Week 6 - oefening 1 (Bis): ListFragment, ArrayAdapter & adapterviews

Doelstelling

Gebruik leren maken van ListFragment, ArrayAdaptar en andere gerelateerde layouts.
Deze oefening is analoog als de Week6 – oefening1, maar met dit verschil dat er met Fragments i.p.v. Activities gewerkt wordt. Deze oefening is een thuisopgave (maar behoort ook tot de verwachte doelstellingen van deze module)

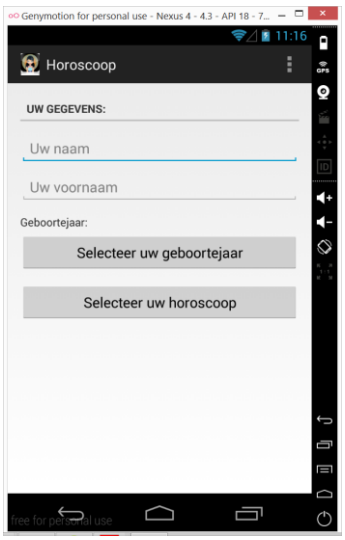
Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen.

Application Name:	Horoscoop
Package Name:	be.howest.nmct
Target SDK:	21
Activity:	MainActivity – *layout: activity_main.xml*
Fragments:	MainFragment – *Layout: fragment_main.xml* HoroscoopFragment (ListFragment, geen layout file) SelectGeboortejaarFragment (ListFragment, geen layout file) *Layout rij: row_horoscoop.xml*
Launch icon:	Zie bijlage

Design

We starten met de uitbouw van de eerste fragment (MainFragment). Van hieruit worden via twee buttons twee andere fragments aangeroepen (zie verder). Gebruik de layout-file uit het bronmateriaal.



Code SelectGeboortejaarFragment

Vanuit de eerste button roepen we een tweede fragment op ('SelectGeboortejaarFragment').

Gebruik de werkwijze uit week 5 om van Fragment te wisselen.

We kiezen ervoor om de SelectGeboortejaarFragment te laten overerven van een **ListFragment** (i.p.v. Fragment). Een ListFragment heeft specifiek tot doel een verzameling van elementen weer te geven. Deze verzameling is heel vaak een array van static data, of een cursor (naar bv. data afkomstig uit resultset bekomen door query op een database).

Een ListFragment hoeft geen layout-file meer te hebben gezien deze onmiddellijk een listview omvat. Wenst men toch nog andere views te tonen (bv. textView bovenaan), dan werkt men op de klassieke wijze met een layoutfile maar dan MOET de aanwezige listview volgend id krijgen: '@android:id/list'. Meer info is te vinden op:
http://developer.android.com/reference/android/app/ListFragment.html

Wanneer men geen afzonderlijke layout-file gebruikt, is de onCreateView-methode overbodig.

Onze data bestaat uit een array van strings (gaande van de geboortejaren starten van 1900 tot nu).

```
private final static List<String> GEBOORTEJAREN;
static {
    GEBOORTEJAREN = new ArrayList<>(Calendar.getInstance().get(Calendar.YEAR) - 1900);
    for (int jaar = 1900; jaar < Calendar.getInstance().get(Calendar.YEAR); jaar++) {
        GEBOORTEJAREN.add("" + jaar);
    }
}
```

Om data en listview te koppelen, maken we gebruik van een ArrayAdapter:

```
private ListAdapter myListAdapter;

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    myListAdapter = new ArrayAdapter<>(getActivity(), android.R.layout.simple_list_item_1, GEBOORTEJAREN);
    setListAdapter(myListAdapter);
```

Meer info over het gebruik van een array-adapter:
http://developer.android.com/reference/android/widget/ArrayAdapter.html

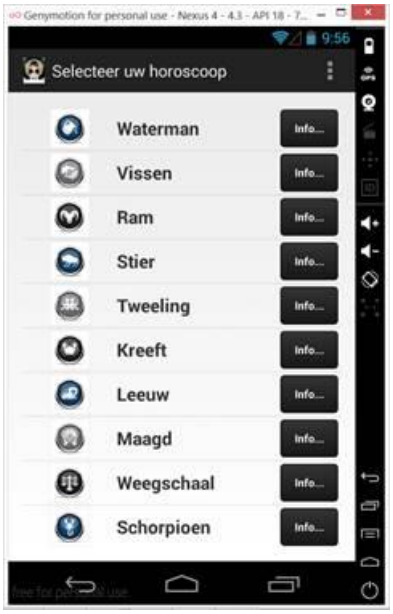
Voeg nu de onListItemClick-methode toe om het geselecteerde jaartal op te vragen. Via de parameter 'position' kan men achterhalen op welk item in de listview er geklikt werd. Geef nu dit jaartal terug aan de MainActivity (zie vorig week).

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    String sGeboortejaar = GEBOORTEJAREN.get(position);
    listener.onNieuwGeboortejaar(sGeboortejaar);
}

public interface SelectGeboortejaarFragmentListener {
    public void onNieuwGeboortejaar(String jaar);
}
```

In de MainActivity zorg je ervoor dat opnieuw MainFragment getoond wordt (zie vorige week). Stuur het jaartal door zodat het daar getoond kan worden.

Code HoroscoopFragment

Vanuit de tweede button op MainFragment roepen we een nieuwe ListFragment ('HoroscoopFragment') aan. Ook hiervan wensen we het geselecteerde resultaat terug te krijgen. Werk op analoge wijze als hierboven om deze fragment aan te roepen.



De HoroscoopFragment zelf bestaat uit een listview waarbij elk listviewitem bestaat uit:

- ImageView
- TextView
- Button

Ook nu kiezen we ervoor om de HoroscoopFragment te laten overerven van een **ListFragment** (i.p.v. Fragment).

List Item View

De layout van een listviewitem verzorgen we via een afzonderlijk xml-file. Maak de file 'row_horoscoop.xml' in de layout-folder aan. Kies een gepaste layout, en bouw één rij uit:

De style van de button is uitbreiding (en wordt hier niet verder beschreven. Zie vorig labo's)

De afbeeldingen vindt u terug in het bronmateriaal. Plaats deze in de juiste mappen van jouw project. Voor de data maakt u gebruik van de gegeven **Data**-klasse (te vinden in het bronmateriaal). In de deze klasse is een enumeration aanwezig. Bestudeer even de code hiervan.

Custom adapter

Om de listview nu op te vullen hebben we een eigen adapter nodig. We gebruiken hiervoor een eenvoudige customadapter 'HoroscoopAdapter'die erft van ArrayAdapter. In de constructor worden o.a. layout en data aan elkaar doorgegeven.

```
class HoroscoopAdapter extends ArrayAdapter<Data.Horoscoop> {
    public HoroscoopAdapter() {
        super(HoroscoopActivity.this, R.layout.row_horoscoop,
                R.id.textViewNaamHoroscoop, Data.Horoscoop.values());
    }
```

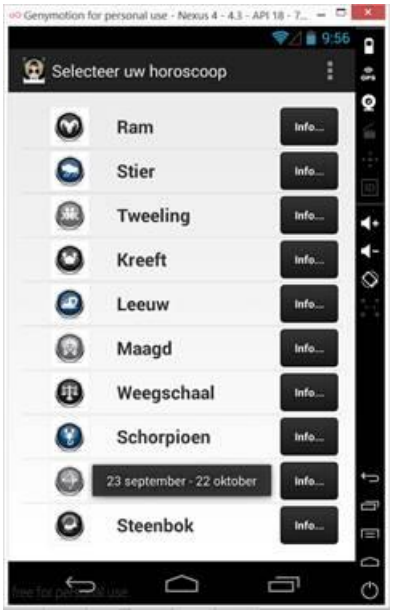
De koppeling gebeurt evenwel in de getView-methode:

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    View row = super.getView(position, convertView, parent);

    Data.Horoscoop horoscoop = Data.Horoscoop.values()[position];

    TextView textViewNaamHoroscoop = (TextView) row.findViewById(R.id.textViewNaamHoroscoop);
    textViewNaamHoroscoop.setText(horoscoop.getNaamHoroscoop());
```

Werk dit verder uit. Zorg dat de juiste afbeelding getoond wordt. Gebruik een hulpmethode dat de juiste afbeelding teruggeeft:

```
private int getResourceId(Data.Horoscoop horoscoop) {
    switch (horoscoop) {
    case WATERMAN:
        return R.drawable.waterman;
    case VISSEN:
```

Bij het klikken op de button toon je via een Toast de begin- en einddatum van de geselecteerde horoscoop.



Opmerking: om ervoor te zorgen dat men op een listviewitem kan blijven klikken, dient men volgende aanpassing in de layout-file aan te brengen.

```
<Button
    android:id="@+id/btnToonInfo"
    style="@style/styleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:background="@drawable/mijn_button"
    android:focusable="false"
    android:focusableInTouchMode="false"
    android:text="@string/toon_info" >
</Button>
```

Zorg ervoor dat bij het selecteren van een listview item het resultaat terugkeert naar MainFragment (werkwijze voor communicatie tss Fragments: zie week 5). Voeg hiervoor opnieuw de onListItemClick-methode in de HoroscoopFragment toe.

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    listener.onNieuwHoroscoop(Data.Horoscoop.values()[position]);
}


public interface SelectHoroscoopFragmentListener {
    public void onNieuwHoroscoop(Data.Horoscoop horoscoopSelected);
}
```

Met het resultaat kan men in de main-activity dezelfde afbeelding in een grotere ImageView tonen.

Optimalisatie

Pas jouw code in de methode 'getView' als volgt aan. Ga in logcat na of je de boodschappen terug vindt. Controleer hoe de list reageert als je erdoor scrollt. Wat stel je vast?

```
Log.d("HOROSCOOP","findViewById");
TextView textViewNaamHoroscoop = (TextView) row
        .findViewById(R.id.textViewNaamHoroscoop);
textViewNaamHoroscoop.setText(horoscoop.getNaamHoroscoop());
```

Minimizing FindByViewId Calls

Lees even volgende info:
http://developer.android.com/training/improving-layouts/smooth-scrolling.html#ViewHolder

Pas dit toe door een nieuwe innerklasse 'ViewHolder' toe te voegen.

```
class ViewHolder {
    public ImageView imageviewHoroscoop = null;
    public TextView textViewNaamHoroscoop = null;
    public Button btnToonInfo = null;

    public ViewHolder(View row) {
        this.imageviewHoroscoop = (ImageView) row
                .findViewById(R.id.imageviewHoroscoop);
        this.textViewNaamHoroscoop = (TextView) row
                .findViewById(R.id.textViewNaamHoroscoop);
        this.btnToonInfo = (Button) row.findViewById(R.id.btnToonInfo);
    }
}
```

Pas vervolgens de getView-methode aan.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    // TODO Auto-generated method stub
    View row = super.getView(position, convertView, parent);

    final Data.Horoscoop horoscoop = Data.Horoscoop.values()[position];

    ViewHolder holder = (ViewHolder) row.getTag();

    if (holder == null) {
        holder = new ViewHolder(row);
        row.setTag(holder);
    }

    TextView textViewNaamHoroscoop = holder.textViewNaamHoroscoop;
    textViewNaamHoroscoop.setText(horoscoop.getNaamHoroscoop());
```

Afwerking

Zorg ervoor dat het laatste gekozen geboortejaar en/of horoscoopafbeelding zichtbaar worden als de app opnieuw zichtbaar wordt.

Meer info

http://developer.android.com/guide/topics/ui/layout/listview.html

http://developer.android.com/guide/topics/ui/layout/listview.html#Loader

http://developer.android.com/training/improving-layouts/smooth-scrolling.html#ViewHolder