Annexe 3– Sérialisation d’objets

Pour faire cette annexe…

* Copier / coller votre projet des mémos que nous avons fait avec des fichiers texte
* Retirer les parties relatives à ces mêmes fichiers texte

Nous avons vu à l’annexe 1 une façon de conserver des données au-delà de la fermeture de l’app, en utilisant des fichiers texte en mémoire interne

Certains problèmes existent avec cette façon :

* Il n’y avait aucun modèle de données dans notre app
* On devait recréer les flux de données à ch. ajout
* Difficulté si nos mémos sont plus complexes (pas seulement du texte, ajouter une date d'échéance par exemple )

Pour résoudre ces défis, on va mettre en œuvre une autre manière de garder ces mémos en mémoire au-delà de la fermeture de l’application :

* **Modèle de données** : classe Memo et singleton pour avoir accès à la liste de n’importe quelle activité
* **Ne pas recréer les flux de données** : utiliser les méthodes du cycle de vie d’une activité
* **Données plus complexes** : utiliser des fichiers de sérialisation d’objets plutôt que de simples fichiers texte

1. **(** *10 min* **) En KOTLIN**, codons une classe Memo qui, en plus du memo/texte comme tel, aura une date d'échéance sous forme d'un objet LocalDate ( **on doit changer le minSdk pour 26 plutôt que 24 si ce n’est pas déjà fait, dans le fichier build gradle**)
2. **(** *35 min* **) En Kotlin**, codons un singleton ayant comme variable d’instance un ArrayList de Mémos. Codons les opérations **ajouter** et **voir les mémos** localement d’abord ( sans persistance ) en travaillant sur l’ArrayList. Voir si ça fonctionne localement sans aucun fichier texte

Rappel sur le code des singletons :

1. ( *15 min* ) Modifiez l'activité pour ajouter un Memo en lui ajoutant un bouton activant un DatePickerDialog permettant de choisir un LocalDate :

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Rappel sur les LocalDate : <https://www.baeldung.com/java-8-date-time-intro>

1. ( *25 min* ) Dans l'activité d'affichage, afficher les mémos en ordre d'échéance ( du plus urgent au moins urgent ) …aurait été plus facile avec une table sqlite mais peut utiliser la méthode sortWith de Kotlin…faites des recherches sur Gemini ou sur l’aide d’Android

Vous pouvez également le faire par vous-même en trouvant le mémo le plus urgent à régler, puis celui le plus urgent parmi ceux qui restent, etc.

Les données sont-elles conservées au-delà de la fermeture de l'app ? NON

## ( 45 min ) Sérialisation d’objets :

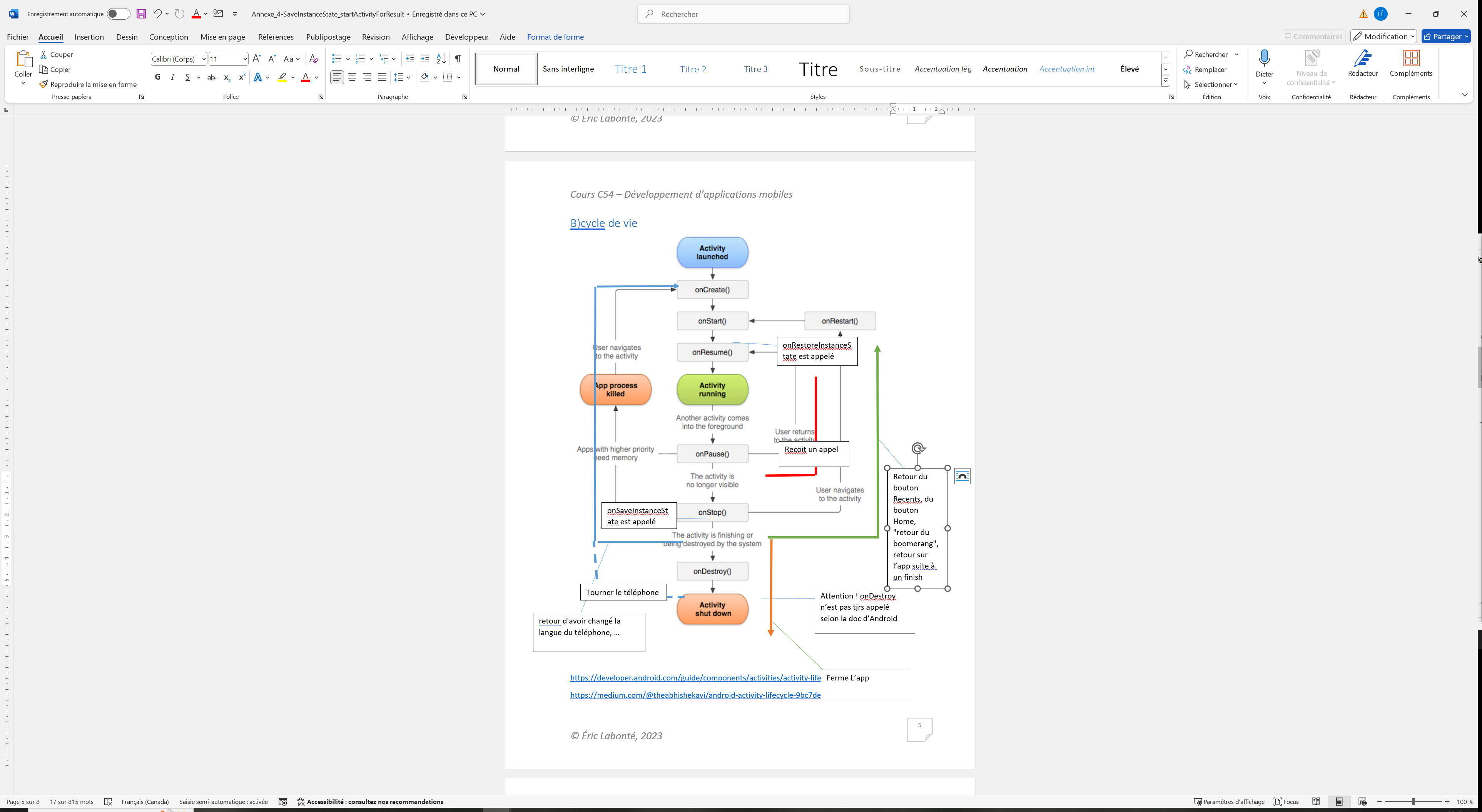
Conservation de l’état d’objets dans un fichier de sérialisation binaire dans le but de récupérer ces objets par la suite

Quels objets sont sérialisables ?

* Pour les classes provenant de l’API : vérifier s’ils implémentent l’interface Serializable et que leurs membres aussi
* Pour nos propres classes, ajouter implements Serializable / : Serializable et s’assurer que les variables d’instance le sont aussi
* C’est une « marker » interface , on a seulement à écrire Serializable pour la mettre en oeuvre

1. Lorsqu’on quitte quelle activité pourrait-on écrire l’ArrayList dans un fichier de sérialisation ?
2. Dans le singleton , codons une méthode permettant de sauvegarder l’état de l’ArrayList dans un fichier de sérialisation et appelons-la dans la méthode du cycle de vie onStop de l’activité choisie en 5
3. Dans le singleton, codons une méthode permettant de récupérer l’objet ArrayList provenant d’un fichier de sérialisation. Où / Comment devrait-t-on appeler cette méthode ?

Cliquez ou appuyez ici pour entrer du texte.



<https://developer.android.com/guide/components/activities/activity-lifecycle>

<https://medium.com/@theabhishekavi/android-activity-lifecycle-9bc7de812dff>