Université Ahmed ZABANA de Relizane

Faculté des Sciences et Technologies

Département d'Informatique

Applications Mobiles

3ème Année Informatique

2021 / 2022

- Chapitre 3 -

Responsable: Dr. Sofiane AMARA

Chapitre 3: Activités et ressources

1. **Notion d'activité**
2. **Cycle de vie d'une activité**
3. **Les ressources**
4. **Organisation des ressources**
5. **Utilisation des ressources**
    - **Les chaînes de caractère**
    - **Les drawables**

1. Notion d'Activité (1/7)

- **Une activité est le point d'entrée pour interagir avec l'utilisateur.**
- **Il représente un écran unique avec une interface utilisateur.**
- **Une ACTIVITÉ est un PROGRAMME (Code JAVA) avec une INTERFACE GRAPHIQUE (XML).**

1. Notion d'Activité (2/7)

Une activité est composée de deux parties:

1. **La logique de l'activité: Définie en Java à l'intérieur d'une classe étendant Activity**

```
1  public class MainActivity extends Activity { //...
2  }
3
```

2. **L'interface utilisateur: Définie généralement à l'intérieur d'un fichier XML (dans le dossier/res/ layout/)**

1. Notion d'Activité (3/7)

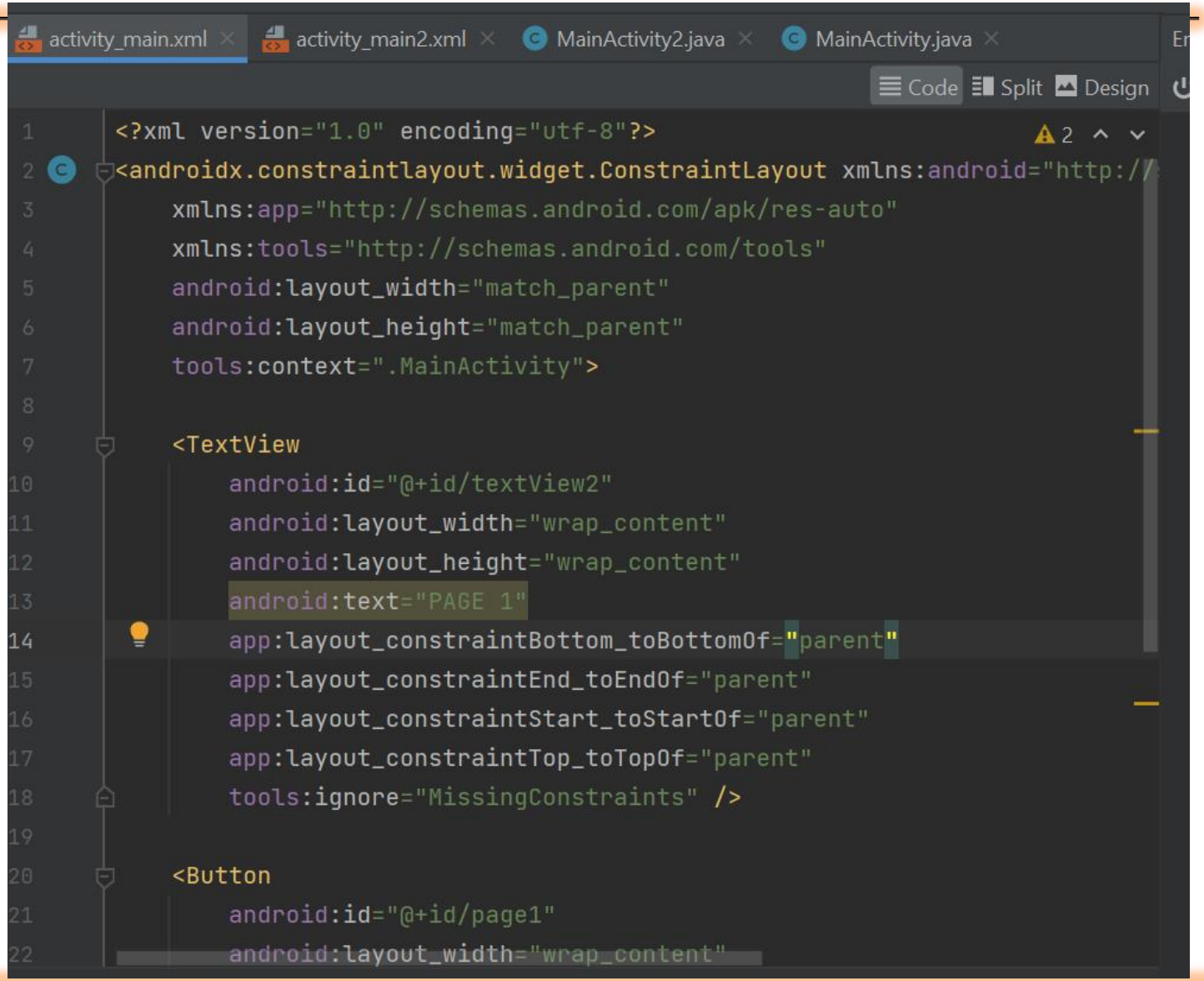
Exemple d'une activité: la partie Logique (Java)

activity_main.xml | activity_main2.xml | MainActivity2.java | MainActivit

```java
package com.example.tp_am;

import android.content.Intent;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {



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

```

1. Notion d'Activité (4/7)

Exemple d'une activité: la partie Interface (Fichier XML)



Pourquoi séparer entre la partie interface et la partie logique?

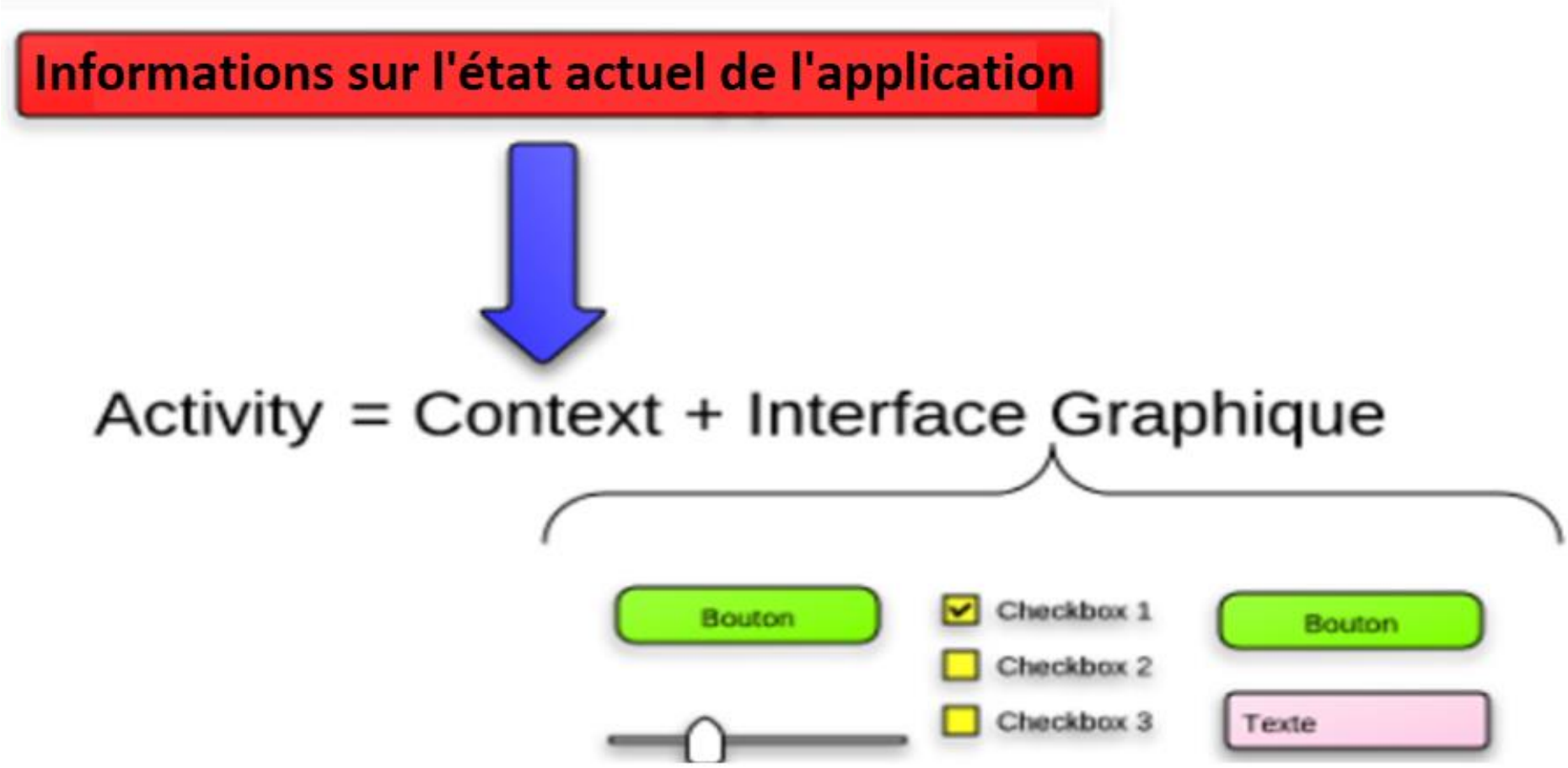
1. Notion d'Activité (5/7)

- **Une application peut avoir une ou plusieurs activités.**

  Par exemple, une application de messagerie peut avoir:

  - ❑ **Une activité qui affiche une liste des e-mails reçus,**
  - ❑ **Une activité pour les e-mail envoyés,**
  - ❑ **Une autre activité pour les messages indésirables (Spam).**

1. Notion d'Activité (6/7)



1. Notion d'Activité (7/7)

- **Une activité contient des informations sur l'état actuel de l'application : ces informations s'appellent le contexte.**
- **Ce contexte constitue un lien avec le système Android ainsi que les autres activités de l'application.**

1.2 Les états d'une activité :

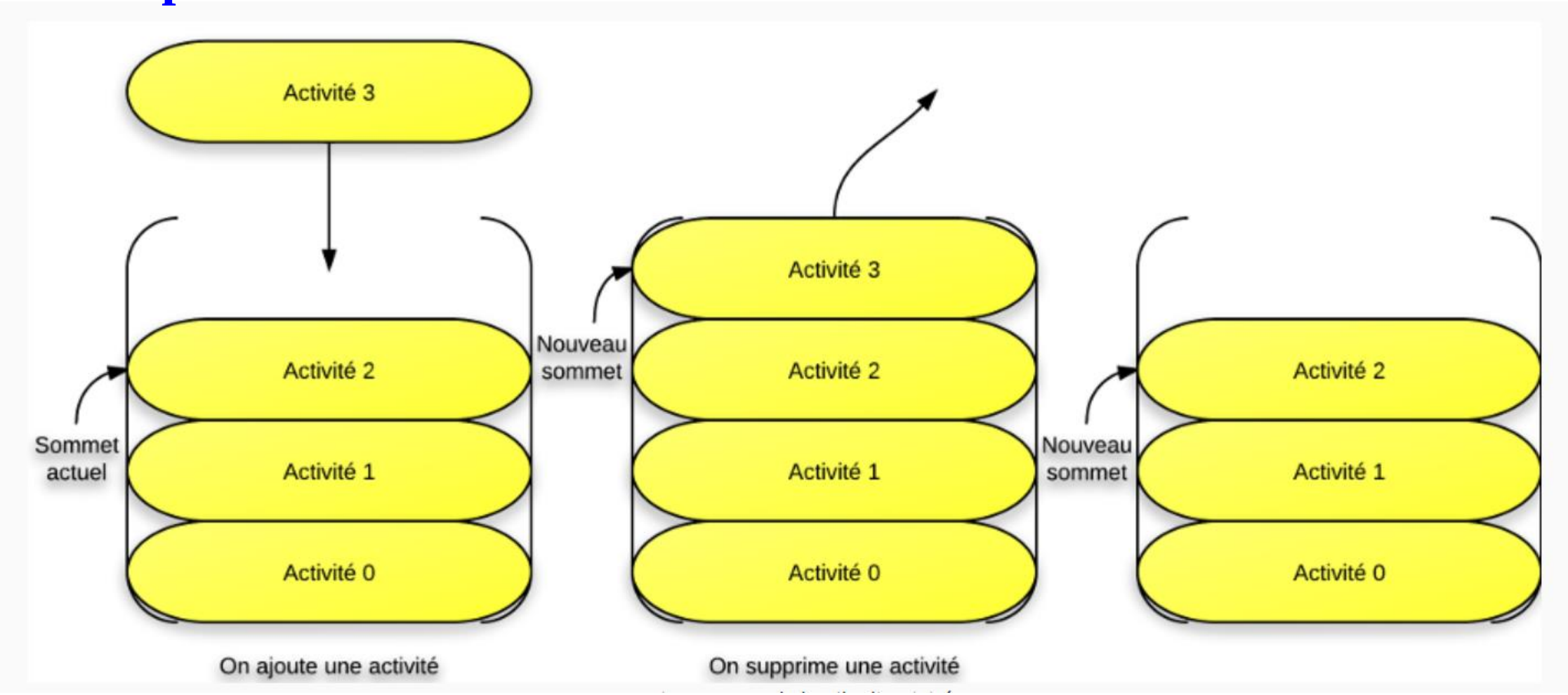
- **À tout moment une activité peut laisser sa place à une autre activité qui a une priorité plus élevée.**
- **EX: lorsque l'utilisateur reçoit un appel téléphonique, il est plus important pour lui de répondre à cet appel que de faire d'autre activité.**
- **Quand une activité se lance, elle se met en haut de la pile d'activités**

1.2 Les états d'une activité :

1.2.2 la pille d'activité :

- **Une pile est une structure de données de type « LIFO =Last In First Out», c'est-à-dire qu'il n'est possible d'avoir accès qu'à un seul élément de la pile, le tout premier élément, aussi appelé sommet.**
- **Quand on ajoute un élément à cette pile, le nouvel élément prendra la première place et deviendra le nouveau sommet**

1.2.2 la pille d'activité :



1.2 Les états d'une activité :

A- État Actif (Active):

- **Une activité en état actif est une activité qui est visible en totalité**
- **Elle est sur le dessus de la pile. Donc c'est cette activité qui est consultée par l'utilisateur à ce moment**
- **Elle a tout le FOCUS (l'utilisateur agit directement avec cette activité)**

1.2 Les états d'une activité :

B- État Suspendu (Paused):

- **Une activité en état suspendu est partiellement visible à l'écran**
- **Elle n'a pas le FOCUS. (l'utilisateur ne peut pas agir directement avec cette activité).**
- **Pour que l'activité puisse récupérer le focus, l'utilisateur doit se débrasser de l'activité qui l'obstrue**
- **Ex: lorsque on reçoit un SMS alors qu'on est entrain de consulter une page.**

1.2 Les états d'une activité :

C- État arrêté (Stopped):

- **Une activité en état arrêté est une activité qui est invisible sur l'écran.**
- **Elle n'a plus le focus (l'utilisateur ne peut pas voir cette activité et il ne peut pas interagir avec)**

1.2 Les états d'une activité :

Pour résumer :

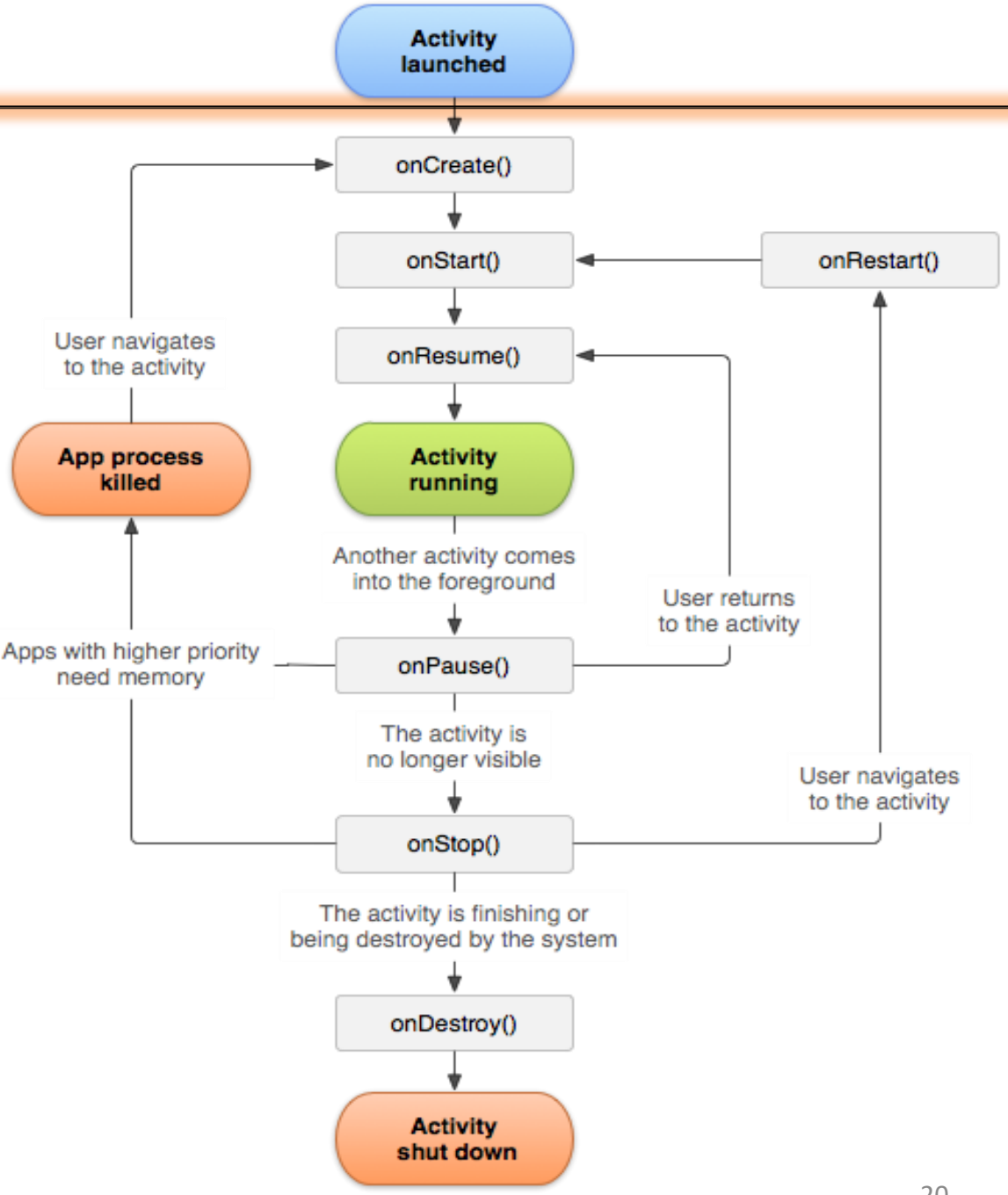
État	Visibilité	Exemple
Active (active ou running)	L'activité est visible en totalité.	L'utilisateur consulte l'application et il peut l'utiliser dans son intégralité
Suspendue (paused)	L'activité est partiellement visible à l'écran.	Quand vous recevez un SMS et qu'une fenêtre semi-transparente se pose devant votre activité pour afficher le contenu du message
Arrêtée (stopped)	L'activité n'est pas visible. Elle est tout masquée par une autre activité	Le système tue l'application pour libérer de la mémoire système.

Est-ce qu'il est possible d'avoir deux applications actives en même temps ?

2. Cycle de vie d'une activité:

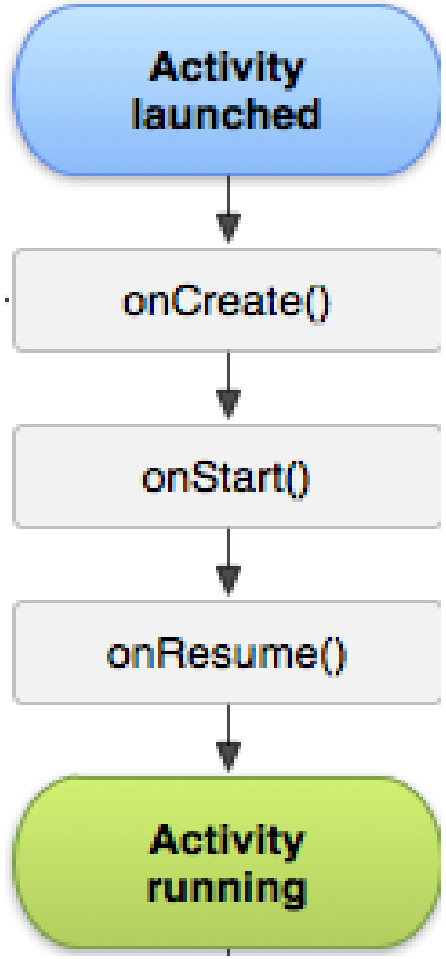
- **Le cycle de vie d'une activité indique les différentes étapes (états) que cette activité peut traverser durant sa vie, depuis sa naissance jusqu'à sa mort.**
- **C'est le système d'exploitation qui est le responsable de la gestion de ces étapes (création, mettre en pause, destruction, etc).**
- **Le passage entre les états de cycle de vie est assuré par l'utilisation des méthodes de « Calback »: onCreate, onStart(), onPause(), onStop()… etc.**

Cycle de vie d'une activité (extrait de la documentation officielle Android)



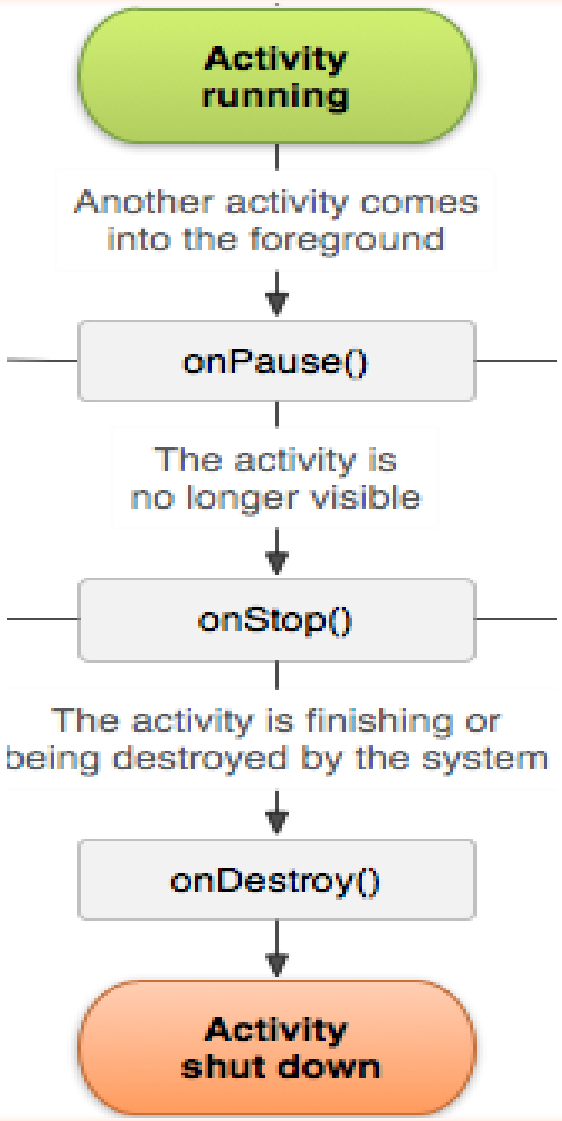
2. Cycle de vie d'une activité:

- **Lors du DÉMARRAGE d'une activité, 3 méthodes sont appelées automatiquement :**
  - **onCreate()**
  - **onStart()**
  - **onResume()**



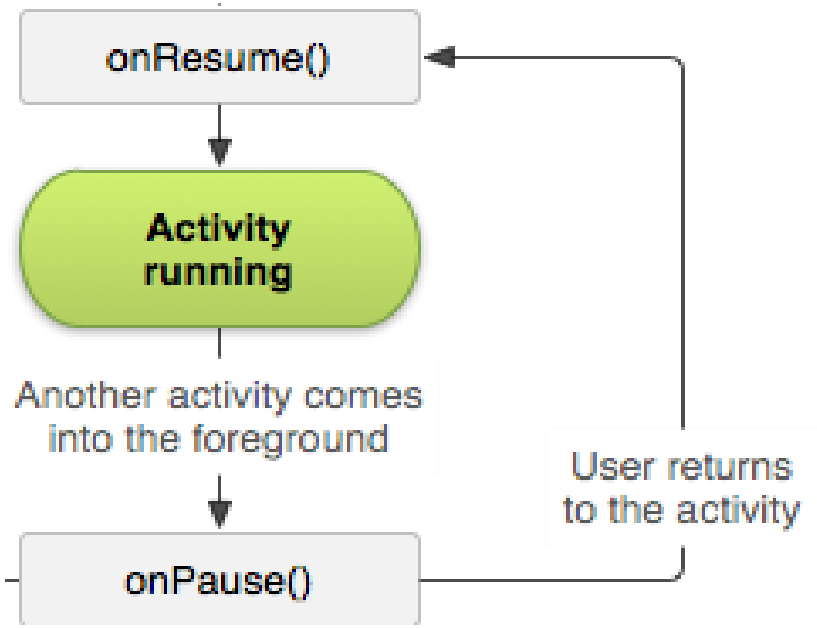
2. Cycle de vie d'une activité:

- **Lorsque on QUITTE une activité, 3 méthodes sont appelées automatiquement :**
- **onPause,**
- **onStop**
- **onDestroy.**



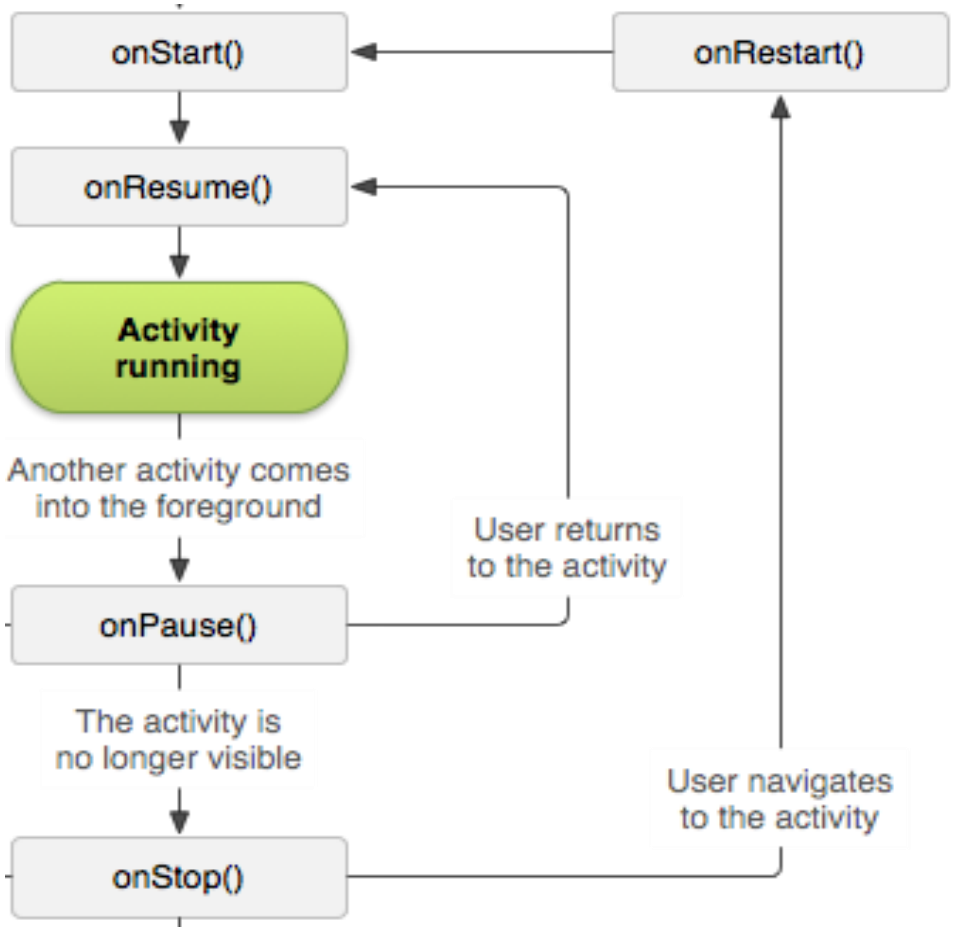
2. Cycle de vie d'une activité:

- **Une activité peut passer d'un état suspendu à un état actif, en appelant la méthode onResume().**



2. Cycle de vie d'une activité:

- **Une activité peut passer d'un état arrêté à un état actif, en appelant la méthode onRestart(), puis la méthode onStart(), puis la méthode onResume().**

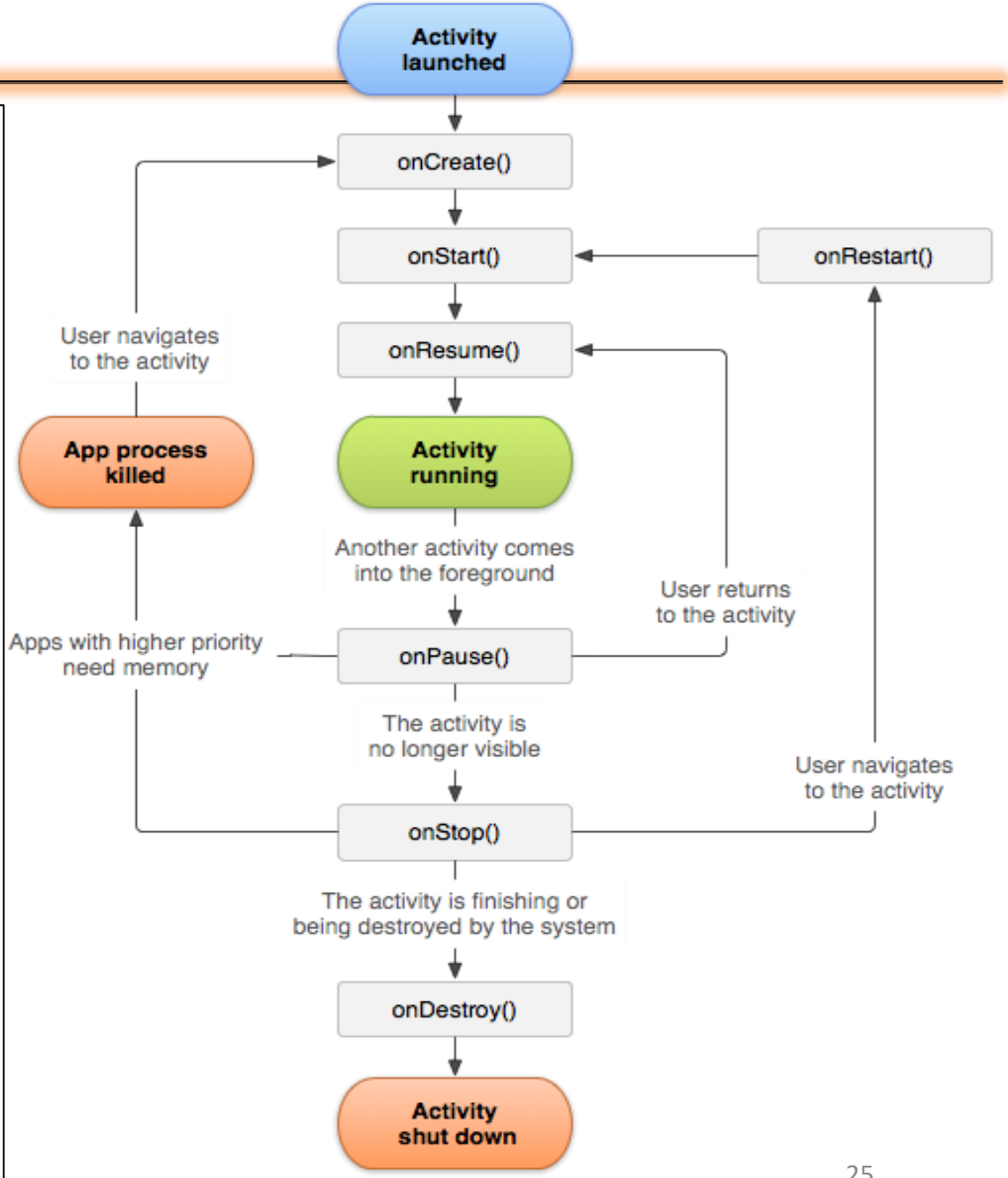


Pour mieux comprendre ce schéma, on propose le scénario suivant:

1- Un utilisateur lance une application de Messagerie qui doit afficher une activité pour présenter la liste des mails. Cette activité est lancée par le système, et ses méthodes onCreate, onStart et onResume sont appelées.

2- Lorsque l'utilisateur consulte la liste de ces mails, une autre activité vient de s'afficher sur l'écran (ex: un nouveau SMS). La première activité (Mails) doit alors se mettre en pause en appelant la méthode onPause.

3- L'utilisateur peut soit ignorer la deuxième activité (SMS) et revenir vers la première activité (Mails) en appelant la méthode onResume, soit rester sur la deuxième activité (SMS) et mettre la première activité dans un état d'arrêt en appelant la méthode onStop.

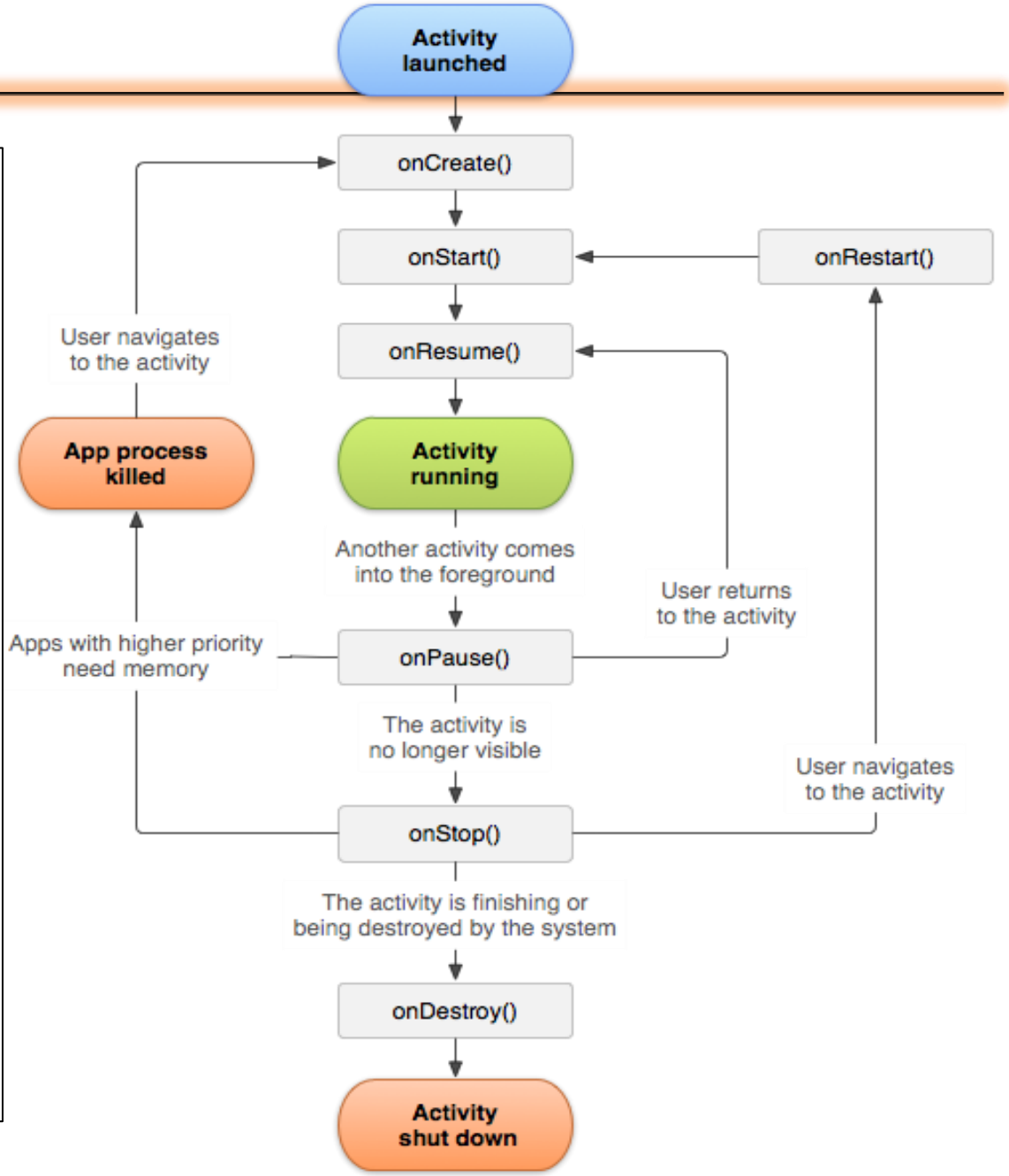


4- la premier activité (Mails) (qui est en état d'arrêt) peut avoir trois directions:

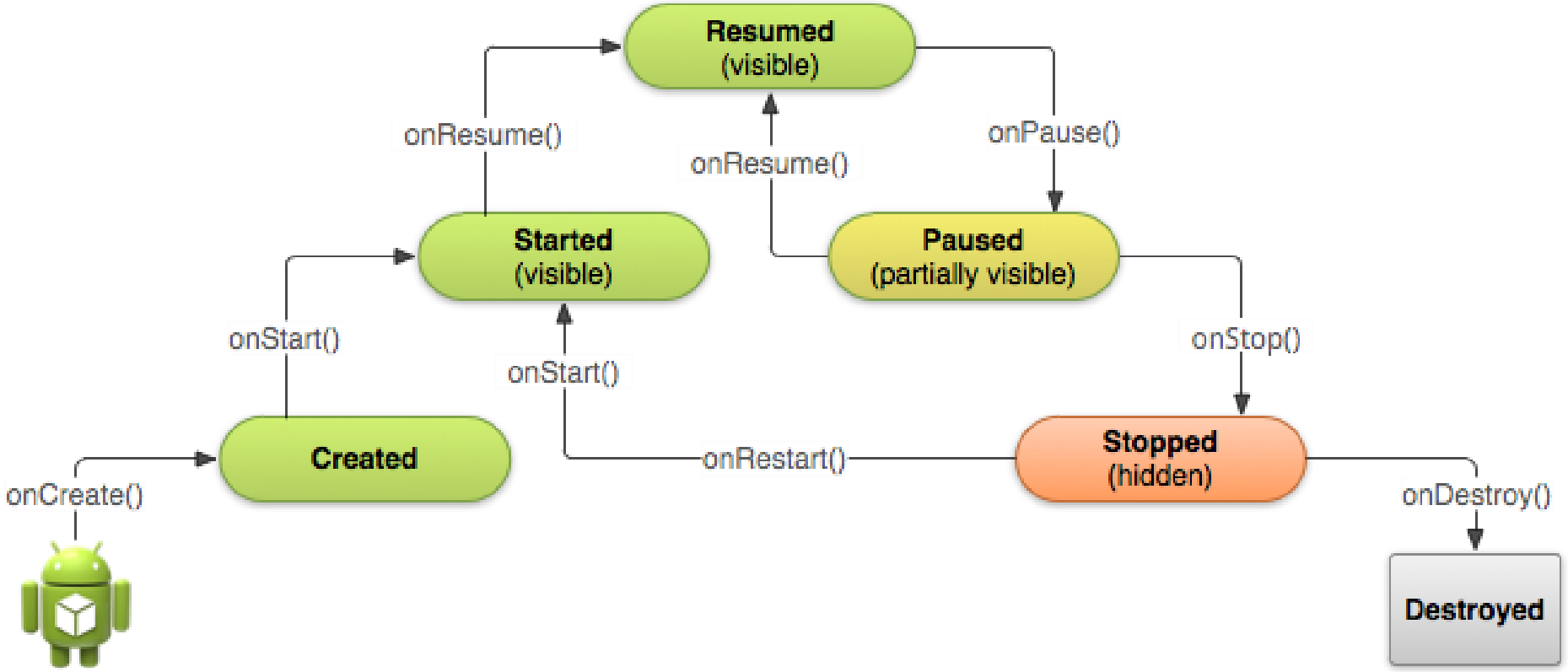
A- L'utilisateur réouvre l'activité. Et donc cette activité doit être à nouveau affichée et ses méthodes onRestart, onStart, et onResume sont appelées avant cela.

B- S'il y'a d'autres applications d'une priorité plus élevée qui ont besoin de la mémoire, le système doit trouver une solution. et il peut tuer les processus de l'activité (Mails). Après cela, l'activité doit entièrement recréer en appelant les méthodes onCreate, onStart, et onResume.

C- l'activité (Mails) peut être arrêtée définitivement soit par l'utilisateur ou par le système en appelant la méthode onDestroy.



Pour résumer:



2.2. Les méthodes de cycle de vie :

onCreate(Bundle) :elle est appelée à la création d'une activité, elle permet l'initialisation de tous les éléments, et un bundle est passé à cette méthode, contenant l'état précédent de l'activité. Cette méthode est toujours suivie par la méthode onStart().

onStrat(): elle est appelée juste avant que l'activité devienne visible, elle est suivie par onResume()

onResume() : elle est appelée juste avant que l'activité devienne en premier plan (l'activité est en haut de la pile). elle est toujours suivie de onPause().

2.2. Les méthodes de cycle de vie :

onPause() :

- Elle est appelée quand le système démarre une autre activité.
- Elle est utilisée typiquement pour enregistrer les données non sauvegardées et pour arrêter les animations ou tout ce qui consomme de la mémoire.
- Elle est Suivie de : onResume si l'activité est rechargée en premier plan et par onStop si l'activité devient invisible.

2.2. Les méthodes de cycle de vie :

onStop() :

- Elle est appelée quand l'activité n'est plus visible a l'utilisateur.
- Elle est suivie par onRestart() si l'activité recommence a interagir avec l'utilisateur et par onDestroy() si l'activité va disparaître.
- Dans cet état, l'activité peut être tuée par le système.

2. Cycle de vie d'une activité: Exemple

```
public class Main extends Activity {

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.acceuil);
   }

 protected void onDestroy() {
       super.onDestroy(); }

 protected void onPause() {
  super.onPause(); }

 protected void onResume() {
       super.onResume(); }

 protected void onStart() {
       super.onStart(); }

 protected void onStop() {
      super.onStop(); }
}
```

Pourquoi implémenter ces méthodes ?

Pourquoi implémenter ces méthodes ?

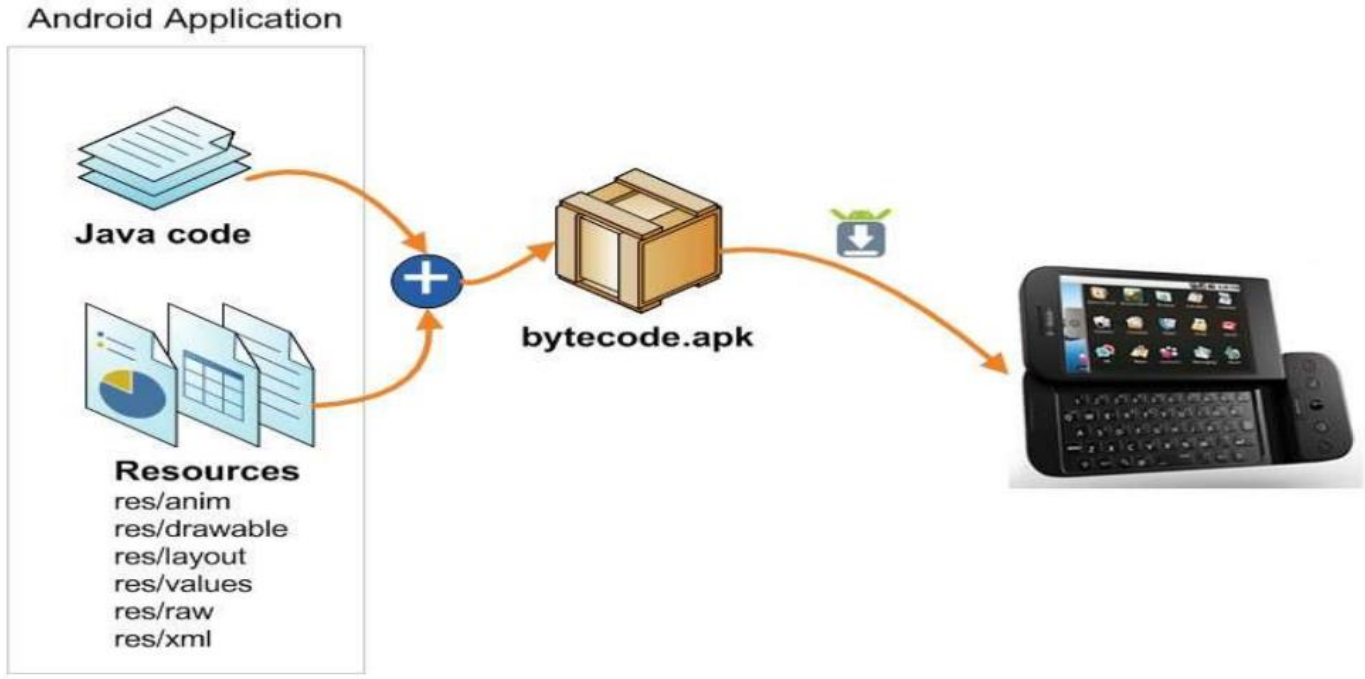
- Pour ne pas avoir de problème lors de la création/ restauration de l'application par le système (lors d'une rotation de l'écran par exemple).
- Pour ne pas consommer trop de ressources système
- Pour permettre à l'utilisateur de basculer rapidement entre les différentes applications en sauvegardant leurs états et données .

3. Les ressources (1/4) :

- Android est destiné à être utilisé sur un très grand nombre de supports différents (taille de l'appareil, la résolution de son écran, la langue d'utilisation, l'orientation de l'écran, etc.)
- Android doit alors s'adapter à ces supports.
- Par exemple, Android doit permettre à une application de s'afficher de la même manière sur un écran 6" que sur un écran 10"
- Ceci est réalisé à l'aide des ressources.
- Ces ressources sont des informations statiques stockées dans le répertoire **res**.

3. Les ressources (2/4):

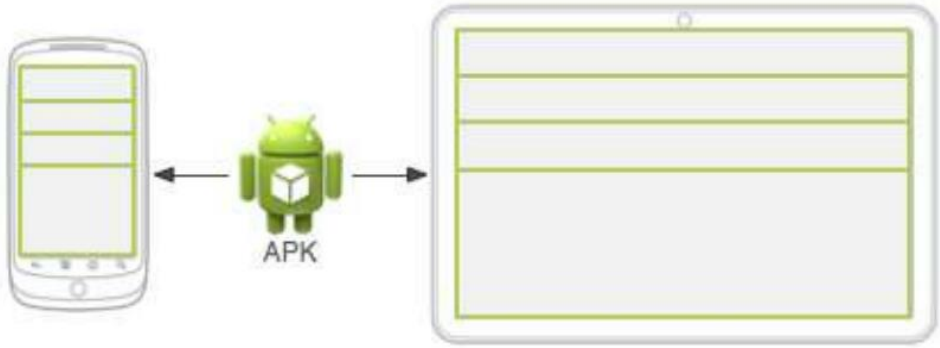
- Parmi les ressources nous pouvons citer : images, vidéo, audio, chaînes de caractères, etc.
- Ces ressources sont regroupées dans le fichier « apk » lors du processus de construction de l'application:



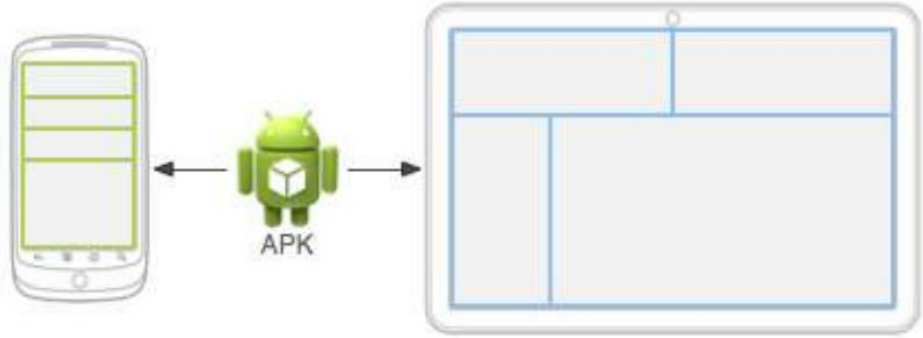
3. Les ressources (3/4) :

- Le SDK Android propose des ressources prédéfinies (par défaut).
- Chaque application doit fournir des ressources alternatives pour prendre en charge des configurations d'appareils spécifiques.
- Par exemple, vous devez inclure des ressources alternatives pour différentes densités d'écran (Ldpi, Mdpi, Hdpi..), et des ressources alternatives pour différentes langues (FR, En, Ar, Es..)
- Au moment de l'exécution, Android détecte la configuration actuelle de l'appareil et charge les ressources appropriées pour l'application.

3. Les ressources (4/4):

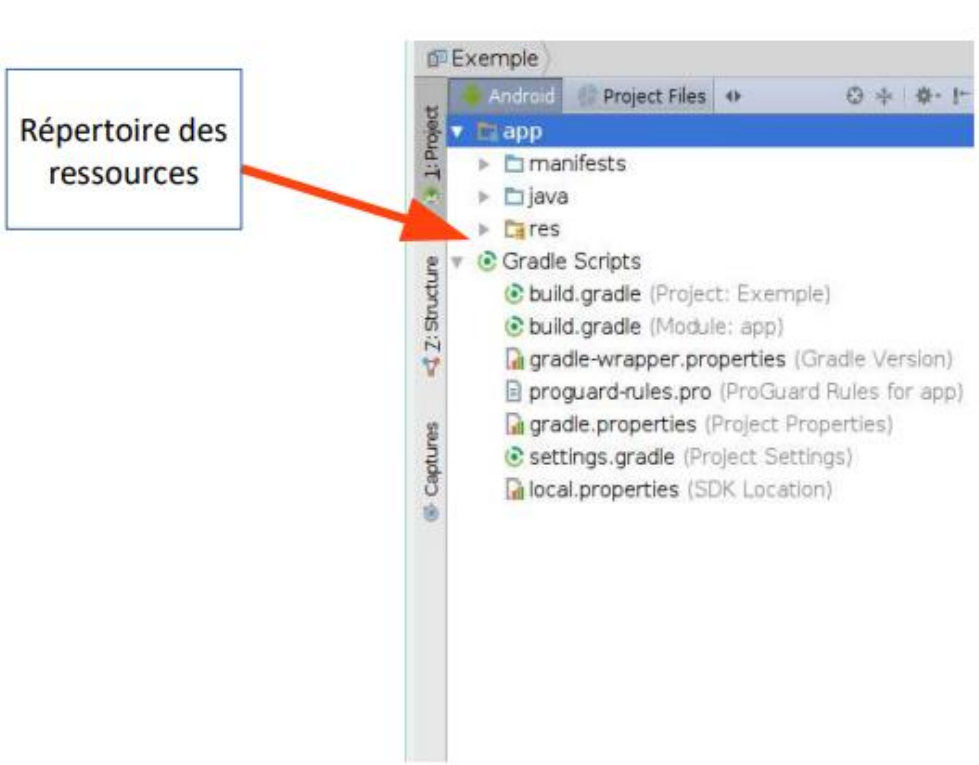


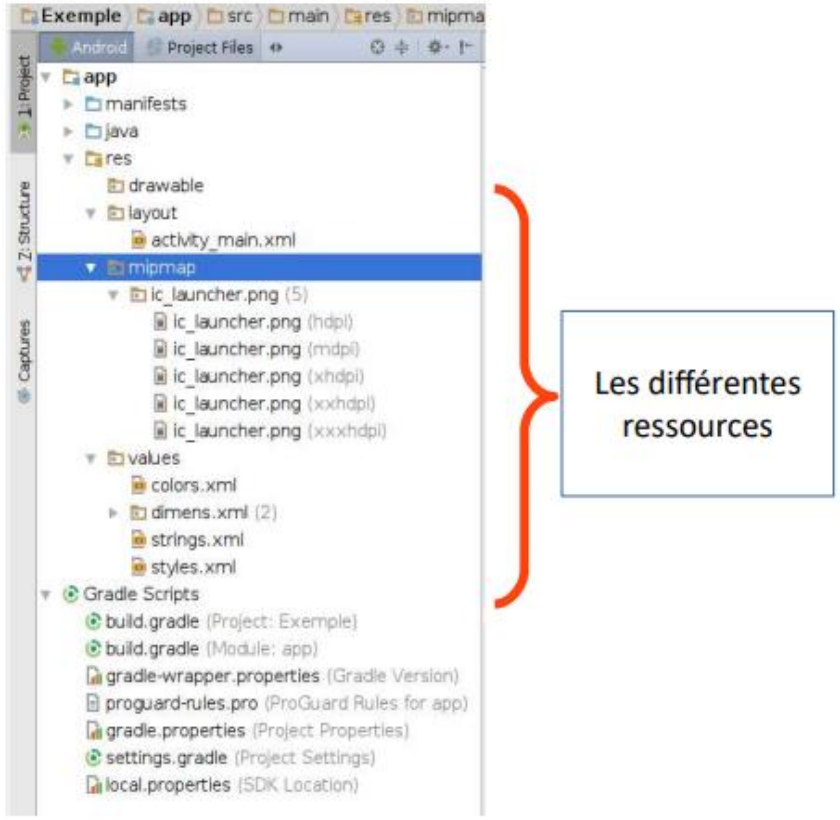
- Pour deux appareils différents où chacun utilise la configuration par défaut, l'application ne fournit pas une mise en page alternative pour un appareil ayant un écran plus grand.



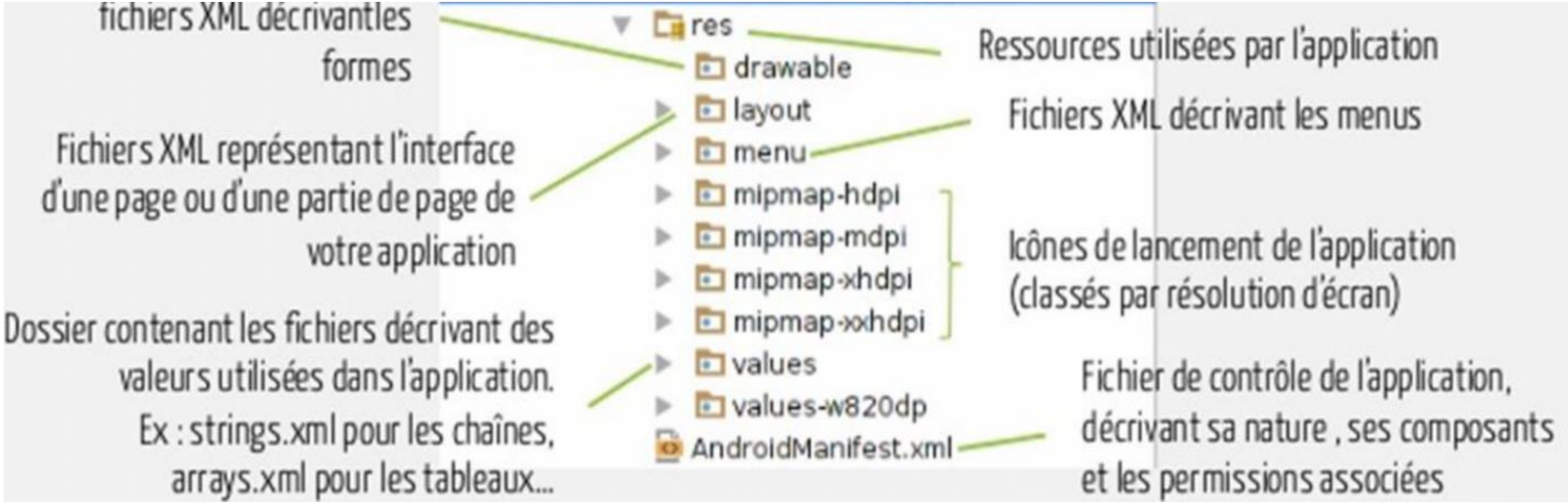
- Pour deux appareils différents où chacun utilise sa propre mise en page, l'application va choisir à chaque fois une mise en page différente associée à la taille de l'écran

4. Organisation des ressources (1/4)



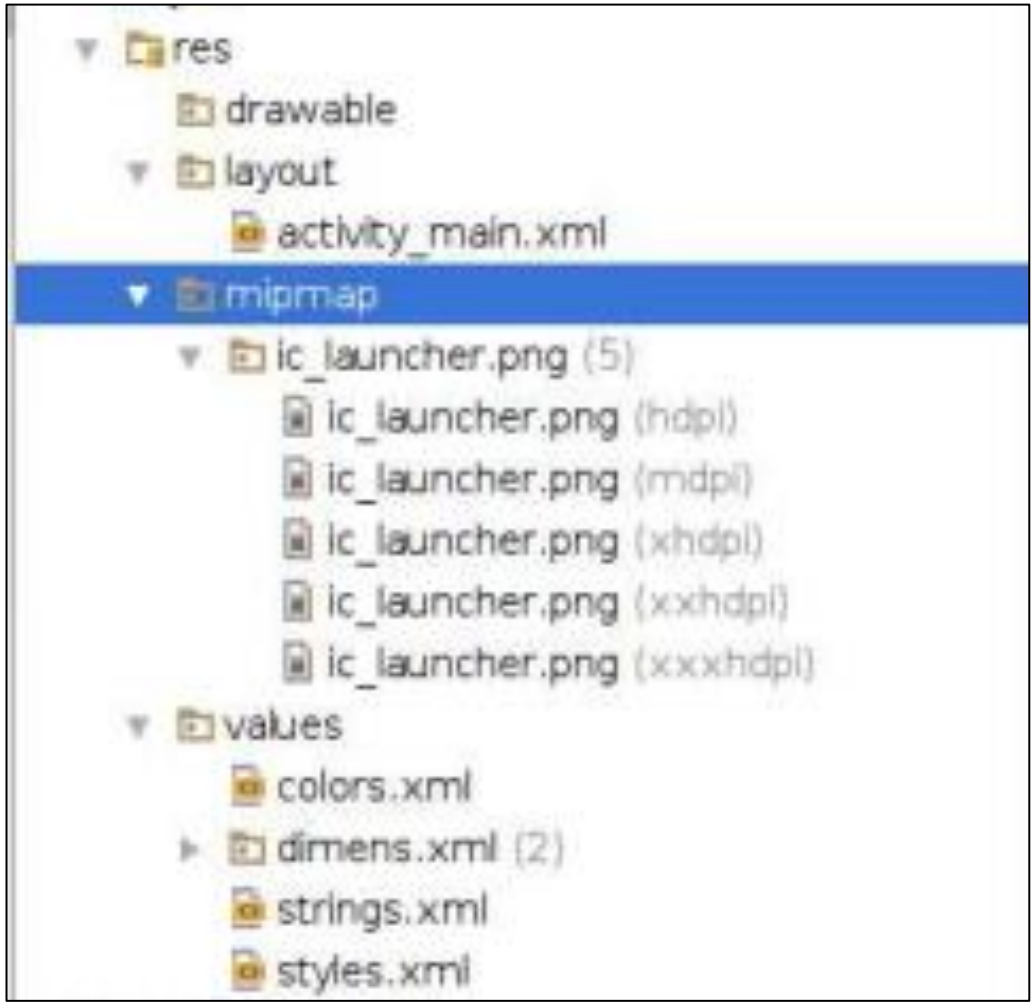


4. Organisation des ressources (2/4)



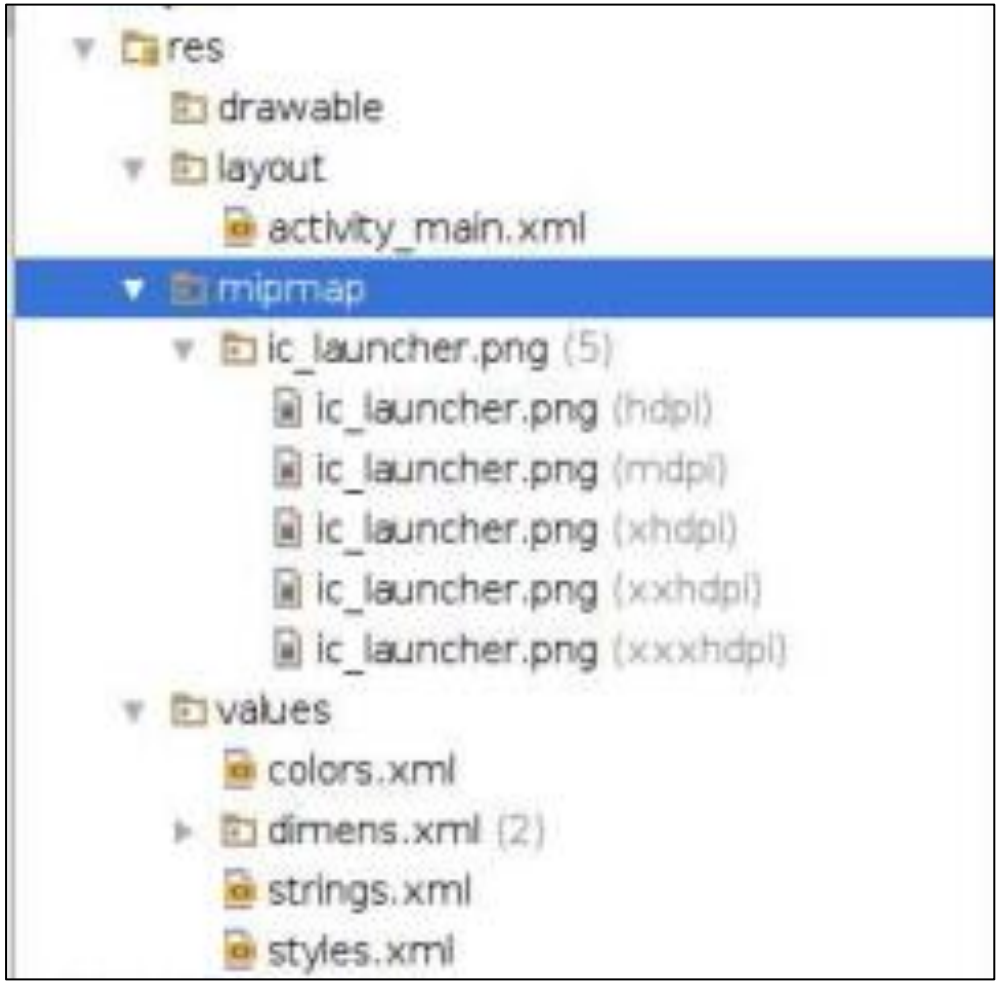
4. Organisation des ressources (3/4)

- « **drawable** » : ce répertoire contient les ressources qui peuvent être dessinées
- « **layout** » : ce répertoire contient les interfaces graphiques de notre application en fichiers xml
- « **mipmap** » : les ressources de différentes densités pour les icônes de lancement.



4. Organisation des ressources (4/4)

- « **values** » : des fichiers XML qui contiennent des valeurs simples, tels que:
  - **colors.xml:** pour les différents couleurs à utiliser par l'app
  - **string.xml:** pour les chaines de caractères
  - **dimension.xml:** pour les dimensions.
  - **styles.xml:** pour les styles



5. Utilisation des ressources

5.1. Les chaines de caractères :

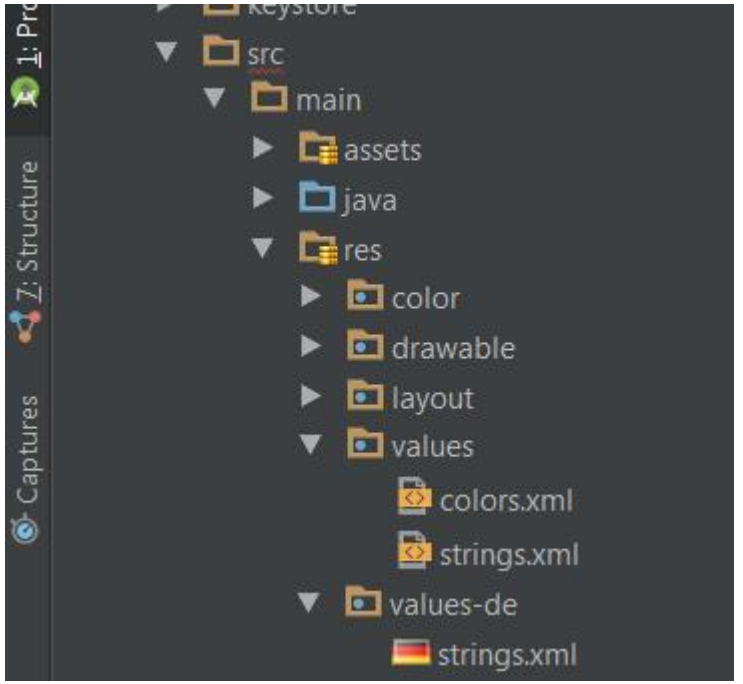
- **Le fichier:** */res/values/**strings.xml*** **permet de stocker les différentes chaines de caractères qui peuvent être utilisées par l'application.**
- **Par exemple, on peut définir une nouvelle chaine de caractères (nommée mon_texte) comme montré sur la figure:**

```
<resources>
    <string name="mon_text">Cours applications mobiles 2021/2022</string>
</resources>
```

- **On peut ensuit utiliser cette chaine sur d'autres parties de l'application**

5.1.1 Traduire une chaine de caractères (1/2):

- **Une chaînes peuvent être utilisée (traduite) en différentes langues du monde en définissant un fichier strings.xml différent pour chaque langue**
- **Pour ajouter une nouvelle langue, on doit créer un nouveau répertoire de valeurs avec le code de langue ISO comme suffixe. Par exemple, lors de l'ajout d'un ensemble allemand, La structure peut ressembler à ceci:**
- **Lorsque le système recherche la chaîne demandée, il vérifie d'abord le fichier XML spécifique à cette langue. S'il n'est pas trouvé, la valeur du fichier strings.xml par défaut est renvoyée**



5.1.1 Traduire une chaine de caractères (2/2) – Exemple - :

- **/res/values/strings.xml**

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

- **/res/values-fr/strings.xml**

```
<resources>
    <string name="hello_world">Bonjour tout le monde !!!</string>
</resources>
```

- **Si la langue du système est française (fr) le deuxième fichier string.xml doit être utilisé par l'application , et donc la chaine « Bonjour tout le monde » sera affichée . Si non (langue du système est arabe par ex) le premier fichier string.xml doit être utilisé, et donc la chaine « Hello World !» sera affichée.**

Les drawables:

- **Ce répertoire contient les ressources qui peuvent être dessinées.**
- **Parmi ces ressources, on peut citer les images (jpeg, gif, etc.), Les formes génériques (cercles, carrés, etc.) définies dans des fichiers XML.**
- **Chaque résolution de la ressource est associée à son propre dossier:**

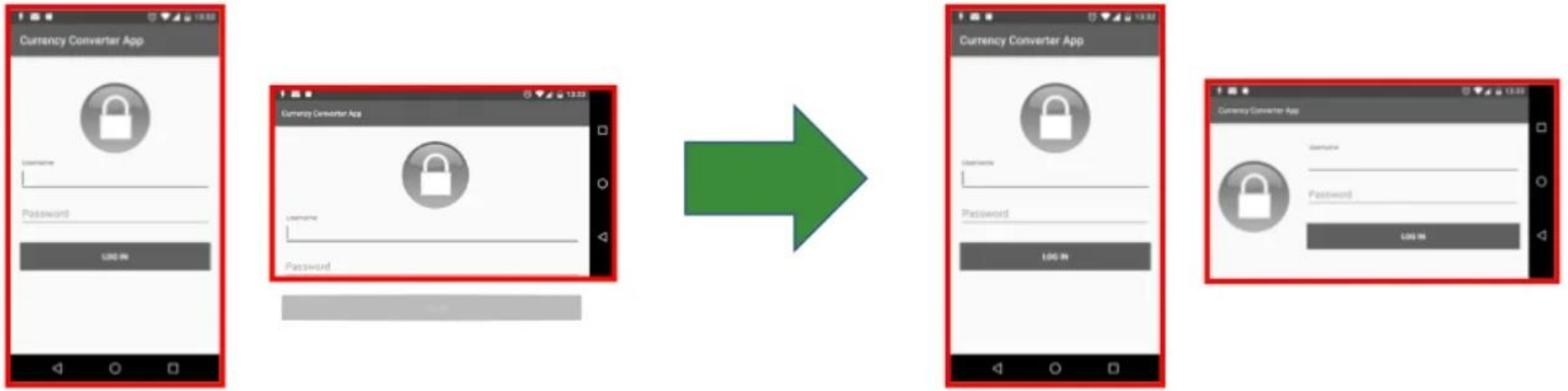
ldpi	mdpi	hdpi	xhdpi	xxhdpi
120 dpi	160 dpi	240 dpi	320 dpi	480 dpi

Dpi= dots per inch (en Français: PPI = Point par pouce)

1 pouce=2.54 cm

Comment faire la différence entre une présentation portrait et une présentation paysage?

Vue portrait vs. vue paysage



- **Pour faire la différence entre une présentation portrait et une présentation paysage, il est conseillé de créer sous le répertoire res :**
  - **Un répertoire layout-land (landscape) avec les fichiers XML d'IHM pour les présentations paysage**
  - **Un répertoire layout-port (portrait) avec les fichiers XML d'IHM pour les présentations portrait**

les fichiers par défaut sont mis dans le répertoire layout

Les icônes (1/2):

- On indique l'icône d'une application dans l'AndroidManifest.xml par la balise android:icon :

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tp_interface">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Tp_interface">
        <activity
```

- ic__launcher est le nom de fichier image sous le répertoire : res/mipmap

- Pour changer l'icône de notre application il suffit d'ajouter un fichier image dans ce répertoire et d'indiquer son nom dans l'AndroidManifest.xml

Comment choisir les ressources (par ex: les icônes) en fonction de la configuration de chaque appareil Android?

Les icônes (2/2):

Pour choisir les icônes appropriés à chaque configuration, il est mieux de prévoir 4 fichiers images de même nom dans chacun des répertoires suivants:

- **mipmap-ldpi** (120 dpi, Low density screen) - icône de 36px x 36px
- **mipmap-mdpi** (160 dpi, Medium density screen) - icône de 48px x 48px
- **mipmap-hdpi** (240 dpi, High density screen) - icône de 72px x 72px
- **mipmap-xhdpi** (320 dpi, Extra-high density screen) - icône de 96px x 96px

Et selon la configuration (la résolution d'écran) de chaque terminale, l'application va automatiquement utiliser le répertoire qui correspond avec cette configuration

Merci de votre attention!

N’hésitez pas à poser vos questions…