Applications mobiles: Interfaces graphiques

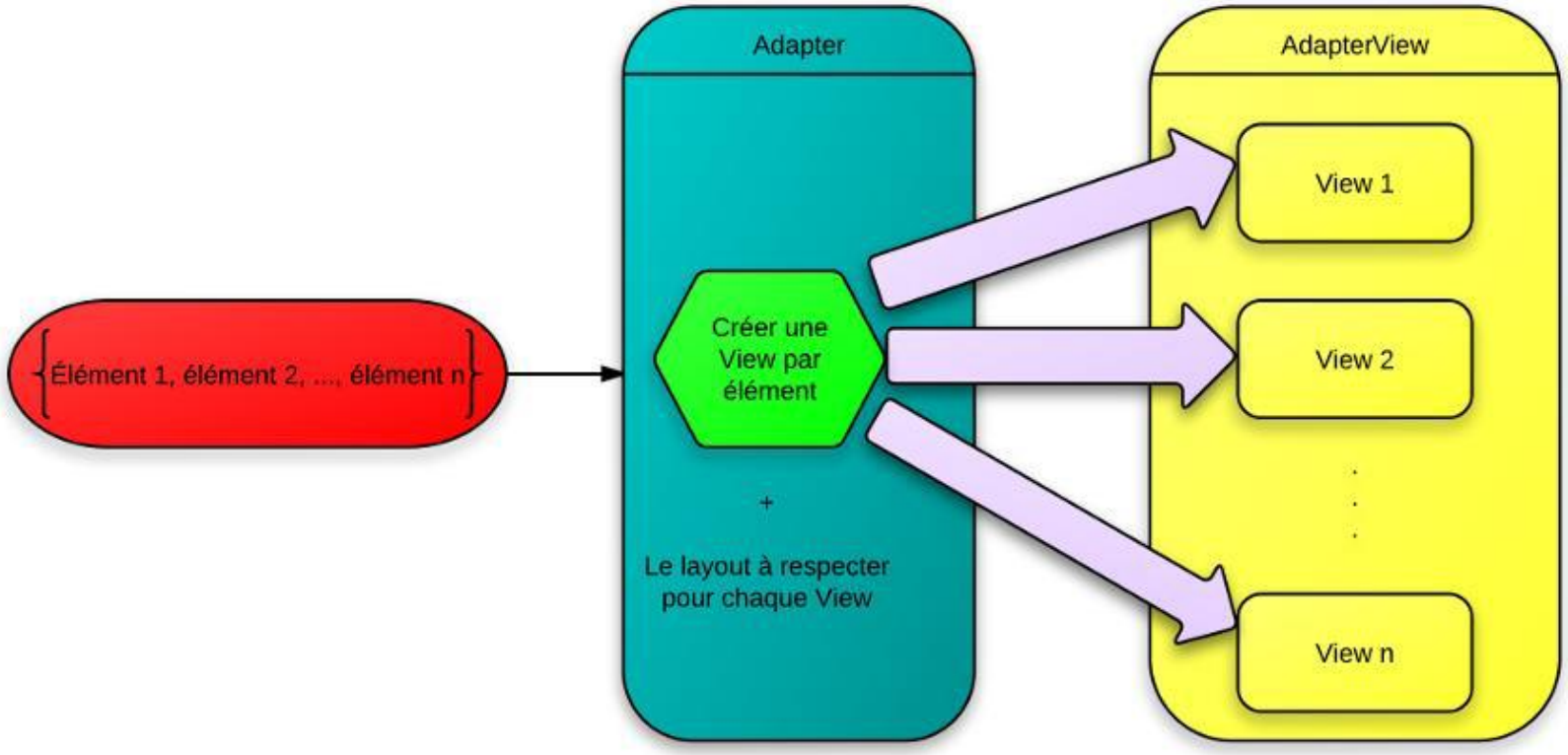
DÉPARTEMENT D'INFORMATIQUE – CENTRE UNIVERSITAIRE DE MILA

2017/2018

1. AdapterView

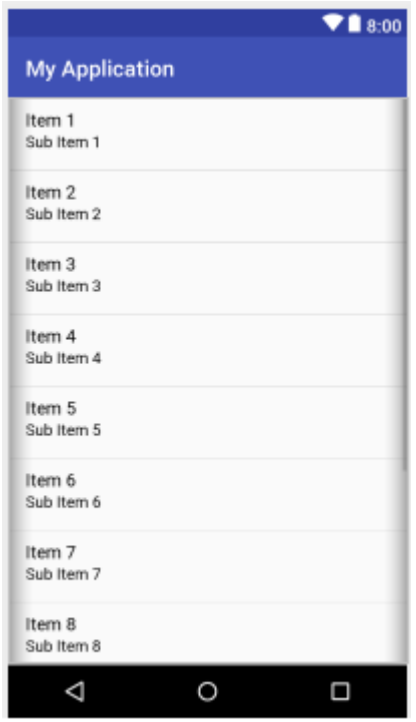
- Un AdapterView est un View Group qui permet d'afficher un contenu dynamique à partir d'une source de données attaché à un **adaptateur.**
- Le rôle de l'adaptateur comme son nom l'indique est de s'adapter à divers structure de données.
- Il joue le rôle d'intermédiaire entre les AdapterView et les données. Il récupère les données à partir des sources de données puis crée une View (ou ViewGroup) pour chaque élément.
-

1. AdapterView



2. Les listes simples (ListView)

- La forme la plus simple d'un ListeView est la liste simple ou la liste de TextView. La source de données dans ce cas est un tableau de chaines de caractères ou d'une liste dynamique de chaine (String[]/ArrayList<String>)



- Exemple
  - Soit le layout suivant qui contient un ListView:

2.1. Les listes simples (ListView)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.guettiche.myapplication.MainActivity">

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listep"/>
</LinearLayout>
```

NI: La liste est initialement vide c'est-à-dire ne contient aucun élément.

2.1 Les listes simples (ListView)

```
String
month[]={"January","February","March","April","May","June",
"July","August","Septembre",
"October","Novembre","Decembre"};
    }
```

- La Création de l'adaptateur

```
ArrayAdapter<String> MyAdapter=new
ArrayAdapter(this,
android.R.layout.simple_list_item_1,month);
```

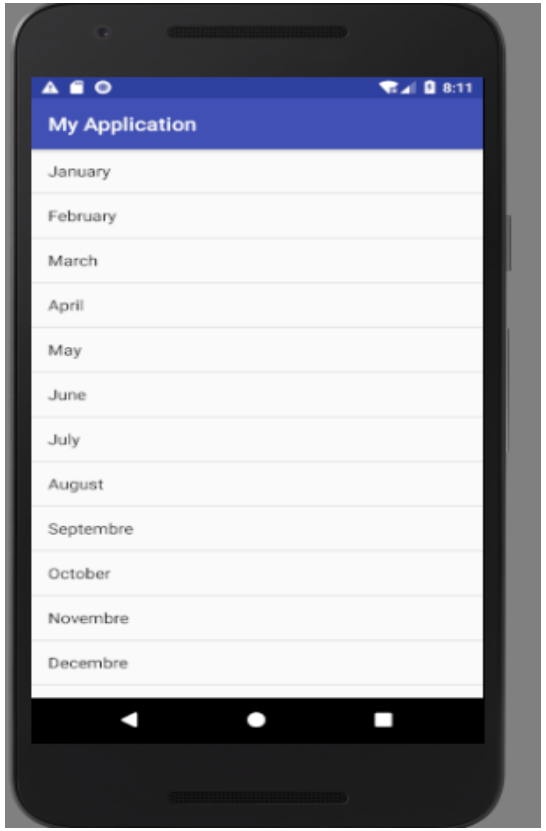
- Fournir l'adaptateur à la liste à l'aide de la méthode <setAdapter>

```
ListView lv=(ListView) findViewById(R.id.listep);
lv.setAdapter(MyAdapter);
```

2.1 Les listes simples (ListView)

- L’ exécution de cet exemple

est illustré sur la figure:



2.2. Listes personnalisées.

- Par exemple, on peut considérer que chaque élément de la liste est un contact et se caractérise par :
  - un nom
  - un prénom
  - un numéro (de téléphone).

2.2. Listes personnalisées.

Etape1: Créer une classe nommée **Contact.java**

```
public class Contact {
    public String nom;
    public String prenom;
    public String telephone;
    public Contact(String aNom, String aPrenom, String
aTelephone) {
        nom = aNom;
        prenom = aPrenom;
        telephone = aTelephone;

 }
```

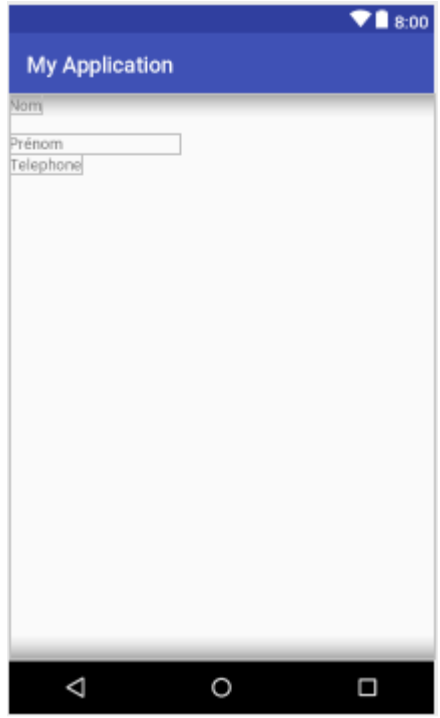
2.2.1 Implémentation de l'adaptateur personnalisé

- Le rôle de l'adaptateur est de créer les views correspondants aux éléments de la liste (ListView) à partir des données d'une source.
  - Etape3: Modifier le fichier **main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent«

tools:context="com.example.guettiche.myapplication.MainActivity">
```

2.2.1 Implémentation de l'adaptateur personnalisé

```
<ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listep"/>
</LinearLayout>
```



2.2.1 Implémentation de l'adaptateur personnalisé

- **Etape 2:** Création d'un fichier **contact_layout.xml**

- **Etape 3:** Nous allons créer un objet qui se chargera de gérer le mapping entre nos données et le layout des items. Ce composant sera basé sur un Adapter, objet qui hérite de base adapter.

```
public class customAdapter extends BaseAdapter {
    ArrayList< Contact > listIt=new ArrayList< Contact >();
    customAdapter(ArrayList<Personne> listIt) {
        this.listIt=listIt;
    }


    @Override
    public int getCount() {
        return listIt.size();
    }

    @Override
    public String getItem(int position) {
        return listIt.get(position).Name;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }
```

2.2.1 Custom Adapter

```
@Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflat=getLayoutInflater();
       View view1=inflat.inflate(R.layout.newview,null);
      TextView tv1= (TextView)view1.findViewById(R.id.textView2);
      TextView tv2= (TextView)view1.findViewById(R.id.textView3);
        tv1.setText(listIt.get(position).Name);
       tv2.setText(listIt.get(position).Tel);
        return view1;

    }

}
```

```
public class MainActivity extends AppCompatActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            Contact cc1=new Contact(« Abid", « Yahia","0658559139");
            Contact cc2=new Contact(« Abid", "Mohammed","0658559149");
            Contact cc3=new Contact("BenChikh", "Sadek","0338565896");
            Contact cc4=new Contact("Ben", « Oussama","0338565898");
            ArrayList<Contact> mylist=new ArrayList<Contact>();
            mylist.add(cc1);
            mylist.add(cc2);
            mylist.add(cc3);
            mylist.add(cc4);
            ListView lv=(ListView) findViewById(R.id.listep);
            ContactAdapter myAdapter;
            customAdapter myadapter=new customAdapter(listIt);
            lv.setAdapter(myAdapter);

        }
    }
```

2.2.2 Autre méthode public sur les adaptateurs

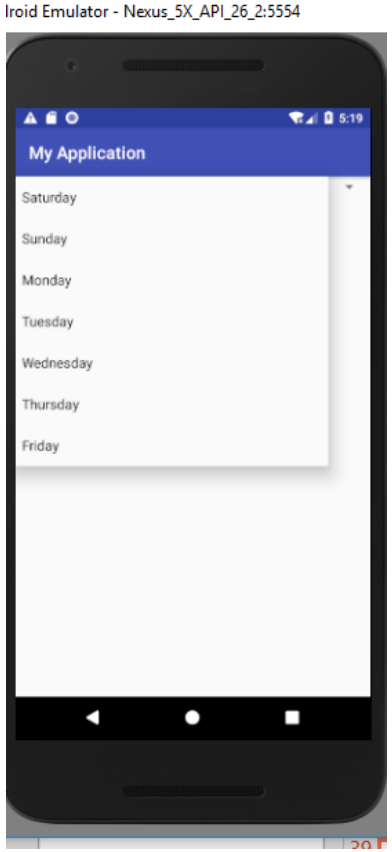
Void add(T object)	**Ajouter un objet à la fin**
Void addAll(T…items)	**Ajouter une liste d'objet à la fin**
T getItem(int position)	**Récupérer les donnée de l'élément à la position spécifiée**
Int getPosition(T item)	**Récupérer la position d'un élément**
Void insert (T object, int index)	**Insérer un objet à une position donnée**
Void notifyDataSetChanged()	**Notifier un changement dans les données Pour rafraichir les view qui les affichent**
Void remove(T object)	**Supprimer un élément**

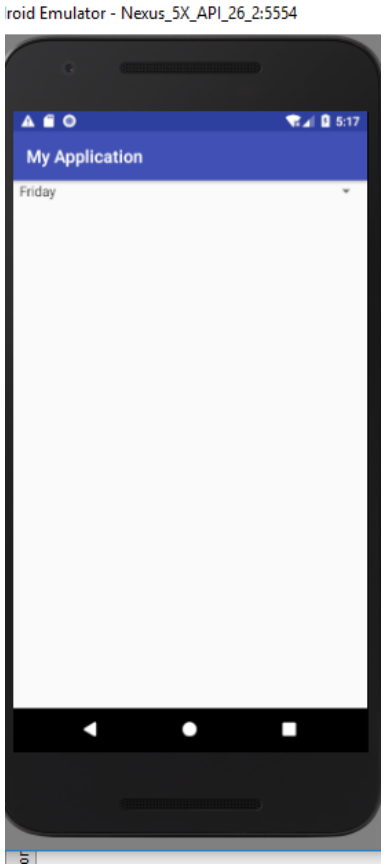
2.2.3 Opération sur les éléments

```
lv.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int position, long
id) {

// do some things


}

} );
```

3. Spinner (Liste déroulante)

```
<Spinner
android:layout_width="match_parent"
    android:layout_height="wrap_content"
         android:id="@+id/sp" />
```





3. Spinner

```
 String
Days[]={"Saturday","Sunday","Monday","Tuesday","Wednesday","Thursda
y","Friday"};
 Spinner sp=( Spinner) findViewById(R.id.sp);
 ArrayAdapter<String> adapter1;
 adapter1=new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1,Days);
adapter1.setDropDownViewResource(android.R.layout.
simple_spinner_item);
 sp.setAdapter(adapter1);
```

3. Spinner

- Notez l’utilisation de deux layout:

- android.R.layout.*simple_spinner_item: pour l’élément sélectionne qui est toujours visible.*

- android.R.layout.*simple_spinner_dropdown_item: pour les élément de la liste déroulante.*