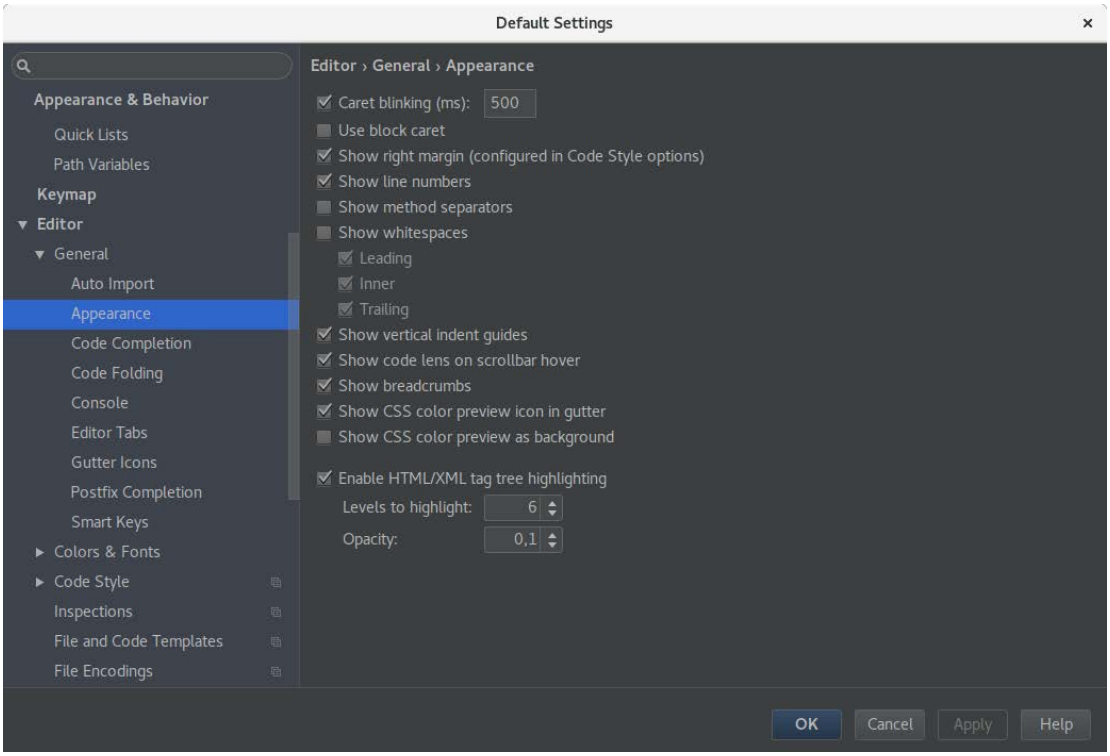
TP Android - Cardview

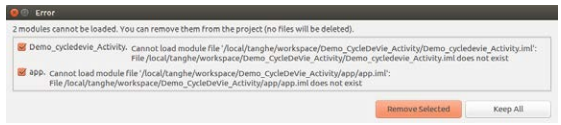
Avant-propos

Tout d'abord, je vous conseille fortement d'activer la numérotation des lignes dans Android Studio.
Allez, sur la page d'accueil d'Android Studio en fermant tous vos projets, cliquez sur *Configure*, puis settings et cochez la case *Show line numbers*, comme sur la capture d'écran ci-dessous.



Pour importer le projet depuis Android Studio, aller sur la page d'accueil et cliquer sur *Checkout project from version control* et choisisser *git*.
Entrer ensuite l'url https://github.com/qvandekadsye/AndroidCardViewTP.git et cliquer sur clone.
Ensuite, il faut se placer sur la branche « debutTP », sinon c'est de la triche !

Après importation, un message d'erreur de la sorte peut apparaître :



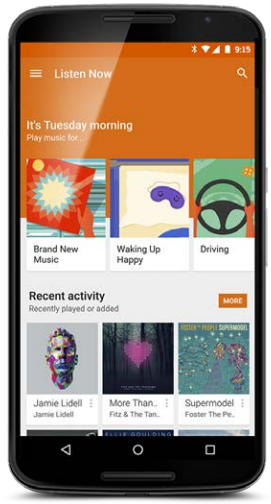
Cliquer sur Remove selected.

Objectif du TP

Bonjour à tous, l'objectif de ce TP est d'apprendre à afficher une ou plusieurs CardView et d'interagir avec.
Ce TP a été testé avec un Android 4.3, il est compatible 4.0 et supérieur.

Qu'est-ce qu'une CardView

La Cardview, vous l'avez forcément déjà vue, elle est notamment utilisée dans l'application Google pour afficher des informations.
C'est à ça qu'elle peut ressembler :



La CardView permet donc d'afficher des informations sous forme de cartes, d'où son nom. ;)

C'est l'heure de coder !

Bon ! Les explications, c'est sympa 30 secondes, mais on sait tous ici

qu'il n'y rien de mieux que de mettre les mains sur le clavier.
A la fin de ce TP, vous aurez une petite bibliothèque musicale, qui affichera une liste d'albums musicaux (mais qui ne jouera pas de la musique, ce n'est pas le sujet du TP).
Vous êtes prêts ?

Gestion des dépendances

la CardView fait partie de la bibliothèque "support V7", on doit préciser que nous allons utiliser cette bibliothèque dans le build.gradle (app). Comme ce qu'il suit :

```
dependencies {
    //Ajouter ces deux lignes dans le build.gradle de l'app
    compile 'com.android.support:cardview-v7:24.1.1'
    compile 'com.android.support:recyclerview-v7:24.1.1'
}
```

build.gradle (app) hosted with by GitHub view raw

La version 24 est utilisée car dans notre projet, le targetSDK est 24.

C'est fait ? Super, vous allez enfin pouvoir coder. ;)

Affichage d'une CardView

Comme dit précédemment, une CardView permet d'afficher des informations sous forme de carte. En réalité, une CardView est un **Layout**, plus précisément un **FrameLayout** (Cela aura son importance plus tard).
Nous allons donc pouvoir y inclure d'autres View. Comme nous voulons afficher les informations d'un album musical, notre CardView sera composée des View suivantes :

1. Une ImageView pour afficher la jaquette de l'album;
2. Un premier TextView pour afficher le titre de l'album;
3. Un deuxième pour afficher l'artiste;
4. Un troisième pour afficher l'année de parution;
5. Et un dernier pour le genre musical.

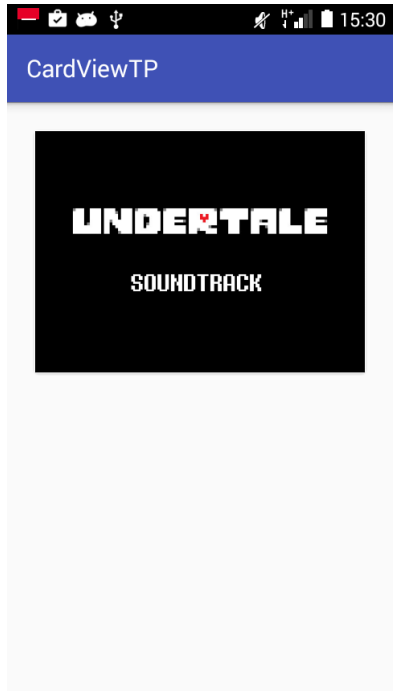
Comme je suis généreux, les layouts portrait et paysage sont déjà codés. Le code doit ressembler à ceci :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.quentinvdk.cardviewtp.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
       <android.support.v7.widget.CardView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp">

                <ImageView
                    android:layout_width="match_parent"
                    android:layout_height="200dp"
                    android:scaleType="centerCrop"
                    android:src="@drawable/undertale_ost"/>

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:text="@string/first_album_name"
                    android:gravity="start"
                    android:textSize="20sp"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:text="@string/first_album_author"/>
                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
```

```
41                      android:text="@string/first_album_year"/>
42                  <TextView
43                      android:layout_width="match_parent"
44                      android:layout_height="wrap_content"
45                      android:text="@string/first_album_type"/>
46          </android.support.v7.widget.CardView>
47      </LinearLayout>
48  </RelativeLayout>
```

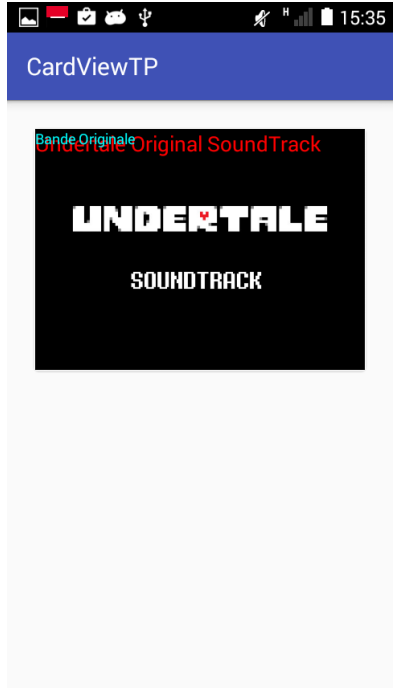
activity_main.xml hosted with by GitHub view raw

Essayer d'executer le code sur votre émulateur/appareil pour voir !
...
Ce n'est pas terrible, n'est-ce pas ?
Vous devriez obtenir ceci :



Comme vous le voyez, il n'y a que l'image... Enfin...

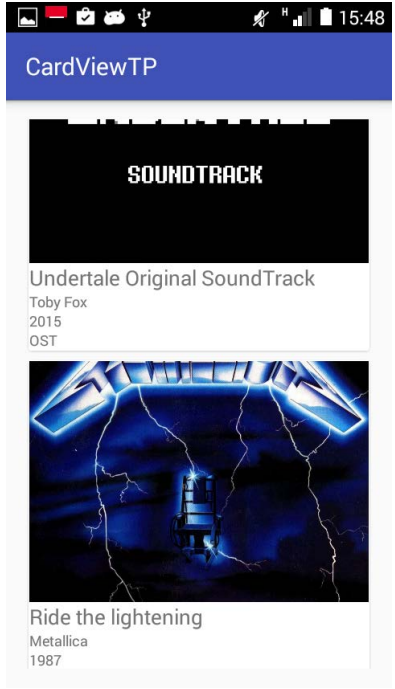
En modifiant les attributs « textColor » des TextView, voici ce qu'on obtient :



Les éléments enfants de la CardView sont en fait superposés les uns sur les autres.
Et là vous devez vous écriez : « Mais pourquoi ?!! »

C'est très simple : Vous vous souvenez qu'une CardView est une **FrameLayout** ?
Eh bien, c'est peut-être un détail pour vous, mais pour la CardWiew ça veut dire beaucoup !

Un **FrameLayout** est un Layout qui n'a **AUCUNE** incidence sur la disposition de ses élements enfants. Ce qui explique la superposition des enfants.
Mais alors, comment corriger cela ? Tout simplement en ajoutant à la CardView un layout type *Linear* et d'y placer les éléments enfants.
Décommenter, les lignes concernant un linear layout ainsi que sa balise fermante et admirer le résultat :

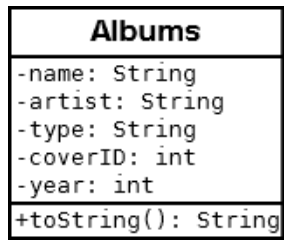


Voilà, vous avez une CardView qui affiche les informations d'un album !
Ce TP est donc ter... Pardon ? Vous avez plus qu'un album dans votre bibliothèque ? Vous voulez donc une liste de CardWiew qui est générée en fonction d'une autre liste ?
...
Bon d'accord !

CardView et RecyclerView

Si le planning des TP a bien été respecté, alors normalement, hier, nous avons dû effectuer le TP sur le RecyclerView.

La classe qui va nous servir de modèle est la classe *Albums*, se trouvant dans le package *com.example.quentinvdk.cardviewtp.JavaClasses*
Voici son descriptif :



La jaquette sera représentée par l'id de la ressource. La méthode *ToString* nous sera utile plus tard.

Dans le fichier *activity-main.xml*, supprimez les lignes de la ligne 13 à 56.
Il ne doit rester que ceci :

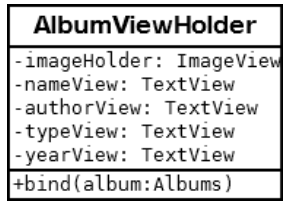
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.quentinvdk.cardviewtp.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/albumRecyclerView">

        </android.support.v7.widget.RecyclerView>
    </LinearLayout>
</RelativeLayout>
```

activity_main.xml hosted with by GitHub view raw

Maintenant que le recyclerView est créé sur le layout principal, il est nécessaire d'avoir un layout dont le rôle sera de servir de motif d'affichage pour le recyclerWiew.
C'est à cela que sert le fichier de layout : *list_cardwiew.xml*
Comme vous le voyez, ce fichier contient le même code que notre fichier *activity-main.xml* auparavant.

Maintenant que notre vue et notre modèle sont prêts, il est temps de les lier !
Pour cela, nous allons utiliser un ***Adapter***. Pour ce TP nous allons utiliser *L'albumAdapter* créé pour l'occasion, que vous trouverez dans le package :
com.example.quentinvdk.cardviewtp.JavaClasses.customAdapter

Cette classe *AlbumAdapter* hérite de la classe *AlbumAdapter.AlbumViewHolder*, où *AlbumAdapter.AlbumViewHolder* est une classe statique interne à la classe *AlbumAdapter*.
En voici le schéma :

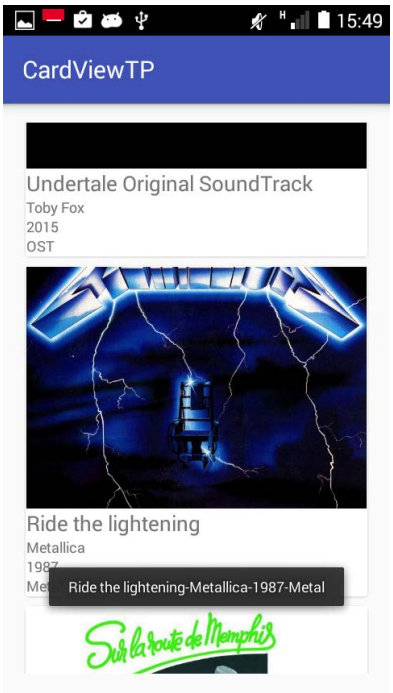


Comme indiqué, la classe *AlbumViewHolder* possède un constructeur et une méthode nommée *Bind*.
Le constructeur va classiquement initialiser les attributs de la classe qui ici sont des Views. Pour ce faire, décommenter le lignes *60* à *64*.
La méthode bind va nous permettre de lier un objet à un ensemble de View. Décommenter les lignes *71* à *76*. Cette méthode va être utiliser dans la méthode *OnBindViewHolder*, pour cela décommenter les lignes *36* et *37*.

Retourner maintenant dans votre *MainActivity.java* décommentez les lignes *24* à *28*. Lancer l'application sur votre émulateur ou votre smartphone et admirez le résultat !
Vous pouvez ensuite retirer un ajout dans la méthode initApp() et voir ce qu'il se passe, (après recompilation).

Il ne nous reste plus qu'une chose à faire : rendre interactive les CardView :
Pour cela dans le fichier *AlbumAdapter.java*, décommenter les lignes *38* à *43*, qui ajouteront un écouteur Onclick à chaque cardView.

Compiler, tester, toucher une CardView et vous devriez avoir un résultat similaire :



Voilà, ce TP est terminé. Vous savez maintenant créer une CardView, en générer selon une liste et gérer le fait de pouvoir cliquer dessus.

Le code final est disponible sur la branche « Master »

Liens utiles

- Le tutoriel, sur lequel je me suis basé en partie : http://tutos-android-france.com/material-design-recyclerview-et-cardview/
- La documentation de Google : https://developer.android.com/training/material/lists-cards.html
- Si vous le souhaitez, le dépot possède une branche "Testing", sur laquelle, le teste le fait de pouvoir rajouter un menu sur chaque CardView.