# Développement mobile

## Android - Activity

# Activity

- Les Activités permettent à l'utilisateur d'interagir avec l'application
- Elles créent une fenêtre dans laquelle se place l'interface décrite par l'Activité

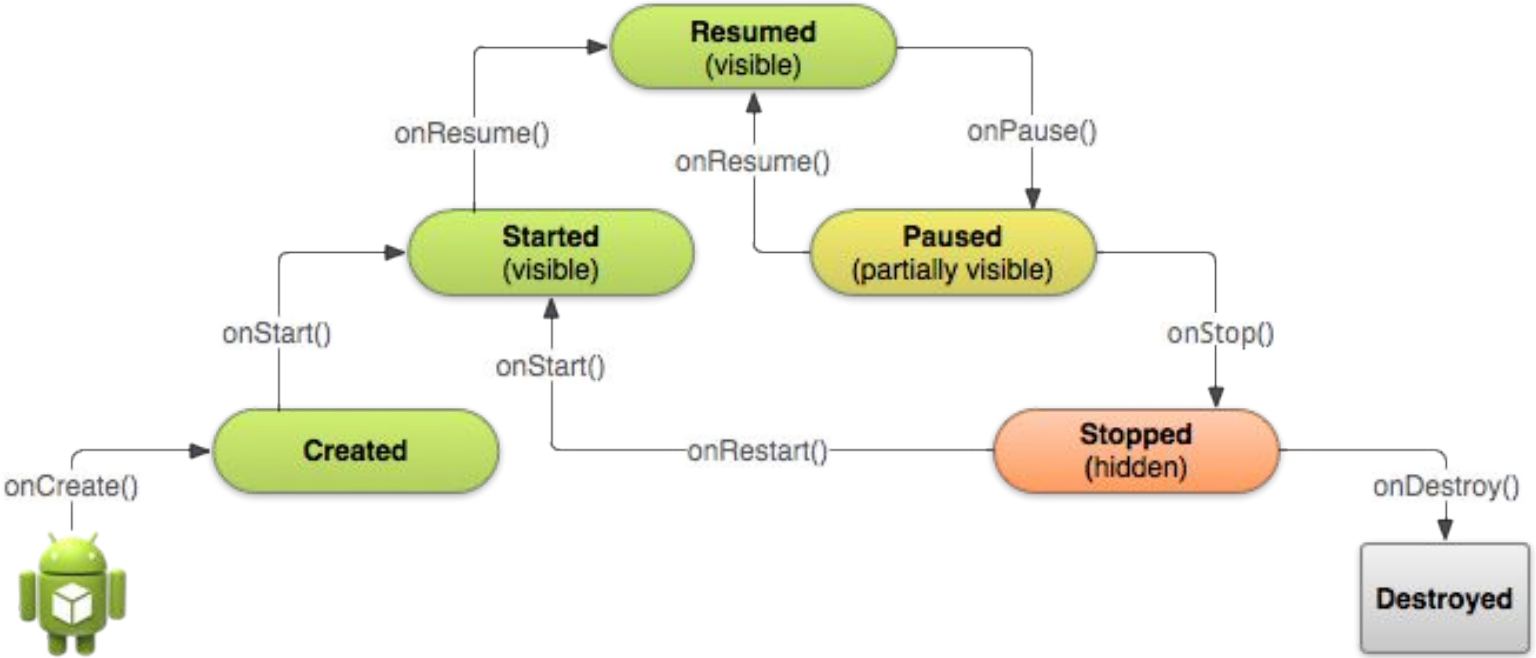
```
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
  }
}
```

# Activity

- Les Activités permettent à l'utilisateur d'interagir avec l'application
- Elles créent une fenêtre dans laquelle se place l'interface décrite par l'Activité
- Elles sont déclarées dans l'AndroidManifest.xml
  - nom de l'Activité
  - un label
  - un thème
  - une orientation
  - etc.

```
<activity
  android:name=".MainActivity"
  android:label="@string/app_name"
  android:theme="@style/AppTheme.NoActionBar">
        <intent-filter>
              <action
        android:name="android.intent.action.MAIN" />
              <category
        android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
</activity>
```

# Activity : cycle de vie



# Activity : cycle de vie

- seule onCreate() est obligatoire, elle instancie l'interface graphique
  - conteneurs,
  - vues,
  - actions
- toutes ces méthodes doivent appeler la méthode parente
- onBackPressed()
  - gère le bouton retour "physique"
  - par défaut, appelle la méthode finish()
- finish() permet de terminer / détruire une Activity

# Sauvegarder l’état d’une Activity

- Pourquoi ?
    - Lors de la rotation d’un écran, l’Activity est détruite et relancée
    - Lorsque l’Activity est “endormie”
- Comment ?
    - En sauvegardant l’état des éléments importants

# Sauvegarder l’état d’une Activity

- Implémenter onSaveInstanceState()
  - Bundle fonctionne comme un HashMap
  - Sauver tous les éléments qui sont modifiés

```
@Override
protected void onSaveInstanceState(Bundle outState)
{
  outState.putString("MA_CLE", String.valueOf(text.getText()));

  super.onSaveInstanceState(outState);
}
```


# Restaurer l’état d’une Activity

- Implémenter onRestoreInstanceState()
  - Récupérer les éléments et les affecter au bon élément

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState)
{
   super.onRestoreInstanceState(savedInstanceState);
   String s = savedInstanceState.getString("MA_CLE");
   text.setText(s);
}
```

# Restaurer l’état d’une Activity

- Modifier onCreate()
  - Vérifier si on crée une nouvelle instance ou si on restaure une instance
  - Récupérer et affecter les données

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

  if(savedInstanceState != null) {
     String s = savedInstanceState.getString("MA_CLE");
     text.setText(s);

  }

}
```

# Mettre en pause et reprendre une Activity

- Implémenter onPause()
  - Stopper les animations
  - Sauvegarder ce que l'utilisateur peut s'attendre à être sauvegardé
  - Libérer les ressources systèmes pour préserver la batterie

```
@Override
protected void onPause() {
  super.onPause();
  Log.i(TAG, "onPause");
}
```

- Implémenter onResume()
  - Redémarrer les animations
  - Remettre en état l'interface (boutons, etc)

```
@Override
protected void onResume() {
  super.onResume();
  Log.i(TAG, "onResume");
}
```

# Création des éléments d’une Activity

Plusieurs méthodes :

- fichier XML (layout)
- code
- mix entre les 2

# Création des éléments d'une Activity

- fichier XML (layout)
  - la racine est forcément un conteneur
  - elle doit définir le namespace (xmlns) d'Android

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

  <TextView
     android:id="@+id/text"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Hello World!"/>

  <EditText
     android:id="@+id/edit_text"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
 />

  <Button
     android:id="@+id/button"
     android:text="Submit"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     />
</LinearLayout>
```

# Création des éléments d'une Activity

- fichier XML (layout)
  - la racine est forcément un conteneur
  - elle doit définir le namespace (xmlns) d'Android
  - associer le fichier XML à l'Activity

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

# Création des éléments d'une Activity

- à partir du code :
  - recréer un à un les éléments et les associer
  - définir les paramètres de chaque élément (taille, texte, alignement, etc)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  LinearLayout layout = new LinearLayout(this);

  text = new TextView(this);
  text.setText("Hello World!");
  layout.addView(text);

  editText = new EditText(this);
  ll.addView(editText);

  Button button = new Button(this);
  button.setText("Submit");
  layout.addView(button);

  setContentView(layout);
}
```

# Création des éléments d'une Activity

- mix des 2 : LayoutInflater
  - permet d'instancier le contenu d'un fichier XML dans un objet View

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        LayoutInflater inflater = LayoutInfalter.from(this);

        View view = inflater.inflate(R.layout.mon_contenu, null);

        text = (TextView) view.findViewById(R.id.text);

        editText = (EditText) view.findViewById(R.id.edit_text);

        setContentView(view);
}
```

# Activities : lancer une nouvelle Activity

- Pourquoi ?
  - permet diviser l'application en plusieurs parties, plusieurs chemins
  - évite de surcharger le layout associé et le code

- Comment ?
  - créer une nouvelle Activity (layout, code)
  - en utilisant un Intent

# Activities : lancer une nouvelle Activity

Qu'est ce qu'un Intent ?

- un Objet faisant le lien entre 2 composants séparés
- On peut associer ça à une "Intention de faire …"
- La plupart du temps utilisé pour lancer une nouvelle Activity
- Permet  aussi de faire le lien avec une autre Application !!

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
```

new Intent :

- Context courant, généralement celui de l'Activity courante
- classe de l'Activity à créer

# Activities : lancer une nouvelle Activity

On veut parfois transmettre des informations à l'Activity créée

- système proche de celui de la sauvegarde d'une Activity :

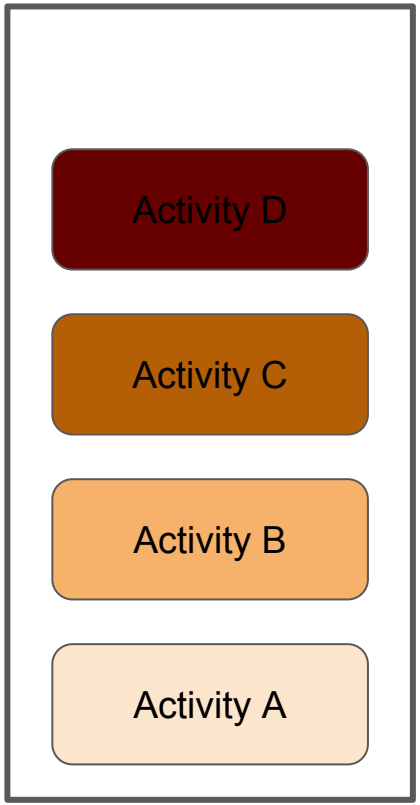
```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
String message = editText.getText().toString();
intent.putExtra("UNE_CLE", message);

startActivity(intent);
```

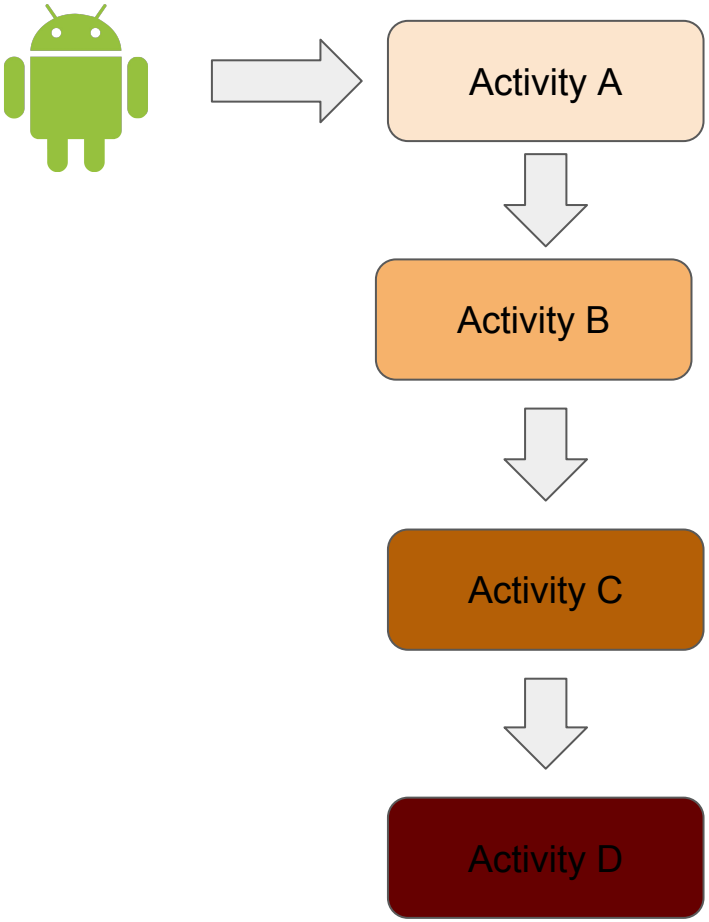
# Activities : lancer une nouvelle Activity

Ok .. mais comment récupérer les données dans la nouvelle Activity ??

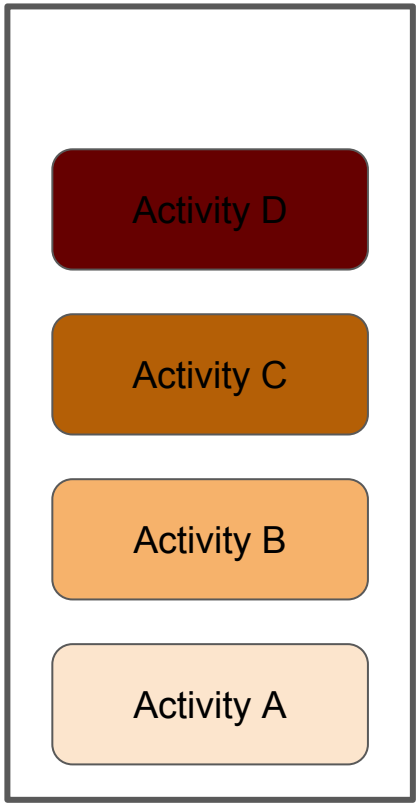
```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_second);

  Intent intent = getIntent();
  String message = intent.getStringExtra("UNE_CLE");
}
```

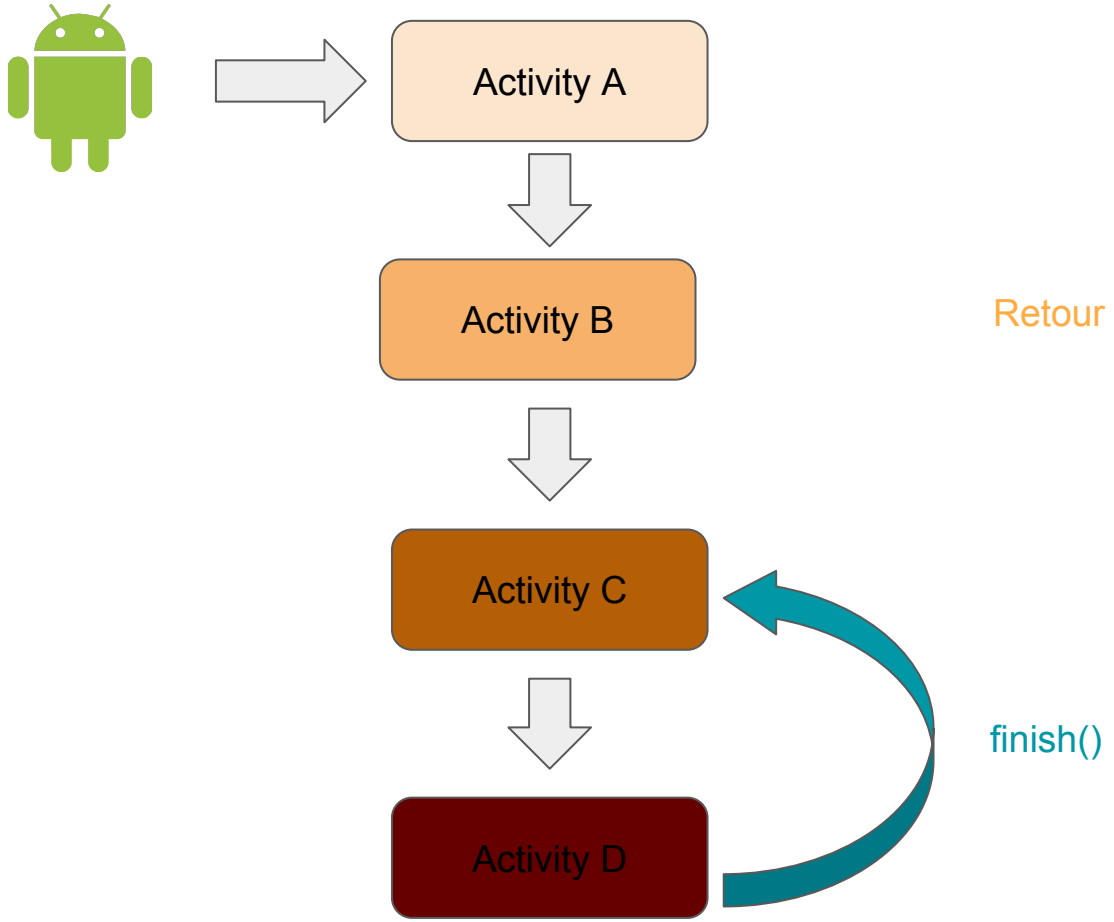
# Pile d’Activity



# Pile d'Activity



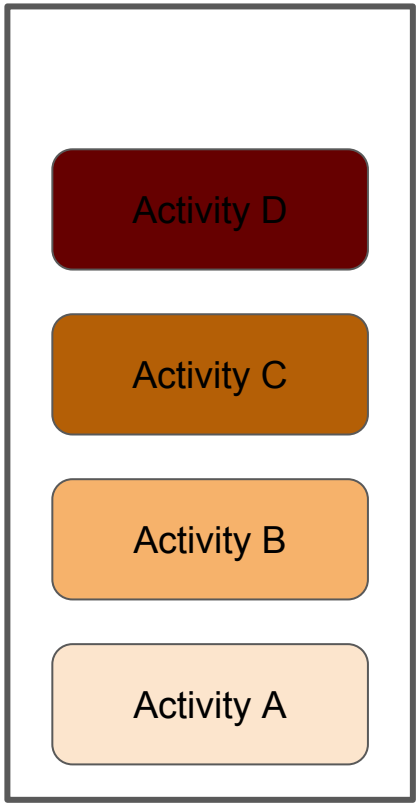



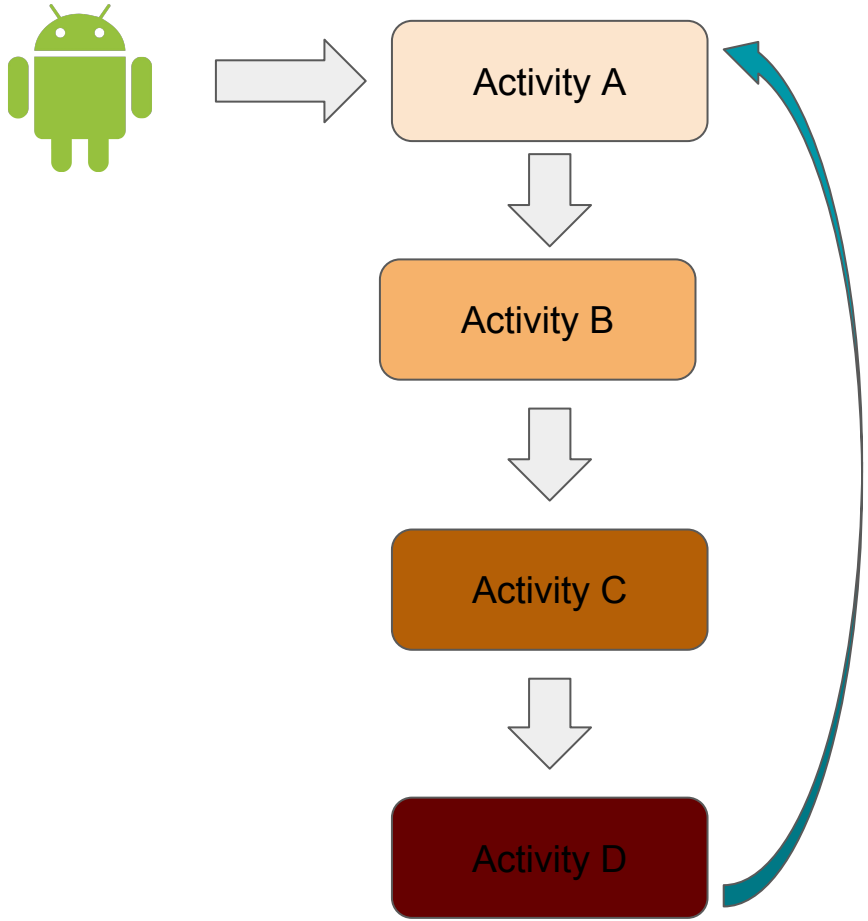
# Pile d’Activity

Jouer avec la pile en passant des FLAG à l’Intent

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
String message = editText.getText().toString();
intent.putExtra("UNE_CLE", message);

intent.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);

startActivity(intent);
```

# Pile d'Activity

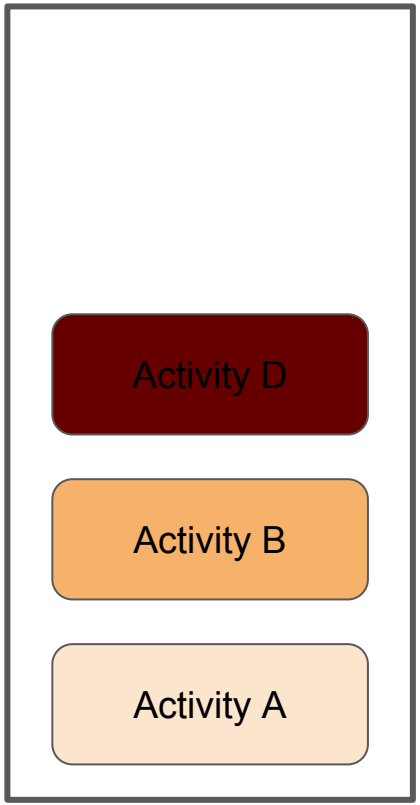


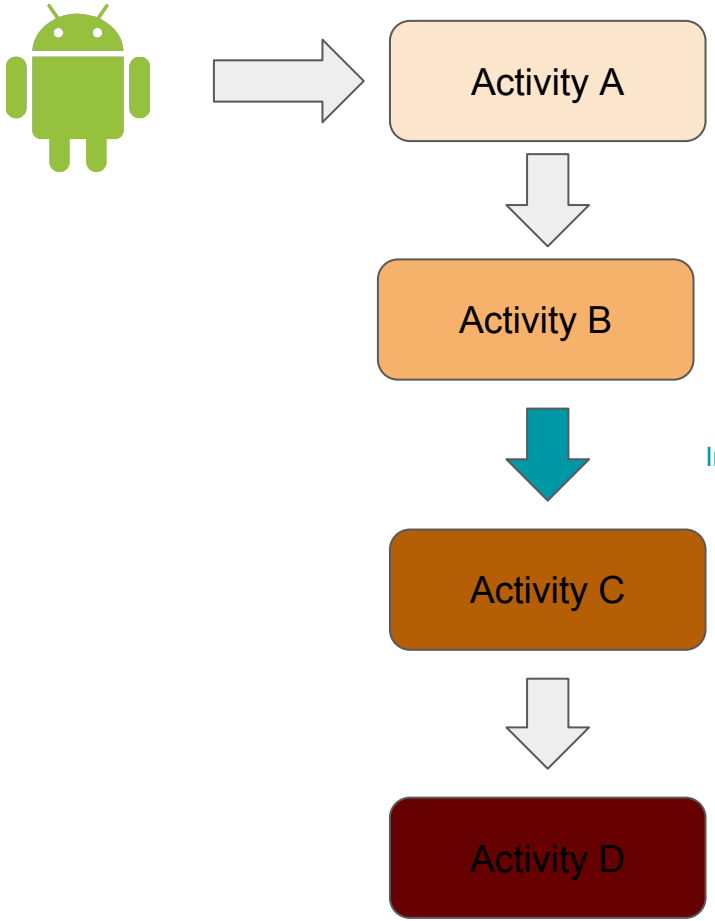




Retour HOME

```
Intent.FLAG_ACTIVITY_C
LEAR_TASK|Intent.FLAG_
ACTIVITY_NEW_TASK
```

# Pile d'Activity




Ajout sans Historique

Intent.FLAG_ACTIVITY_NO_HISTORY

# Pile d’Activity

Plein d’autres FLAGS disponibles :
http://developer.android.com/reference/android/content/Intent.html