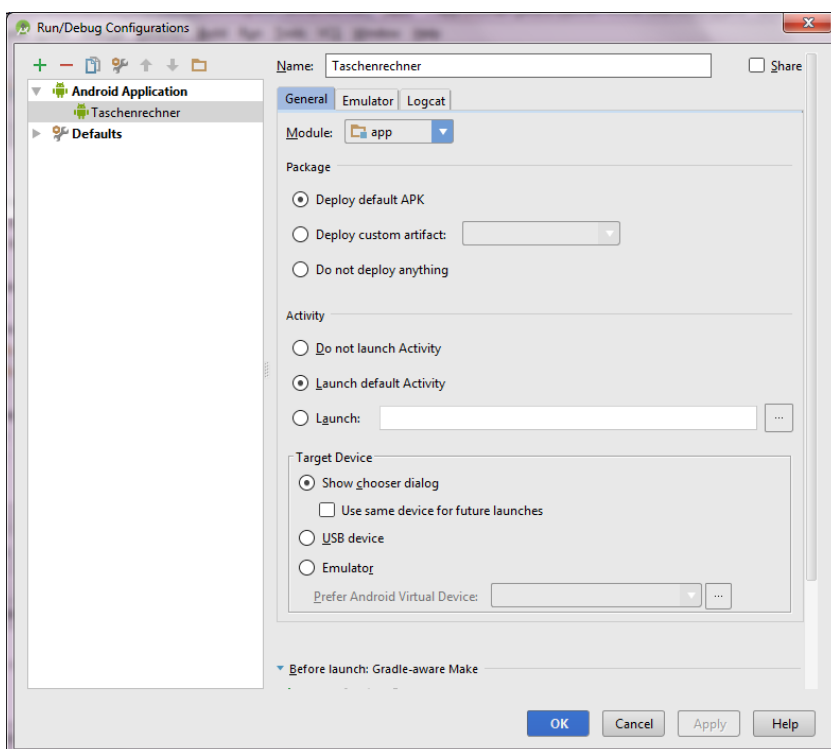
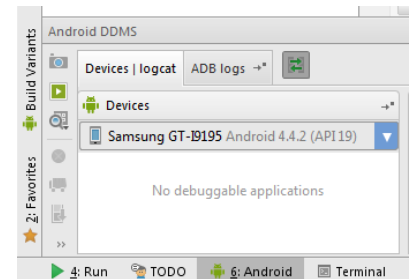
Der PC muss das Tablet als Android ADB Interface erkennen. Dies kann man im Geräte-Manager überprüfen (*System-Steuerung* → *Geräte-Manager*). Dort sollte das Mobilgerät mit der Kennzeichnung "ADB Device" (oder ähnlich je nach Gerät) eingetragen sein.

Sollte dies nicht der Fall sein, muss der ADB-Treiber installiert werden. Diesen installieren Sie entweder per Rechtsklick auf das Gerät oder über die Android-Developer-Website (Linkliste).

Das Installationsprogramm fragt nach der Modellnummer. Diese findet man auf der Rückseite des Tablets oder in den *Einstellungen* → *Allgemein* → *Modellnummer* des Tablets.

Bei Erfolg fragt das Tablet, ob das Debugging für diesen Rechner zugelassen werden soll. Beantworten Sie mit Ja.

In Android Studio muss nun der ADB-Modus über das Menü *Tools* → *Android* → *Enable ADB-Integration* eingeschaltet werden. Im unteren Fenster *Android* wird das Gerät angezeigt und ist damit bereit, in Android Studio erstellte Apps auszuführen.



Im Menü *Run* → *Edit Configurations* können je nach Bedarf Einstellungen bezüglich des Zusammenspiels zwischen Android Studio und dem Tablet eingestellt werden. (Die dargestellten Einstellungen sollten eingestellt werden)

Package: Deploy default APK

Eine APK (Android Package) ist das Standardformat mit dem Android eine App intern ablegt.

Activity: Launch default Activity

Startet die MainActivity des aktuellen Projektes.

Target Device: Show Choose Dialog

Vor dem Starten wird man gefragt welches Gerät (Tablet oder AVD) benutzt werden soll.

3 Erste Schritte mit Android Studio

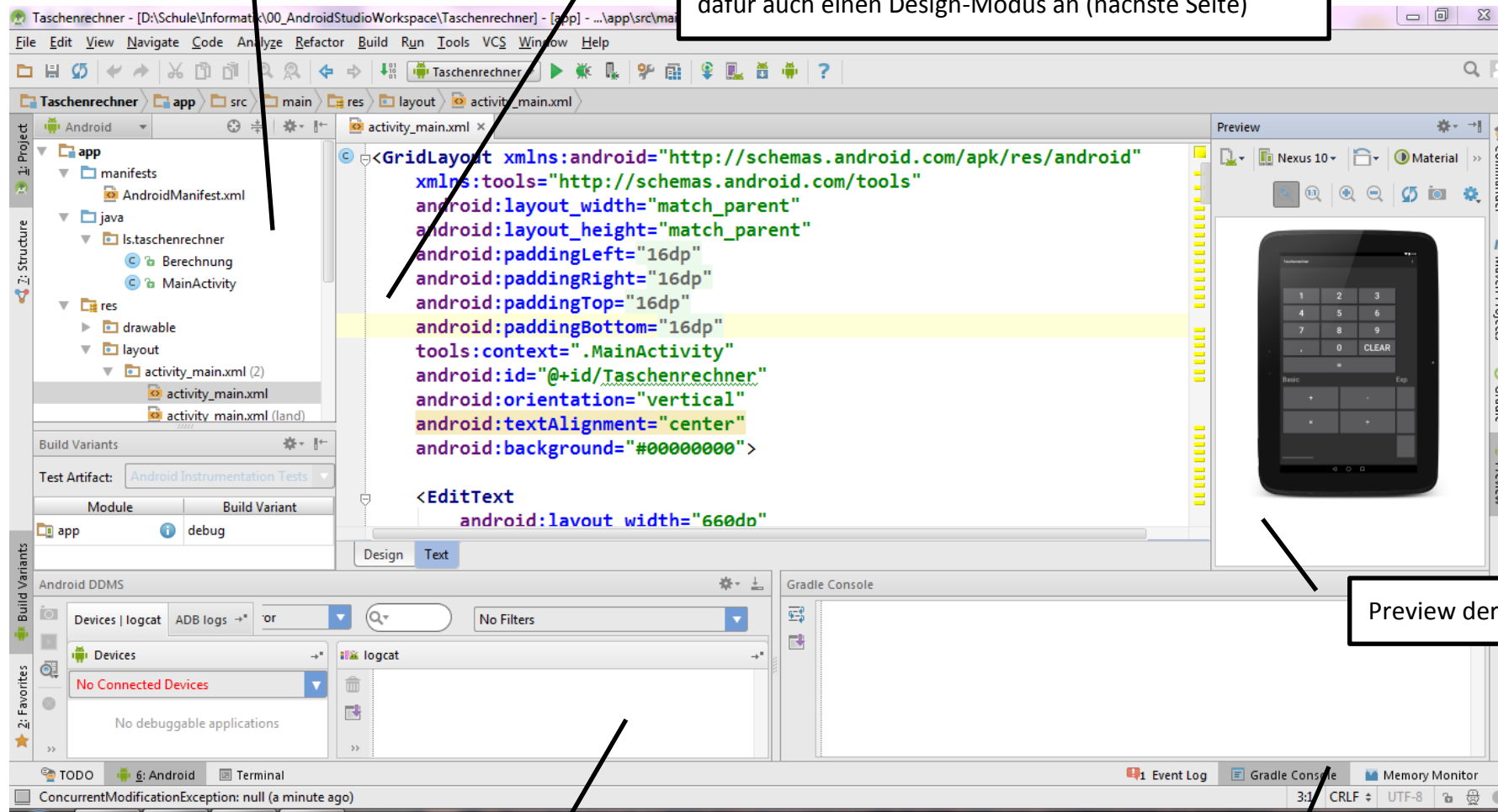
3.1 Die Bedienoberfläche von Android Studio

Die Bedienoberfläche des Android Studios ähnelt der Oberfläche anderer Entwicklungsumgebungen. Wer mit einer IDE (z.B. Eclipse) vertraut ist, wird sich auch hier leicht zurechtfinden.

Siehe Abbildungen nächste Seiten:

Ansicht des Projektbaums

Quellcode-Ansicht:
Hier wird beispielhaft eine XML-Datei zur Oberflächen-
gestaltung im Textmodus gezeigt. Android Studio bietet
dafür auch einen Design-Modus an (nächste Seite)



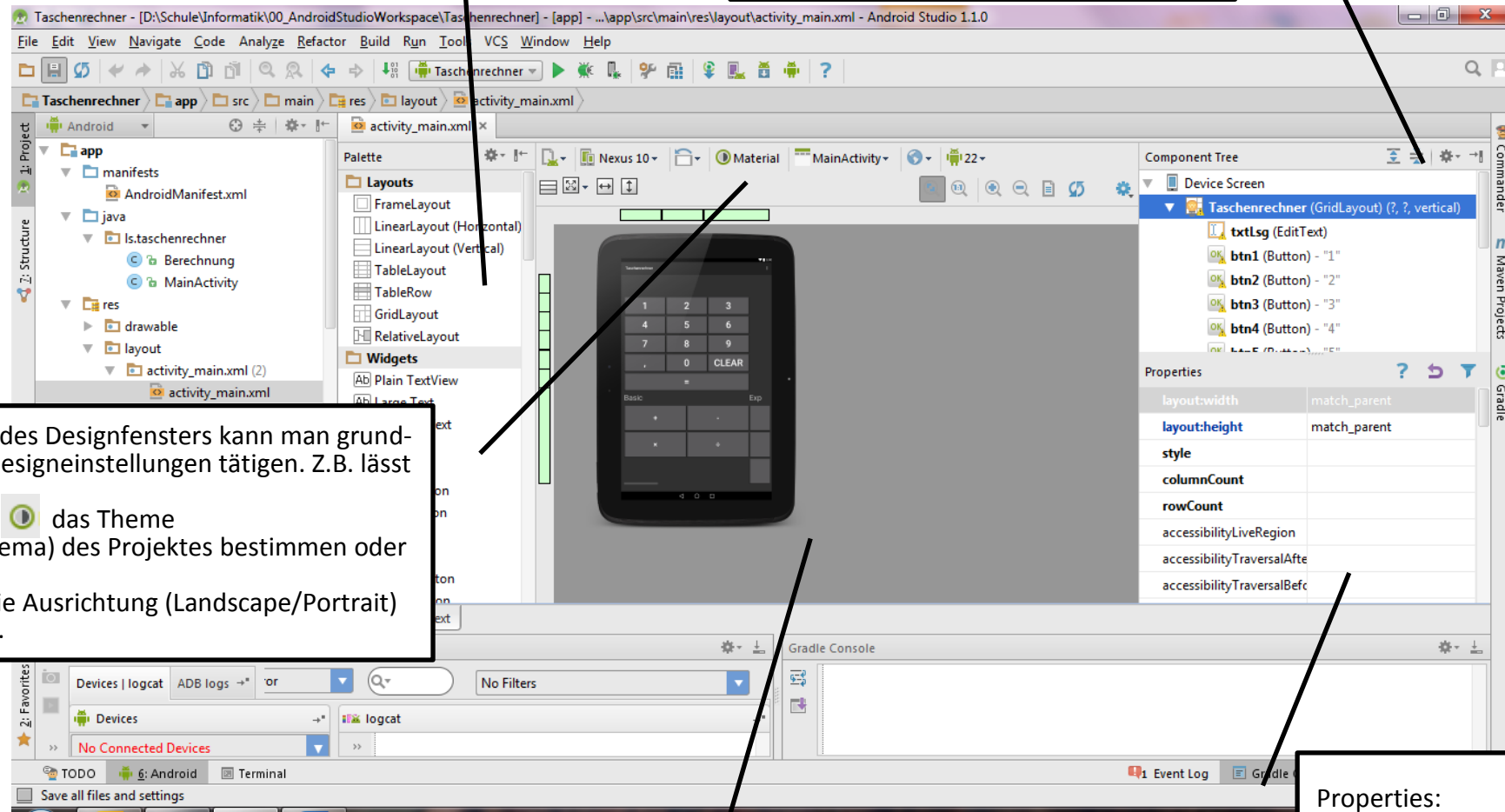
Preview der Benutzeroberfläche.

Im unteren Bereich lassen sich verschiedene Konsolenfenster für Informationen über Fehlermeldungen, Speicherverwaltung, Tablet-Anbindung u.ä. einblenden.

Für jeden Bildschirmbereich kann man über diese Schaltflächen das entsprechende Fenster ein- oder ausblenden (markiert: ein; nicht markiert: aus)

Die Palette hält diverse Design-Objekte bereit, die der Activity hinzugefügt werden können.

Component Tree:
Hier werden die bereits hinzugefügten GUI-Objekte (mit Angabe der Klasse) in einer hierarchischen Baumstruktur angezeigt.



WYSIWYG-Designfenster:
Hier werden die GUI-Objekte auf der Activity mit der Maus platziert.

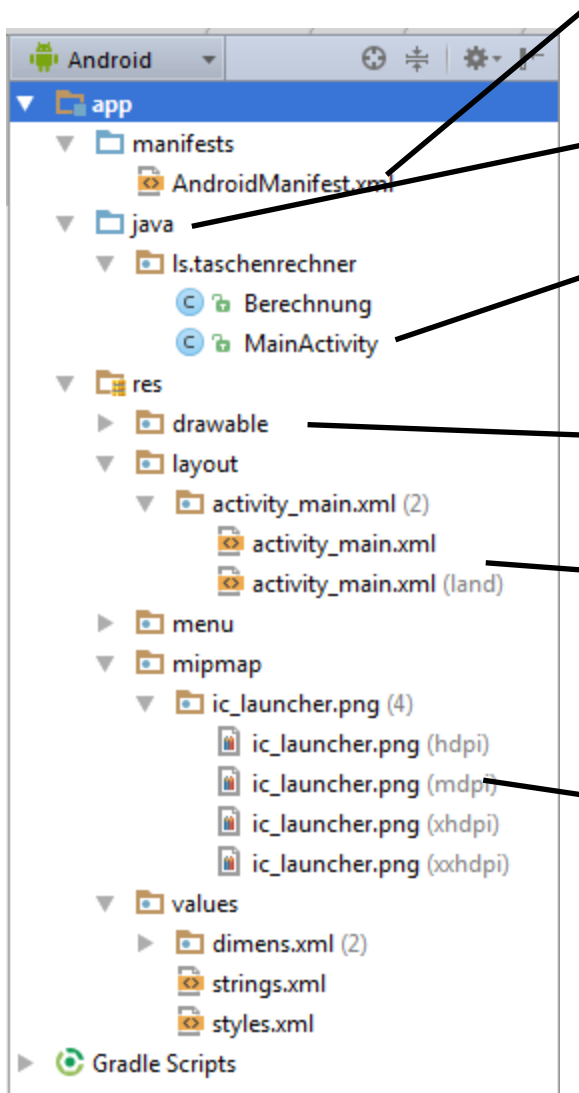
Properties:
Hier werden zum ausgewählten GUI-Objekt die verfügbaren Attribute angezeigt.

3.2 Grundgerüst eines Android-Projektes

Ein Android-Projekt besitzt eine vorgegebene Ordnerstruktur, die beim Erstellen des Projektes zu Beginn einer Sitzung automatisch erzeugt wird. Sie wird nachfolgend erläutert.

Diese Projektstruktur ist nicht ganz identisch mit der physischen Ordnerstruktur auf dem Datenträger.

Dateien, die Projekt-Einstellungen oder die Benutzeroberfläche beschreiben, liegen im XML-Format vor. Kenntnisse im grundsätzlichen Umgang mit XML-Dateien sind bei der App-Entwicklung unter Android empfehlenswert (Linkliste).



AndroidManifest.xml:
Grundsätzliche Einstellungen zum Projekt.

Ordner java:
Hier liegen die Pakete mit den Java-Klassen

MainActivity.java:
(Beschreibung siehe unten)

drawable
In diesem Ordner werden die Icons gesammelt, die auf der GUI verwendet werden.

Activity_main.xml
(Beschreibung siehe unten)

ic_launcher.png
Hier wird das Icon für die Android-Oberfläche gespeichert. Es sollte für mehrere Auflösungen zur Verfügung gestellt werden.

Die beiden zentralen Dateien sind:

Activity_main.xml

In dieser XML-Datei werden die GUI-Elemente als XML-Tags eingetragen. Auch die Attribute und deren Werte finden sich hier wieder. Diese Datei ist somit der zentrale Ort, der das Layout der App festlegt. Diese Datei wird beim Start automatisch erzeugt. Die Einträge für die GUI-Elemente werden automatisch

hinzugefügt, wenn man im Designmodus die GUI-**Objekte** auf die Activity zieht. Trotzdem kann es nötig sein, bei komplexeren Projekten auch direkt in dieser Datei zu arbeiten.

Im Projektbaum finden sich zwei *Activity_main.xml*-Dateien:

Mit Zusatz (*land*): Hier wird das Design für die Horizontalansicht festgelegt

Ohne Zusatz: Hier wird das Design für die Vertikalansicht festgelegt.

Dies bedeutet, dass man für beide Bildschirmorientierungen völlig unterschiedliche Designs festlegen kann. Unsere App kann also auf das Kippen des Tablets reagieren. Der Nachteil ist, dass man manche Aufgaben (z.B. Hinzufügen eines neuen Buttons) zweimal erledigen muss. Deshalb bietet es sich an, zuerst die Vertikalansicht fertig zu stellen und erst dann die Horizontalansicht zu aktivieren.

MainActivity.java

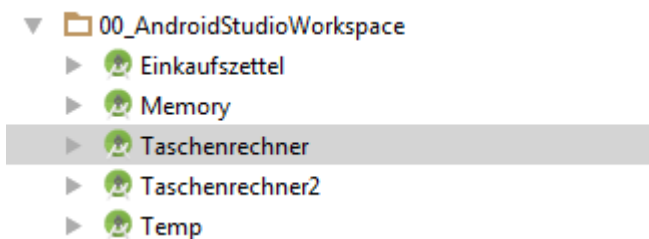
Diese Java-Klasse repräsentiert die MainActivity, die unsere App steuert. Sie ist somit die zentrale Datei, was die Steuerung der App über die GUI-Elemente angeht. Sie erbt Funktionalität von anderen Activity-Klassen, je nachdem was bei der Projekterstellung als voreingestellte Activity ausgewählt wurde.


Diese Klasse besitzt die Methode *onCreate()*, die beim Starten der App ausgeführt wird. In ihr wird zum einen ein Exemplar der Superklasse erzeugt, zum anderen die *Activity_main.xml* als Layout-Datei eingefügt. Diese wird im weiteren Verlauf durch die Klasse *R* repräsentiert.

Ähnlich einer GUI-Klasse bei der Swing-Programmierung werden hier auch die GUI-Objekte deklariert und deren Eventhandler hinzugefügt.

3.3 Laden eines bestehenden Projektes

Ein bestehendes Android-Studio-Projekt wird über *File* → *Open* geöffnet



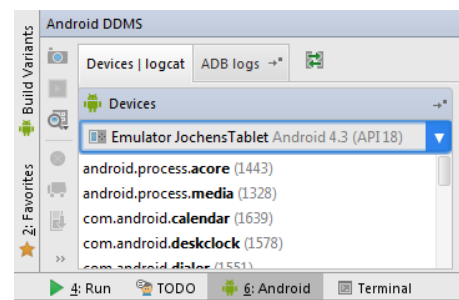
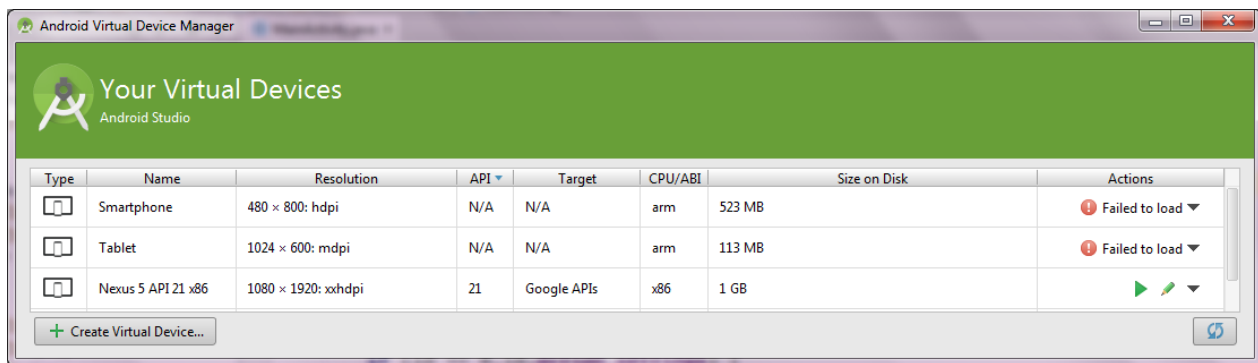
Im anschließenden Öffnen-Dialog wählt man das Projektverzeichnis aus, welches durch das Android Studio-Icon  gekennzeichnet ist.

Ein bestehendes Eclipse-Projekt lässt sich über *File* → *Import Project* einladen.

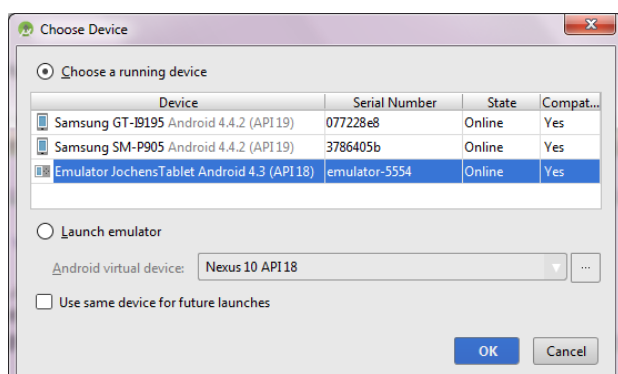
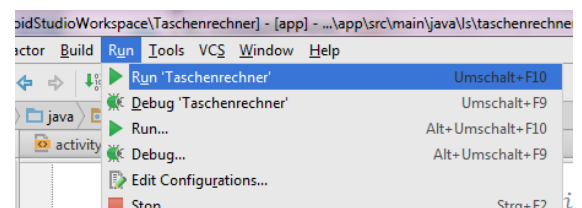
4 Testen einer App

4.1 Testen einer App im Emulator (Android Virtual Device)

Ein AVD wird über *Tools* → *Android* → *AVD Manager* gestartet. Im anschließenden Fenster wird ein AVD ausgewählt und über den grünen Pfeil in der Spalte *Actions* gestartet. Der Start einer AVD kann mehrere Minuten benötigen.



Wenn die AVD läuft, ist sie im Fenster *Android* unten als Device sichtbar. Dann kann die App über *Run* → *Run ...* gestartet werden. Der Vorgang kann ein Weilchen dauern.

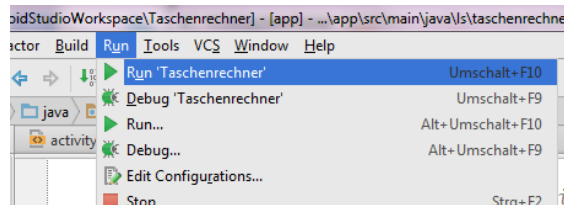
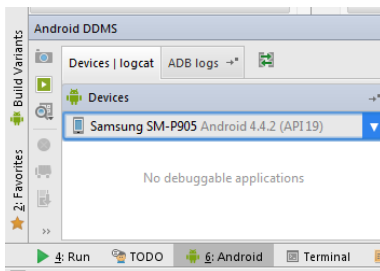


Im *Choose Device*-Dialog sind alle Geräte (AVD's aber auch reale angeschlossene Geräte) sichtbar. Wählen Sie hier Ihre AVD aus.

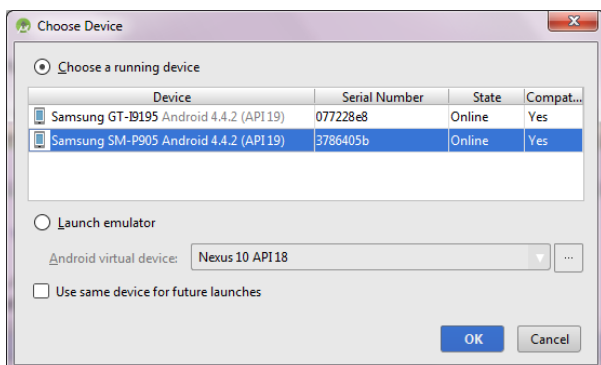
Etwaige Fehler im Programmablauf werden in Android Studio im unteren Fensterbereich *Messages* angezeigt.

4.2 Testen einer App direkt auf dem Tablet

Das Tablet muss gemäß obiger Beschreibung eingerichtet sein und über ein USB-Kabel mit dem PC verbunden werden.



Das Gerät ist dann im Fenster *Android* unten als Device sichtbar. Dann kann die App über *Run* → *Run ...* gestartet werden. Der Vorgang kann ein Weilchen dauern.



Im *Choose Device*-Dialog sind alle Geräte (reale angeschlossene Geräte, aber auch AVD's) sichtbar. Wählen Sie hier Ihr Tablet aus.

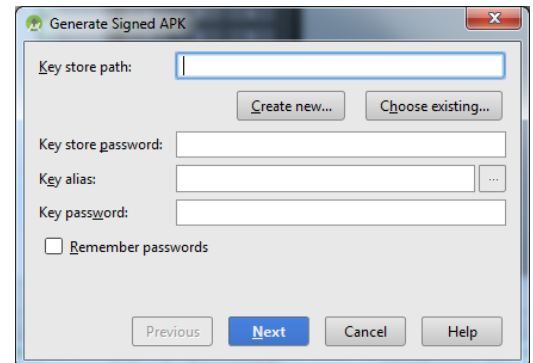
Etwaige Fehler im Programmablauf werden in Android Studio im unteren Fensterbereich *Messages* angezeigt.

4.3 Testen einer App mittels Deployment

Deployment bedeutet, dass die in Android Studio erstellte App als apk (Android Package) gespeichert und direkt auf dem Tablet installiert wird. Dies kommt dem offiziellen Vertriebsweg nahe, nämlich dem Download aus dem Playstore. Diesen extremen Umweg muss man hier nicht gehen, aber man muss eine apk erzeugen und signieren. Eine apk ist eine ausführbare Installations-Datei, die auf Android-Betriebssystemen lauffähig ist.

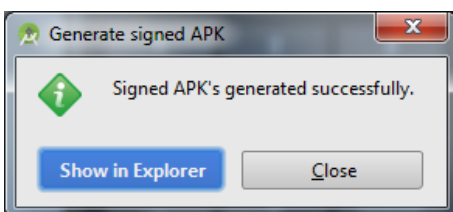
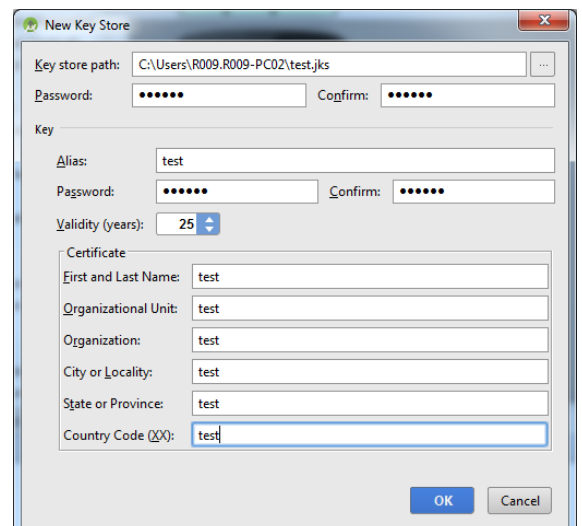
Wählen Sie im Menü *Build* → *Generate Signed apk...*

Über den Button *Create new...* legen Sie eine neue jks-Datei an (Java Keystore). Diese Datei ist ein Container für Zertifikate, private Schlüssel u.ä. In unserem Beispiel ist der Inhalt dieser Datei egal, da wir die App nur testen, aber nicht veröffentlichen wollen. Trotzdem muss diese Datei angelegt werden.



Im folgenden Dialog müssen alle Felder ausgefüllt werden, sei es nur mit Dummy-Werten. Diese Informationen wären wichtig, wollten wir die App im Google Playstore veröffentlichen. Das Passwort muss mindestens 6 Zeichen haben, ist aber für die weitere Vorgehensweise nicht relevant.

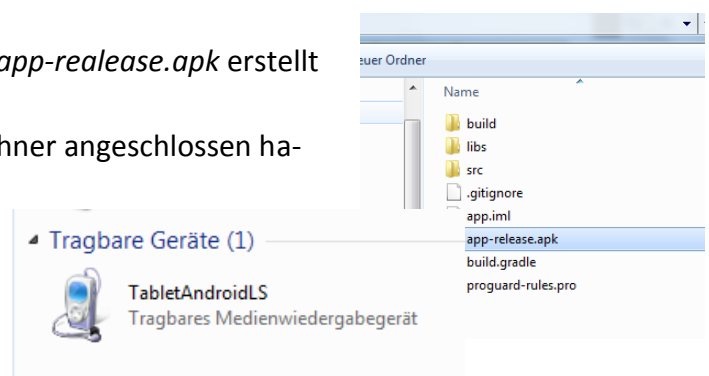
Mit *OK* wird die apk erstellt.



Wechseln Sie in das Verzeichnis, in dem die Datei *app-release.apk* erstellt wurde.

Wenn Sie Ihr Android-Tablet über USB an den Rechner angeschlossen haben, können Sie dies im Windows-Explorer sehen.

Kopieren Sie dann die apk-Datei auf das Tablet z.B. in das *Download* oder *Document*-Verzeichnis.

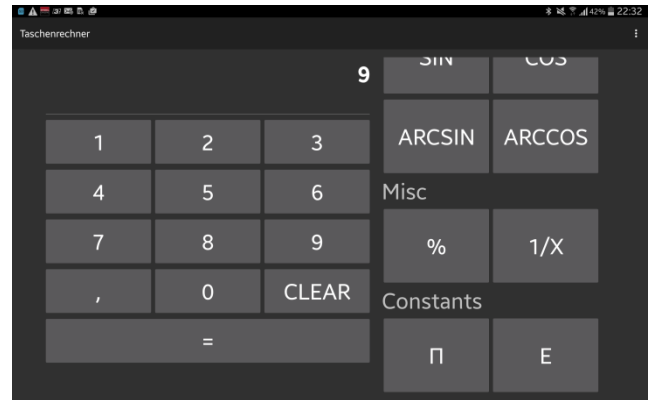
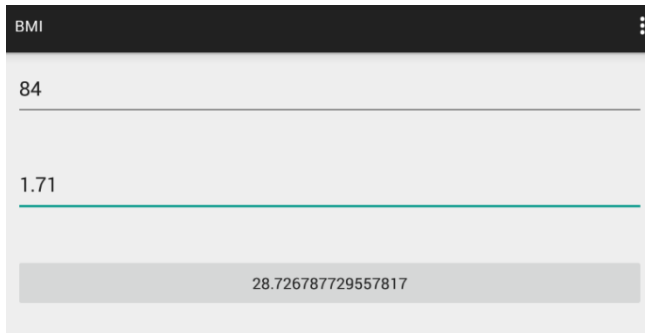


Klicken Sie auf dem Tablet die apk-Datei an und beantworten Sie die Sicherheitsabfragen mit *Ja* oder *Akzeptieren*. Dann können Sie die App testen.

5 Weitere Vorgehensweise

Die Unterrichtseinheiten *UE BMI-App* und *UE Taschenrechner-App* führen konkret durch die Erstellung von Apps.

Erstere behandelt ein simples Projekt zum Einstieg, was die wichtigsten Schritte zur fertigen App erklärt. Die zweite erläutert ein komplexeres Projekt mit Tablet-typischen Funktionen wie Kippen und Scrollen.



6 Linkliste

Download Java SDK

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Download Android Studio

<https://developer.android.com/sdk/index.html>

Download ADB-Treiber

<http://developer.android.com/tools/extras/oem-usb.html#Drivers>

XML-Kurse (Beispiele)

SelfXML (Autor: Julian Bart)

<http://www.selfxml.de/>

XML Tutorial #1 #Grundlagen #Einführung (Autor: JavaWebAndMore)

<https://www.youtube.com/watch?v=9gA0f8u26A4>

XML (Einführung) | Informatik Lernvideo (Autor: Lernvideos und Vorträge)

<https://www.youtube.com/watch?v=UHABhalsicA>

XML-Technologien (1) – Einführung (Autor: FredFreeEducation)

<https://www.youtube.com/watch?v=5G3SDjEcpAI>