Rapport de contribution - EC Développement Logiciel Libre

Axel LABARRE Théo MONTAIGU
25 février 2023

Introduction

Il s'agit d'une application mobile Android open source d'une simple calculatrice "Simple-Calculator" faisant partie d'une suite complète de plusieurs autres applications "SimpleMobileTools".

Repository Github

- **URL** : GitHub Simple-Calculator
- **Nom de l'issue** : Input is lost when the screen rotates #299
- **Lien de l'issue** : https://github.com/SimpleMobileTools/Simple-Calculator/issues/299
- **Lien Pull Request** : https://github.com/SimpleMobileTools/Simple-Calculator/pull/305

Description de l'issue

Lorsque l'on utilise la calculatrice et qu'on affiche une formule ou un résultat, lors de la rotation du mobile, toutes les donnés précédentes sont perdues.

Cela provient du fait que la rotation du téléphone engendre la fin de l'activité courante (qui contient les données de calcul), et la création d'une nouvelle activité. Cette suite d'évènements est normale pour une application mobile.

Cependant il est nécessaire d'implémenter une sauvegarde et une transmission des données entre les activités si nous ne voulons pas que le calcul courant soit écraser lors de la rotation.

Parmis d'autres, les ressources suivantes nous ont permis de mieux comprendre le problème et ses solutions :

- Une réponse StackOverflow décrivant ce qu'implique une rotation de l'écran ainsi que les manières de sauvegarder et de restaurer des variables entre les instances : **ici**
- La documentation Android décrivant le cycle de vie des activités : **ici**

Environement technique

- **Logiciel IDE** : Android Studio 2022
  (Runtime version : 11.0.15
  VM : Op enJDK 64-Bit Server VM by JetBrains s.r.o.)
- **Langage** : Kotlin 221-1.8.0-release-for-android-studio-AS5591.52
- **Conventions et règles de contribution** :

  Les mainteneurs nous indiquent des conventions et des règles de contribution ici : Règles de contribution

  - Le projet utilise CamelCase comme convention de nommage.

  - Formattage de code : le projet Simple_Calculator utilise un fichier .editorconfig. Ce dernier décrit des règles syntaxiques qui sont utilisées par l'IDE pour formater le code lors d'un lancement de formatage automatique.

  - Optimisation des imports : les mainteneurs du projet nous conseil de lancer un "Optimize imports" afin de, par exemple, supprimer des importations inutiles ou d'organiser les déclarations d'importations.

Fil de contribution

Une première résolution insuffisante

Dans un premier temps nous avons cherché à conservé l'affichage du calcul courant ainsi que du calcul précédent. Après avoir trouvé quelles variables étaient concernées, nous pouvions les sauvegarder via *onSaveInstanceState()* puis les récupérer via les méthodes de *savedInstanceState*. Ensuite nous pouvions les donner au constructeur de l'objet Calculator afin qu'il initialise certaines de ses variables en fonction.

Cela nous a bien permis de conserver l'affichage à la suite d'une rotation. Cependant, nous nous sommes rendu compte que cela était insuffisant.

Une solution plus complète

En effet, nous avons constaté plusieurs suites d'évènements provoquant un résultat non souhaités :

- "1 + 3" -> Rotation -> "=" : donne 0 et non 4.
- "1 + 2" -> "=" (ce qui donne 3) -> Rotation -> "=" : donne 0 et non 5. On attend 5 car le comportement normal d'une calculatrice est de réappliquer la dernière opération (ici "+ 2") dans l'application de plusieurs "=" successifs.

Pour régler cela nous avons étendu la sauvegarde et la restauration de l'état de l'instance à toutes les attributs de Calculator.

Aussi, nous utilisions une méthode déjà existante pour récupérer la variable *secondValue* : getSecondValue(). Néanmoins, dans le cas ou notre calcul courant était "1 + 2", la méthode getSecondValue(), effectuait l'opération "1 + 2" et renvoyait 3. Ce comportement créais un bug car nous attentions 2.

Après avoir réglé ces deux problèmes, l'application se comportait tel que nous le souhaitions.

Propreté du code

Refactorisation ?

Afin de simplifier le constructeur de l'objet Calculator nous avons choisis de ne plus lui donner les variables de sauvegarde en argument. A la place de cela, nous passons par une fonction *set* pour réaffecter les variables sauvegardés : setFromSaveInstanceState().

De plus, pour utilisons un JsonObject pour contenir nos variables de sauvegarde et ainsi n'avoir qu'à stocker ce dernier lors de l'appel à la fonction on SaveInstanceState().

vers une Pull Request

Conclusion personnelle