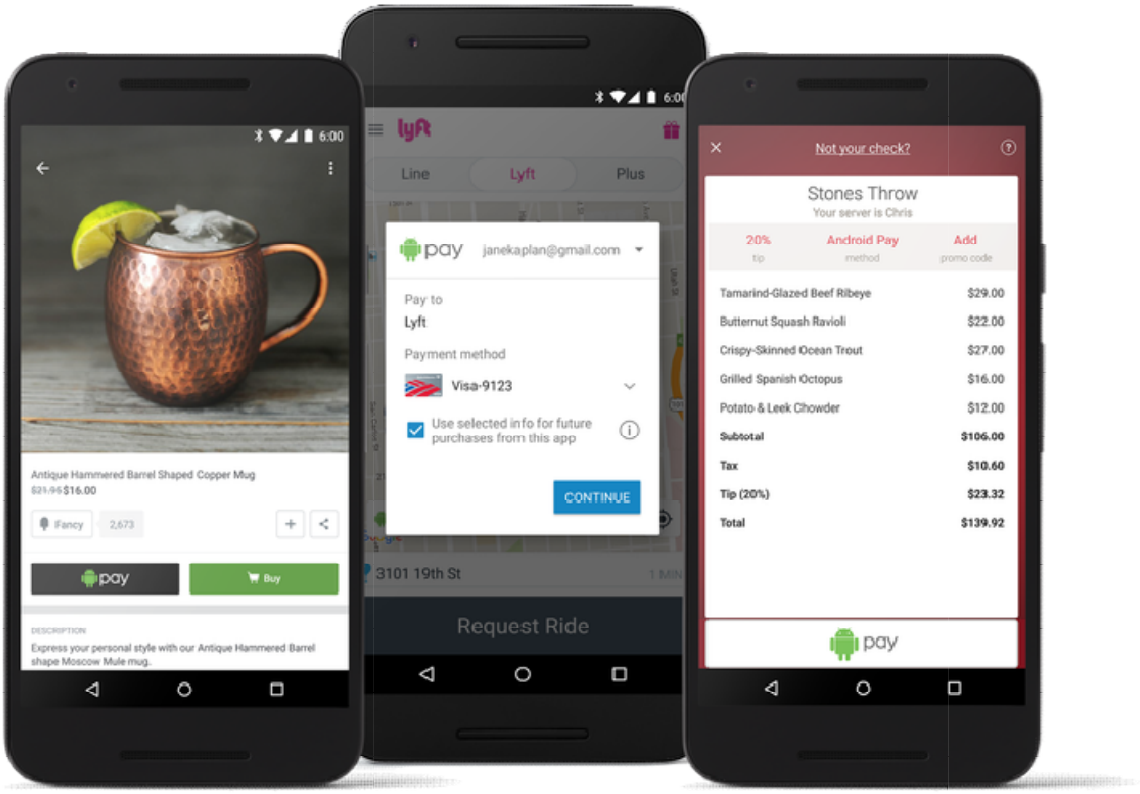
Android Studio : les premiers pas.

Concepts de BASE

Activité et VIEWS(vues)

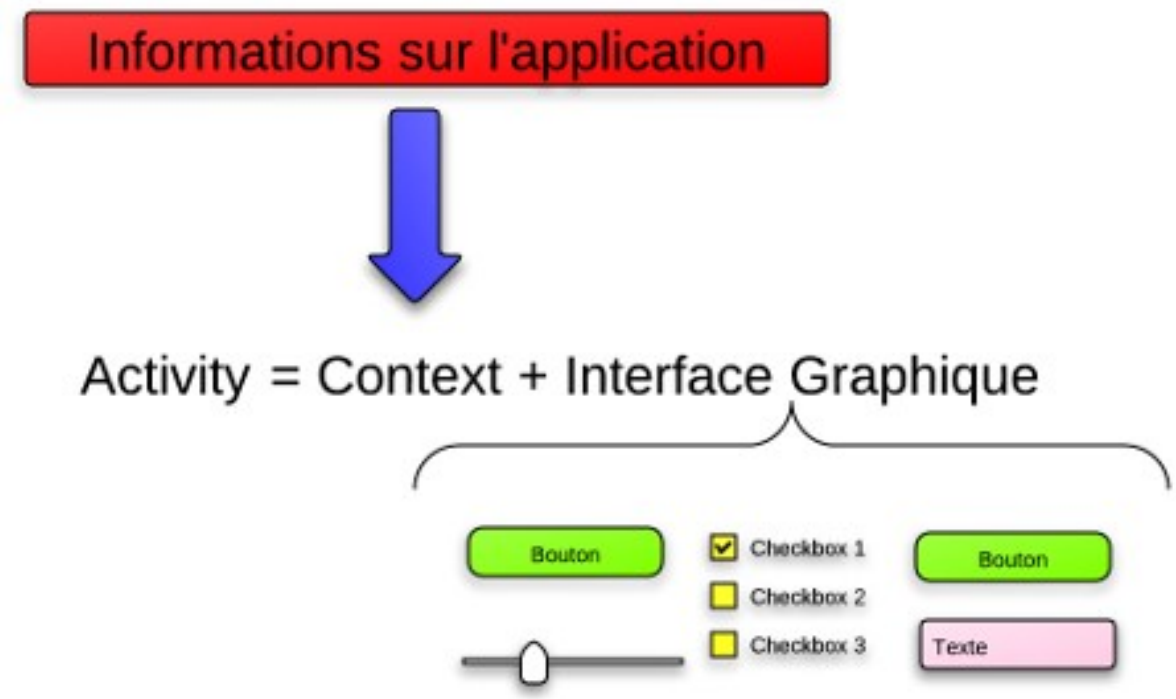
Qu'est-ce qu'une activité ?

Si vous observez un peu l'architecture de la majorité des applications Android, vous remarquerez une construction toujours à peu près similaire. Prenons par exemple l'application du Play Store. Vous avez plusieurs fenêtres à l'intérieur même de cette application : si vous effectuez une recherche, une liste de résultats s'affichera dans une première fenêtre et si vous cliquez sur un résultat, une nouvelle fenêtre s'ouvre pour vous afficher la page de présentation de l'application sélectionnée. Au final, on remarque qu'une application est un assemblage de fenêtres entre lesquelles il est possible de naviguer.



Ces différentes fenêtres sont appelées des activités

Une activité contient des informations sur l'état actuel de l'application : ces informations s'appellent le `context`. Ce `context` constitue un lien avec le système Android ainsi que les autres activités de l'application, comme le montre la figure suivante.



Une activité est constituée du contexte de l'application et d'une seule et unique interface graphique

États d'une activité

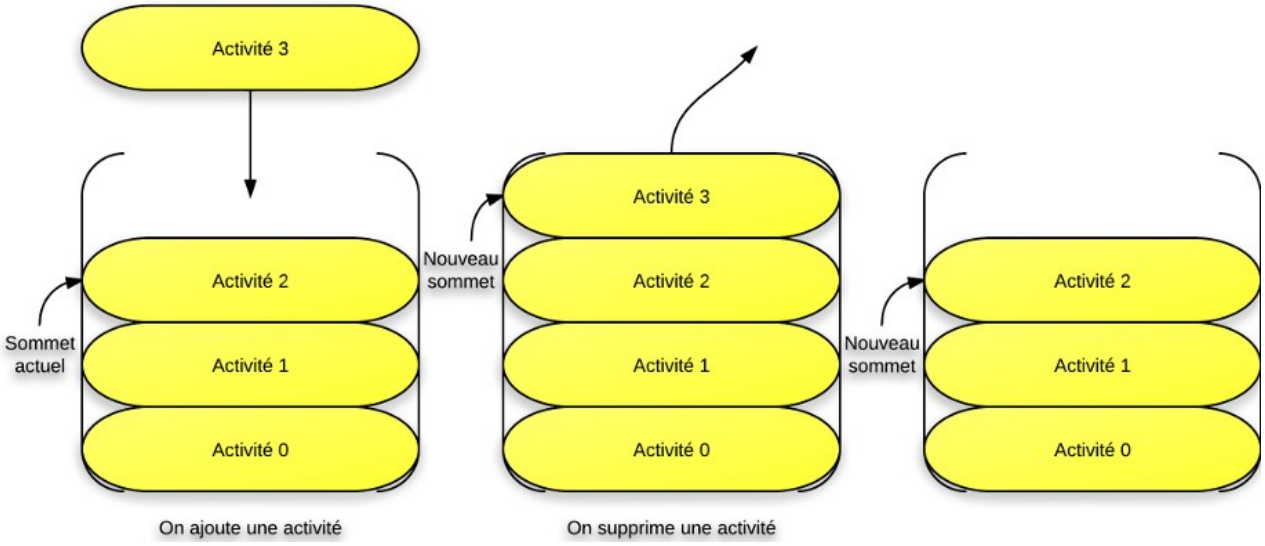
Si un utilisateur reçoit un appel, il devient plus important qu'il puisse y répondre que de faire en sorte qu'il puisse écouter la chanson que votre application diffuse. Pour pouvoir toujours répondre à ce besoin, les développeurs d'Android ont eu recours à un système particulier :

À tout moment votre application peut laisser place à une autre application, qui a une priorité plus élevée. Si votre application utilise trop de ressources système, alors elle empêchera le système de fonctionner correctement et Android l'arrêtera sans vergogne.

Votre activité existera dans plusieurs états au cours de sa vie, par exemple un état actif pendant lequel l'utilisateur l'exploite, et un état de pause quand l'utilisateur reçoit un appel.

Pour être plus précis, quand une application se lance, elle se met tout en haut de ce qu'on appelle la pile d'activités.

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. Quand on veut récupérer un élément, ce sera le sommet qui sera récupéré, sorti de la liste et l'objet en deuxième place deviendra le nouveau sommet, comme illustré à la figure suivante.



Fonctionnement de la pile d'activités

L'activité que voit l'utilisateur est celle qui se trouve au-dessus de la pile. Ainsi, lorsqu'un appel arrive, il se place au sommet de la pile et c'est lui qui s'affiche à la place de votre application, qui n'est plus qu'à la deuxième place. Votre activité ne reviendra qu'à partir du moment où toutes les activités qui se trouvent au-dessus d'elle seront arrêtées et sorties de la pile. On retrouve ainsi le principe expliqué précédemment, on ne peut avoir qu'une application visible en même temps sur le terminal, et ce qui est visible est l'interface graphique de l'activité qui se trouve au sommet de la pile.

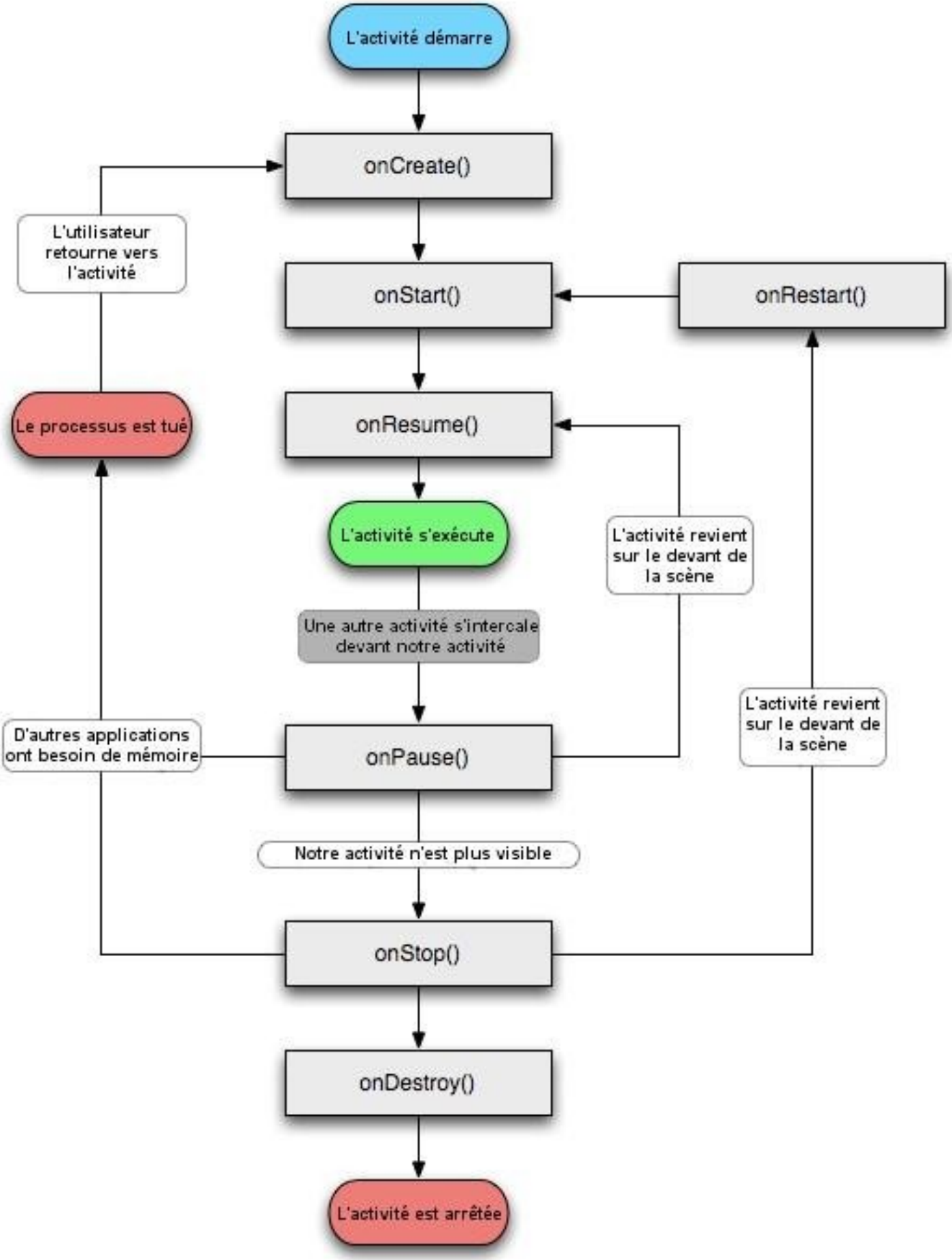
Une activité peut se trouver dans trois états qui se différencient surtout par leur visibilité :

État	Visibilité	Description
Active (« `active` » ou « `running` »)	L'activité est visible en totalité.	Elle est sur le dessus de la pile, c'est ce que l'utilisateur consulte en ce moment même et il peut l'utiliser dans son intégralité. C'est cette application qui a le *focus*, c'est-à-dire que l'utilisateur agit directement sur l'application.
Suspendue (« `paused` »)	L'activité est partiellement visible à l'écran. C'est le cas quand vous recevez un SMS et qu'une fenêtre semi-transparente se pose devant votre activité pour afficher le contenu du message et vous permettre d'y répondre par exemple.	Ce n'est pas sur cette activité qu'agit l'utilisateur. L'application n'a plus le focus, c'est l'application au-dessus qui l'a. Pour que notre application récupère le focus, l'utilisateur devra se débarrasser de l'application partiellement au-dessus pour que l'utilisateur puisse à nouveau interagir avec notre activité. Si le système a besoin de mémoire, il peut très bien tuer l'application.
Arrêtée (« `stopped` »)	L'activité est tout simplement invisible pour l'utilisateur, car une autre activité prend toute la place sur l'écran.	L'application n'a évidemment plus le focus, et puisque l'utilisateur ne peut pas la voir, il ne peut pas agir dessus. Le système retient son état pour pouvoir reprendre, mais il peut arriver que le système tue votre application pour libérer de la mémoire système.

Cycle de vie d'une activité

Une activité n'a pas de contrôle direct sur son propre état (et par conséquent vous non plus en tant que programmeur), il s'agit plutôt d'un cycle rythmé par les interactions avec le système et d'autres applications. Voici un schéma qui présente ce que l'on appelle **le cycle de vie d'une activité**, c'est-à-dire qu'il

indique les étapes que va traverser notre activité pendant sa vie, de sa naissance à sa mort. Vous verrez que chaque étape du cycle est représentée par une méthode. Nous verrons comment utiliser ces méthodes en temps voulu.



Cycle de vie d'une activité

Les activités héritent de la classe `Activity`. Or, la classe `Activity` hérite de l'interface `Context` dont le but est de représenter tous les composants d'une application. On les trouve dans le package `android.app.Activity`.

Pour rappel, un package est un répertoire qui permet d'organiser notre code source, un récipient dans lequel nous allons mettre nos classes de façon à pouvoir trier votre code et différencier des classes qui auraient le même nom.

Les vues (que nos amis anglais appellent view), sont ces fameux composants qui viendront se greffer sur notre échafaudage, il s'agit de l'unité de base de l'interface graphique. Leur rôle est de fournir du contenu visuel avec lequel il est éventuellement possible d'interagir. À l'instar de l'interface graphique en Java, il est possible de disposer les vues à l'aide de conteneurs, nous verrons comment plus tard.

Les vues héritent de la classe View. On les trouve dans le package android.view.View