# Klausur zur Veranstaltung
# Einführung in die Anwendungsprogrammierung mit Android
# im Sommersemester 2017

Florin Schwappach 2. August 2017

---

**Nachname** .....

**Vorname** .....

**Matrikelnr.** ..... **Studiengang, HS (Abk.)** .....

## Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie können 64 Punkte erreichen.
2. Verwenden Sie für das Lösen der Aufgaben die Programmiersprache Java. Alle gegebenen Code-Fragmente sind ebenfalls in Java verfasst. Halten Sie sich an die bekannten Regeln zur Codequalität.
3. Schreiben Sie Ihren Namen und Ihre Matrikelnummer leserlich unten auf jedes Angabenblatt bevor Sie mit der Bearbeitung beginnen! Blätter ohne diese Angaben werden nicht gewertet. Wenn Sie die Rückseite eines Blattes verwenden, notieren Sie dies bitte auf der Vorderseite. Kennzeichnen Sie eindeutig, zu welcher Aufgabe eine Lösung gehört.
4. Benutzen Sie keine Bleistifte, keine rot schreibenden Stifte und kein TippEx (oder ähnliche Produkte).
5. Die Klausur ist als *Closed Book*-Klausur angelegt. Sie dürfen keine mitgebrachten Quellen oder Notizen zur Bearbeitung der Aufgaben verwenden. Technische Hilfsmittel sind ebenfalls nicht erlaubt.
6. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Klausuraufsicht (Hand heben). Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für den gesamten Hörsaal durchgegeben. Aussagen unter vier Augen sind ohne Gewähr.
7. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung zu einer Aufgabe ergeben also null Punkte. Das gilt auch für die *Multiple Choice*-Fragen. Kreuzen Sie bei einer Frage zwei richtige und zwei falschen Möglichkeiten an, ergibt das in der Summe null Punkte.

# Teil 1: Theorie

**Hinweis** Falsche Antworten führen zu **Punktabzug** innerhalb einer Aufgabe. Eine richtige und eine falsche Antwort bei einer Aufgabe geben in der Summe also null Punkte. Bitte kreuzen Sie nur die Antworten an, bei denen Sie sich sicher sind.
Bitte beachten Sie, dass eine korrekte Antwort **zwei Punkte** wert ist.

## 1) Kommunikation (2 Punkte)

Welcher Mechanismus wird in Android verwendet, um Ereignisse zu kommunizieren, die das Gesamtsystem betreffen (etwa niedriger Akkustand)?

- □ NotificationBuilder
- □ Broadcasts
- □ LocationListener mit LocationEvent
- □ Services
- □ Shared Preferences

## 2) Adapter (2 Punkte)

Welche der folgenden Aussagen für Adapter trifft zu?

- □ Adapter müssen über die Art der Änderung an der Datenquelle informiert werden.
- □ Adapter können Datenquellen nur mit Instanzen von ViewGroup-Unterklassen verbinden, die das Interface AdapterView implementieren.
- □ Das zu befüllende View-Objekt muss beim Adapter durch die Methode setView(View v) registriert werden.
- □ Adapter benötigen für die Verbindung zu einer Datenquelle einen OpenGL-Kontext.

## 3) Datenbanken (2 Punkte)

Zu welchem Bestandteil der Android-Systemarchitektur gehört SQLite?

- □ Applications
- □ Libraries
- □ Android Runtime
- □ Linux-Kernel

## 4) Datenübertragung (2 Punkte)

Wieso sind weniger, aber dafür größere Datenübertragungen meist sinnvoller als mehrere kleine?

- □ Geringerer Stromverbrauch aufgrund der Schaltung der Zustände der Mobilfunkantenne
- □ Größere Datenpakete verursachen höhere Mobilfunkgebühren
- □ Diese Aussage ist falsch, es kommt nur auf die gesamte Datenmenge an
- □ Damit wird Prefetching ermöglicht

## 5) Services (4 Punkte)

Wie unterscheiden sich Services von AsyncTasks?

- □ Services sind nicht an Activities gebunden
- □ Services sind für längerfristige Hintergrundaufgaben
- □ Services werden mit der App, die sie gestartet hat, beendet
- □ Services haben keinen eigenen Lebenszyklus

## 6) Displays (4 Punkte)

Welche Auswirkungen hat die Pixeldichte (dpi / ppi) von Endgeräten?

- □ Je höher die Pixeldichte, desto kleiner ist ein Element einer bestimmten Pixelgröße auf dem Display
- □ Aufgrund der stark unterschiedlichen Pixeldichten bei Android-Geräten sind Pixel keine geeignete Maßeinheit
- □ Je niedriger die Pixeldichte, umso mehr Rechenleistung ist für die Darstellung der GUI erforderlich
- □ Je niedriger die Pixeldichte, desto schärfer löst ein Bildschirm auf

## 7) Ereignisse (4 Punkte)

Wie kann zur Laufzeit auf Ereignisse, wie dem Antippen eines Buttons, gelauscht werden?

- □ Mit (anonymen) inneren Klassen, die als "Lauscher" eingesetzt werden
- □ Durch die Implementierung eines passenden Interfaces bei der Activity
- □ Durch Java-Code, der in der XML-Definition des UI-Elementes eingebettet wird
- □ Gar nicht, jedes UI-Element muss in der XML-Definition auf die dazu gehörende Methode verweisen (z.B. mit dem "onClick"-Attribut)

## 8) Maßeinheiten (4 Punkte)

Welche der folgenden Maßeinheiten sollten möglichst nicht verwendet werden?

- □ pt
- □ px
- □ dp
- □ sp

## 9) Logging (2 Punkte)

Wie nennt sich das Logging-System von Android?

- □ logdog
- □ logdroid
- □ logcat
- □ logfrog

## 10) Lifecycle (2 Punkte)

Versehen Sie die unten genannten Methoden mit Ziffern entsprechend ihrer Reihenfolge, in der sie beim Beenden einer Activity aufgerufen werden!
Eine richtig eingeordneten Methode wird mit einem Punkt bewertet, wenn alle korrekt eingeordnet sind, gibt es zwei Punkte. Jede Ziffer ist nur einmal zu nennen, bei doppelter Vergabe gibt es Punktabzug.

_____ onStop()

_____ onDestroy()

_____ onPause()

# Teil 2: Code implementieren

## 11) Klausur-Star-Rating (Intents) (16 Punkte)

In dieser Aufgabe sollen Sie eine App vervollständigen, die Nutzern eine Sterne-Bewertungsleiste bietet, mit der die Klausur bewertet werden kann.
Die `MainActivity` besitzt dazu eine `RatingBar` (siehe Abb. 1). Nutzer können mit dieser `RatingBar` interagieren und visuell eine bestimmte Anzahl an Sternen auswählen, die sie der Klausur geben möchten. Zudem enthält das Layout einen `Button`, mit dem die `ResultActivity` aufgerufen wird. In der `ResultActivity` wird dann in einem Textfeld ausgegeben, wie viele Sterne der Klausur gegeben wurden (In einer späteren Version könnte mit dieser Information z.B. eine Datenbank gefüllt werden).

**Hinweis** Gehen Sie davon aus, dass der gegebene Code korrekt ist und dass alle notwendigen `import`-Anweisungen bereits vorhanden sind.

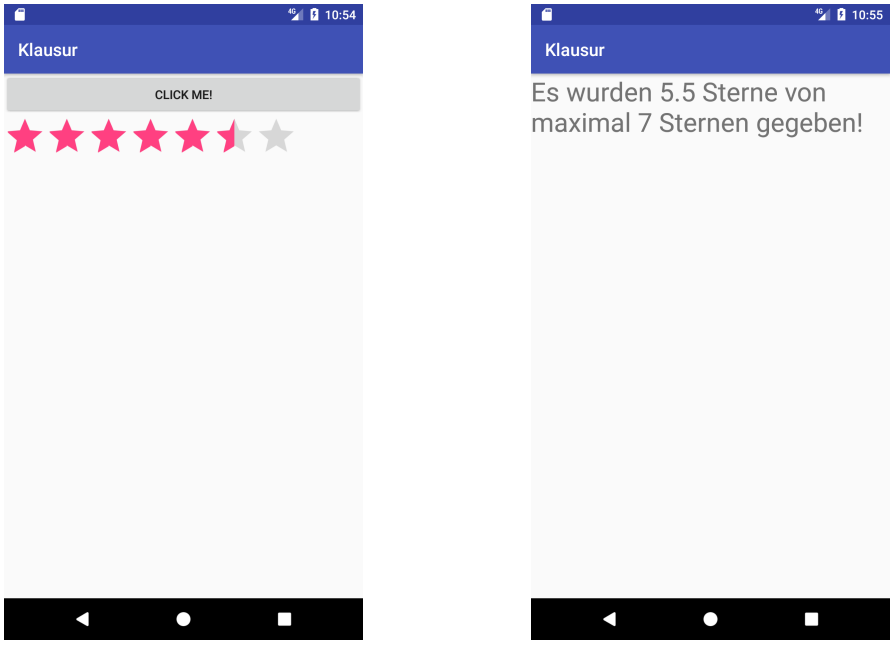


Abbildung 1: Links: Ansicht der MainActivity, Rechts: Ansicht der ResultActivity

```java
public class MainActivity extends AppCompatActivity {

    private Button    clickButton;
    private RatingBar ratingBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Code hier ergänzen
    }
}
```

```java
public class ResultActivity extends AppCompatActivity{

    TextView result;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Code hier ergänzen
    }
}
```

**11a) Implementierung der Klasse MainActivity** Als Erstes ist es Ihre Aufgabe, die Klasse `MainActivity` zu vervollständigen. Beginnen Sie damit, in der `onCreate`-Methode die Member-Variablen mit den korrekten UI-Elementen zu belegen. Die Button-ID ist 'click_button', die RatingBar-ID ist 'rating_bar'.
Anschließend müssen Sie für den `Button` einen `OnClickListener` setzen, mit dessen Hilfe die `ResultActivity` aufgerufen werden kann. Dazu können Sie z.B. eine Instanz einer anonymen Klasse verwenden, die Sie mit `new View.OnClickListener() {...}` erzeugen. Diese Klasse muss dann eine Methode `public void onClick(View v)` implementieren.
Zuletzt gilt es noch, die `ResultActivity` aufzurufen. An diese soll über einen `Intent` die Information gesendet werden, wie viele Sterne gewählt wurden. Instantiieren Sie einen `Intent` mit den korrekten Parametern (Anwendungs-`Context` und Klassentyp der Ziel-Activity).

Über die Methoden `getRating()` und `getNumStars()` der `RatingBar`-Klasse erhalten Sie die Bewertungsinformationen. Legen Sie diese als zusätzliche Informationen im `Intent` ab und rufen Sie die `ResultActivity` auf.

**Lösung zu a)**

**11b) Implementierung der Klasse ResultActivity** Nun muss noch die `ResultActivity` implementiert werden. Auch hier gilt als Erstes: die Member-Variable `result` mit dem richtigen UI-Element belegen. Die ID des Textfeldes ist 'result_view'.
Anschließend belegen Sie eine Variable vom Typ `Bundle` mit den Informationspaket aus dem Intent. Um das Textfeld mit den Informationen zu füllen, müssen diese aus dem `Bundle` geholt werden. Beachten Sie, dass je nach Typ der Information eine andere Methode zum Einsatz kommen kann.
Zuletzt soll das Textfeld einen Text erhalten, der dem aus Abb. 1 entspricht, und bei dem die Informationen aus dem `Intent` eingesetzt werden.
**Lösung zu b)**

## 12) Locations aus dem Web (AsyncTask) (20 Punkte)

In dieser Aufgabe geht es um einen Abruf von Daten aus dem Internet, der im Hintergrund ablaufen soll. Die geplante Android-Anwendung soll Informationen zu bestimmten Orten aus einer Web-Datenbank abrufen. Bei diesen Orten handelt es sich um `Locations`, die zuvor von Nutzern auf einer Karte markiert wurden.
In der gegebenen Klasse `MainActivity` wird bereits eine `ProgressBar` realisiert. Bei Klick auf einen Startbutton werden die gewählten Locations geholt und an den `BackgroundTask` übergeben. Gehen Sie davon aus, dass das UI und Click-Handler bereits implementiert sind.
Bei Vollendung des `BackgroundTasks` wird die Fortschrittsanzeige versteckt und eine Ergebnisanzeige aktualisiert. Ihre Aufgabe ist es, sich um den `BackgroundTask` zu kümmern.

**Hinweis** Gehen Sie davon aus, dass der gegebene Code korrekt ist und dass alle notwendigen `import`-Anweisungen bereits vorhanden sind.

```
interface LocationInfoListener {
    public void onProgress(int indexOfNewInfo);
    public void onResult(ArrayList<LocationInformation>
       informationList);
}
```

```
public class MainActivity extends Activity implements
   LocationInfoListener {

    private ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // (...) Hier wird das UI initialisiert
    }

    private void onStartButtonClicked() {
        Location[] locations = getSelectedLocations();
        // (...) Hier wird die progressBar initialisiert
        progressBar.setVisibility(View.VISIBLE);
        new BackgroundTask(this).execute(locations);
    }

    @Override
    public void onProgress(int indexOfNewInfo) {
        progressBar.setProgress(indexOfNewInfo);
```

```
20      }
21
22      @Override
23      public void onResult(ArrayList<LocationInformation>
            informationList) {
24          progressBar.setVisibility(View.GONE);
25          updateResultView(informationList);
26      }
27  }
```

**12a) Implementieren Sie das Gerüst einer eigenen Klasse BackgroundTask**
Diese soll von `AsyncTask<Location, Integer, ArrayList<LocationInformation»` erben und die Funktionsdefinitionen aller notwendigen Methoden beinhalten. Die Klasse benötigt folgende Funktionen, deren Köpfe Sie in dieser Aufgabe definieren sollen:

- Konstruktor
- Funktion, welche die Arbeit erledigt (`doInBackground`)
- Funktion, welche sich um die Kommunikation des Fortschritts kümmert (`onProgressUpdate`)
- Funktion, die nach getaner Arbeit ausgeführt wird (`onPostExecute`)

Ordnen Sie den Methoden die korrekten Parameter- und Rückgabetypen zu und beachten Sie, dass die `MainActivity` das Interface `LocationInfoListener` implementiert. Über die Methoden dieses Interfaces kann der `BackgroundTask` mit dem UI-Thread kommunizieren. Hierfür muss der `BackgroundTask` eine Instanz übergeben bekommen und auch speichern, damit er später damit arbeiten kann.
**Lösung zu a)**

**12b) Implementieren Sie die Methoden von BackgroundTask**

Nun sollen die Methodenrümpfe implementiert werden. Bei Ausführung des `BackgroundTask` soll für jedes übergebene `Location`-Objekt (siehe Z. 14 der `MainActivity`) jeweils ein `LocationInformation`-Objekt erzeugt werden.

Die Anfrage an den Server, sowie das Verarbeiten der Serverantwort muss nicht selbst implementiert werden, verwenden Sie stattdessen die Methode `public static LocationInformation getInformationFor(Location location)` der Klasse `OnlineDBHelper`.

Abschließend sollen alle erhaltenen `LocationInformation`-Objekte gebündelt in einer ArrayList zurückgegeben werden. Darüber hinaus soll der UI-Thread für jedes erzeugte `LocationInformation`-Objekt über den Fortschritt informiert werden. Dabei wird jeweils ein Zahlenwert übergeben, der das erzeugte `LocationInformation`-Objekt repräsentiert.

Für die Kommunikation zwischen `BackgroundTask` und `MainActivity` implementiert `MainActivity` das Interface `LocationInfoListener`. Denken Sie daran, den Listener an geeigneter Stelle in `BackgroundTask` zu registrieren, beachten Sie hierfür den Aufruf des `BackgroundTask` in der Methode onStartButtonClicked() in `MainActivity`. Nutzen Sie die Methoden des Listeners, um sowohl Fortschritt als auch Endresultat in den entsprechenden `BackgroundTask`-Methoden zu kommunizieren.

**Lösung zu b)**