Landesinstitut für Schulentwicklung

Thema:	**App-Entwicklung mit Android Studio**
Name der Autorin/ des Autors:	Jochen Pogrzeba,StR Max-Weber-Schule, Freiburg
Fach:	Informatik, Wirtschaftsinformatik
Klasse/Jahrgangsstufe:	JG1
Schulart:	Berufliches Gymnasium
Lehrplanbezug:	LPE 9: Objektorientierte Systementwicklung (Fortführung) LPE 9: Objektorientierte Systemanalyse und -entwicklung
Zeitumfang:	-
Betriebssystem/e:	Android
Apps:	-
Technische Settings:	Schüler-PC's mit Windows und Android Studio Ggf. Schüler-Tablets mit Android

Fachliche Ziele:

- Kenntnisse der Besonderheiten der App-Entwicklung.
- Umsetzung des ereignisorientierten Ansatzes der GUI-Entwicklung.
- Vertiefung der Kenntnisse in einer objektorientierten Programmiersprache.
- Vertiefung der Kenntnisse in einer strukturierenden Auszeichnungssprache.

Sonstige Ziele:

- Erweiterung der Sozialkompetenz durch zusammenführende Gruppen- oder Projektarbeit.
- Schulung der gestalterischen Kreativität.
- Förderung der Medienkompetenz.
- Evtl. Schulung der Präsentationskompetenz.
- Evtl. individuelle Förderung.

Diese Unterrichtseinheit ist nicht als direkt umsetzbares Material konzipiert. Vielmehr soll dieses Beispiel der Lehrerin/dem Lehrer die Möglichkeit geben, sich ggf. schnell in das Thema einzuarbeiten. Die direkte Umsetzung des Beispiels mit Schülern ist möglich, aber nicht Hauptziel. Diese Unterrichtseinheit soll den Lehrer motivieren, ein eigenes, evtl. profilbezogenes Beispiel, mit den Schülern umzusetzen.
Aus diesem Grunde wurde darauf verzichtet einen Verlaufsplan hinzuzufügen.

Landesinstitut für Schulentwicklung

App-Entwicklung mit Android Studio
- Entwicklung einer BMI-App –
(Ein Beispiel für Einsteiger)

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

Inhaltsverzeichnis:

jochen.pogrzeba@max-weber-schule.de

1 Vorbemerkungen

In dieser Unterrichtseinheit wird mit Hilfe der Entwicklungsumgebung *Android Studio* ein kleiner BMI-Rechner entwickelt, der auf Android-Tablets lauffähig ist.

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.

Als Programmierbeispiel wurde ein kleiner BMI-Rechner gewählt. Dieses Projekt ist bewusst simpel gewählt, da hiermit die grundlegenden Eigenschaften eines Android-Projektes dargestellt werden sollen. Es enthält keine typischen Tablet-Funktionalitäten wie Scrollen, keine Fachklasse und keine aufwändige GUI-Gestaltung. Für ein komplexeres Projekt siehe die Unterrichtseinheit *UE Taschenrechner-App.*

Hauptaugenmerk dieser Beschreibung liegt in der Vorstellung der grundsätzlichen Prinzipien der Android-Entwicklung. Hauptziel dieser Unterrichtseinheit ist es, der Lehrerin/dem Lehrer eine Anleitung mitzugeben, mit der er sich leicht und schnell in die Thematik einarbeiten kann. Eine direkte Umsetzung des vorgestellten Projektes im Unterricht ist zwar möglich, aber nicht primäres Ziel. Der Leser ist also eingeladen, eine Unterrichtseinheit mit einer eigenen Softwareidee durchführen.

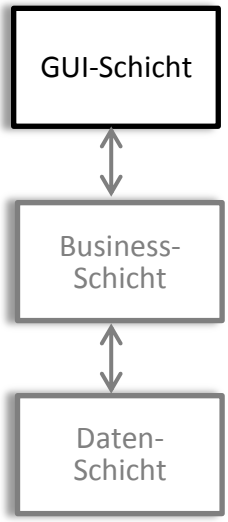
Das zugehörige Android Studio-Projekt finden Sie im Archiv *Projekt BMI.*

Bei der Lektüre dieser Unterrichtseinheit wird folgendes vorausgesetzt:
- Ein installiertes und eingerichtetes Android Studio ist verfügbar (siehe *Handreichung - Installation und Einrichtung Android Studio*)
- Die Grundstruktur eines Android-Projektes ist bekannt (siehe *Handreichung - Installation und Einrichtung Android Studio*)
- Kenntnisse in Java und idealerweise Swing.

Sinnvoll wären Kenntnisse in:
- Umgang mit einer IDE wie z.B. Eclipse.
- XML

2 Beschreibung der GUI-Schicht

activity_main.xml
MainActivity.java



Ein- und Ausgabe der Werte auf der Benutzeroberfläche und Weiterleitung an die Business-Schicht

Nicht vorhanden, da dieses Projekt nur die Funktionalität auf der GUI zeigen soll.

Nicht vorhanden, da eine persistente Datenhaltung hier nicht erforderlich ist.

Die beiden zentralen Dateien der GUI-Schicht 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 Bildschirmausrichtungen völlig unterschiedliche Designs festlegen kann. Die App kann also auf das Kippen des Tablets reagieren. Für dieses einfache Projekt wird nur die Vertikalansicht benutzt.

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 (siehe Kap. 4).

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

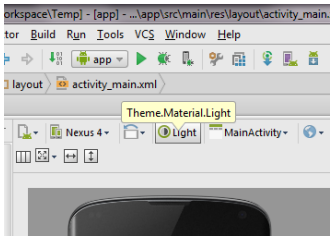
3 Erstellung der Benutzeroberfläche

3.1 Layout

Eine Activity wie diese kann nahezu vollständig mittels Drag-and-Drop im Designmodus erstellt werden. Dabei wird dann im Hintergrund die *activity_main.xml* mit den entsprechenden Tags gefüllt.
Bei komplexen Projekten wird man aber nicht darum herumkommen direkt in dieser Datei zu arbeiten, da bei GUIs mit vielen GUI-Objekten und Scroll-Bereichen die Handhabung im Design-Modus sehr unübersichtlich werden kann.

Bei einem neu erstellten Projekt sind bereits ein Layout und ein Textfeld mit dem Inhalt "Hello world" vorbereitet.

Hinweis: Sollte der Designmodus hier "Rendering Problems" anzeigen, so empfiehlt es sich das Grafikthema zu ändern. Ändern Sie diesem Falle *ProjectTheme* auf ein anderes Thema, z.B. *MaterialLight*.



in

Bevor die Elemente auf der Activity platziert werden können, muss ein Layout platziert werden. Ein Layout kann man sich wie einen Teppichboden vorstellen, auf dem Möbelstücke nach bestimmten Mustern angeordnet werden. Das voreingestellte Layout ist das *RelativeLayout*, hier werden die GUI-Objekte relativ zu der Position der anderen Elemente platziert (ähnlich wie HTML ohne css).

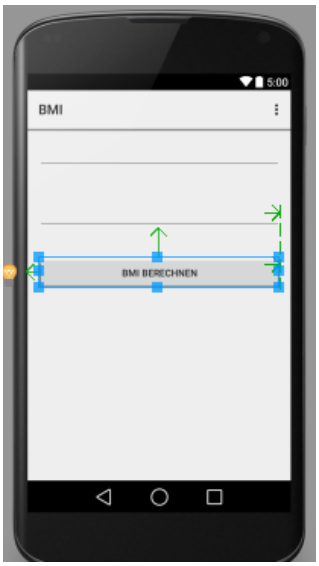
Für komplexere Projekte empfiehlt sich ein leichter handhabbares Layout wie das *GridLayout*, hier wird allerdings das *RelativeLayout* beibehalten.

In der *activity_main.xml* wird nun das *RelativeLayout* als XML-Tag angezeigt. Zwischen `<RelativeLayout… >` und `</RelativeLayout>` werden dann später die XML-Tags für die anderen GUI-Objekte stehen.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">
```

Ein GUI-Objekt lässt sich nun im Designmodus per Drag-und-Drop in das *RelativeLayout* platzieren.

3.2 Platzierung der GUI-Objekte



Auf die Activity werden nun zwei Textfelder (Text Fields Number Decimal) und ein Button gesetzt. Die Ausgabe des BMI wird später auf dem Button gezeigt.

Im Code der *main_activity.xml* ist bei der Höhe und Breite der Wert *wrap_content* eingetragen. Dies sorgt dafür, dass das GUI-Objekt genauso hoch/breit ist, wie es Platz hat.

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:ems="10"
    android:id="@+id/txtGewicht"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
```

Wichtig ist vor allem der Name des Objektes. Vergeben Sie bei *id* (im Properties-Bereich) oder `android:id="@+id/txtNameDesObjektes"` (in der *activity_main.xml)* einen aussagekräftigen Namen.
z.B. *txtGewicht, txtGroesse* und *btnBMI*

4 Programmierung in der 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 wird beim Erstellen des Projektes mit einigen vorbereiteten Methoden bereitgestellt.

4.1 Die MainActivity im Ausgangszustand

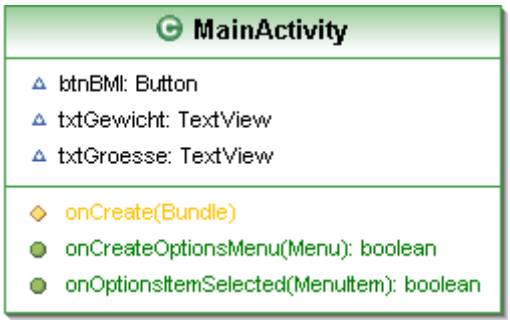
Die Klasse *MainActivity* erbt von der Klasse *ActionBarActivity*.
Die Methoden *onCreateOptionsMenu()* und *onOptionsItemSelected()* werden nur benötigt, wenn die App ein Menü bekommen soll und werden deshalb hier nicht weiter behandelt.

Die Methode *onCreate()* wird ausgeführt, wenn die App ein Exemplar der *MainActivity* erstellt, soll heißen, wenn die App gestartet wird. Sie führt folgendes durch:

- Erstellung eines Exemplars der Superklasse *ActionBarActivity.*
- Der Klasse *R* wird die Datei *activity_main.xml* als Attribut hinzugefügt. Die Klasse *R* repräsentiert somit die dort erstellte GUI. Die GUI-Objekte wie Buttons, Textfelder usw. sind somit Attribute dieser Klasse *R* und können nun in der *MainActivity* verwendet werden.
- Die GUI-Objekte wie Buttons, Textfelder usw. werden registriert und mit einem Eventhandler ausgestattet der Ereignisse wie Klicken o.ä. verarbeitet.

Der Methode *onCreate()* wird beim Start vom System ein Bundle-Objekt mitgegeben. In ihm können Zustände einer App gespeichert werden, was aber hier nicht benötigt wird.

4.2 Die MainActivity für die BMI-App



Wir haben der Klasse drei neue Attribute spendiert. Sie repräsentieren die drei GUI-Objekte.
Sie werden hier aufgeführt, da die Methode der inneren Klasse des Eventhandler diese Objekte benötigt.

Die neue Funktionalität wird der Methode *onCreate()* hinzugefügt.

Die Methode *onCreate()*: (Die neuen Inhalte sind fett gedruckt)

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    txtGewicht = (TextView) findViewById(R.id.txtGewicht);
    txtGroesse = (TextView) findViewById(R.id.txtGroesse);
    btnBMI = (Button) findViewById(R.id.btnBMI);

    btnBMI.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v)
      {
          double gewicht = Double.parseDouble(txtGewicht.getText().toString());
          double groesse = Double.parseDouble(txtGroesse.getText().toString());

          double bmi = gewicht/(groesse*groesse);

          btnBMI.setText(String.valueOf(bmi));
      }
    });
}
```

Registrierung der GUI-Objekte

ActionListener für den Button *btnBMI*

Eingegebene Werte erfassen

BMI berechnen

Ergebnis auf die Schaltfläche des Buttons schreiben

Registrierung der GUI-Objekte:
Hier werden die GUI-Objekte mit Hilfe der Klasse *R* erstellt. Ähnlich wie bei JavaScript werden die in der Klasse *R* repräsentierten GUI-Elemente über eine Methode *findViewById* als GUI-Objekt gespeichert. Ein Cast übernimmt die Umwandlung der allgemeinen GUI-Elemente in ein spezielles GUI-Objekt.

ActionListener für den Button *btnBMI:*
Ein ActionListener "horcht", ob ein Ereignis auf dem Button ausgeführt wird, z.B. ein Klick. Ist dem so, wird der Inhalt der Methode *onClick()* ausgeführt. *onClick()* ist dabei die von uns fertig programmierte Methode eines Interface.

Eingegebene Werte erfassen:
Die Werte der Textfelder werden ähnlich wie in Swing ausgelesen.
Der Unterschied ist, dass die Methode *getText()* keinen String zurückgibt, sondern eine CharSequence, die noch mittels *toString()* in einen String umgewandelt werden muss.
Mit *Double.parseDouble()* wird dieser String in eine Kommazahl umgewandelt.

5 Weitere Vorgehensweise

Sie können nun die App in Ihrer Testumgebung testen. Siehe dazu *Handreichung - Installation und Einrichtung Android Studio*.

Wenn Ihnen oder Ihren Schülern dieses Einfachbeispiel zu einfach ist, empfehlen wir die Taschenrechner-App zum Nachbauen. Siehe dazu *UE Taschenrechner-App*.