Android
Übungsbeispiele Q2

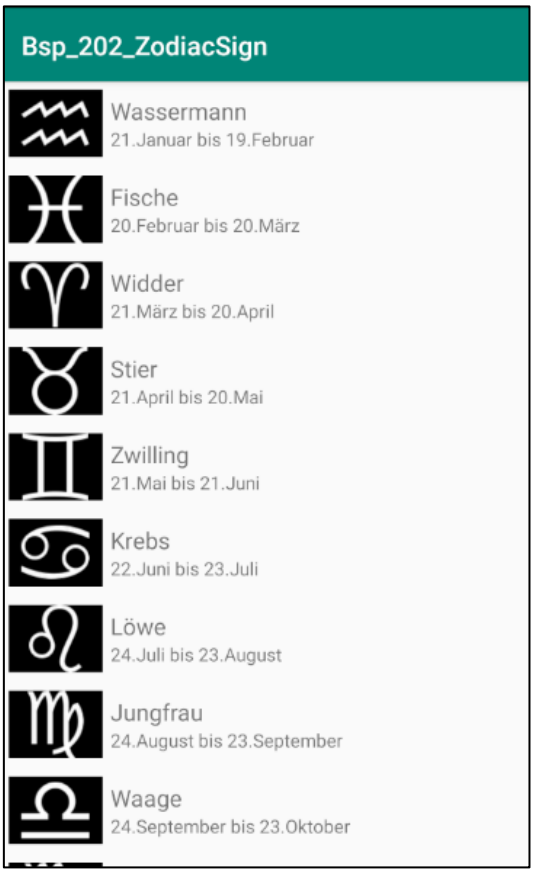
Collections, Listen, Inheritance Lambdas, Streams
Fragments, Intents, Serialization, Unit-Tests

Dr. Heinz Schiffermüller
HTBLA-Kaindorf
Abteilung EDVO

Erstellung: August 2019
Letzte Überarbeitung: November 2019

Exa_201_ZodiacSign

☆☆



Erstelle eine App zur Anzeige einer Liste aller Sternzeichen.

Programmablauf:
Bei Programmstart wird eine Liste aller Sternzeichen angezeigt. Durch Klicken auf ein Sternzeichen leitet die App weiter zur Wikipedia-Seite des jeweiligen Sternzeichens.

Programmbeschreibung:

Datei **`activity_main.xml`**:

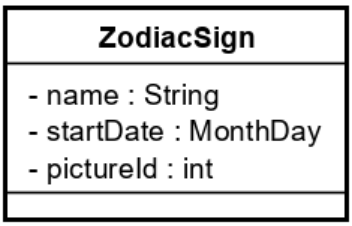
Verwende ein **`RecyclerLayout`** um die Liste von Sternzeichen der Abbildung zu erstellen.

Datei **`zodiac_item.xml`**:

Verwende zur Anzeige der Daten eines Sternzeichens eine **`ImageView`** und zwei **`TextView`**-Elemente.

Klasse **`ZodiacSign.java`** :

Datenklasse entsprechend dem Klassendiagramm. Die Methode **`getIdFromDrawable()`** liefert die id des jeweiligen Bildes zurück.



Klasse **`ZodiacViewHolder.java`** :

`ViewHolder`-Klasse für jeden Listeneintrag/ für jedes Sternzeichen.
`onClick`-Event für die Liste. Implementiere das **`RecyclerView.OnClickListener`**-Interface über die ViewHolder-Klasse. Bei einem Klick auf ein Listen-Element verzweigt die App zur Wikipedia-Seite des jeweiligen Sternzeichens.

Klasse **`ZodiacSignAdapter.java`** :

`Adapter`-Klasse für die Anzeige aller Listeneinträge.
Verwende eine geeignete Datenstruktur zum Speichern der Listenelemente.

Klasse **`MainActivity.java`** :

Hinweis:
Um auf ein neues Fenster weiterleiten zu können wird in Android ein **`Intent`**-Objekt verwendet:

```
String url = getString(R.string.wikipedia_url, sign.getName(this));
Intent viewIntent = new Intent("android.intent.action.VIEW",
Uri.parse(url));
startActivity(viewIntent);
```

Der Link ist in der Datei **`strings.xml`** definiert:

```
<string name="wikipedia_url">http://de.m.wikipedia.org/wiki/%s_(Tierkreiszeichen)</string>
```

Kompetenzen: RecyclerView, Listen, Events

Exa_202_BookListApp ☆☆

Erstelle eine App zur Anzeige einer Liste von Büchern.

Programmablauf:
Bei Programmstart wird eine Liste aller Bücher aus der Datei **`booklist.csv`** gelesen und in einer Liste angezeigt.
Alle Konten werden aufsteigend nach dem Titel sortiert.

Programmbeschreibung:

Datei **`activity_main.xml`**:
Verwende eine **`RecyclerView`** um die Liste aller Bücher anzuzeigen.

Datei **`book_item.xml`**:
Erstelle die Oberfläche zur Darstellung der Daten eines Buches entsprechend der Abbildung.

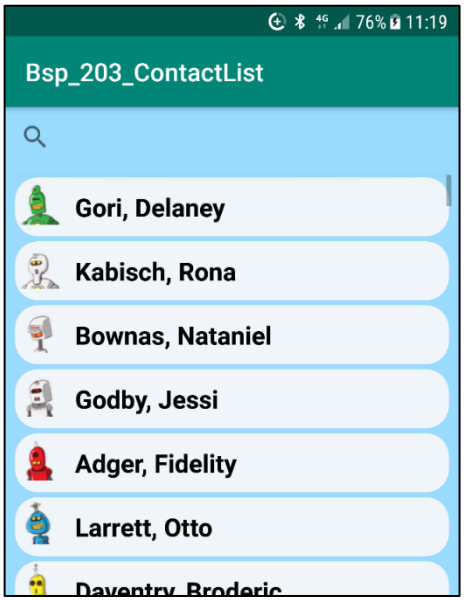
Klasse **`BookViewHolder.java`** :
`ViewHolder`-Klasse für die Bücher.

Klasse **`BookAdapter.java`** :
`Adapter`-Klasse zur Anzeige aller Bücher.
Verwende eine geeignete Datenstruktur zum Speichern der Listenelemente. Verwende zum Sortieren der Bücher nach dem Titel einen Streaming-Ausdruck.

Klasse **`IO_Access.java`** :
Implementiere die statische Methode **`loadBooks()`** die alle Bücher von der Datei **`booklist.csv`** aus dem Assets Verzeichnis lädt und in geeigneter Form zurückgibt.

Kompetenzen: RecyclerView, Listen, Events

Exa_203_ContactsApp ☆☆



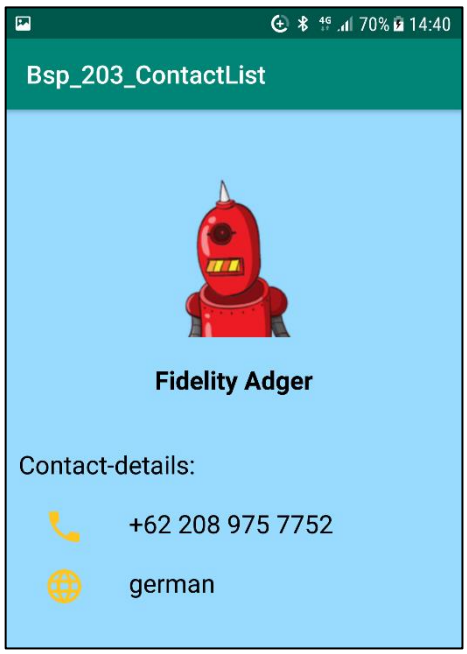
Erstelle eine App zur Anzeige einer Liste von Kontaktdaten.

Programmablauf:
Bei Programmstart wird eine Liste aller Kontakte aus der Datei **`contact_data.csv`** gelesen und in einer Liste angezeigt.
Die Liste kann mit einer SearchView nach dem Namen gefiltert werden und nach verschiedenen Kriterien sortiert werden. Beim Klicken auf eine Kontakt werden die Detailinformationen in einer eigenen Activity angezeigt.
Mit Wischen nach rechts kann der jeweilige Kontakt aus der Liste entfernt werden.

Programmbeschreibung:

Datei **`activity_main.xml`**:

Füge oben eine SearchView ein, um die Liste nach dem Namen filtern zu können.
Verwende **`RecyclerView`** um die Liste aller Kontake anzuzeigen.
Implementiere eine FastScrollbar.



Datei **`contact_item.xml`**:

Verwende zur Anzeige der Daten eines Kontakts eine **`ImageView`** für das Profil-Bild und eine **`TextView`** für den Namen.

Klasse **`Contact.java`** :

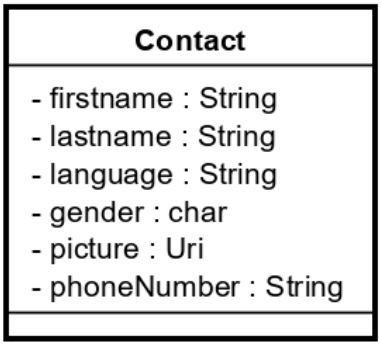
Datenklasse entsprechend dem Klassendiagramm.

Klasse **`ContactViewHolder.java`** :

`ViewHolder`-Klasse für jeden Listeneintrag/ für jeden Kontakt.
Erweiterung: Lade das Profilbild von der angegebenen URL.
Bei einem Klick auf ein Listen-Element verzweigt die App zu einer Activity mit der Detailansicht der Daten.
Erweiterung: Durch Wischen nach rechts wird das Element aus der Liste gelöscht.

Klasse **`ContactAdapter.java`** :

`Adapter`-Klasse für die Anzeige aller Listeneinträge.
Verwende eine geeignete Datenstruktur zum Speichern der Listenelemente. Lade im Konstruktor die Kontakt-Daten von der Datei **`contact_data.csv`** und speichere sie in einer geeigneten Datenstruktur.
Implementiere die Methode **`filterContacts(String filter)`** um die Liste der Kontake zu filtern. Verwende zum Filtern die **`removeIf()`**-Methode. Um die **`RecyclerView`** zu aktualisieren verwende die entsprechende **`notify`**-Methode.

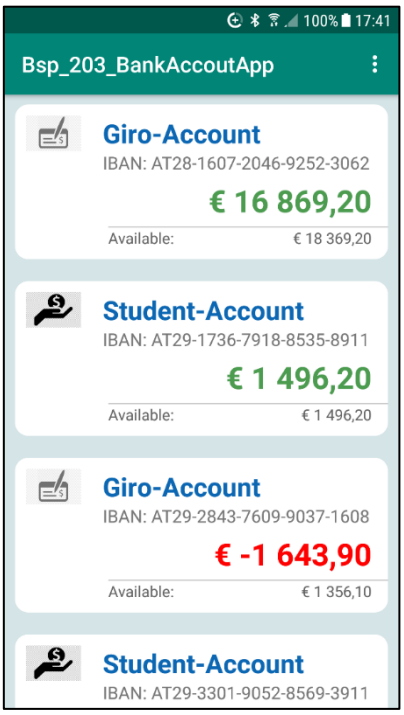


Klasse **`MainActivity.java`** :

Implementiere für die SearchView das Interface **`SearchView.OnQueryTextListener`**.

Kompetenzen: RecyclerView, SearchView, Listen, Files, Lambdas, Streams, Events

Exa_204_BankAccountApp ☆☆



Erstelle eine App zur Anzeige einer Liste von Konto-Daten.

Programmablauf:
Bei Programmstart wird eine Liste aller Kontakte aus der Datei **`account_data.csv`** gelesen und in einer Liste angezeigt.
Über das Option-Menü kann die Liste gefiltert werden: es werden entweder alle Konten, oder nur die Giro-Konten oder nur die Schüler-Konten angezeigt. Unter ‚Available' wird angezeigt wie viel Geld noch verfügbar ist. Positive Kontostände werden grün, negative rot angezeigt.
Alle Konten werden aufsteigend nach dem IBAN sortiert.
Beim Klicken auf ein Konto wird eine neue Activity gestartet über, um einen Betrag von vorhanden ist, zu überweisen. Bei der Abbuchung darf der Rahmen nicht überschritten werden.

Programmbeschreibung:

Datei **`activity_main.xml`**:
Füge ein Options-Menü ein, um die Liste nach der Kontoart filtern zu können.
Verwende eine **`RecyclerView`** um die Liste aller Konten anzuzeigen.
Implementiere eine FastScrollbar.
Mit einem Long-Click auf ein Konto kommt man zur TransferActivity.

Datei **`filter_options.xml`**:

Option-Menü zum Filtern der Konten mit den drei Items: All Accouts - Student Accounts - Giro Accounts

Datei **`account_item.xml`**:

Erstelle die Oberfläche zur Darstellung der Daten eines Kontos entsprechend der Abbildung.

Klasse **`AccountViewHolder.java`** :

`ViewHolder`-Klasse für die Konten.
Implementiere den **`OnLongClickListener`** um bei einem LongClick zur Transfer-Activity zu wechseln. Alle notwendigen Daten werden auf den Intent gelegt. Starte die Activity mit **`startActivityForResult`** um Daten von der Transfer-Activity zurückzubekommen.

Klasse **`AccountAdapter.java`** :

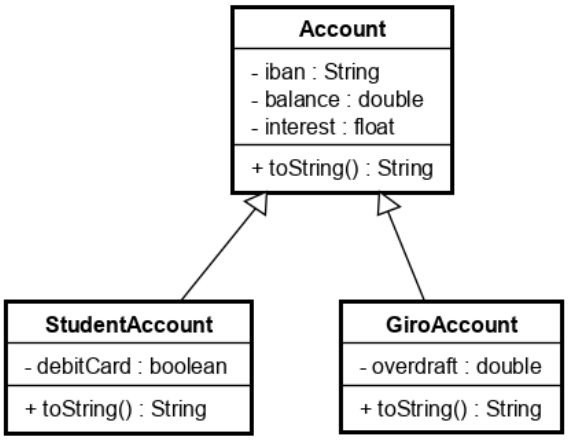
`Adapter`-Klasse zur Anzeige aller Konten.
Verwende eine geeignete Datenstruktur zum Speichern der Listenelemente. Verwende zum Sortieren der Konten nach dem Iban einen Streaming-Ausdruck.
Implementiere die Methode **`filterAccount(String type)`** um die Liste der Konten zu filtern. Verwende dazu einen Streaming-Ausdruck.
Implementiere die Methode **`transferMoney(Account fromAccount, String toIban, double amount)`** um den Betrag **`amount`** vom Konto **`fromAccount`** auf das durch **`toIban`** spezifizierte Konto.

Klasse **`IO_Access.java`** :

Implementiere die statische Methode **`loadAccounts()`** die alle Kontodaten von der Datei **`account_data.csv`** aus dem Assets Verzeichnis lädt und in geeigneter Form zurückgibt.

Klasse **`MainActivity.java`** :

Implementiere das Option-Menu.
Implementiere die Callback-Methode **`onActivityResult`** um die Änderungen der beiden Konten von der Transfer-Activity zu bekommen und die Daten im **`AcountAdapter`** zu aktualisieren.

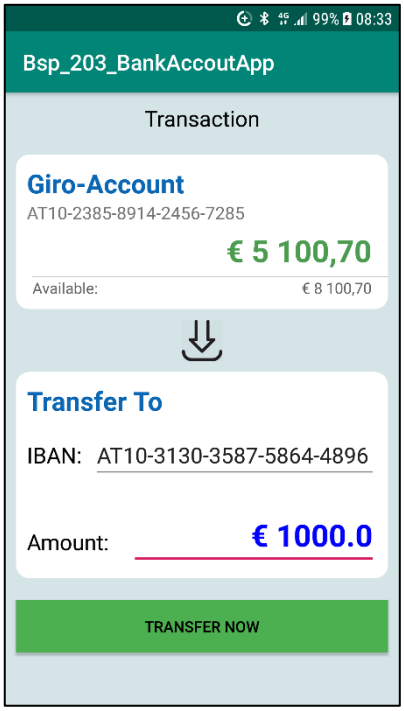


Die Klassen **`Account, StudentAccout, GiroAccount`**

Datenklassen entsprechend dem Klassendiagramm.
Zum Übertragen der Konto-Daten von einer Activity zur nächstens ist für alle drei Klassen das Paracelable Interface zu implementieren.

Datei **`activity_transfer.xml`**:

Zweite Activity zum Überweisen eines Betrags auf ein anderes Konto. Im oberen Bereich werden die Daten des geklickten Kontos angezeigt. Im unteren Bereich können ein Iban in eine **`AutoCompleteTextView`** und ein Betrag in ein **`EditText`**-Feld eingegeben werden. Der Transfer-Button ist disabled und wird erst aktiviert wenn ein gültiger Iban und ein gültiger Transferbetrag eingegeben wurden.
Der eingegebene Betrag wird beim oberen Konto abgezogen, die beiden **`TextView`**-Elemente werden bei jeder Änderung aktualisiert. Wenn der Überweisungsbetrag den verfügbaren Betrag überschreitet wird er rot angezeigt und der Transfer-Button disabled. Der Transferbutton kann nur geklickt werden wenn ein gültiger Iban und ein verfügbarer Betrag eingegeben wurden.
Nach Klicken auf den Transfer-Button wird der zur Main-Activity zurückgeleitet und die Beträge auf beiden Konten aktualisiert.



Klasse **`TransferActivity.java`** :

Implementiere die **`AutoCompleteTextView`** für den Iban mit einer Adapter-Klasse.
Implementiere die **`EditText`**-View mit einem TextWatcher um den Button zu aktivieren und zu deaktivieren und die EditText-Elemente für das Konto zu aktualisieren.

Kompetenzen: Vererbung, Option-Menu, RecyclerView, Listen, Files, Lambdas, Streams, Intent, Parcelable

Exa_205_Game2048 ☆☆☆

Erstelle eine App für das Spiel 2048.

Programmablauf:
Erstelle das Spiel nach den offiziellen Regeln: beim Start werden zwei Zahlen, entweder 2 oder 4, zufällig auf den Feldern verteilt. Ein Zug wird durch eine Wisch-Bewegung nach links, rechts, oben oder unten durchgeführt. Alle Zahlen werden mit dem Zug in die jeweilige Richtung bis zum Rand verschoben. Jeweils 2 gleiche Zahlen werden aufsummiert und zu einer Zahl zusammengefasst. Der Spieler erhält dann die Punkte der beiden Werte. Die aktuelle Punktezahl wird oben rechts angezeigt. Nach jedem Zug wird eine weitere Zahl (2 oder 4) zufällig auf einen freien Platz gesetzt.
Das Spiel ist beendet wenn entweder kein Zug mehr gemacht werden kann, weil das Spielfeld voll ist, oder wenn auf einem Feld die Zahl 2048 erreicht wurde. Mit dem Renew-Button wird ein neues Spiel gestartet.

Programmbeschreibung:

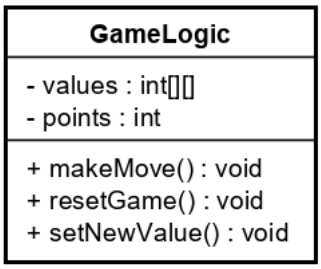
Datei **`activity_main.xml`**:
Erstelle die Oberfläche entsprechend der Abbildung.

Klasse **`MainActivity.java`**

Implementiere einen Gesture-Listener für die Wisch-Events.
Implementiere ein onClick-Event für den Renew-Button.

Datei **`GameLogic.java`**

Erstelle die Klasse als Business-Layer für die Spiel-Logik.
Speichere die Werte auf einem 2D-int Feld.
`makeMove(String direction)` berechnet die Zahlenverteilung bei einem Zug nach links, rechts, oben oder unten.
`resetGame()` setzt das Spiel wieder in den Startzustand und die Punkte auf 0.
`setNewValue()` setzt zufällig eine neue Zahl auf einen freien Platz am Spielfeld. Die Zahl ist entweder 2 oder 4, wobei 2 mit doppelt so hoher Wahrscheinlichkeit kommt.



Datei **`ColorScheme.java`**

Erstelle eine Enum mit Konstanten für den Wert, die Hintergrundfarbe und die Schriftfarbe aller unterschiedlicher Zahlenwerte.