IUT de Paris - Rives de Seine
Université Paris Cité

R4.Real.11 – Développement pour applications mobiles

– Cours 4 –

Boites de dialogue et notion de fragments

Pr Chaouche A.-C.
ac.chaouche@gmail.com

Résumé

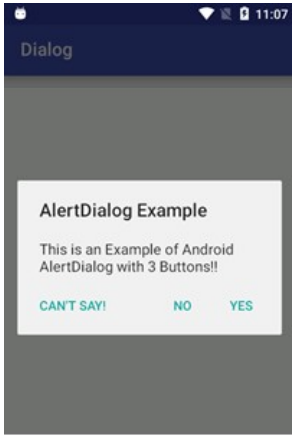
Prérequis

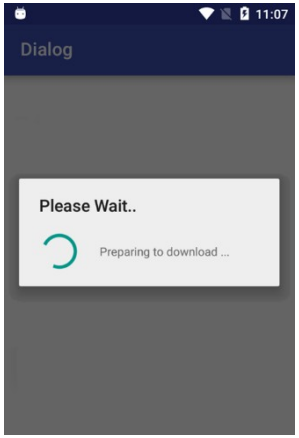
- Gestion des vues et des layouts
- Gestion des évènements

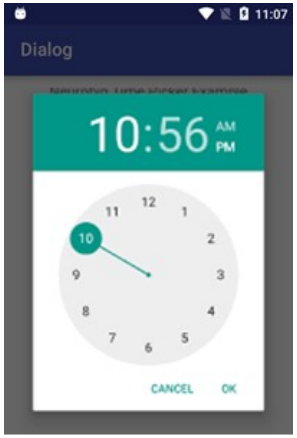
Objectifs du cours

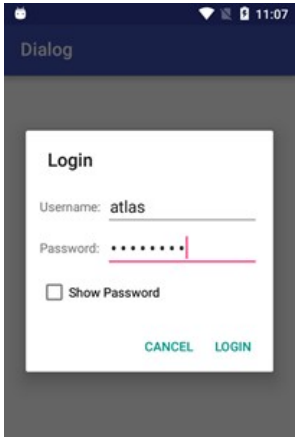
- Créer et afficher une boite de dialogue
- Comprendre le principe de fonctionnement des fragments
- Assimiler le cycle de vie d'un fragment

Boites de dialogue





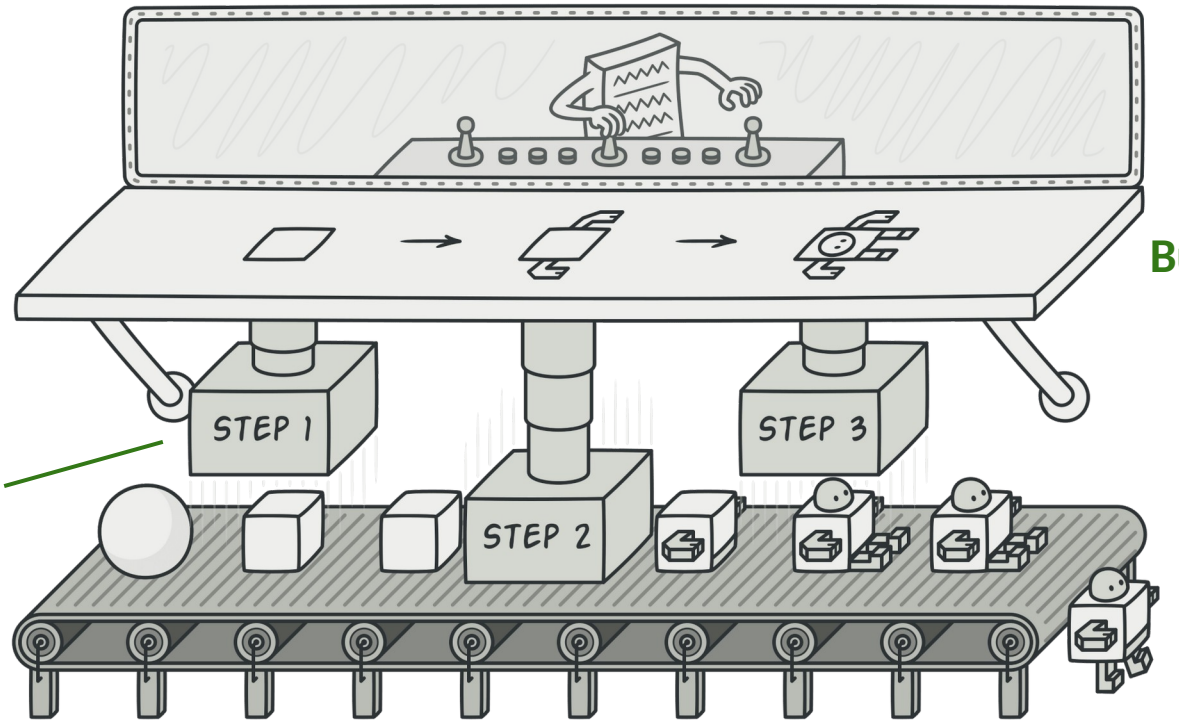




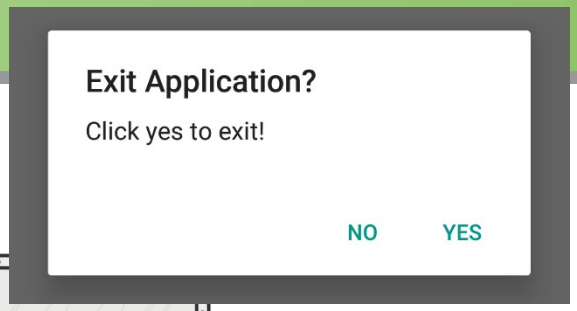
Boites de dialogue (`Dialog`)

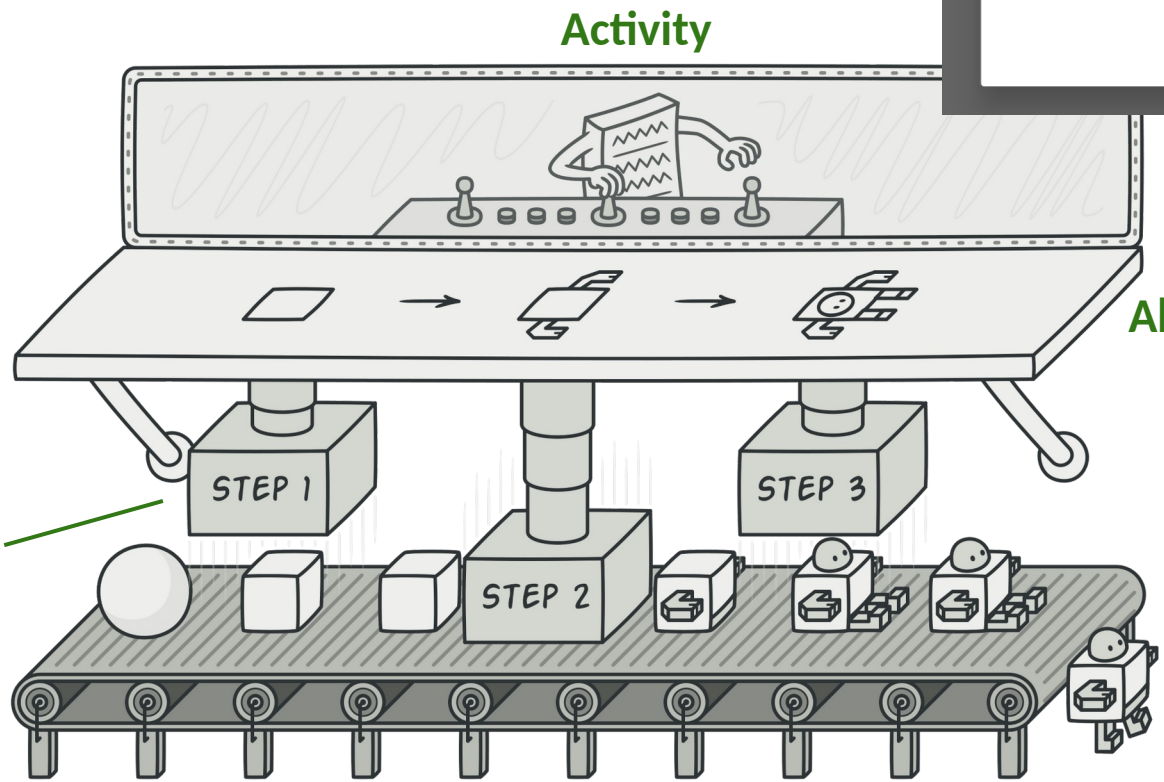
- **Fenêtre modale** qui possède des boutons pour interagir avec l'utilisateur
- **Les boites de dialogues** héritent de la classe **`android.app.Dialog`**
- Android fournit des boites de dialogues prédéfinies :
  - **`AlertDialog`** : alerte
  - **`ProgressDialog`** : information sur un traitement en cours
  - **`DatePickerDialog`** : choix d'une date
  - **`TimePickerDialog`** : choix d'une heure
  - **`... extends Dialog`** : personnalisée

Design pattern : Builder

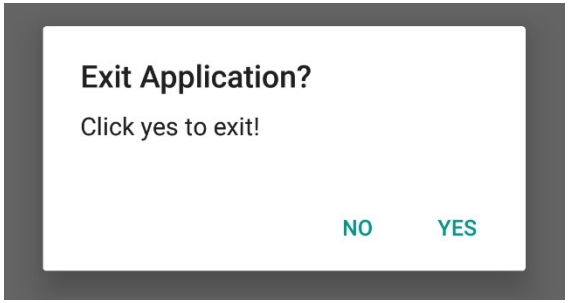


Design pattern : Builder



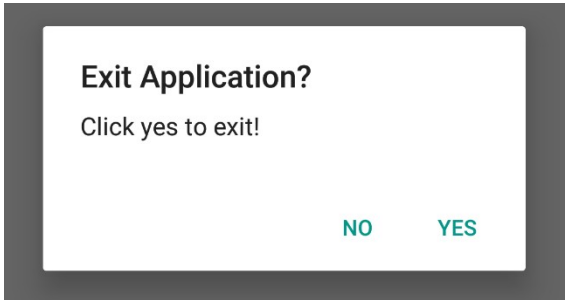


AlertDialog (1/2)



- La **classe interne `Builder`** permet de simplifier la construction de la boite de dialogue
- De nombreuses méthode permettent d'ajouter **un message, un titre, une liste d'items, etc.**
- L'affichage de la boite de dialogue est effectué avec **`show()`**
- La boite de dialogue est fermée avec **`dismiss()`** et **`cancel()`**

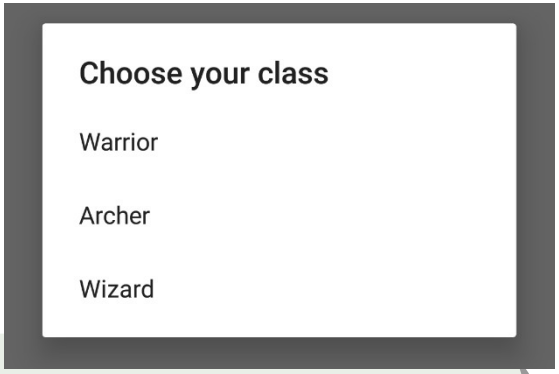
AlertDialog (2/2)



/java/MainActivity.java

```
...
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Exit Application?");
builder.setMessage("Click yes to exit!");
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener(){
          public void onClick(DialogInterface dialog, int id) { ... }
   });
builder.setNegativeButton("No", new DialogInterface.OnClickListener(){
          public void onClick(DialogInterface dialog, int id) { ... }
   });
AlertDialog dialog = builder.create();
dialog.show();
...
```

AlertDialog avec choix

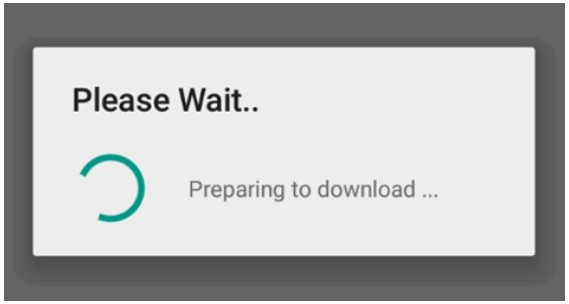


/java/MainActivity.java

```
...
String[] list = {"Warrior", "Archer", "Wizard"};

AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Choose your class");
builder.setItems(list, new DialogInterface.OnClickListener(){
          @Override
          public void onClick(DialogInterface dialog, int id) {
               Log.i("Dialog", "Position = " + id);
          }
    });
AlertDialog dialog = builder.create();
dialog.show();
...
```

ProgressDialog



/java/MainActivity.java

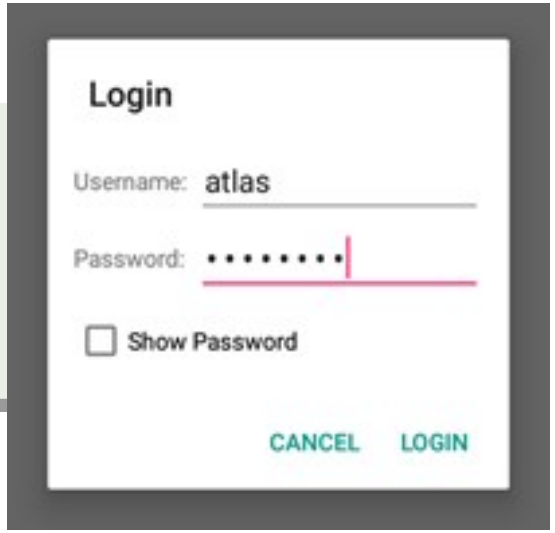
```
...
ProgressDialog dialog = new ProgressDialog(this);
dialog.setCancelable(true);
dialog.setIndeterminate(true);
dialog.setTitle("Please Wait...");
dialog.setMessage("Preparing to download...");
dialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
dialog.show();
...
```

Dialog personnalisés

1. Créer une boite de dialogue

/res/layout/dialog_custom.xml

```
<LinearLayout xmlns:android="http://schemas...
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    ...
</LinearLayout>
```

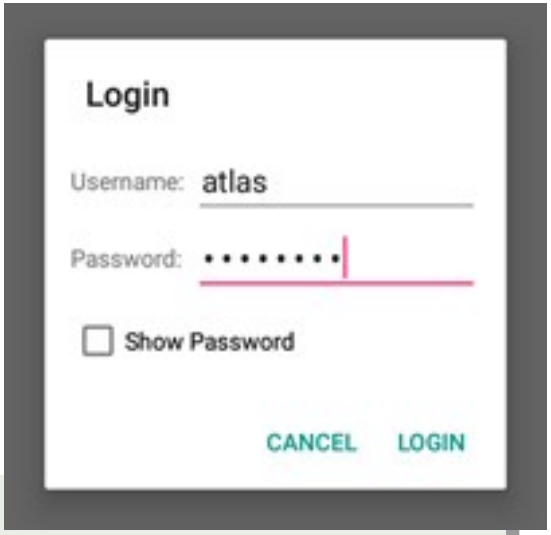


/java/CustomDialog.java

```
public class CustomDialog extends Dialog {
    public CustomDialog(Context context){ super(context); ... }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.dialog_custom);
        ...
    }
}
```

Dialog personnalisés

2. Lancer et fermer la boite de dialogue



/java/MainActivity.java

```
...
CustomDialog customDialog = new CustomDialog(MainActivity.this);
customDialog.show();
...
customDialog.dismiss();
```

Débogage sous Android Studio (1/2)

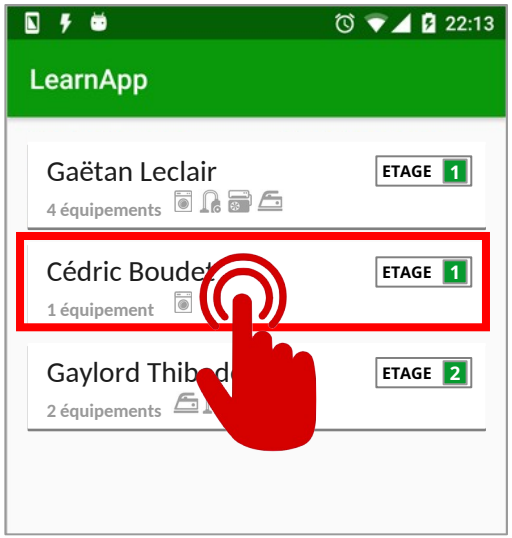
Etapes

1. **Dans l'éditeur Java > Mettre un point d'arrêt (breakpoint)**

```
40          friendsLV.setOnItemClickListener(new AdapterView.OnItemClickListener() {
41              @Override
42              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
43                  Log.i(TAG, "onItemClick: position " + position);
44              }
45          });
46      }
```

1. **Débuguer l'application en cliquant sur**

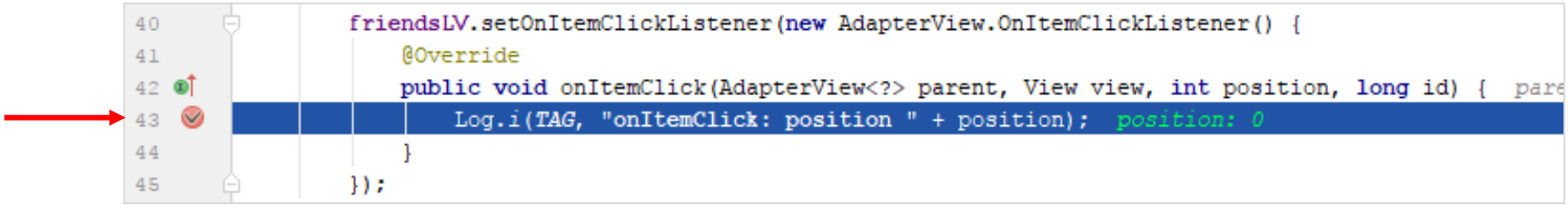
1. **Après l'execution de l'application > cliquer sur un élément**



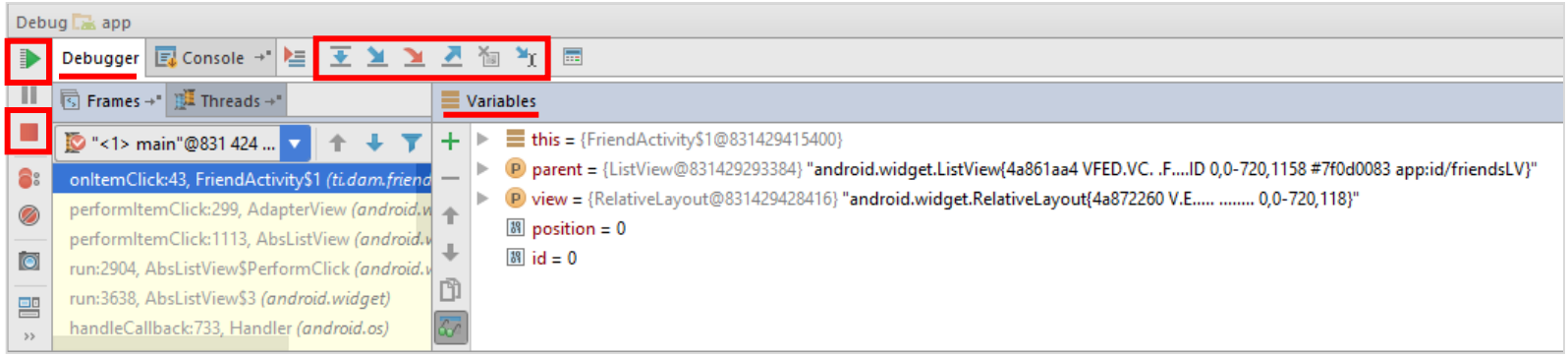
Débogage sous Android Studio (2/2)

Etapes

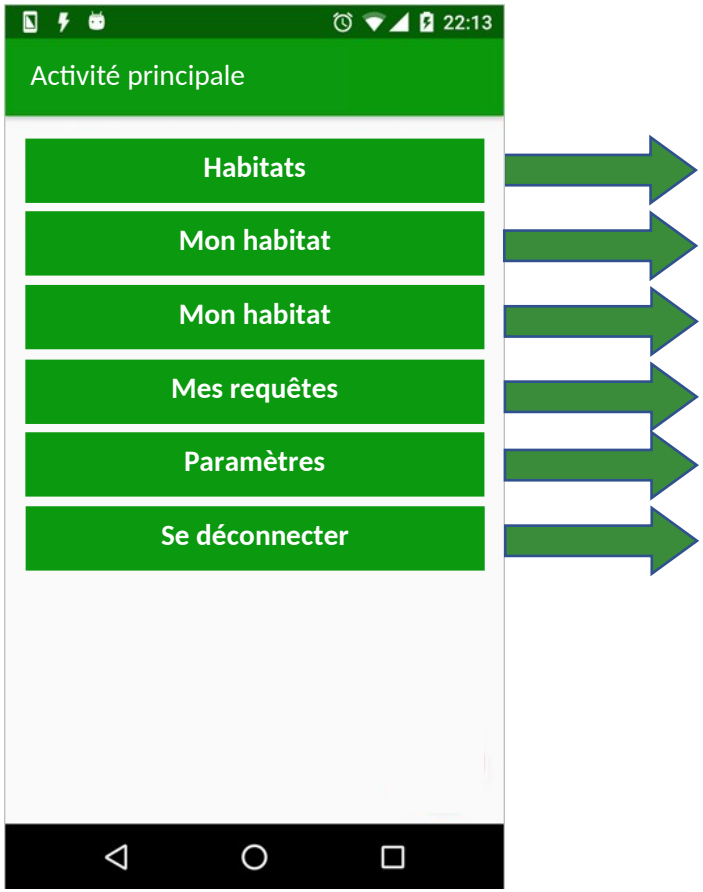
4. **Dans le code source > L'application est arrêtée dans le point d'arrêt**



4. **Dans Android Studio > une fenêtre "Debug" apparaît**



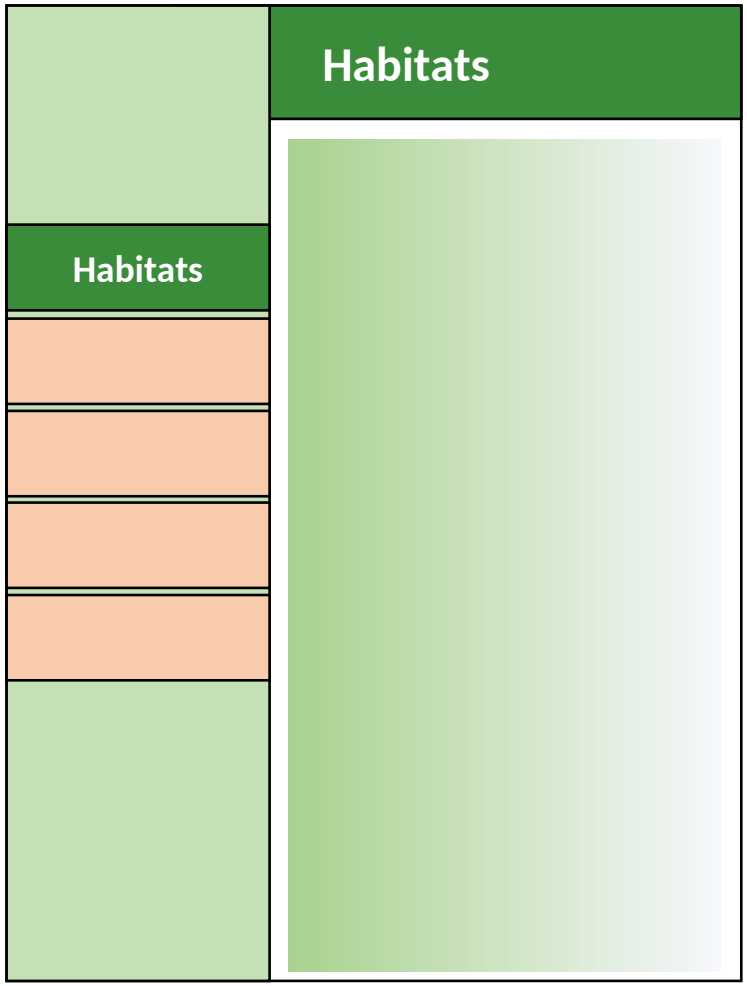
Problème : Utilisation des activités simples



Solution : Un menu déroulant principal

Création d'un `DrawerLayout` avec un menu :

- **Habitats (`HabitatsFragment`)**
- **Mon habitat,**
- **Mes requêtes,**
- **Paramètres,**
- **Se déconnecter, …**



Fragments

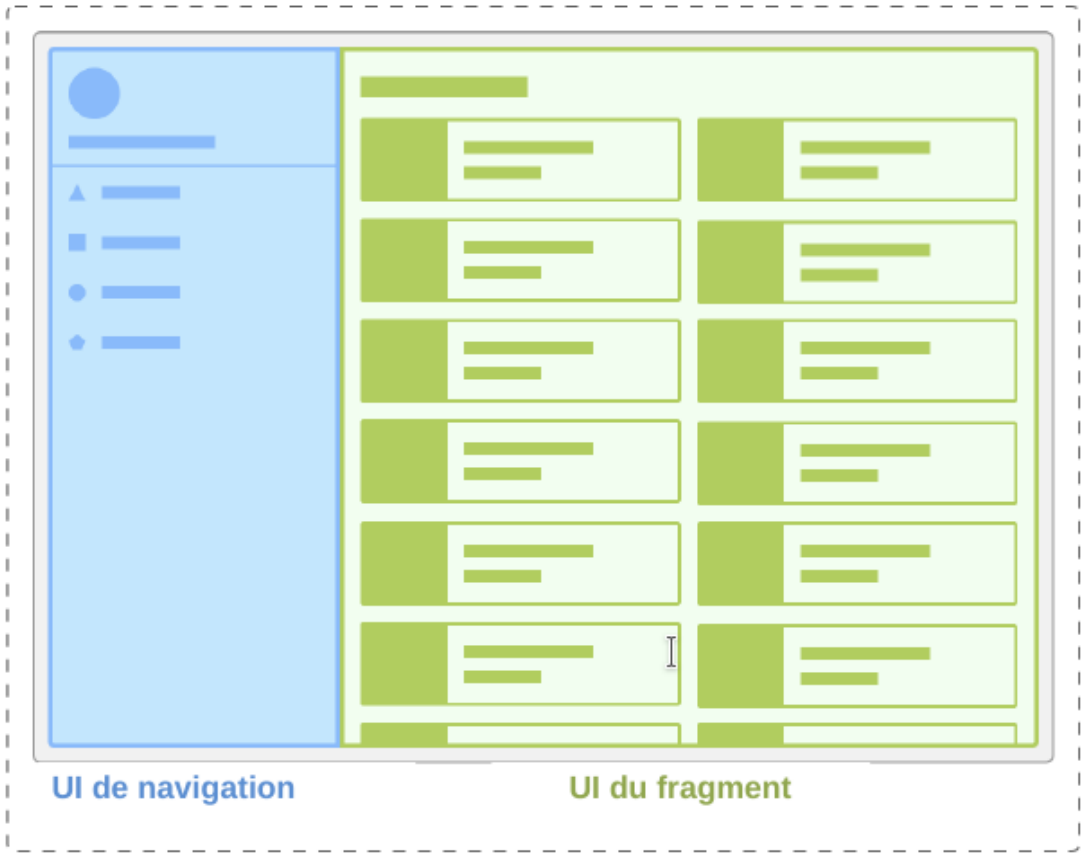
DrawerLayout / NavigationView

CoordinatorLayout

SwipeRefreshLayout

TabLayout / ViewPager

Fragments (1/3)



(a) Large écran

(b) Petit écran

Fragments (2/3)

Un Fragment représente une **portion d'interface utilisateur** dans une activité

- Il est ajouté dans un `ViewGroup` au sein de la hiérarchie d'une activité
- Sert à **adapter les interfaces** aux différentes résolutions des appareils
- Disponible depuis **Android 3.0 (API 11)**
  - grâce à l'**AppCompat** générée avec un projet Android, il peut être utilisé sur des **versions plus anciennes**
- Plusieurs fragments peuvent être combinés dans une seule activité et un même fragment peut être réutilisé dans plusieurs activités

Fragments (3/3)

Pour créer un fragment, il faut créer une classe qui hérite de **`androidx.fragment.app.Fragment`** ou une de **ses sous-classes**

Sous-classes utiles héritant de Fragment :

- **`DialogFragment`** (Affiche une boîte de dialogue flottante)
- **`ListFragment`** (Affiche une liste d'éléments gérés par un adaptateur)
- **`PreferenceFragment`** (Affiche une hiérarchie de Préférences pour créer une activité Settings)

2 façons pour ajouter un fragment à l'activité :

- Référencé de façon statique **dans le layout de l'activité**
- Chargé dynamiquement (par programmation) **dans l'activité**

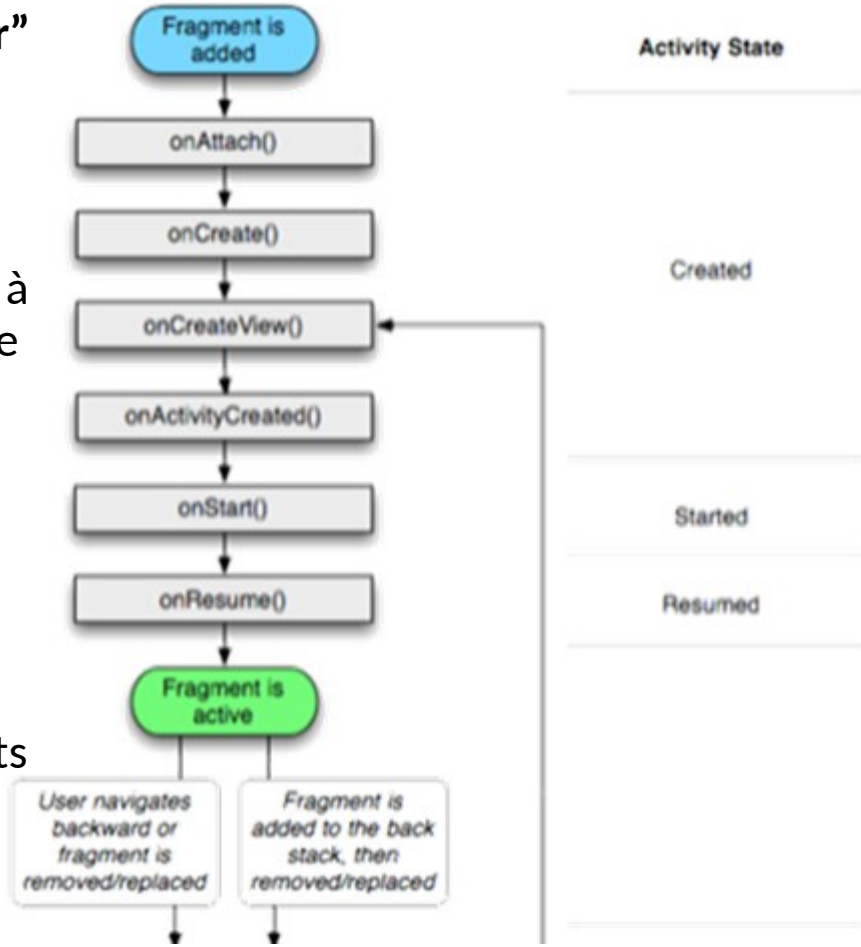
Cycle de vie (1/2)

Un fragment possède **son propre cycle de vie**

- Isoler son comportement et **rendre plus "léger"** le code de l'activité

Méthodes callback :

- **`onAttach()`** : (quand le fragment est associé à une activity) pour récupérer l'activité contenante
- **`onCreate()`** : pour instancier les objets non graphiques
- **`onCreateView()`** : pour instancier les objets graphiques
- **`onStart()`** : pour lancer les traitements
- **`onResume()`** : pour s'abonner aux évènements et récupérer le contexte
- **...**

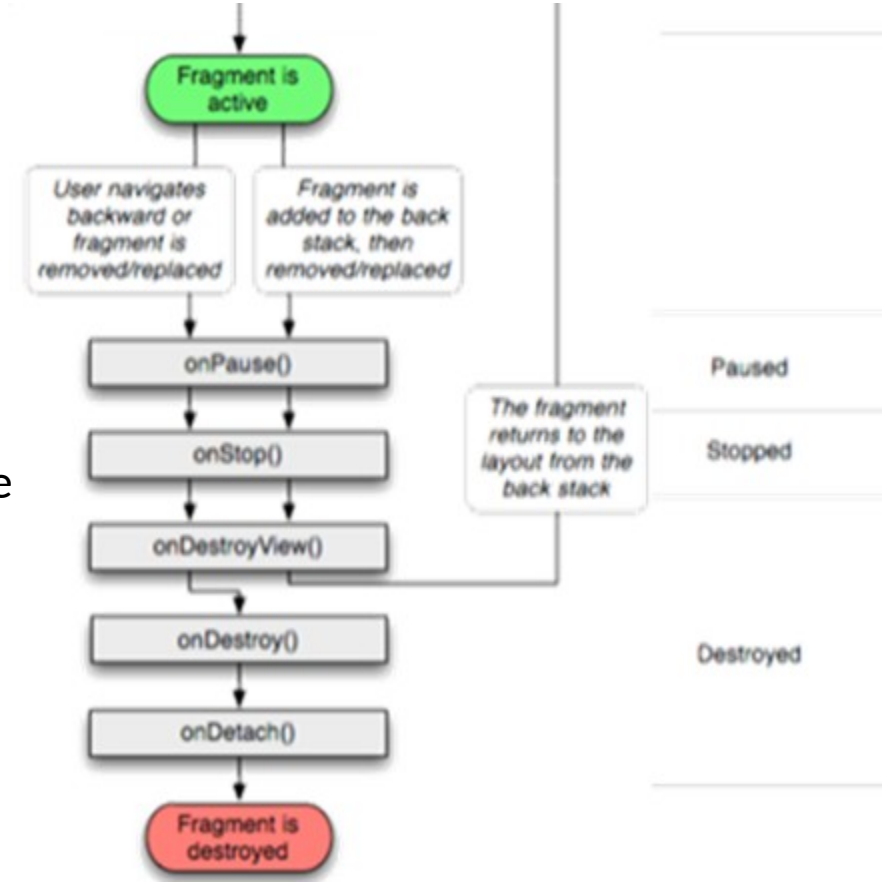


Fragments

Cycle de vie (2/2)

Méthodes callback :

- **…**
- **`onPause()` :** pour se désabonner aux évènements et sauvegarder le contexte
- **`onStop()` :** pour arrêter les traitements (threads, …) et désallouer les ressources
- **`onDestroyView()` :** pour libérer la mémoire des objets graphiques
- **`onDestroy()` :** (les ressources restants sont libérées automatiquement)
- **`onDetach()` :** (quand la vue associée au fragment est détachée de l'activité)



Fragments

Création d'un fragment

/res/layout/fragment_my.xml

```
<LinearLayout xmlns:android="..." ...>
    <!-- content -->
</LinearLayout>
```

/java/MyFragment.java

```
public class MyFragment extends Fragment{
  public MyFragment(...){ ... }
  @Override
  public View onCreateView(LayoutInflater inflater,
                           ViewGroup container,
                           Bundle savedInstanceState) {
    View layout = inflater.inflate(R.layout.fragment_my, container, false);
    ...
    return layout;
  }
}
```

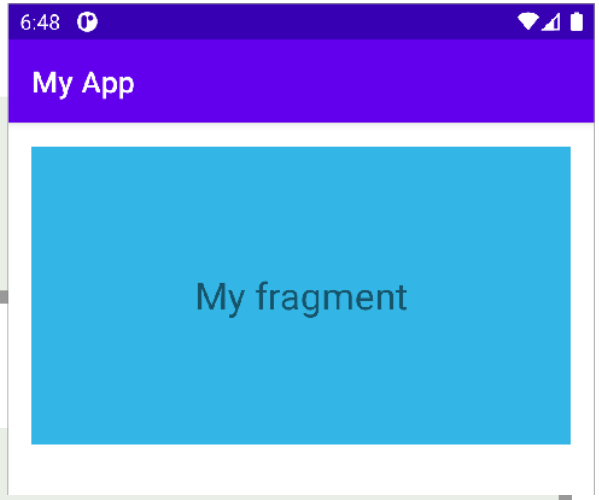
Fragments

Ajout à l'activité (1/2)

Méthode 1 : Chargé dynamiquement (par programmation)

/res/layout/activity_main.xml

```
...
<FrameLayout
    android:id="@+id/fl_container" ...>
```



/java/MainActivity.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    getSupportFragmentManager().beginTransaction()
            .replace(R.id.fl_container, new MyFragment())
            .commit();
}
```

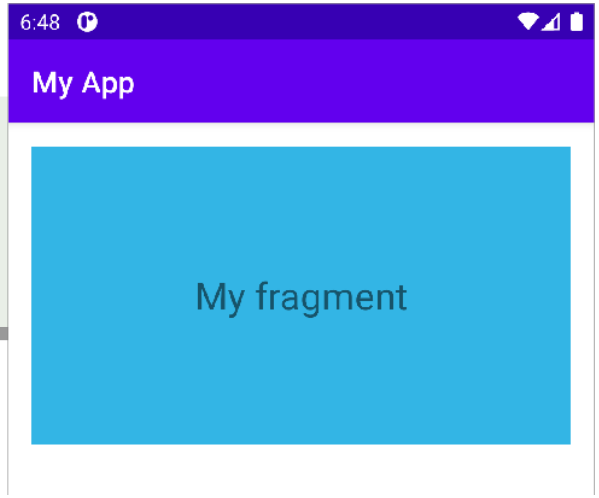
Fragments

Ajout à l'activité (2/2)

Méthode 2 : Référencé de façon statique dans le layout

/res/layout/activity_main.xml

```
...
<androidx.fragment.app.FragmentContainerView
    android:name=".MyFragment" ...>
```



`FragmentManager` et gestion des transactions

Le **`FragmentManager`** permet de :

- Manipuler des fragments existants en utilisant **`findFragmentById(…)`** ou **`findFragmentByTag(…)`**
- Gérer la pile accessible via le bouton **"Back"** :
  - Dépiler un fragment de la pile de fragments, avec la méthode **`popBackStack()`**
  - Associer un Listener aux changement dans la pile avec la méthode **`addOnBackStackChangedListener(…)`**
- Un **`FragmentTransaction`** permet d'ajouter, de supprimer, ou de remplacer un fragment dans une activité,
- Il est possible de sauvegarder une transaction dans la pile de fragments grâce à la métode **`addToBackStack(…)`**

Menu déroulant principal

Création d'un `DrawerLayout` avec un menu :

- **Habitats (`HabitatsFragment`)**
- **Mon habitat,**
- **Mes requêtes,**
- **Paramètres,**
- **Se déconnecter, …**