

## Week 6 - oefening 1: ListActivities, ArrayAdapters & custom adapterviews


### Doelstelling

Gebruik leren maken van ListActivities, ArrayAdapter en andere gerelateerde layouts.

Opmerking: in deze versie gebruiken we uitsluitend activities. In versie 'Week6Oefening1Bis' vindt u dezelfde opgave maar dan met Fragments. De communicatie tussen activities werd in labo week 4 behandeld.

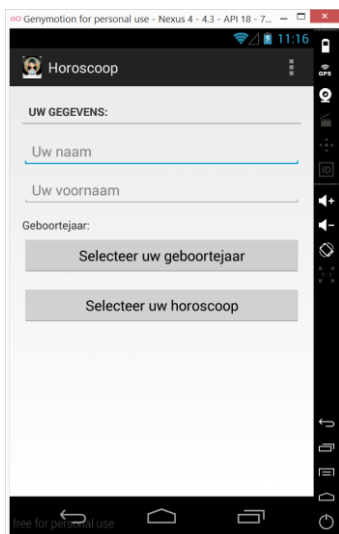
### Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen.

Application Name:	Horoscoop
Package Name:	be.howest.nmct
Target SDK:	21
Activity:	MainActivity ( <i>layout: activity_main.xml</i> ) HoroscoopActivity ( <i>ListActivity, geen layout file</i> ) SelectGeboortejaarActivity ( <i>ListActivity, geen layout file</i> ) <i>Layout rij: row_horoscoop.xml</i> 
Fragments:	
Launch icon:	Zie bijlage

### Design

We starten met de uitbouw van de eerste activity (MainActivity). Van hieruit worden via twee buttons twee andere activities aangeroepen (zie verder). Bouw onderstaande MainActivity na.



## Code SelectGeboortejaarActivity

Vanuit de eerste button roepen we een tweede activity ('SelectGeboortejaarActivity') aan. Gebruik hiervoor een explicit of implicit intent. Deze tweede activity zal een jaartal moeten teruggeven. Gebruik de methode uit vorig labo:

- aanmaak van final String constante in de MainActivity  

```
public static final String EXTRA_BIRTHYEAR = "be.howest.nmct.week5oef1.BIRTHYEAR";
```
- gebruik maken van de methode  

```
public void selecteerGeboortejaar(View v) {
    Intent intent = new Intent(MainActivity.this,
        SelectGeboortejaarActivity.class);
    startActivityForResult(intent, REQUEST_BIRTHDAY);
}
```

De activity 'SelectGeboortejaarActivity' zelf bestaat uit een listview waarbij de gebruiker zijn geboortejaar kan selecteren.



We kiezen ervoor om de SelectGeboortejaarActivity te laten overerven van een **ListActivity** (i.p.v. Activity). In de onCreate-methode maken we geen gebruik van 'setContentView' en laden dus geen layout-file in.

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

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

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

<http://developer.android.com/reference/android/widget/ArrayAdapter.html>

```
private ListAdapter myListAdapter;

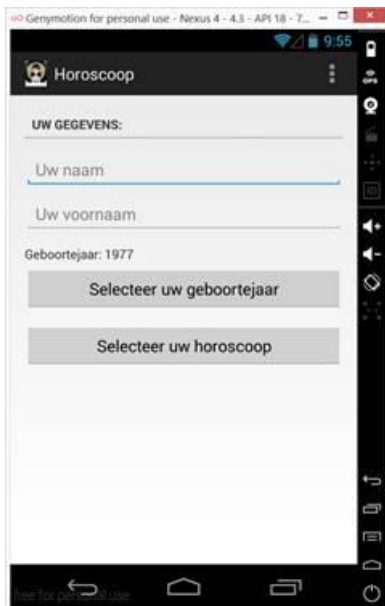
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    myListAdapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, GEBOORTEJAREN);
    setListAdapter(myListAdapter);
}
```

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
protected void onListItemClick(ListView l, View v, int position, long id) {
    // TODO Auto-generated method stub
    String sGeboortejaar = GEBOORTEJAREN.get(position);

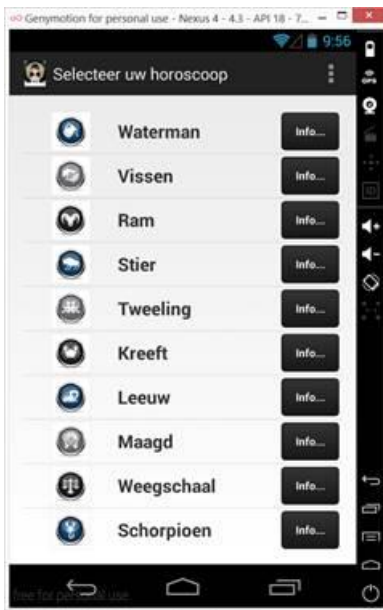
    Intent intent = new Intent();
    intent.putExtra(MainActivity.EXTRA_BIRTHYEAR, sGeboortejaar);
    setResult(RESULT_OK, intent);
    finish();
}
```

In de MainActivity vang je het resultaat via de 'onActivityResult'-methode op (zie labo week 4). De textview wordt aangevuld met het jaartal.



## Code HoroscoopActivity

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



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

- ImageView
- TextView
- Button

Ook nu kiezen we ervoor om de HoroscoopActivity te laten overerven van een **ListActivity** (i.p.v. Activity).

Alternatief:

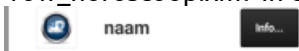
In sommige gevallen wenst men naast de listview ook nog andere views te tonen. Dit is perfect mogelijk, maar dan moet het id van de listview in de layoutfile volgend id krijgen: '@android:id/list'

```
<ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:choiceMode="singleChoice" />
```

Opmerking: enkel nodig indien je de klasse laat overerven van ListActivity...

### 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 simpele custom adapter '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);

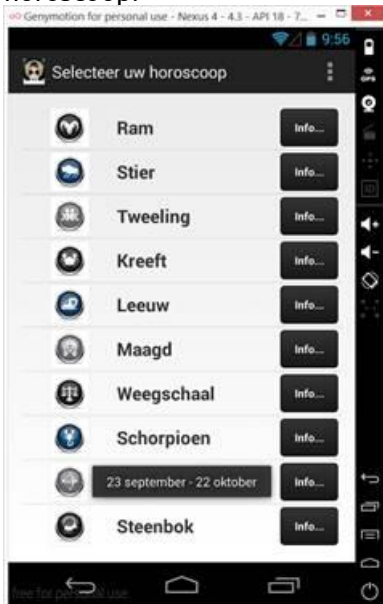
    Data.Horoscoop horoscoop = Data.Horoscoop.values()[position];

    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 VTSSRM:
            return R.drawable.vtssrm;
    }
}
```

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. Voeg hiervoor opnieuw de onItemClick-methode in de HoroscoopActivity toe. Je kan het nummer van het geselecteerde element uit de enumeration teruggeven. Hiermee kan in de main-activity dezelfde afbeelding in een grotere ImageView getoond worden.



## 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 FindViewById 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 geboortjaar en/of horoscoopafbeelding zichtbaar worden als de MainActivity 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>