Cours : Activity et intents

Présenté par :
Dr ZEMALI Elamine

Introduction aux activités

- La classe Activity est un composant crucial d'une application Android.

- Contrairement aux paradigmes de programmation dans lesquels les applications sont lancées avec une méthode main (), le système Android initie le code dans une instance d'activité en invoquant des méthodes spécifiques qui correspondent à des étapes spécifiques de son cycle de vie.

Le concept

- Dans une app mobile, l'utilisateur ne commence pas forcement l'interaction du meme endroit.
- Par exemple,
  - app de messagerie :
    - Accès direct -> boite de reception
    - Acces via une autre app -> envoyer un email
- La classe Activity est concue pour faciliter ce paradigme:
  - Lorsqu'une application en appelle une autre, l'application appelante appelle une activité dans l'autre application, plutôt que l'application dans son ensemble atomique.

Concept

- Une activité fournit la fenêtre dans laquelle l'application trace son interface utilisateur.

- L'activity remplit généralement l'écran, mais peut être plus petite que l'écran et flotter au-dessus d'autres fenêtres.

- La plupart des applications contiennent plusieurs écrans, ce qui signifie qu'elles comprennent plusieurs activités.

- Chaque activité peut ensuite démarrer une autre activité afin d'effectuer différentes actions.

Concept

- Chaque activité n'est liée que de manière faible aux autres activités; il existe généralement des dépendances minimales entre les activités d'une application.
- les activités peuvent démarrer des activités appartenant à d'autres applications.
- Pour utiliser des activités :
  - Enregistrer des informations sur les activity dans le manifeste de l'application.
  - Gérer les cycles de vie des activités de manière appropriée

Configuration du manifeste

- Pour que votre application puisse utiliser des activités, vous devez déclarer les activités et certains de leurs attributs dans le manifeste.


```
<manifest … >
 <application … >
    <activity android:name=".ExampleActivity" />
    …
 </application … >
 …
</manifest >
```

- Le seul attribut requis pour cet élément est android: name, qui spécifie le nom de classe de l'activité. Vous pouvez également ajouter des attributs qui définissent des caractéristiques d'activité telles que:
- android:label
- Android:icon
- Android :theme : le thème de l'interface utilisateur.

Les filtres d'intention(intent-filter)

- Ils offrent la possibilité de lancer une activité basée non seulement sur une demande explicite, mais aussi implicite. (plus de détails dans la partie intent implicite).

- Vous pouvez tirer parti de cette fonctionnalité en déclarant un attribut <intent-filter> dans l'élément <activity> :

```
<activity android:name=".ExampleActivity" android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

Gérer le cycle de vie de l'activité

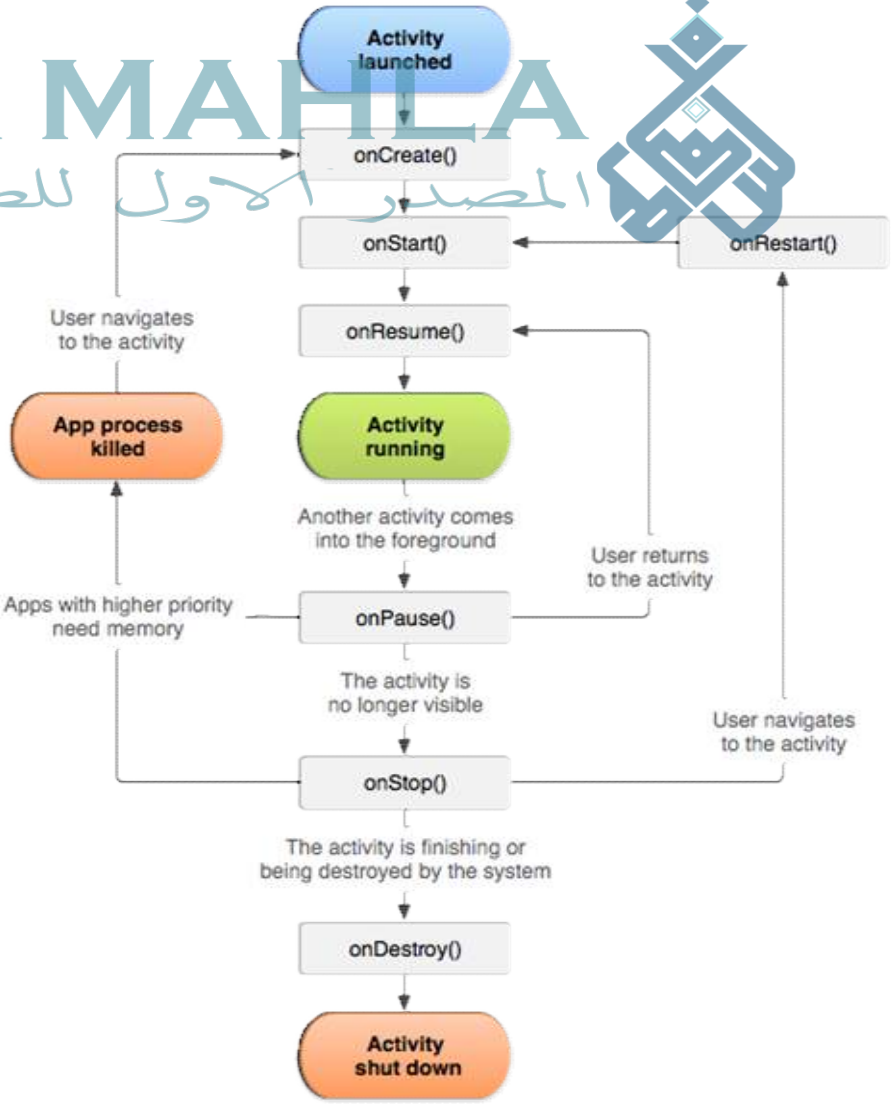
- Au cours de sa vie, une activité passe par plusieurs états. Vous utilisez une série d'appels pour gérer les transitions entre les états.

- La classe Activity fournit un certain nombre d'Appels qui permettent à l'activité de savoir qu'un état a changé:
  - que le système crée, arrête ou reprend une activité, ou détruit le processus dans lequel l'activité réside.
  - Dans les méthodes de Appel du cycle de vie, vous pouvez déclarer le comportement de votre activité lorsque l'utilisateur quitte et revient l'activité., ou lors de démarrage.

Gérer le cycle de vie de l'activité

- Par exemple: Un lecteur vidéo en streaming.
- une bonne implémentation des appels du cycle de vie peut aider à garantir que votre application évite:
  - Plantage si l'utilisateur reçoit un appel téléphonique ou passe à une autre application lors de l'utilisation de votre application.
  - Consommer de précieuses ressources système lorsque l'utilisateur ne l'utilise pas activement.
  - Perdre la progression de l'utilisateur s'il quitte votre application et y revient plus tard.
  - Plantage ou perte de la progression de l'utilisateur lorsque l'écran pivote entre l'orientation paysage et portrait.

Concepts du cycle de vie des activités



- Pour parcourir les transitions entre les étapes du cycle de vie d'une activité, la classe Activity fournit un ensemble de base de six appels: onCreate (), onStart (), onResume (), onPause (), onStop () et onDestroy ().
- Le système appelle chacun de ces rappels lorsqu'une activité entre dans un nouvel état.

onCreate ()

- L'appel a cette méthode se déclenche lorsque le système crée l'activité pour la première fois.
- Lors de la création de l'activité, l'activité passe à l'état Créée.
- Ce qui faut mettre dans cette méthode : les instructions et les déclaration qui ne doivent se produire qu'une seule fois pendant toute la durée de vie de l'activité.
- Par exemple :
  - lier des données à des listes.
  - associer l'activité à un ViewModel .
  - instancier certaines variables de portée de classe.
- Cette méthode reçoit le paramètre savedInstanceState, qui est un objet Bundle contenant l'état précédemment enregistré de l'activité. Si l'activité n'a jamais existé auparavant, la valeur de l'objet Bundle est nulle.

onStart ()

- Lorsque onCreate () se ferme, l'activité entre dans l'état Démarré et l'activité devient visible pour l'utilisateur.
- L'appel onStart () rend l'activité visible pour l'utilisateur, car l'application se prépare à ce que l'activité entre au premier plan et devienne interactive.

- La méthode onStart () se termine très rapidement et, comme pour l'état Created, l'activité ne reste pas résidente dans l'état Started. Une fois ce rappel terminé, l'activité entre dans l'état Reprise et le système appelle la méthode onResume ().

onResume()

- Lorsque l'activité entre dans l'état Reprise, elle revient au premier plan, puis le système appelle le rappel onResume ().
- L'application reste dans cet état jusqu'à ce que quelque chose se détache de l'application.
- C'est là que les composants du cycle de vie peuvent activer toute fonctionnalité devant être exécutée lorsque le composant est visible et au premier plan.
- Lorsqu'un événement interruptif se produit;
  - l'activité entre dans l'état Pause et le système appelle la méthode onPause ().
- Si l'activité revient à l'état Repris à partir de l'état Suspendu, le système appelle à nouveau la méthode onResume ().
- Pour cette raison, vous devez implémenter onResume () pour initialiser les composants que vous relâchez pendant onPause () et effectuer toutes les autres initialisations qui doivent se produire chaque fois que l'activité entre dans l'état Reprise.

onPause ()

- Le système appelle cette méthode comme la première indication que l'utilisateur quitte votre activité (bien que cela ne signifie pas toujours que l'activité est détruite);
- Utilisez la méthode onPause () pour suspendre ou ajuster les opérations qui ne devraient pas continuer.
- Il existe plusieurs raisons pour lesquelles une activité peut entrer dans cet état:
  - Un événement interrompt l'exécution de l'application,
  - Une nouvelle activité semi-transparente

- Vous pouvez utiliser la méthode onPause () pour libérer les ressources système, des capteurs (comme le GPS) ou toutes les ressources qui peuvent affecter la durée de vie de la batterie pendant que votre activité est en pause et que l'utilisateur n'en a pas besoin.

- L'exécution de onPause () est très brève et ne donne pas nécessairement suffisamment de temps pour effectuer des opérations de sauvegarde. Pour cette raison, vous ne devez pas utiliser onPause () pour enregistrer des données d'application ou d'utilisateur, effectuer des appels réseau ou exécuter des transactions de base de données; ce travail peut ne pas se terminer avant la fin de la méthode.

onRestart ()

- Le système appelle ce rappel lorsqu'une activité à l'état Arrêt est sur le point de redémarrer. onRestart () restaure l'état de l'activité à partir du moment où elle a été arrêtée.

- Ce rappel est toujours suivi de onStart ().

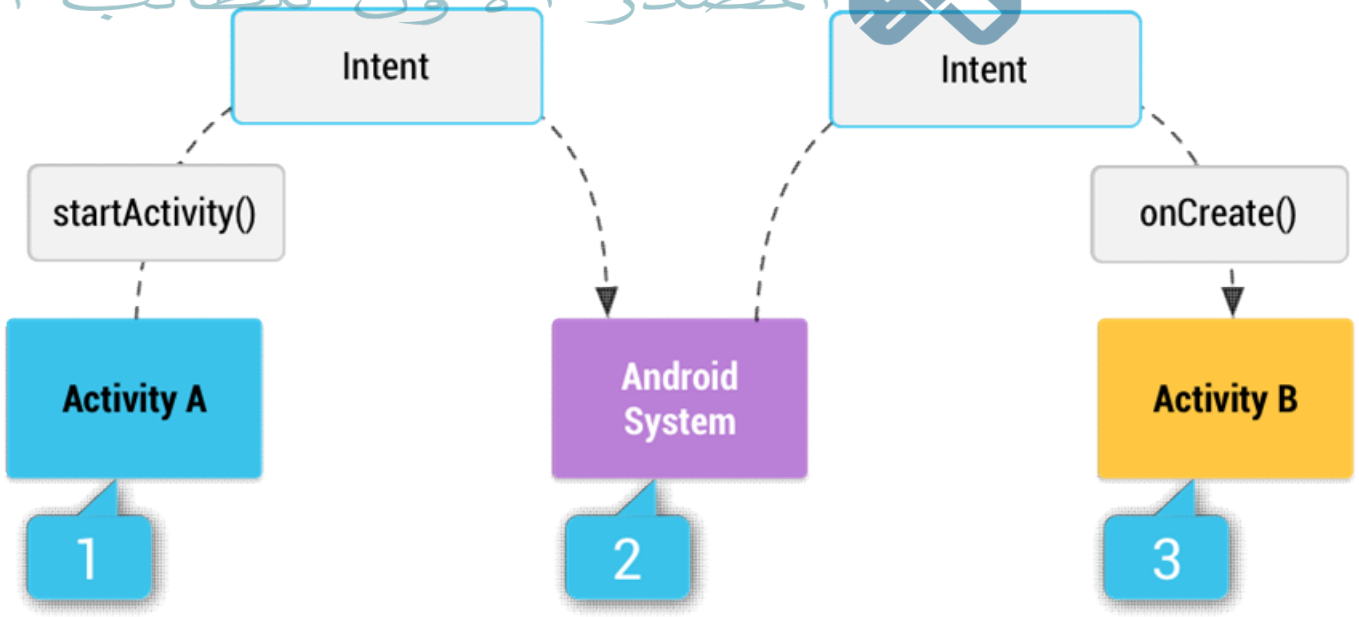
onDestroy ()

- onDestroy () est appelé avant que l'activité ne soit détruite.
- Le système appelle cette méthode:
  - l'activité est en cours de fin (en raison de la fermeture complète de l'activité par l'utilisateur,
  - le système détruit temporairement l'activité en raison d'un changement de configuration (comme la rotation de l'appareil).
- C'est là que les composants du cycle de vie peuvent nettoyer tout ce dont ils ont besoin avant que l'activité ne soit détruite.

Une intention (Intent)

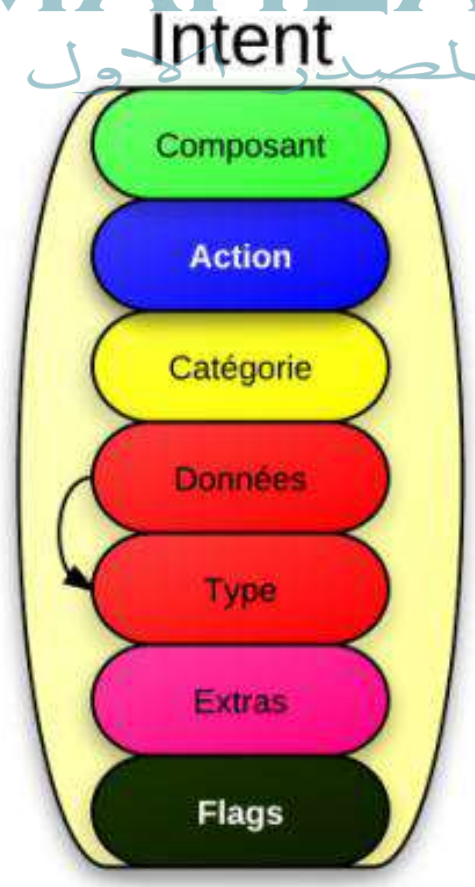
- Une intention est un objet de messagerie que vous pouvez utiliser pour demander une action à un autre composant d'application.
- il existe trois cas d'utilisation fondamentaux :
  - Démarrage d'une autre activité
  - Démarrage d'un service (*Un service est un composant qui effectue des opérations en arrière-plan*)
  - Diffusion d'une émission (BroadCast) (Batterie faible)

Exemple illustratif



Intent

- Un intent est un objet qui contient plusieurs champs.



Champs d'intent

- **Nom du composant:** c'est l'élément d'information essentiel qui rend une intention explicite.
- **Action:** Une chaîne qui spécifie l'action générique à effectuer (telle que la vue ou la sélection). *«L'action détermine en grande partie la structure du reste de l'intention»*
- **La catégorie** : fournit des informations supplémentaires sur l'action à exécuter et le type de composant qui devra gérer l'intent.
- **Les données** l'URI (un objet Uri) qui référence les données sur lesquelles agir et / ou le type MIME de ces données. Le type de données fournies est généralement dicté par l'action de l'intention. Par exemple, si l'action est ACTION_EDIT, les données doivent contenir l'URI du document à modifier.
- **Le type :** indique le type des données incluses. En précisant cet attribut vous pouvez imposer un type particulier.
- **Les extras :** pour ajouter du contenu à vos intents afin de les faire circuler entre les composants.
- **Les flags :** indiquer au système Android comment lancer une activité **.**

Type d'intentions

- Il existe deux types d'intentions:
  - Les intentions explicites spécifient quelle application satisfera l'intention, en fournissant soit le nom du package de l'application cible, soit un nom de classe de composant complet. Vous utiliserez généralement une intention explicite pour démarrer un composant dans votre propre application, car vous connaissez le nom de classe de l'activité ou du service que vous souhaitez démarrer.
  - Les intentions implicites ne nomment pas un composant spécifique, mais déclarent à la place une action générale à effectuer, qui permet à un composant d'une autre application de le gérer.

Intent Explicite

- Pour créer un intent explicite, il suffit de donner un Context qui appartient au package où se trouve la classe de destination
  - Intent I = **new Intent (Context context, Class<?> cls);**

- Pour lancer l'intent il existe **deux façons:**
  - Sans Retour
  - Avec Retour (le composant de destination nous renvoie une réponse).

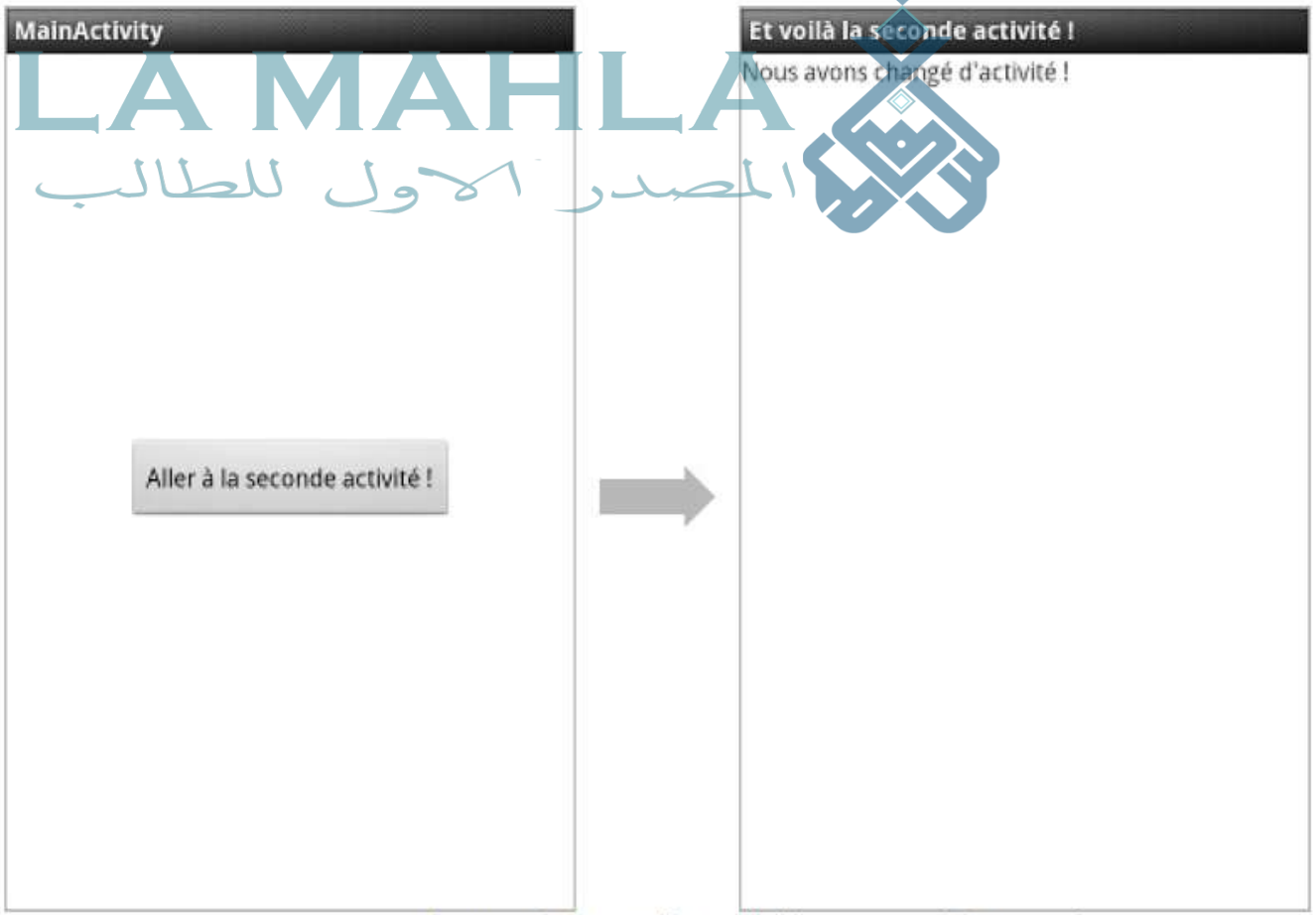
1) Activité Sans Retour

- Pour démarrer une activité *(sans retour);*


```
Intent intent= new I (this, Activity2.class);
startActivity (I);
```

- Avant de pouvoir démarrer l'activité, il faut au préalable la déclarer dans le fichier **AndroidManifest.xml de l'application.**
  - Sans cette déclaration, une erreur du type ***ActivityNotFoundException*** sera générée lors de l'appel de la méthode startActivity.
- Si on souhaite démaeer un service :
  - StartService(I)

Exemple



bouton de la première activité, on passe à la seconde

2) Activité Avec Retour

- Pour démarrer une activité qui renvoie un petit *feedback au retour, on utilise la méthode:*
  - ***void startActivityForResult(Intent intent, int requestCode)***
- La méthode ***startActivityForResult permet de communiquer une valeur de retour à l'activité parent.***

- La *'sous-activité' est identifiée avec un* ***requestCode.***
  - ***i.e identifier de manière unique un Intent***
- la méthode **setResult de la classe *Activity est utilisée pour renvoyer une valeur de retour, en indiquant comme paramètre le code de retour.***
- Android prévoit plusieurs valeurs par défaut telles que: *RESULT_OK; RESULT_CANCELED ;*

Activité Avec Retour

Les valeurs constantes (codes de retours)

Constante	Valeur
DEFAULT_KEYS_DIALER	1
DEFAULT_KEYS_DISABLE	0
DEFAULT_KEYS_SEARCH_GLOBAL	4
DEFAULT_KEYS_SEARCH_LOCAL	3
DEFAULT_KEYS_SHORTCUT	2
RESULT_CANCELED	0
RESULT_FIRST_USER	1
RESULT_OK	-1

2) Activité Avec Retour

- Pour envoyé un resultat; la seconde activity utilise :
  - **void setResult (int resultCode, Intent data)**
- Une fois retournée à la premiere activity :
  - La méthode **void onActivityResult (int requestCode, int resultCode, Intent data) est la première méthode de *callback appelée dans l'activité initiale quand l'activité appelée s'arrêtera*.**
  - La méthode ***onActivityResult utilise trois paramètres pour identifier l'activité et ses valeurs de retour :***
  - **requestCode: valeur identifiant quelle activité a appelé la méthode (le même code que celui passé dans *startActivityForResult*)**
  - **resultCode: valeur de retour envoyée par la sous-activité pour indiquer comment elle s'est terminée.**
    - **C'est une constante définie dans la classe *Activity (RESULT_OK « si l'activité s'est terminée normalement », RESULT_CANCELED « s'il y a eu un problème ou qu'aucun code de retour n'a été précisé », etc.)***
  - **Intent data : cet objet permet d'échanger des données**

Exemple

- **Exemple:**
  - La première activité contient un bouton qui lance une deuxième activité.
  - La seconde activité proposera à l'utilisateur de cliquer sur deux boutons.
  - Cliquer sur un de ces boutons retournera à l'activité précédente en lui indiquant lequel des deux boutons a été choisit.

Intent Implicite

- Une intention implicite spécifie une action qui peut appeler n'importe quelle application sur l'appareil capable d'exécuter l'action.
- Utile lorsque votre application ne peut pas effectuer l'action.

  Exemple : on souhaite appeler un numéro

```
Uri uri = Uri.parse("055055555");
Intent intent = new Intent(Intent.ACTION_DIAL,uri)
startActivity(intent);
```

Intent Implicite

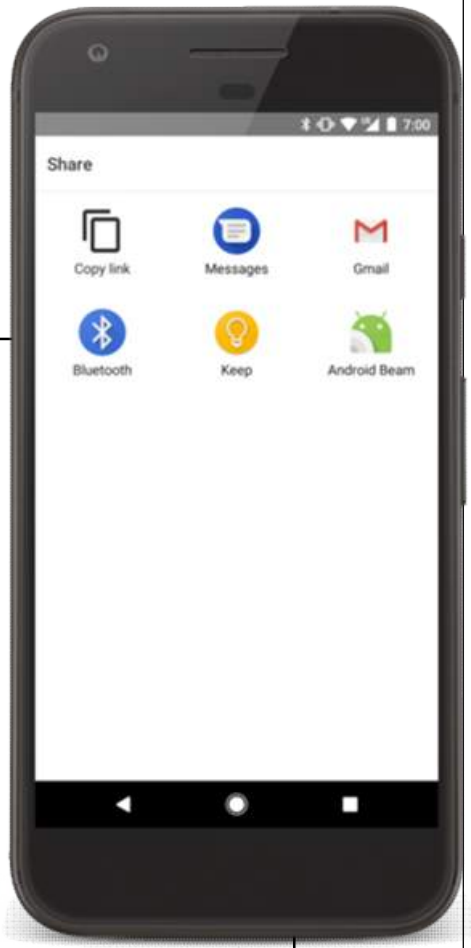
- Il est possible qu'un utilisateur ne dispose d'aucune application qui gère l'intention implicite que vous envoyez à startActivity (). Ou, une application peut être inaccessible en raison des restrictions de profil.
- Pour vérifier, utiliser resolveActivity comme suit ;

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

Intent Implicite

- Si plusieurs applications peuvent répondre à l'intention et que l'utilisateur souhaite utiliser une application différente à chaque fois, vous devez explicitement afficher une boîte de dialogue de sélection


```
Intent sendIntent = new Intent(Intent.ACTION_SEND);
// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show the chooser dialog
Intent chooser = Intent.createChooser(sendIntent, title);

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```



Intent Implicite

- Recevoir une intention implicite
  - Pour annoncer les intentions implicites que votre application peut recevoir, déclarez un ou plusieurs filtres d'intention pour chacun de vos composants d'application avec un élément <intent-filter> dans votre fichier manifeste.
  - Chaque filtre d'intention spécifie le type d'intentions qu'il accepte en fonction de l'action, des données et de la catégorie de l'intention.
  - Chaque filtre d'intention est défini par un élément <intent-filter> dans le fichier manifeste de l'application, imbriqué dans le composant d'application correspondant (tel qu'un élément <activity>).

Intent Implicite

- Dans le <intent-filter>, vous pouvez spécifier le type d'intentions à accepter en utilisant un ou plusieurs de ces trois éléments:

<action>

- Déclare l'action d'intention acceptée, dans l'attribut name. La valeur doit être la valeur de chaîne littérale d'une action, pas la constante de classe.

<data>

- Déclare le type de données acceptées, en utilisant un ou plusieurs attributs qui spécifient divers aspects de l'URI de données (schéma, hôte, port, chemin) et le type MIME.

<catégorie>

- Déclare la catégorie d'intention acceptée, dans l'attribut name. La valeur doit être la valeur de chaîne littérale d'une action, pas la constante de classe.
- Remarque: Pour recevoir des intentions implicites, vous devez inclure la catégorie CATEGORY_DEFAULT dans le filtre d'intention. Les méthodes startActivity () et startActivityForResult () traitent toutes les intentions comme si elles déclaraient la catégorie CATEGORY_DEFAULT. Si vous ne déclarez pas cette catégorie dans votre filtre d'intention, aucune intention implicite ne se résoudra à votre activité.