Développement d'Applications Mobiles

Android Navigation :  
Launching an activity from another activity

Patrick Audriaz, Gauthier Abel

HEIA-FR 2019  
Télécommunications – T3F

Table des matières

[1. Introduction 1](#_Toc3753262)

[2. Questions 1](#_Toc3753263)

[1.1. Question 1 1](#_Toc3753264)

[1.2. Question 2 3](#_Toc3753265)

[1.3. Question 3 5](#_Toc3753266)

[1.4. Question 4 6](#_Toc3753267)

[1.5. Question 5 8](#_Toc3753268)

[3. Conclusion 11](#_Toc3753269)

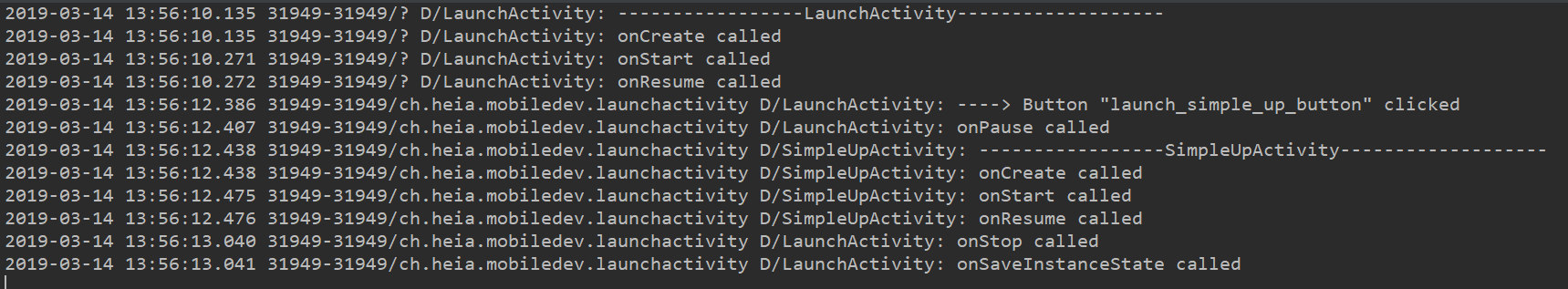
# Introduction

Concevoir des applications sur Android nécessite de maîtriser différents types de navigation. Il faut savoir lancer de nouvelles activités depuis les autres, mais aussi revenir de la nouvelle activité vers les autres parties de l'application. Pour cela, deux types de références aux différentes activités sont disponibles : Le backstack et l'activité parente. Nous explorerons ceci ainsi que les différents modes de lancement d'activités.

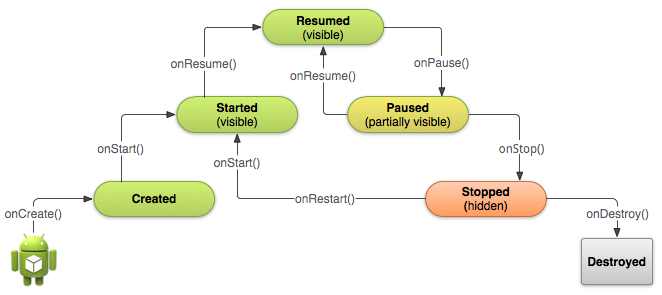
# Questions

## Question 1

*Affichez le contenu de la fenêtre « logcat » démontrant les appels aux méthodes de cycle de vie des activités « LaunchActivity » et « SimpleUpActivity » lorsque l’utilisateur démarre l’application puis clique sur le bouton « Launch Activity ». Expliquez le comportement des deux activités à l’aide du diagramme du cycle de vie des activités.*

**

Logcat lors du démarrage de l'application et appuie du bouton "Launch SimpleUp Activity"



Rappel du lifecycle des activités Android

On peut constater grâce à Logcat que les deux activités suivent bien la logique du lifecycle ci-dessus (le "save" et "restore" instance state ne sont pas modélisés) :

**LaunchActivity :**

**🡪** onCreate 🡪 onStart 🡪 onResume 🡪 onPause 🡪 onStop 🡪 onSaveInstanceState

Etat final : STOPED

**SimpleUpActivity :**

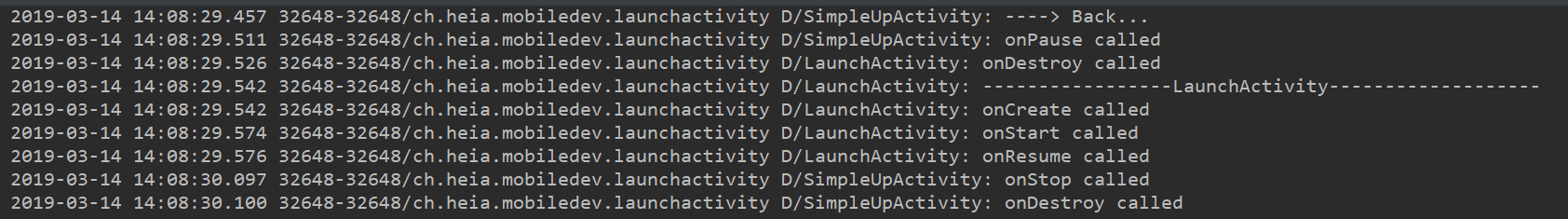
**🡪** onCreate 🡪 onStart 🡪 onResume

Etat final : RESUMED (visible)

## Question 2

*Affichez le contenu de la fenêtre « logcat » démontrant les appels aux méthodes de cycle de vie des activité « LaunchActivity » et « SimpleUpActivity » lorsque l’utilisateur retourne à l’activité hôte (« LaunchActivity » depuis l’activité « SimpleUpActivity », pour les deux cas de « Back » et « Up » navigation. Expliquez le comportement à l’aide du diagramme du cycle de vie des activités.*

**Bouton UP :**

**

Logcat lors du retour au LaunchActivity avec le bouton "UP"

**SimpleUpActivity :**

**🡪** onPause 🡪 onStop 🡪 onDestroy

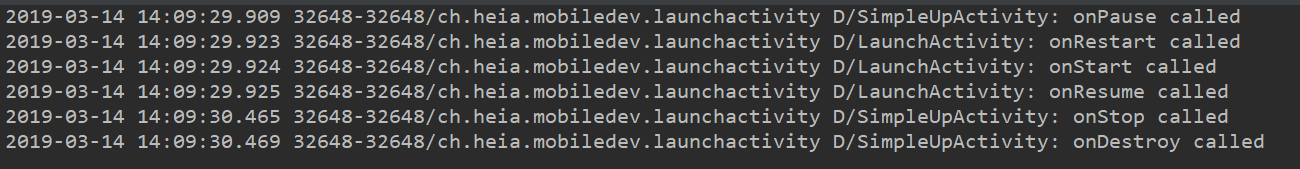
Etat final : DESTROYED

**LaunchActivity :**

**🡪** onDestroy **🡪** onCreate 🡪 onStart 🡪 onResume

Etat final : RESUMED (visible)

**Bouton BACK :**

**

Logcat lors du retour au LaunchActivity avec le bouton "BACK"

**SimpleUpActivity :**

**🡪** onPause 🡪 onStop 🡪 onDestroy

Etat final : DESTROYED

**LaunchActivity :**

**🡪** onRestart 🡪 onStart 🡪 onResume

Etat final : RESUMED (visible)

On peut constater une différence flagrante entre les deux méthodes, le UP détruit l'activité avant de la relancer depuis le début tandis que le BACK la restart.

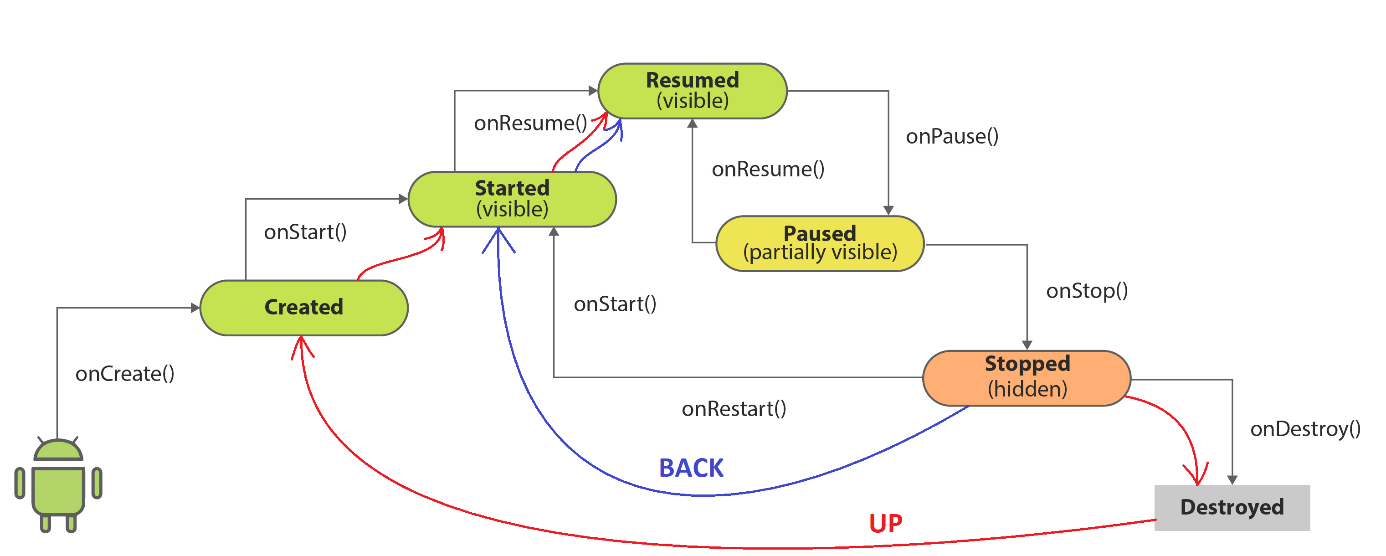
**

Schéma du cycle de vie de LaunchActivity dans le cas du UP et du BACK

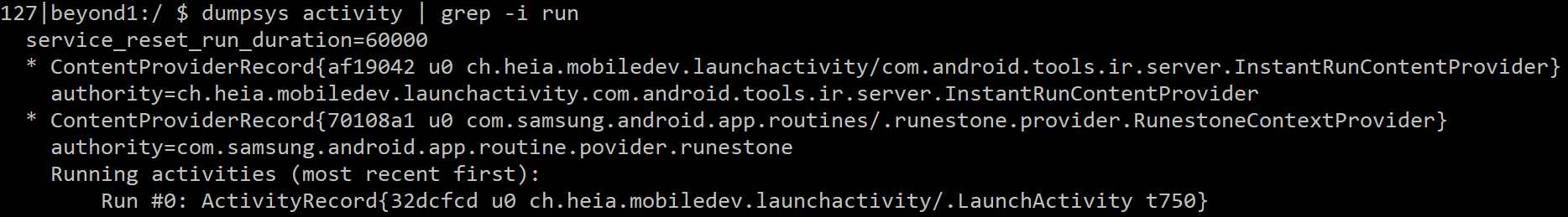
## Question 3

*Documentez l’état du « back stack » dans chacun des états de l’application.*

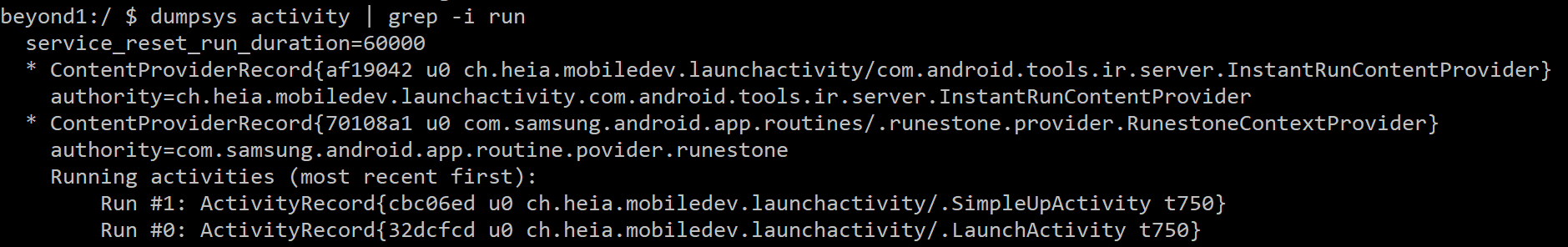
Le back stack contient toujours, de bas en haut, toutes les activités que nous lançons dans l'ordre chronologique. Cela signifie que la tâche en cours et toujours sur le haut de la pile et l'action "BACK" effectue un "pop" de cette activité et remonte celle juste en-dessous.

Les activités plus anciennes sont toujours conservées en arrière-plan tant que l'activité en cours d'exécution n'est pas détruite. L'entièreté du stack, à l'exception de l'activité parente, est vidée lorsque l'action "UP" est invoquée.

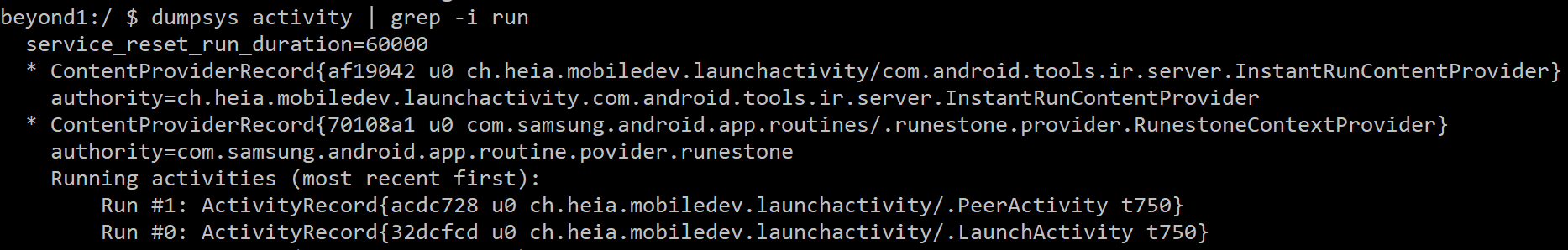
* Avec seulement l'activité de départ affichée



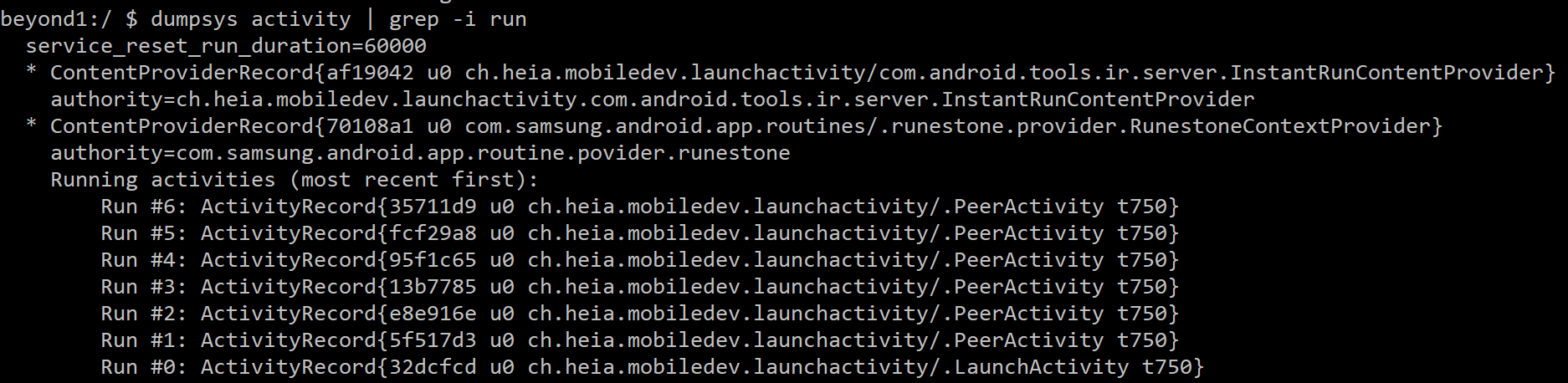
* Avec SimpleUpActivity affichée



* Avec une seule PeerActivity lancée

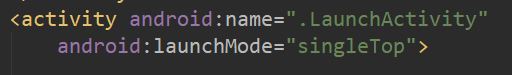


* Avec plusieurs PeerActivity lancées



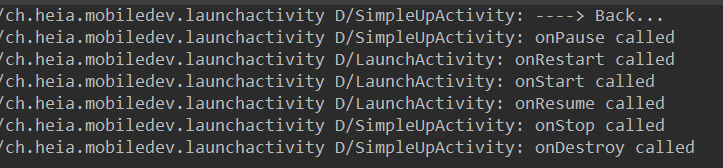
## Question 4

*Même question que la question 2, après avoir effectué le changement de l’étape 9.*

**

Modification effectuée : mode singleTop pour LaunchActivity, on le définit comme un attribut d'une activité dans le manifest

**Bouton UP :**

**

*Logcat lors du retour au LaunchActivity avec le bouton "UP"*

**SimpleUpActivity :** pas de changements

**LaunchActivity :**

**🡪** onRestart 🡪 onStart 🡪 onResume

Etat final : RESUMED (visible)

On constate qu'avec la modification du launch mode, le bouton UP se comporte comme le bouton BACK. C’est-à-dire qu'il ne destroy pas l'activité avant de la recréer, il la restart simplement.

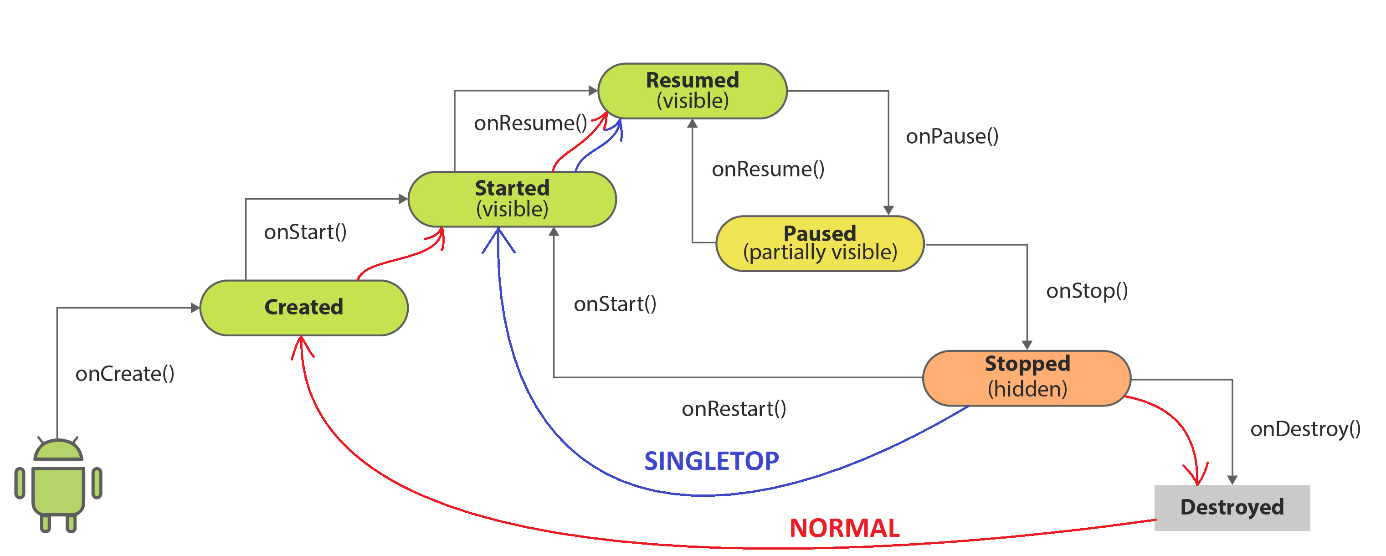
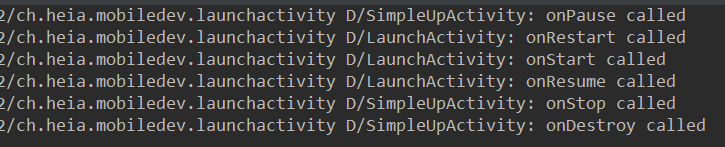


Schéma du cycle de vie de LaunchActivity dans le cas du launchmode en normal ou singleTop

**Bouton BACK :** pas de changements comparé à la question 2

**

Logcat lors du retour au LaunchActivity avec le bouton "BACK"

## Question 5

*Expliquez les différents « launch » modes des activités sous Android.*

🡪 Source :

<https://medium.com/@ankit.sinhal/understand-activity-launch-mode-with-examples-721e85b6421e>

<https://medium.com/@iammert/android-launchmode-visualized-8843fc833dbe>

<https://inthecheesefactory.com/blog/understand-android-activity-launchmode/en>

Il existe 4 types de launchMode :

**standard**

Mode par défaut, place une nouvelle instance sur le stack quand une nouvelle activité est créée, même si elle existe déjà dans celui-ci. Il ajoute "bêtement" chaque nouvelles activités à la stack et détruit la précédente.

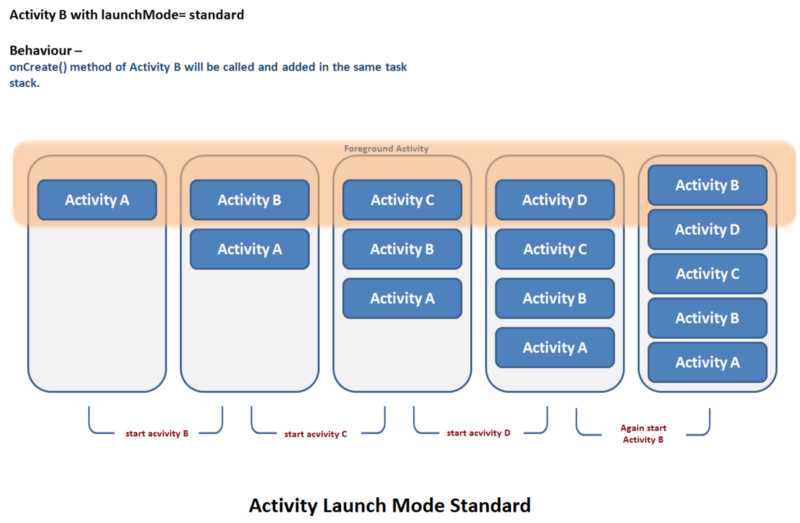


Schéma de la stack pour le launchmode standard qui montre le comportement décrit ci-dessus

**singleTop**

Comportement similaire au mode standard ou chaque activité créé une nouvelle instance mais si l'activité existe déjà dans la stack, il ne va pas créé une nouvelle instance mais plutôt chercher l'activité existante avec la méthode onNewIntent().

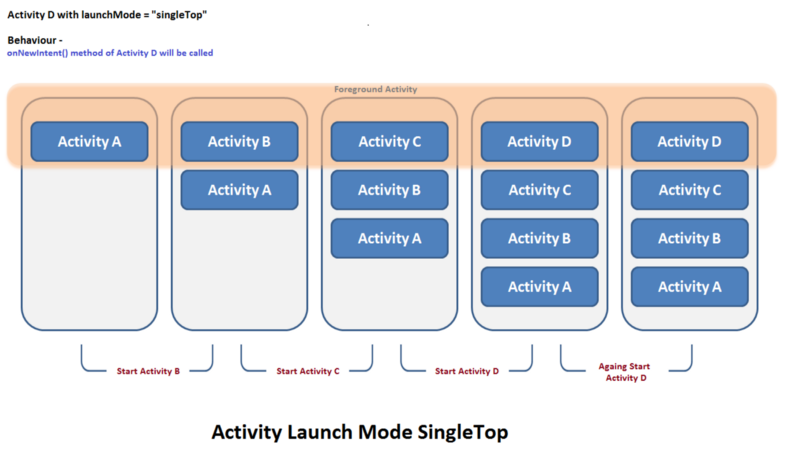
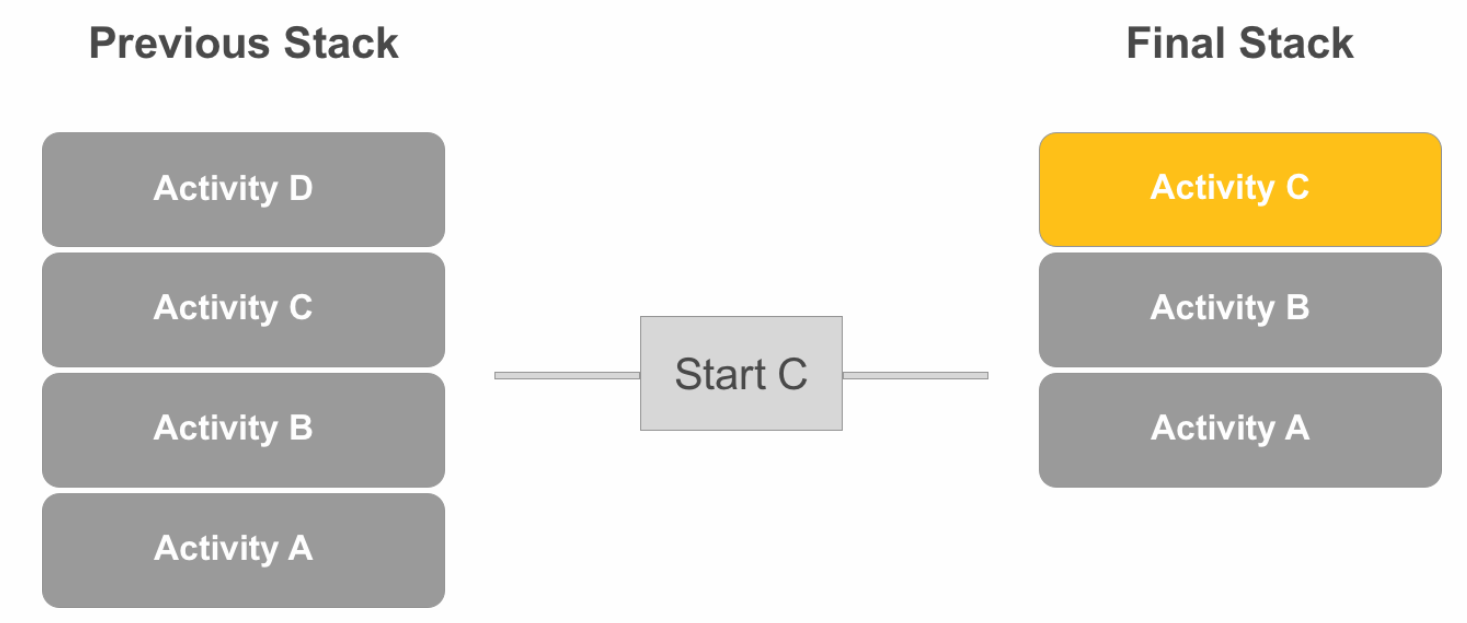


Schéma de la stack pour le launchmode singleTop qui montre le comportement décrit ci-dessus

**singleTask**

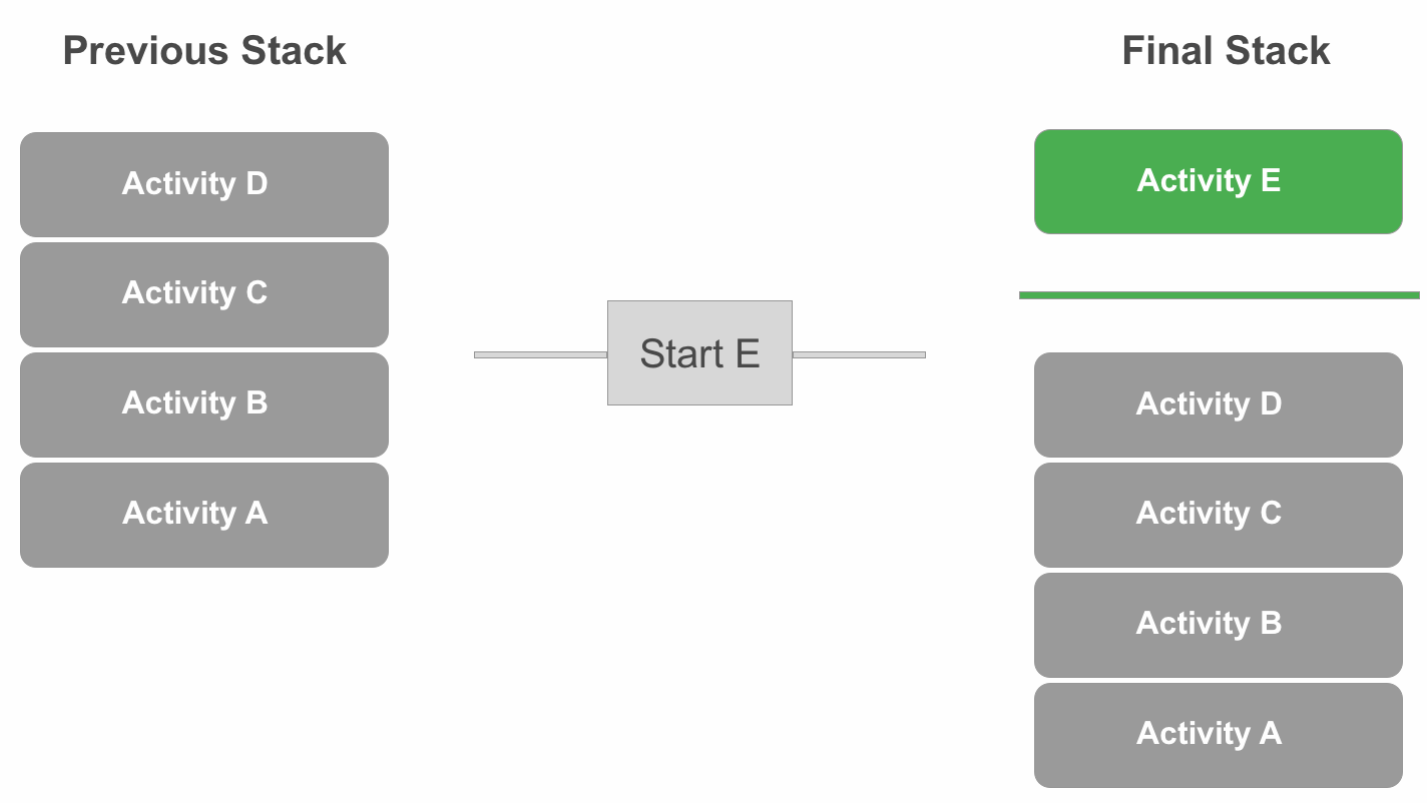
Lorsqu'une nouvelle activité est créée dans ce mode, le système va créer la nouvelle activité par-dessus le backstack si aucune instance de celle-ci n'existe au moment de l'exécution de l'Intent. Si une instance existe déjà dans le backstack, toutes les activités au-dessus seront détruites et la nouvelle instance remplacera l'ancienne en conservant le bas du stack.



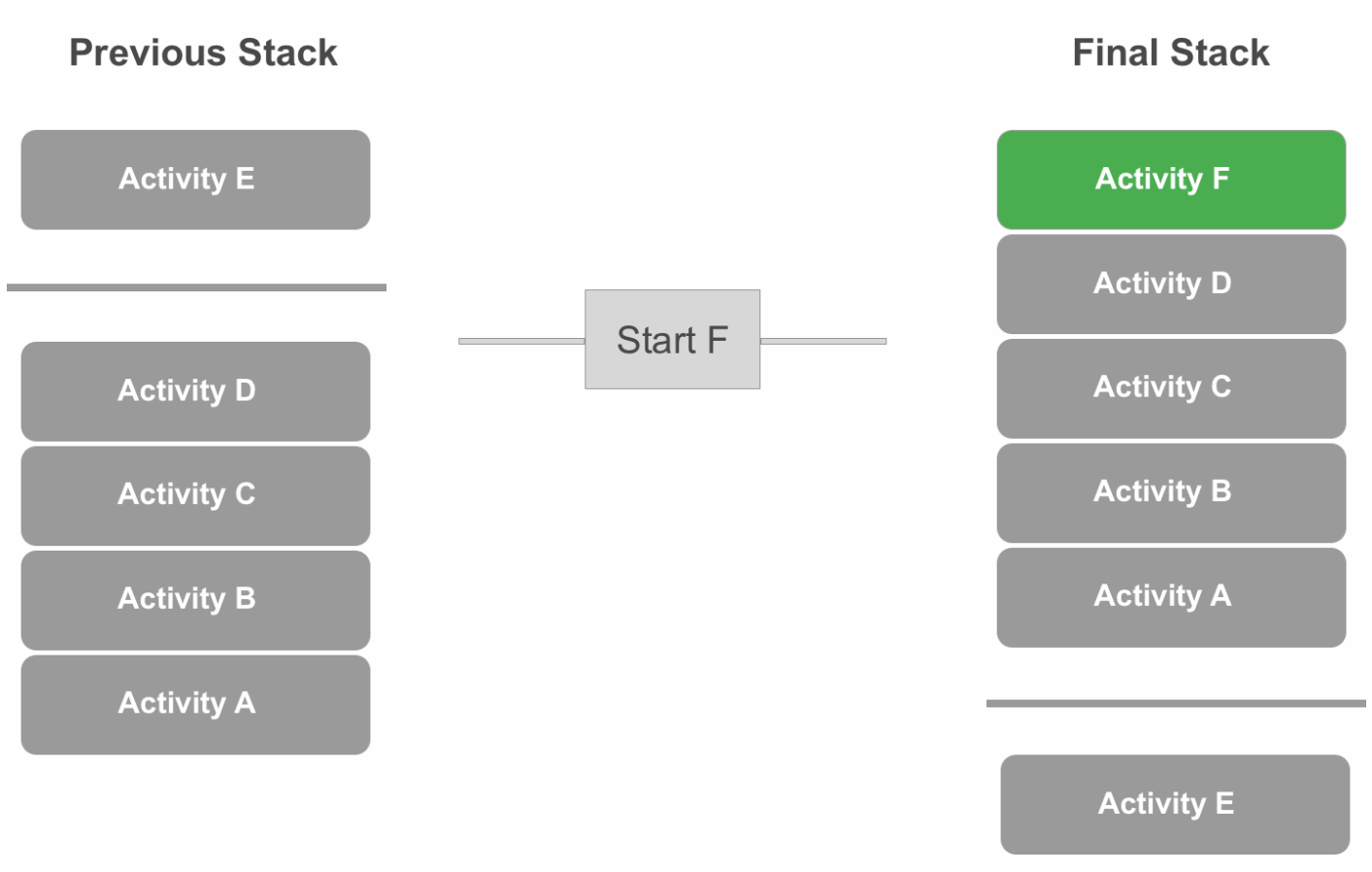
L'activité D est détruite et la nouvelle activité C est instanciée à la place de l'ancienne

**singleInstance**

Dans ce mode, lancer une nouvelle activité lance une nouvelle tâche. Celle-ci n'est pas incluse dans le backstack de l'ancienne activité. Cela veut dire que le backstack de l'ancienne activité peut recevoir de nouvelles activités indépendamment de la nouvelle.



Démarrer une nouvelle activité dans l'ancienne tâche ajoutera cette nouvelle activité au-dessus de l'ancien stack, pas au-dessus de E.



# Conclusion

Android offre de nombreuses différentes façons de naviguer. Il faut utiliser ces différentes techniques avec discernement pour créer exactement l'expérience utilisateur recherchée, car une erreur dans ce genre d'aspects d'une application peut frustrer et faire perdre du temps à l'utilisateur.

Chacune de ces méthodes ayant ses propres particularités en matière de conservation des données actives, de gestion de la mémoire et de performances, les utiliser judicieusement peut réellement faire la différence entre une application de qualité et un programme inutilisable.

Cet aspect du développement devra toujours être maîtrisé et contrôlé dans nos projets à venir.