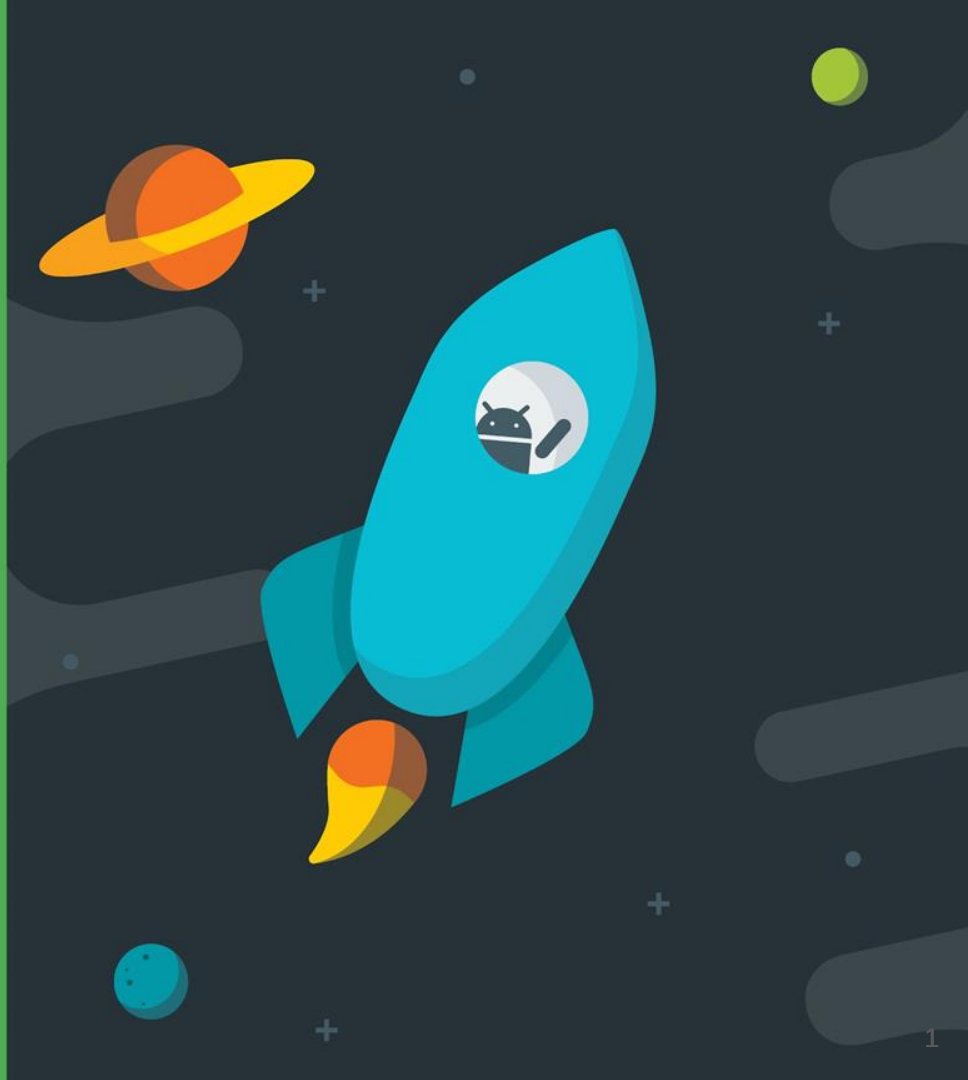
UNITÉ D ’ENSEIGNEMENT (UE) :

DÉVELOPPEMENT MOBILE

CHAPITRE III :

Activité et Intent



Plan

- Activités
- Définir une activité
- Démarrer une nouvelle activité avec une intention
- Passer des données entre les activités avec des extras
- Naviguer entre les activités

Activités

Qu'est-ce qu'une activité ?

- Une activité (`Activity`) est un composant d'application
- Représente une fenêtre, une hiérarchie de vues
- Remplit généralement l'écran, mais peut être intégré à une autre activité ou apparaître comme une fenêtre flottante
- Généralement une activité par fichier Java,

A quoi sert une activité ?

- Représente une activité, telle que la commande, l'envoi d'e-mails ou l'obtention d'un itinéraire
- Gère les interactions des utilisateurs, telles que les clics sur les boutons, la saisie de texte ou la vérification de connexion
- Peut démarrer d'autres activités dans la même ou d'autres applications
- A un cycle de vie : est créé, démarré, exécuté, mis en pause, repris, arrêté et détruit

Applications et activités

- Les activités sont vaguement liées entre elles pour constituer une application
- La première activité que l'utilisateur voit est généralement appelée « activité principale » `MainActivity`
- Les activités peuvent être organisées en relations parent-enfant dans le manifeste pour faciliter la navigation

Layouts et activités

- Une activité a généralement une interface utilisateur (layout)
- Le layout est généralement définie dans un ou plusieurs fichiers XML
- L'activité charge (inflates) le layout dans le cadre de la création

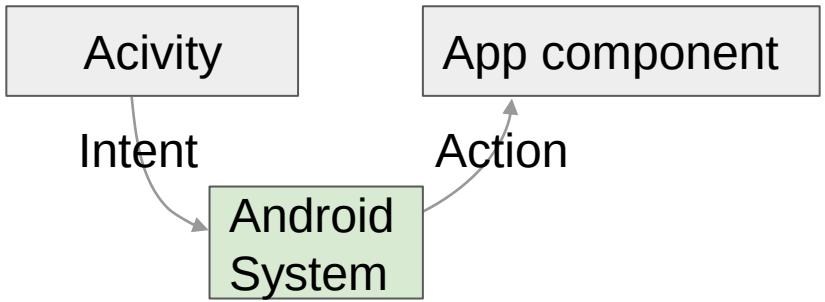
Implémenter une nouvelle activité

1. Définir le layout en XML
2. Définir la classe Java `Activity`
   - **`extends AppCompatActivity`**
3. Connecter l'activité au layout
   - Définir la vue du contenu dans `onCreate()`
     `setContentView(R.layout.activity_main);`
4. Déclarer l'activité dans le `manifest.xml`
   - **`<activity android:name=".MainActivity">`**

Intents

Qu'est-ce qu'une intention ?

Une intention est une description d'une opération à effectuer.
Une intention est un objet utilisé pour demander une action à un autre composant d'application via le système Android.



Que peut faire un intent?

- Démarrer une activité
  - Un clic sur un bouton démarre une nouvelle activité
  - Cliquer sur Partager ouvre une activité ou une application qui vous permet de publier une photo
- Démarrer un service
  - Lancer le téléchargement d'un fichier en arrière-plan
- Diffuser un `Broadcast`
  - Le système informe tout le monde que le téléphone est maintenant en charge

Intentions explicites et implicites

Intention explicite

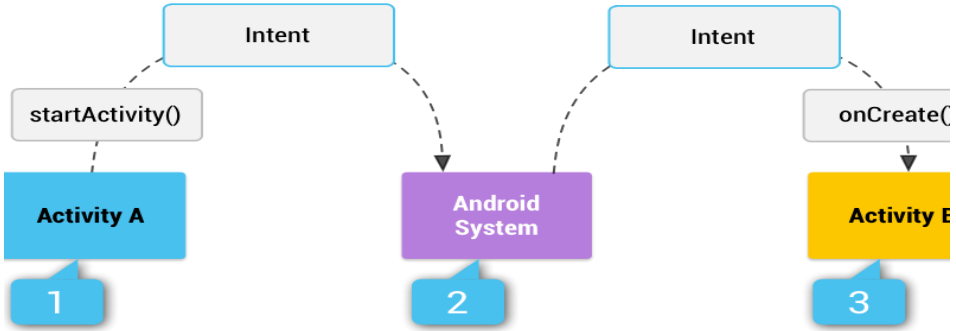
- Commence une activité spécifique
  - L'activité principale démarre l'activité ViewShoppingCart

Intention implicite

- Demande au système de trouver une activité qui peut gérer cette demande
  - Cliquer sur Partager ouvre un sélecteur avec une liste d'applications

Comportement d’un Intent Implicite

1. Activité A crée un Intent avec une action et le passe en paramètre à startActivity

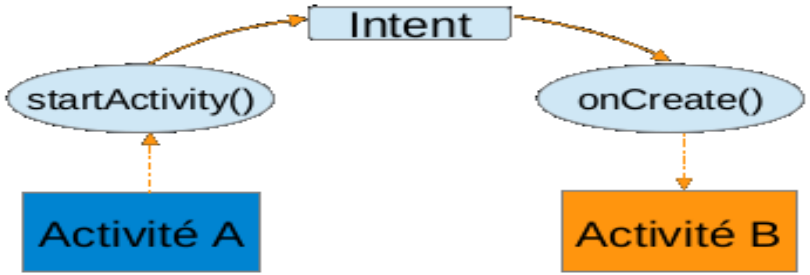


2. Le système Android cherche toutes les applications pour trouver un Intent Filter qui correspond à cet Intent

3. Quand une correspondance est trouvée, le système démarre l’activité (Activity B) en invoquant sa méthode onCreate et en lui passant l’intent

Les Intentions explicites

1. Message adressé à un composant connu
    - On donne le nom de la classe correspondante
    - Généralement réservé aux composants appartenant à la même application ...
2. Le composant est démarré immédiatement



Démarrer une activité

Démarrer une activité avec une intention explicite

Pour démarrer une activité spécifique, utilisez une intention explicite

1. Créer une intention
   - `Intent intent = new Intent(this, ActivityName.class);`
2. Utiliser l'intention pour démarrer l'activité
   - `startActivity(intention);`

Démarrer une activité avec une intention implicite

Pour demander à Android de trouver une activité pour traiter votre demande, utilisez une intention implicite

1. Créer une intention
   - `Intent intention = new Intent(action, uri);`
2. Utiliser l'intention pour démarrer l'activité
   - `startActivity(intention);`

Intentions implicites - Exemples

Afficher une page Web

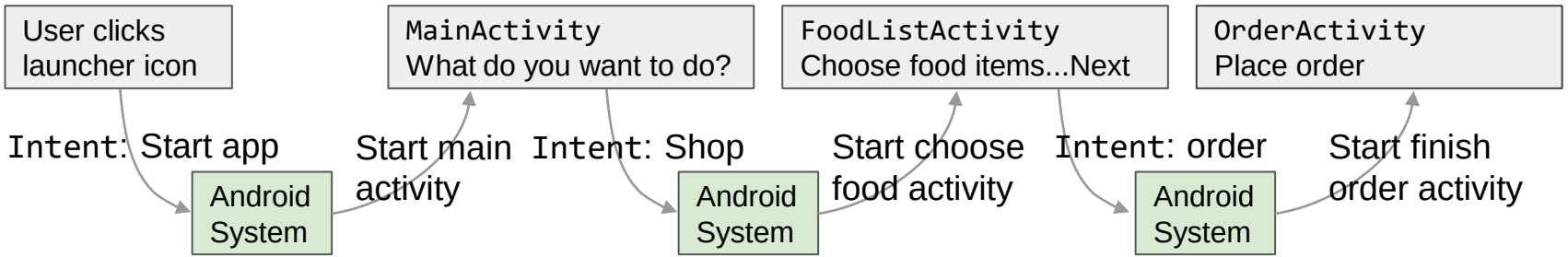
```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

Composez un numéro de téléphone

```
Uri uri = Uri.parse("tel:98989898");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

Comment se déroulent les activités

- Toutes les instances d'activité sont gérées par le runtime Android
- Démarré par un "Intent", un message à l'environnement d'exécution Android pour exécuter une activité



Envoi et réception de données

Types d'envoi de données

- Données : une information dont l'emplacement des données peut être représenté par un URI

- Extras : un ou plusieurs éléments d'information en tant que collection de paires clé-valeur dans un `Bundle`

Envoi et récupération de données

Dans la première activité (d'envoi) :

1. Créer l'objet Intent
2. Mettre des données ou des extras dans cette intention
3. Démarrer la nouvelle activité avec `startActivity()`

Dans la deuxième activité (réception) :

1. Obtenez l'objet Intent, l'activité a été démarrée avec
2. Récupérer les données ou extras de l'objet Intent

Mettre un URI en tant que données d'intention

```
// URL page web
intent.setData(
    Uri.parse("http://www.google.com"));

// URI fichier
intent.setData(
     Uri.fromFile(new
File("/sdcard/exemple.jpg")));
```

Mettre des informations dans les extras d'intention

- `putExtra(String nom, int valeur)`
  ⇒ `intent.putExtra(“niveau", 406);`
- `putExtra(String nom, String[] value)`
  ⇒
```
String[] foodList = {“Riz", “Pain", “Fruit"};
intent.putExtra("food", foodList);
```
- `putExtras(bundle);`
  ⇒ s'il y a beaucoup de données, créez d'abord un bundle et transmettez le bundle.
- Voir la documentation

Envoi de données à une activité avec des extras

```
public static final String EXTRA_MESSAGE_KEY =
    "com.example.android.twoactivities.extra.MESSAGE";

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

String message = "Hello Activity!";

intent.putExtra(EXTRA_MESSAGE_KEY, message);

startActivity(intent);
```

Obtenir des données à partir des intentions

- `getData();`
  `⇒ Uri locationUri = intent.getData();`
- `int getIntExtra (String name, int defaultValue)`
  `⇒ int level = intent.getIntExtra(“niveau", 0);`
- `Bundle bundle = intent.getExtras();`
  ⇒ Obtenez toutes les données à la fois sous forme de bundle.
- Voir la documentation

Retour des données à l'activité de départ

1. Utilisez `startActivityForResult()` pour démarrer la deuxième activité
2. Pour renvoyer les données de la deuxième activité :
    - Créer une (**new**) intention
    - Mettez les données de réponse dans l'intention à l'aide de `putExtra()`
    - Définissez le résultat sur `Activity.RESULT_OK` ou `RESULT_CANCELED`, si l'utilisateur a annulé
    - appelez `finish()` pour fermer l'activité
3. Implémenter `onActivityResult()` dans la première activité

startActivityForResult()

`startActivityForResult(intent, requestCode);`

- Démarre `Activity` (`intent`), lui attribue un identifiant (`requestCode`)
- Renvoie des données via `Intent` extras
- Une fois terminé, pop stack, revenir à l'activité précédente et exécuter le rappel `onActivityResult()` pour traiter les données renvoyées
- Utilisez `requestCode` pour identifier quelle activité est « retournée »

Exemple

```
public static final int CHOOSE_FOOD_REQUEST = 1;

Intent intent = new Intent(this, ChooseFoodItemsActivity.class);
startActivityForResult(intent, CHOOSE_FOOD_REQUEST);
```

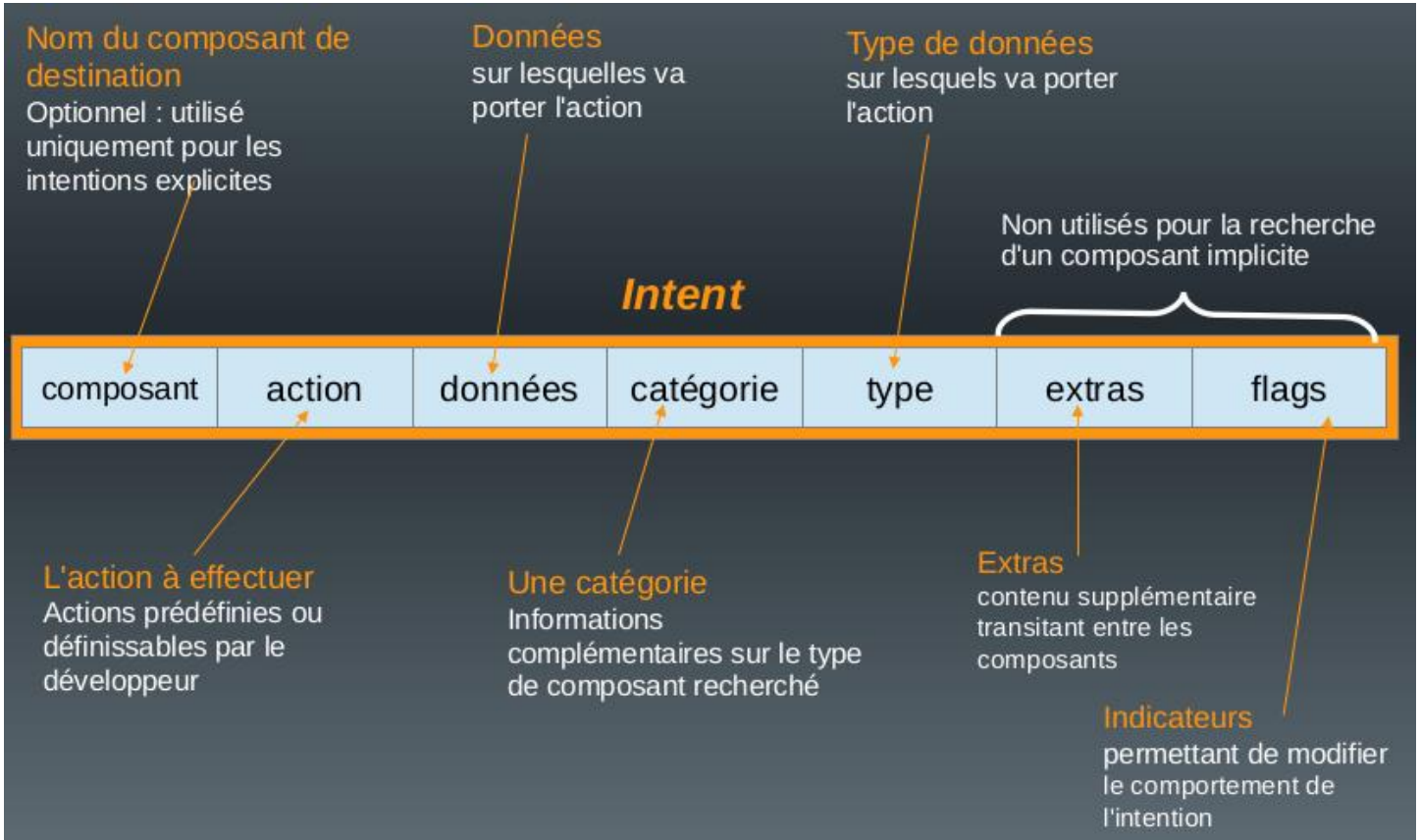
Renvoyer les données et terminer la deuxième activité

```
 // Créer une intention
Intent replyIntent = new Intent();

// Mettez les données à retourner dans l'extra
replyIntent.putExtra(EXTRA_REPLY, reply);

// Définissez le résultat de l'activité sur RESULT_OK
setResult(RESULT_OK, replyIntent);

// Terminer l'activité en cours
finish();
```

Implémenter onActivityResult()

```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  if (requestCode == TEXT_REQUEST) { // Identifier l'activité
    if (resultCode == RESULT_OK) { // Activité réussie
      String reply =
data.getStringExtra(SecondActivity.EXTRA_REPLY);
      // … faire quelque chose avec les données
 }}}
```

Structure générale des Intentions



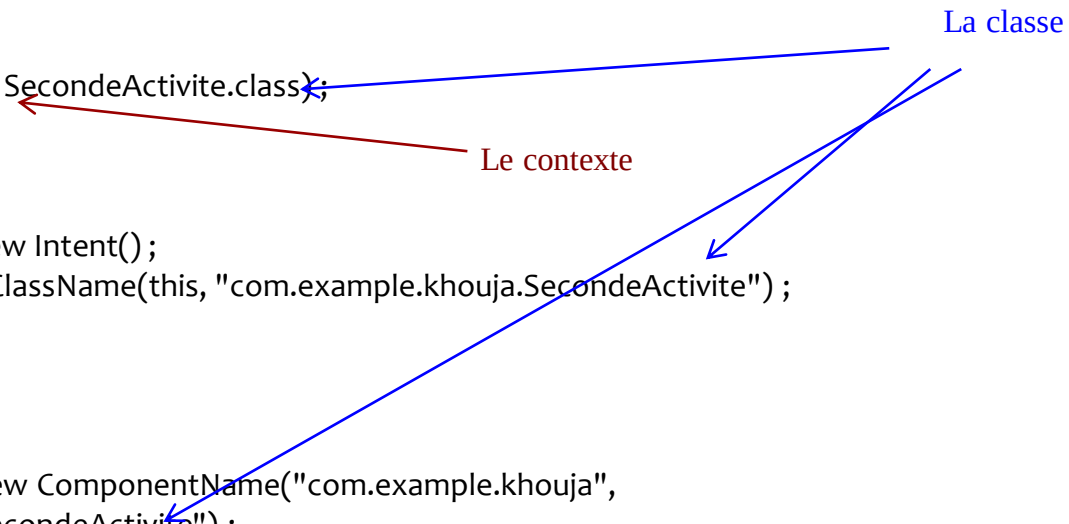
Nom du composant

- Différentes méthodes utilisables :

–Intent(...) : le constructeur

–SetComponent(...), setClassName(...), setClass(...), etc.

```
...
Intent = new Intent(this, SecondeActivite.class);
...
startActivity(intent) ;
...
                Intent = new Intent() ;
                Intent.setClassName(this, "com.example.khouja.SecondeActivite") ;
...
startActivity(intent) ;
...
Intent = new Intent() ;
Intent.setComponent(new ComponentName("com.example.khouja",
"com.example.khouja.SecondeActivite") ;
```



Action à effectuer

- Nombreuses constantes prédéfinies dans la classe Intent :
  - ACTION_MAIN : démarre l'activité comme point d'entrée principal
  - ACTION_VIEW : affiche les données fournies à l'utilisateur
  - ACTION_EDIT : permet l'édition des données fournies par l'utilisateur
  - ACTION_SEND : permet l'envoi des données (mail, réseau social, …)
  - …
- Possibilité de définir ses propres actions comme constante de classe
  - static final String ACTION_MON = "com.example.khouja.Mon";

Données

- Formatées à l'aide des URI (Uniform Ressource Identifier)
  - Chaîne de caractères représentant un endroit ou une ressource
  - Syntaxe normalisée :

<schéma> : <information> { ? <requêtes> } { # <fragment> }

  - Nature de l'information : http, https, content, geo, file, tel, voicemail, sms, mms, …
  - L'information dépend du schéma : tel:98000001, Geo:123.456,12.3456, http://www.google.fr

Le type

- Normalement implicite dans les données
- Absence de données/recherche plus précise :

–peut être précisé en utilisant les codes MIME : text, audio, video

–Et des sous-types : text/plain, text/xml, text/html, audio/mp3, audio/ogg, audio/wav, video/mpeg, video/mp4, ...

```
Intent intention = newIntent() ;
...
Intention = intention.setType("audio/mp3") ;
 ...
String mime = intention.getType() ;
```

Catégorie

- Précise le type de composant qui doit prendre en charge l'intention
- Exemples :

  – CATEGORY_LAUNCHER : le composant doit être visible dans le lanceur d'applications

  – CATEGORY_CAR_MODE : le composant doit être utilisable en mode conduite de véhicule

- Remarques

  –Nombre quelconque de catégories dans une intention

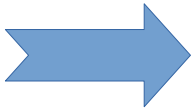
  –La plupart des intentions ne nécessitent pas de catégories

Les extras

- Permet d'insérer des données dans l'intention
  - Communications inter-composants
  - Structure : (clé, valeur)
    - Intent putExtra(String cle, type valeur) ; //Clé définie par l'utilisateur
    - type get{type}Extra(String cle, type vdefaut) ; //vdefaut : valeur par défaut renvoyée si la clé n'est pas trouvée dans l'intention

Recherche d'un composant

- Utilisation d'intents implicites :
  - Le système recherche le(s) composant(s) possible(s)
  - Le système doit connaître les types d'intents que peut recevoir chaque composant
- Nécessité de préciser pour chaque composant les intents réceptionnables



Intent filters

IntentFilters

- Un Intent Filter est une expression dans le fichier Manifest d'une application qui spécifie le type d'intents que le composant veut recevoir
- Permet aux autres activités de lancer directement votre activité en utilisant un certain Intent
- Si vous ne déclarez pas d'Intent Filters à votre activité, elle ne pourra être déclenchée que par un Intent Explicite
- Il est recommandé de ne pas déclarer d'Intent Filters pour les services, car cela peut causer des problèmes de sécurité

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".main"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name="ReponseIntent"></activity>
<activity android:name="Test"></activity>
</application>
```

Navigation

Pile d'activités

- Lorsqu'une nouvelle `Activity` est démarrée, l'activité précédente est arrêtée et poussée sur la pile arrière d'activité
- Pile dernier entré, premier sorti : lorsque l'activité en cours se termine ou que l'utilisateur appuie sur le bouton Précédent, elle est extraite de la pile et l'activité précédente reprend

Deux formes de navigation

 Navigation temporelle ou arrière

- fourni par le bouton Retour de l'appareil
- contrôlé par la pile arrière du système Android

 Navigation ancestrale ou ascendante

- fourni par le bouton haut dans la barre d'action de l'application
- contrôlé en définissant des relations parent-enfant entre les activités dans le manifeste Android

Navigation en arrière

- La pile arrière préserve l'historique des écrans récemment visionnés
- La pile arrière contient toutes les instances d'activité qui ont été lancées par l'utilisateur dans l'ordre inverse pour la tâche en cours
- Chaque tâche a sa propre pile arrière
- Commutation entre les tâches active la pile arrière de cette tâche

Up navigation

- Va au parent de l'activité actuelle
- Définir un parent d'activité dans le manifeste Android
- Définir `parentActivityName`

```
<activity
    android:name=".ShowDinnerActivity"
    android:parentActivityName=".MainActivity" >
</activity>
```