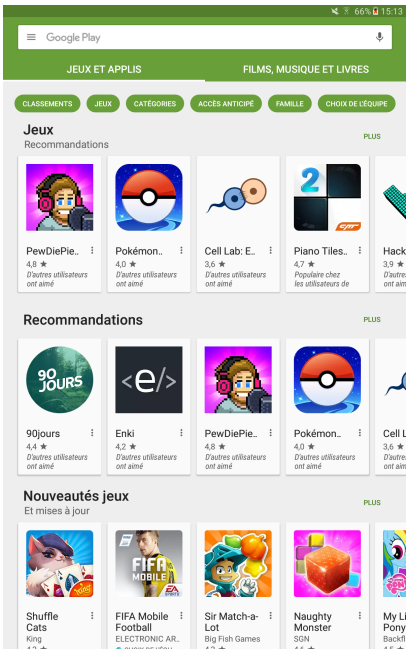
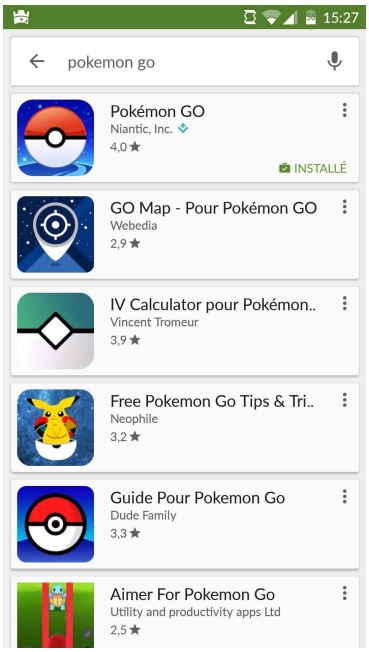
Les CardView

L'utilisation des CardView est un moyen simple de rendre l'affichage de données plus beau, que ce soit pour un seul élément ou pour tous les éléments présents dans une ListView ou un RecyclerView.
Si vous avez déjà utilisé un appareil Android, vous avez sûrement déjà vu des CardView. L'exemple le plus simple d'utilisation de CardView est le Play Store qui les utilise sous différentes formes pour afficher ses applications.



PAGE D'ACCUEIL DU PLAY STORE



PAGE DE RECHERCHE DU PLAY STORE

Le contenu du TP

Le TP contient 2 dossiers : le dossier « TPCardView » qui est le projet Android à ouvrir dans Android Studio, et le dossier « Code final » qui contient le code final des fichiers à modifier. Après avoir ouvert le projet TPCardView dans Android Studio, commencez par lancer le projet pour vérifier que tout se passe bien et qu'il n'y a pas de problème avec les SDK minimum et maximum configurés. Si c'est le cas, créez un nouveau projet et copiez/collez les fichiers de TPCardView à l'intérieur de celui-ci. (ne pas oublier d'ajouter

```
compile 'com.android.support:cardview-v7:23.2.0'
```

et

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

dans le build.gradle de l'application)

Le but du TP

L'application que nous allons réaliser dans ce TP est une application très simple qui affiche une liste de quelques monuments (Tour Eiffel, Taj Mahal, etc…) rangés dans des CardViews avec leur photo et leur nom.

RÉSULTAT FINAL

1ère partie : le layout

Si vous cliquez sur le bouton « RecyclerView simple », une nouvelle Activity s'ouvre avec le contenu et une gestion de clic basique géré. Cependant, si vous cliquez sur le bouton « RecyclerView avec CardView », rien ne s'affiche. C'est normal.

Nous allons commencer par aller dans le fichier item_card.xml dans les layouts. Ce fichier représente le layout de chacun des éléments qui se trouvera dans la liste (chacune des lignes). On y trouve un élément CardView qui comporte les mêmes attributs que n'importe quel élément (layout_width, layout_height, layout_margin, etc…) ainsi que d'attributs spécifiques comme cardCornerRadius (application d'un arrondi plus ou moins important sur les coins), cardBakgroundColor (pour la couleur de fond de la CardView), etc…
Dans cet élément, on trouve un RelativeLayout dans lequel se trouve le contenu (ici une ImageView et un TextView). Décommentez le RelativeLayout et ouvrez à nouveau la deuxième page.
Vous pouvez maintenant voir apparaître les CardView (elles sont vides pour le moment, c'est normal).

2ème partie : le code

MainActivity.java

Commencez tout d'abord par ouvrir le fichier MainActivity.java. C'est dans cette Activity qu'est géré le clic sur les boutons renvoyant vers le RecyclerView.

PhotoActivity.java

Cette Activity est une classe toute simple qui ne contient qu'une ImageView permettant d'afficher une photo en plus grand que dans la RecyclerViewActivity.

RecyclerViewActivity.java

Ouvrez ensuite le fichier RecyclerViewActivity.java. C'est dans cette Activity que l'on récupère le RecyclerView, que l'on initialise les données qui seront à afficher dans notre liste et que l'on crée l'Adapter qui lui sera fourni, en fonction du type d'affichage à utiliser (simple ou CardView).

SimpleListAdapter.java et CardsListAdapter.java

Ouvrez maintenant le fichier CardsListAdapter (les deux fichiers sont semblables), c'est ici que tout se passe. Les Adapter fonctionnant uniquement avec des ArrayList, le contenu de notre HashMap est stocké sous forme d'EntrySet dans l'ArrayList mData.
L'affectation des valeurs des ImageView et TextView de notre CardView se passe dans la méthode onBindViewHolder().

Décommentez la première partie du code (de final Resources ligne 57 à descriptionTextView.setText() ligne 71).
Si vous relancez le projet, vous verrez que la liste s'affiche maintenant bien avec la photo et le nom de chaque monument.

Nous allons maintenant voir comment gérer le clic sur une CardView. Le but est, lorsque l'on clique sur une CardView, d'afficher une boîte de dialogue nous proposant deux actions : soit ouvrir la photo en plus grand dans une nouvelle Activity, soit lancer une recherche sur internet à propos de ce monument.
Pour ce faire, décommentez le code des lignes 74 à 101 (de holder.cardView.setOnClickListener() à la fin de la fonction onBindViewHolder()).

Vous pouvez relancer le projet et constater que vous pouvez en effet cliquer sur un monument et choisir d'afficher sa photo dans une nouvelle Activity.
Cependant, en cliquant sur le bouton pour effectuer une recherche, rien ne se passe. C'est en effet maintenant à vous de jouer !
(pour ouvrir internet, on crée un Intent avec en paramètres Intent.ACTION_VIEW et l'Uri à ouvrir. L'url d'une recherche Google est de la forme « http://www.google.fr/search?q=a+vous+de+jouer »)

URL ayant servi à la réalisation du TP

https://developer.android.com/training/material/lists-cards.html