TP 5

Interfaces utilisateurs avancées

-RecyclerView-

Pré requis:

Il est indispensable de bien connaître la notion de « Custom listView », RecyclerView, LinearLayoutManagere et StaggeredGridLayoutManager.

.

Objectifs:

- Utiliser les RecyclerView ;
- Implémenter l'événement OnItemTouch ;
- Utiliser le Layout Manager pour présenter le RecyclerView(Horizontal Vertical & Staggered) ;
- Utiliser les images Rondes (circleimageview) ;

Néto Graphie :

https://www.thedroidsonroids.com/blog/how-to-implement-a-recyclerview

https://github.com/googlesamples/android-RecyclerView/

https://www.androidhive.info/2016/01/android-working-with-recycler-view/

Table des matières

1 *ListView Simple & BackgroundColor*

Le but de cette application est d'ajouter une ListView offrant à l'utilisateur la possibilité de modifier la couleur d'arrière-plan de l'interface.

Pour se faire, vous aurez besoin de la méthode suivante :

```
View.setBackgroundColor(getResources().getColor(R.color.color_name))
```

DEMO de l'application [1]

1) Créez un projet Android et ajoutez à son interface une ListView.
2) Implémentez l'Adapter relatif à cette liste permettant d'avoir le résultat suivant (utilisez les fichiers ressources pour le contenu de la liste et pour les couleurs).

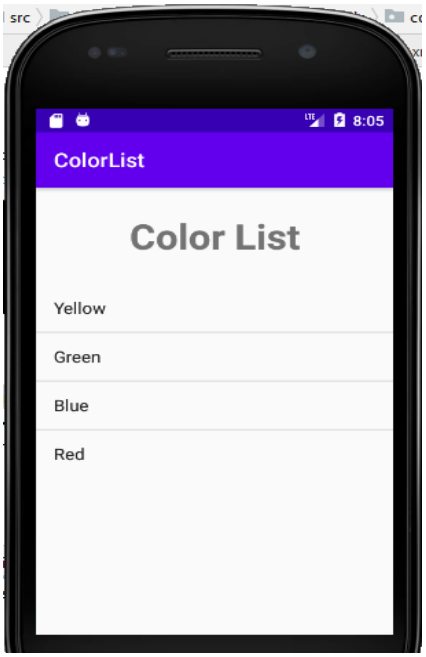


Figure 1: UI

3) Implémentez l'évènement onClick sur les Items de la liste permettant de changer la couleur de l'arrière-plan de l'interface.

Informations utiles :

a- L'événement onClick sur les Items est : **setOnItemClickListener.**

b- Pour récupérer l'item cliqué utilisez : adapterView.getItemAtPosition(i)

c- Ressource de type tableau :

Fichier res/values/strings.xml

La syntaxe est :

```
<string-array name="mon_tableau">
    <item>Element 1 </item>
    <item>Element 2</item>
</string-array>
```

Fichier java

Pour assigner cette valeur à un champ de votre interface :

```
String[] nom-tableau-java = getResources().getStringArray(R.array.mon_tableau_res);
```

Application 2 :Custom Movie RecyclerView

2

I.1. Implémentation du RecyclerView

Le but de cette partie est de créer l'application suivante => **TP1-video1.avi**

1) Créez un nouveau projet qu'on nommera « RcyclerViewMovie ».
2) Ajoutez à votre projet 3 classes comme suit :

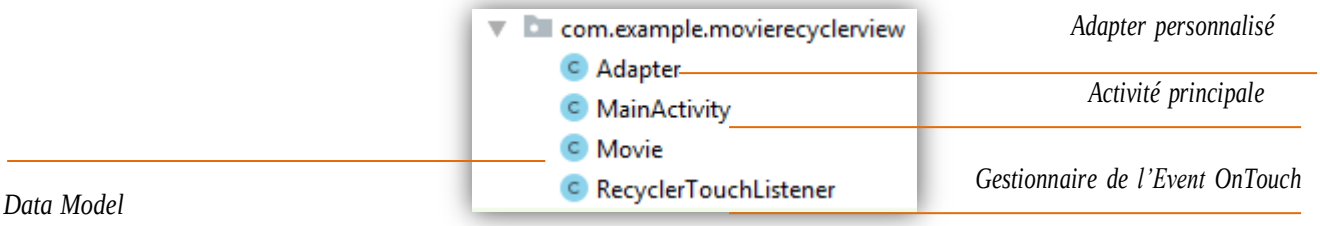


Figure 1: Structure du projet

DataModel : « Movie.java »

Cette application permet d'afficher la liste des films stockés dans une liste sous la forme suivante :

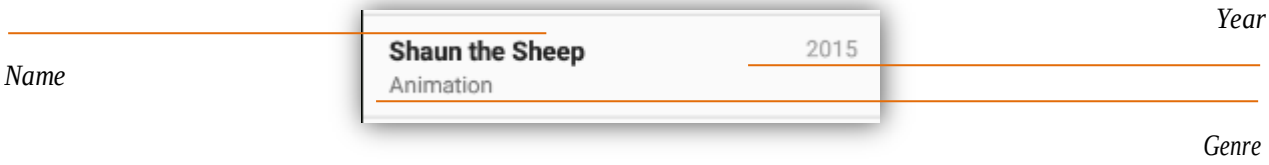


Figure 2: Ligne de la liste

3) Ajoutez à votre DataModel les attributs nécessaires (type String) et générez le constructeur et les getters.
4) Ajoutez aux ressources de type Layout un nouveau layout (my_row.xml) présentant une ligne de cette liste.

Activité principale : « MainActivity.java »

5) Pour pouvoir travailler avec les RecyclerView vous devez ajouter aux dépendances de votre projet (build.gradle-> Dependencies) la ligne suivante :

```
// RecyclerView
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

6) Ajoutez à votre interface un RecyclerView
7) Déclarez ce RecyclerView au niveau de votre activité Main.
8) Déclarez une liste de Movie :

```
List<Movie> movieList = new ArrayList<>();
```

9) Ajoutez à cette liste les films suivants :

```
Movie movie = new Movie("Mad Max: Fury Road", "Action & Adventure",
"2015");
movieList.add(movie);

movie = new Movie("Inside Out", "Animation, Kids & Family", "2015");
movieList.add(movie);

movie = new Movie("Star Wars: Episode VII - The Force Awakens",
"Action", "2015");
movieList.add(movie);

movie = new Movie("Shaun the Sheep", "Animation", "2015");
movieList.add(movie);

movie = new Movie("The Martian", "Science Fiction & Fantasy",
"2015");
movieList.add(movie);

movie = new Movie("Mission: Impossible Rogue Nation", "Action",
"2015");
movieList.add(movie);

movie = new Movie("Up", "Animation", "2009");
movieList.add(movie);

movie = new Movie("Star Trek", "Science Fiction", "2009");
movieList.add(movie);

movie = new Movie("The LEGO Movie", "Animation", "2014");
movieList.add(movie);

movie = new Movie("Iron Man", "Action & Adventure", "2008");
movieList.add(movie);

movie = new Movie("Aliens", "Science Fiction", "1986");
movieList.add(movie);

movie = new Movie("Chicken Run", "Animation", "2000");
movieList.add(movie);

movie = new Movie("Back to the Future", "Science Fiction", "1985");
movieList.add(movie);

movie = new Movie("Raiders of the Lost Ark", "Action & Adventure",
"1981");
movieList.add(movie);

movie = new Movie("Goldfinger", "Action & Adventure", "1965");
```

```
movieList.add(movie);

movie = new Movie("Guardians of the Galaxy", "Science Fiction &
Fantasy", "2014");
movieList.add(movie);
```

Classe Adapter: «Adapter.java »

10) Cette classe doit hériter de :

```
public class Adapter extends RecyclerView.Adapter<Adapter.MyViewHolder> {
}
```

11) Corrigez les erreurs en implémentant les méthodes nécessaires, ajoutant la classe MyViewHolder ainsi que le constructeur :

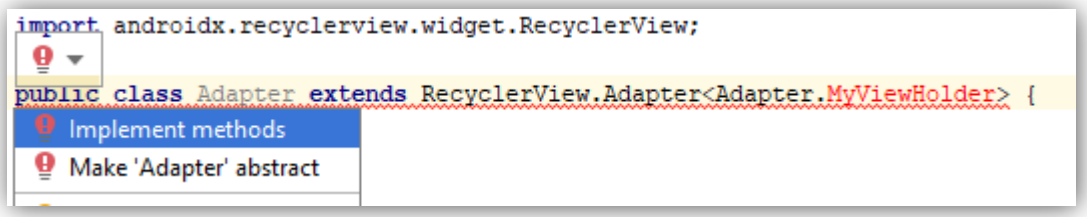


Figure 3: Implémentation des méthodes

12) Implémentez le code source relatif aux différentes méthodes :

- Constructeur :

```
public class Adapter extends RecyclerView.Adapter<Adapter.MyViewHolder> {
    private List<Movie> moviesList;
    //Constructor
    public Adapter(List<Movie> moviesList)
    {
        this.moviesList = moviesList;
    }
```

Figure 4: Code source du constructeur

- Méthode onCreateViewHolder
- Classe MyViewHolder
- Méthode onBindViewHolder
- Méthode getItemCount()

Activité principale : « MainActivity.java »

13) Complétez votre activité en créant une instance de l'Adaptateur, un LinearLayoutManager, assigner ce Layout au RecyclerView et enfin assigner l'adaptateur au RecyclerView.

14) Testez votre application

I.2. Implémentation de l'Evénement OnItemTouch

Classe « RecyclerTouchListener.java »

Le but maintenant est de gérer l'événement Touch.

Pour se faire, soit le code source de la classe « RecyclerTouchListener.java » :

```
public class RecyclerTouchListener implements
RecyclerView.OnItemTouchListener {
    private GestureDetector gestureDetector;
    private ClickListener clickListener;

public RecyclerTouchListener(Context context, final RecyclerView
recyclerView, final ClickListener clickListener) {
        this.clickListener = clickListener;
        gestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }
           @Override
            public void onLongPress(MotionEvent e) {
                View child = recyclerView.findChildViewUnder(e.getX(),
e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child,
recyclerView.getChildPosition(child));
                }
            }
        });
    }
    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null &&
gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildPosition(child));
        }
        return false;
    }
    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }
    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean
disallowIntercept) {
    }
    public interface ClickListener {
        void onClick(View view, int position);
        void onLongClick(View view, int position);
    }
}
```

15) Cette classe doit implémenter l'interface suivante:

```
public class RecyclerListener implements RecyclerView.OnItemTouchListener  {
}
```

Activité principale : « MainActivity.java »

Si l'utilisateur clique sur un Item de la liste, un Toast sera affiché comme le montre la vidéo « **TP1_video2.avi** »

16) Ajoutez l'événement OnItemTouchListener à votre RecyclerView comme suit :

```
// row click listener
recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getApplicationContext(), recyclerView, new RecyclerTouchListener.ClickListener() {
    @Override
    public void onClick(View view, int position) {
        Movie movie = movieList.get(position);
        Toast.makeText(getApplicationContext(), text: movie.getTitle() + " is selected!", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onLongClick(View view, int position) {

    }
}));
```

Figure 5: Event OnOtemTouchListener

17) Vérifiez le bon fonctionnement de votre application.

Application 3 : Horizontal and straggered Scroll 3

Le but de cet exercice est d'utiliser le layout Manager pour afficher une liste de films.

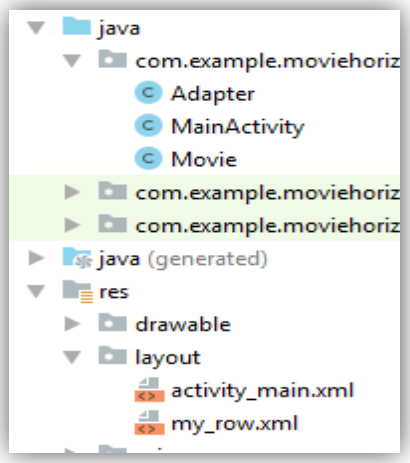
Chaque film est décrit par :

- **Title** : Titre du film de type String.
- **ImgURL** : Image du film sous la forme d'un lien de la ressource (dans le dossier Drawable) de type Integer.

  Exemple : `R.drawable.themartian.`

Soit le dossier Assests contenant les images à utiliser pour ce TP.

Votre projet aura la structure suivante :



Soit la liste des films à utiliser pour votre application :

```
//Déclaration de la liste
List<Movie> movieList = new ArrayList<>();
//Ajout des données à la liste
Movie movie = new Movie("Mad Max: Fury Road",R.drawable.madmax);
movieList.add(movie);

movie = new Movie("The Martian", R.drawable.themartian);
movieList.add(movie);

movie = new Movie("Shaun the Sheep",R.drawable.shaun);
movieList.add(movie);

movie = new Movie("Star Wars",R.drawable.starwars);
movieList.add(movie);

movie = new Movie("Inside Out",R.drawable.insideout);
movieList.add(movie);
```

II.1. RecyclerView Vertical

1) Implémentez le code source des différentes classes/activité pour avoir un rendu comme la figure 6 suivante :

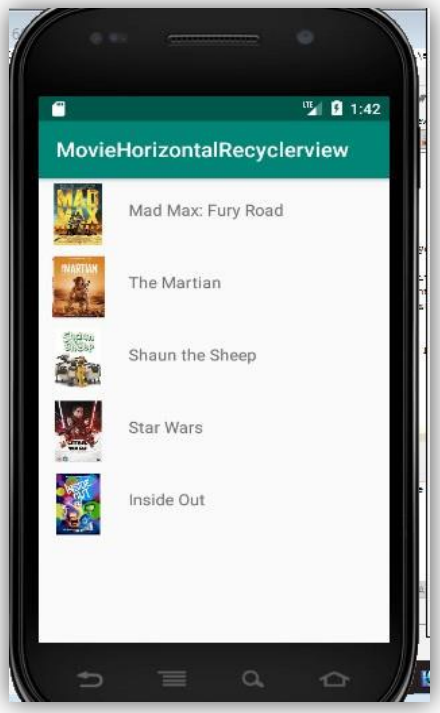


Figure 6: RecyclerView Verticale

II.2. Round/Circle Images

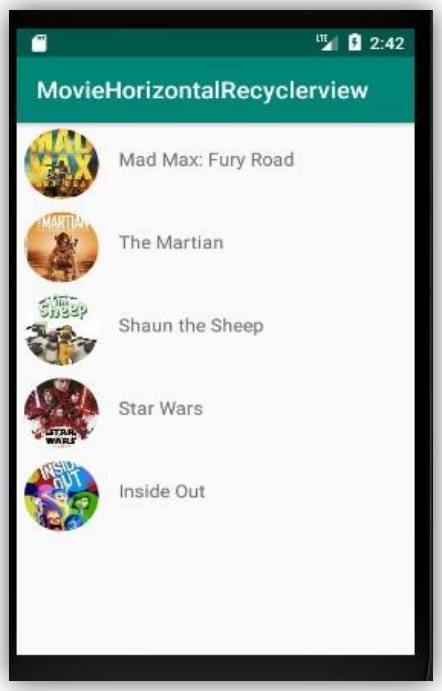


Figure 7: Images circulaires

Il est possible d'utiliser des ImageView circulaires sous Android. Pour se faire :

Gradle

```
dependencies {
    ...
implementation
'de.hdodenhof:circleimageview:2.2.0'

}
```

Utilisation

```
<de.hdodenhof.circleimageview.CircleImageView
…/>
```

Au lieu de la balise ImageView

2) Modifiez votre code source pour avoir le rendu comme la figure 7.

II.3. RecyclerView Horizontal

3) Modifiez votre code source pour avoir le rendu comme dans la vidéo « **TP1-video3.avi** ».

II.4. RecyclerView Straggered

4) Modifiez votre code source pour avoir le rendu comme la figure 8.

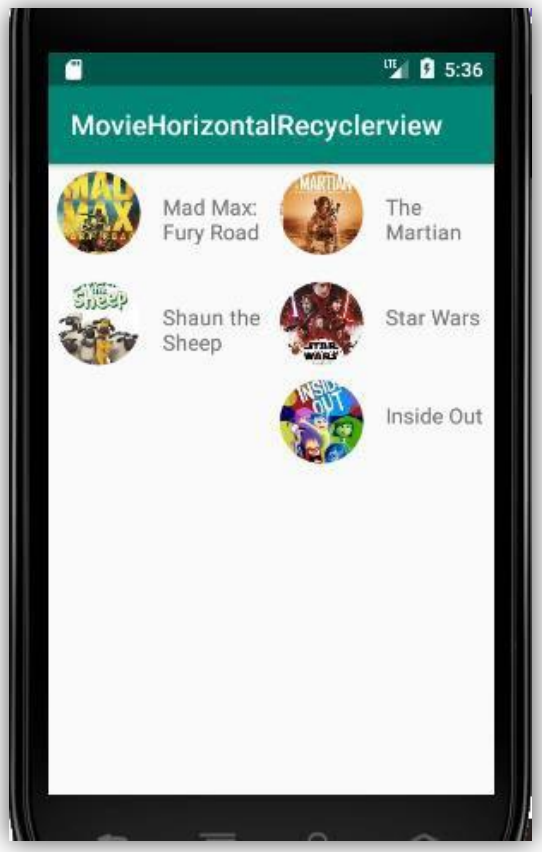


Figure 8: Manager Straggered

4

Liste Personnalisée : Android Version

Le but de cette application est de créer une liste personnalisée présentant les versions suivantes d'Android.

Utilisez la méthode suivante pour assigner une image à une ImageView :

```
View.setImageResource(int resource);
```

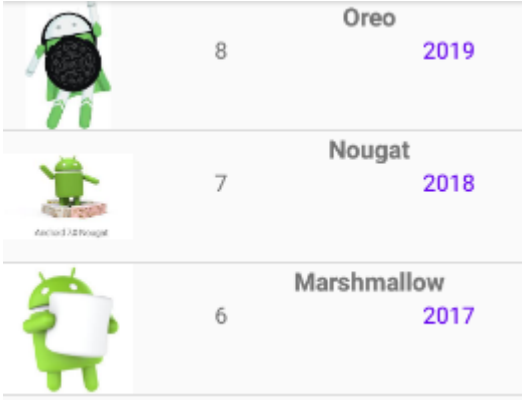


Figure 2: UI Android version

5

GridView & Liste personnalisée

Le but de cette application est de créer une galerie photo pour afficher des images en utilisant une GridView pour avoir le rendu suivant :

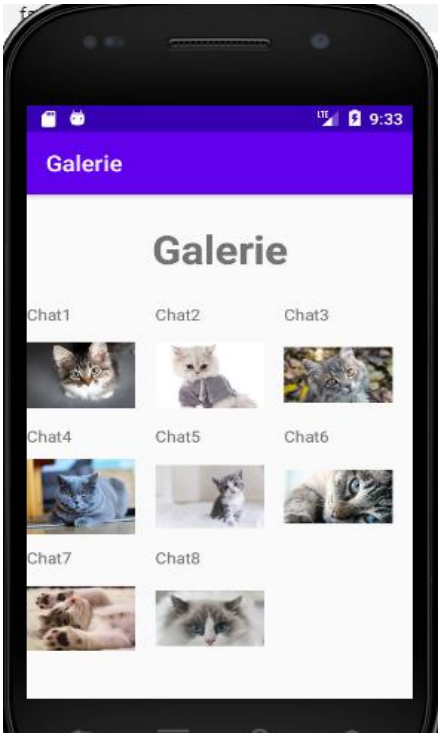


Figure 3: Rendu Galerie

"We may have all come on different ships, but we're in the same boat now."

-Martin Luther King Jr.