Week 6 - oefening 2: CursorAdaptar & Loader

Doelstelling

Gebruik leren maken van ListViews en andere gerelateerde layouts.

Voorbereiding

Maak een nieuw Android Application Project aan met volgende instellingen.

Application Name:	Scores studenten
Package Name:	be.howest.nmct.evaluationstudents
Target SDK:	21
Activity:	StudentsActivity *(layout: activity_students.xml)*
Fragments:	StudentsFragment *(layout: fragment_student.xml)* *Layout rij: row_student.xml*
Launch icon:	Zie bijlage

Vooraf

We gebruiken de data-klassen die we in week 1 opgave 3 ontwikkeld hebben. Je kan ook gebruik maken van de klassen Student en Modulepunt uit het bronmateriaal.

In deze opgave wordt verondersteld dat alle methodes aanwezig zijn. Plaats de klassen Student en ModulePunt in een afzonderlijke package 'be.howest.nmct.admin'.

De klasse **Student** werd uitgebreid zodat ook naam & voornaam bijgehouden worden. Via een tweede constructor worden dat naast de andere parameters ook een naam en voornaam doorgeven. Een extra static methode geeft een lijst met alle unieke modulenamen terug. Deze worden gehaald uit een lijst van studenten.

```
public static List<String> getModuleNamen(List<Student> studenten) {
    List<String> uniekeNamenModules = new ArrayList<String>();

    for (Student student : studenten) {
        for (String modulenaam : student.getScoresStudent().keySet()) {
            if (!uniekeNamenModules.contains(modulenaam)) {
                uniekeNamenModules.add(modulenaam);
            }
        }
    }
    return uniekeNamenModules;
}
```

Voeg de klasse **StudentAdmin**.java toe aan. In deze klasse werd het Singleton Design Pattern toegepast. Deze methode geeft ons later test-data voor handen.

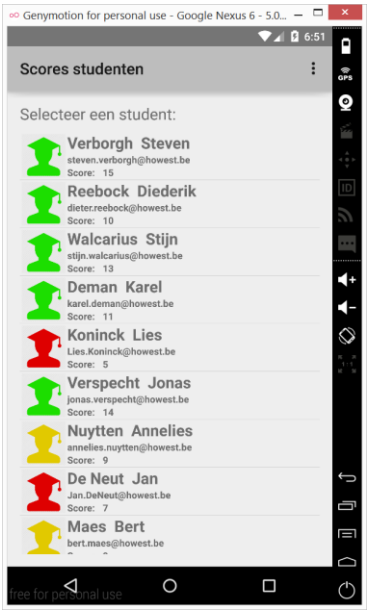
Plaats bovenstaande drie klassen in een afzonderlijke package 'be.howest.nmct.evaluationstudents.data'

Kopieer alle afbeeldingen uit het bronmateriaal in de juiste mappen.

StudentsFragment

We starten met de uitbouw van het eerste fragment (StudentsFragment). Werk op gelijkaardige manier als opgave 1bis: laat deze klasse erven van ListFragment.

Merk op dat dit fragment naast een listview ook een textview bevat. Hier hebben we dus wel de bijhorende layoutfile nodig. Zorg ervoor dat in de layoutfile de listview als id de waarde '@android:id/list' krijgt!



Werk de layout van een listview-item in de layout-file 'row_student.xml' uit:

Opmerking: gebruik verschillende TextViews voor resp. naam, voornaam, email en totale score.

Binding to Data

In deze oefening gebruiken we een **CursorAdaptar** (meer specifiek een **SimpleCursorAdaptar**) om doorheen de data te lopen.

Het opvragen van de data doen we asynchroon m.b.v. AsyncTaskLoader. Eenmaal ingeladen is het resultaat een MatrixCursor: dit type cursor overloopt een tweedimensioneel data model bestaande uit kolomnamen en bijhorende rijen.

http://developer.android.com/guide/components/loaders.html
http://developer.android.com/reference/android/content/AsyncTaskLoader.html

Stap 1: maak een nieuw package aan: 'be.howest.nmct.evaluationsstudents.loader'

Stap 2: Om een matrixCursor te kunnen afleveren houden we de verschillende kolomnamen in een afzonderlijke (gecentraliseerde) klasse bij:

```java
package be.howest.nmct.evaluationstudents.loader;

import android.provider.BaseColumns;

/**
 * Created by Stijn on 19/03/2015.
 */
public final class Contract {

    public interface StudentColumns extends BaseColumns {
        public static final String COLUMN_STUDENT_NAAM = "student_naam";
        public static final String COLUMN_STUDENT_VOORNAAM = "student_voornaam";
        public static final String COLUMN_STUDENT_EMAIL = "student_email";
        public static final String COLUMN_STUDENT_SCORE_TOTAAL = "student_score_totaal";
    }

}
```

Stap 3: maak de klasse 'StudentsLoader' in een nieuwe package 'be.howest.nmct.evaluationsstudents.loader'. Laat deze klasse erven van AsyncTaskLoader<Cursor>. De verplicht uit te werken methode is 'loadInBackground'. Andere nuttige methode is 'onStartLoading' waar kan nagegaan worden of er reeds data ingeladen werd en/of de content gewijzigd is.

```java
public class StudentsLoader extends AsyncTaskLoader<Cursor> {

    private Cursor mCursor;

    private final String[] mColumnNames = new String[]{
            BaseColumns._ID,
            Contract.StudentColumns.COLUMN_STUDENT_NAAM,
            Contract.StudentColumns.COLUMN_STUDENT_VOORNAAM,
            Contract.StudentColumns.COLUMN_STUDENT_EMAIL,
            Contract.StudentColumns.COLUMN_STUDENT_SCORE_TOTAAL};


    private static Object lock = new Object();

    public StudentsLoader(Context context) {
        super(context);
    }


    @Override
    protected void onStartLoading() {
        if (mCursor != null) {
            deliverResult(mCursor);
        }
        if (takeContentChanged() || mCursor == null) {
            forceLoad();
        }
    }

    @Override
    public Cursor loadInBackground() {
        if (mCursor == null) {
            loadCursor();
        }
        return mCursor;
    }
```

Onderstaande methode creëert een MatrixCursor.
http://developer.android.com/reference/android/database/MatrixCursor.html
Via de newRow-methode wordt een nieuwe rij gecreëerd. Vul deze volledig op, in overeenstemming met de opgegeven kolommen.

```
private void loadCursor() {
    synchronized (lock) {
        if (mCursor != null) return;

        MatrixCursor cursor = new MatrixCursor(mColumnNames);
        int id = 1;
        for (Student student : StudentAdmin.getInstance().getStudenten()) {
            MatrixCursor.RowBuilder row = cursor.newRow();
            row.add(id);
            row.add(student.getNaamStudent());
            row.add(student.getVoornaamStudent());
            row.add(student.getEmailStudent());
            row.add(Math.round(student.getTotaleScoreStudent()));
            id++;
        }
        mCursor = cursor;
    }
}
```

De methode 'loadStudenten' genereert testdata. Deze methode vindt u in het bronmateriaal terug (zie txt-file).

Stap 4: bouw in de StudentFragment een innerklasse '**StudentAdaptar**'. Deze klasse erft van 'SimpleCursorAdapter' en staat in voor het visualiseren van data afkomstig van een Cursor.

http://developer.android.com/reference/android/widget/SimpleCursorAdapter.html

Waarom is een eigen versie nodig? We wensen naast het mappen van de data-velden naar de juiste Textviews ook de afbeelding doen differentiëren. Indien dit niet nodig is, hoeft een eigen klasse niet.

```
class StudentAdaptar extends SimpleCursorAdapter {

    private int layout;

    public StudentAdaptar(Context context, int layout, Cursor c, String[] from, int[] to, int flags) {
        super(context, layout, c, from, to, flags);
        this.layout = layout;
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        final LayoutInflater inflater = LayoutInflater.from(context);
        View row = inflater.inflate(layout, parent, false);
        ImageView icon = (ImageView) row.findViewById(R.id.imageViewStudent);

        int colnr = cursor.getColumnIndex(Contract.StudentColumns.COLUMN_STUDENT_SCORE_TOTAAL);

        if (cursor.getDouble(colnr) < 8) {
            icon.setImageResource(R.drawable.student_red);
        } else if (cursor.getDouble(colnr) < 10) {
            icon.setImageResource(R.drawable.student_orange);
        } else {
            icon.setImageResource(R.drawable.student_green);
        }

        return row;
    }
}
```

Stap 5: zorg ervoor dat de Fragmentklasse uitgerust wordt om asynchroon data in te laden. Hiervoor maken we gebruik van de klassen Studentsloader en StudentAdaptar. Alles start met het fragment de juiste interface te laten implementeren:

```
public class StudentsFragment extends ListFragment implements LoaderManager.LoaderCallbacks<Cursor> {
```

Hierdoor kan het Fragment communiceren met zijn loader.
http://developer.android.com/reference/android/app/LoaderManager.LoaderCallbacks.html

Baseer je op het voorbeeld op onderstaande link om jouw eigen Fragment-klasse verder aan te vullen.
http://developer.android.com/guide/components/*loaders*.html

- In de methode *onActivityCreated*: maak een lege StudentAdapter dat we zullen gebruiken om de data te tonen. Koppel deze al aan de listview. Roep de methode initLoader van de LoaderManager aan (om zo er zeker van te zijn dat de loader bestaat, indien niet wordt deze aangemaakt, in het andere geval wordt de laatste hergebruikt)
- In de methode *onCreateLoader* (door interface): deze methode zal een nieuwe loader aanmaken. Maak hier gebruik van jouw klasse Studentsloader. De loader geeft een cursor naar de weer te geven data terug.
- In de methode *onLoadFinished* is data opgehaald en wordt de adaptar aan de cursor gekoppeld.
- De methode *onLoaderReset* wordt opgeroepen wanneer eerder aangemaakte cursor gesloten wordt. Data is onbeschikbaar nu.

Test alles voldoende uit.

In het volgende labo wordt de detail verder uitgewerkt.