Atelier 3 – Patron de conception Observer

On peut résumer ce patron comme ceci ; "Un sujet avertit ses dépendants ( Observateurs ) automatiquement d'un changement d'état en appelant une de leurs méthodes "

## Le sujet :

* Maintient une liste de ses observateurs
* Fournit des méthodes pour inscrire et désinscrire des observateurs
* Fournit une méthode pour avertir les observateurs d’un changement dans une de ses variables d’instance ( état )
* Ces méthodes peuvent être établies dans une interface que le sujet implémentera

## Un / des Observateur(s)

* On définit une autre interface que les observateurs devront implémenter, de cette manière tous les observateurs auront une façon commune de recevoir les nouvelles du changement de l’état de l’objet
* L’observateur répond au changement de l’état du sujet par le contenu de la méthode de l’interface

## Exemple :

Ouvrez le projet Atelier 3 présent sur LEA. Roulez le programme et regarder le code attentivement.

Quel est le sujet ? un objet de la classe Modèle, mettant en œuvre l’interface Sujet

Quel est l'observateur ? l’activité ObservateurActivity, mettant en œuvre l’interface ObservateurChangement

Lorsque l'état de l’objet de la classe Modele change, on avertir le ou les observateurs du changement en appelant la méthode de l'interface.

Il y a un seul observateur donc pas nécessaire de maintenir une liste d’observateurs dans le modèle dans ce cas-ci

Vous remarquez que le modèle utilise un Thread séparé pour gérer le délai de 5 secondes et la modification de la variable. On utilise un fil d’exécution ( thread ) séparé afin de ne pas bloqué l’affichage de l’activité pendant le délai de 5 secondes, ce qui ferait en sorte que la référence à l’observateur ( obs ) dans le modèle serait null. On verra plus en détail l’utilisation de thread ( de fils d’exécutions séparés ) plus tard.

## Appliquer à notre TP Musique

Pourquoi est-ce utile dans le tp actuel ? L’observateur ( Activité ) doit être mis au courant que nous avons reçu la réponse du serveur ( Modèle )

Quel est le sujet ? classe qui interagit avec le serveur par la librairie Volley

Quelle est la variable dont le changement d'état doit être transmis ? liste de chansons : null 🡪 pleine, boolean false 🡪 true

Quel est l'observateur ? Activité qui contient l’exoPlayer

Par quoi peut-on remplacer l’activité dans la création de l’objet RequestQueue ( requête Volley ) dans le sujet ? par le contexte que vous passez en paramètre, remplacé par this

Pour plus d'infos : <https://www.geeksforgeeks.org/observer-pattern-set-1-introduction/>

https://www.baeldung.com/kotlin/observer-pattern

NB : Des classes / interfaces existent pour représenter ce patron ( Observer, Observable ) mais certaines sont « deprecated » car elles ne sont pas assez riches pour représenter tout l’éventail de ramifications pouvant survenir. Cela ne veut pas dire que le patron n’est plus utilisé, au contraire !