M1 Informatique - Ingénierie Logicielle
Réalisation d'un framework intégrant le schéma `Modèle-Vue-Contrôleur` (MVC) et appliquant le schéma `Observateur` (*Observer*).

Le schéma Modèle-Vue-Contrôleur est largement utilisé depuis plus de 30 ans pour architecturer intelligemment les applications dotées d'une partie graphique. Il est intégré dans la grande majorité des frameworks pour le web (lui ou une de ses variantes, MVVM ou MVP - voir par exemple ici, où son utilisation est souvent automatisée et avec lesquels les programmeurs, même s'ils l'utilisent, peuvent ne pas voir ou ne pas comprendre son implantation.

Le but de ce TP est de réaliser une implantation (minimale, comme toujours dans cette UE) du framework MVC puis de l'utiliser pour réaliser une application.

Récupérez le zip du code Java qui va avec le sujet. Il est défini dans un package nommé `mvc2`. Chargez le tout dans votre environnement préféré. Le code n'est pas opérationnel ; il manque quelques méthodes que vous devrez ajouter.

— **Le schéma MVC.**

Etudiez le schéma MVC. Vous avez notamment à votre disposition l'article original qui présentait ce schéma dans le *Journal of Object-Oriented Programming* en 1988 : `KrasnerPope88.pdf`. Etudiez notamment le modèle d'interaction de la page 2. Complétez votre lecture par la description de ce schéma sur le site de votre choix.

Question 1.

Qu'apporte la solution logicielle décrite par le schéma MVC ? Quels sont ses avantages principaux ?

— **Le schema Observateur**

Etudiez le schéma Observateur (*Observer* an anglais), par exemple ici : `https://www.oodesign.com/observer-pattern`.

Question 2.

Explicitez en quoi le schéma Observateur a un rapport avec le schéma MVC. Lequel utilise lequel ?

— **Un framework pour les applications MVC**

Relisez la section du cours sur les frameworks.

Question 3.

Parmi les classes qui vous ont été fournies, lesquelles réalisent un framework permettant d'implanter une architecture de type MVC ?

Question 4. A quoi sert la classe `MV_Association` ? Pourquoi ne pas avoir un attribut de type `List<Vue>` dans la classe `Model` ?

Question 5. Un modèle (du MVC) est-il un observé ou un observateur (selon le schéma Observer) ?

Sur la classe `Model`, écrivez la méthode nommée `changed(Object how)` selon le schéma MVC. C'est conceptuellement la même que celle nommée `notify()` dans le schéma `Observateur`. Utilisez `changed(Object how)` , c'est le nom qui est utilisé dans le reste de notre framework. `how` est un objet qui explicite en quoi le modèle a changé et que l'on passe aux observateurs.

Question 6.

Dans cette implantation de MVC, le patron `Observateur` est utilisé deux fois de deux façons différentes. Explicitez lesquelles.

— **Etendre le framework : réaliser une application.**

`CompteurApp` est une application concrète qui étend le framework .

Etudier les classes `Compteur`, `CompteurController` et `CompteurView1`.

Question 7.

Dans la méthode `changeValeur(int i)`, le compteur courant dit “j’ai changé”. Parle-t-il tout seul ? Sinon, à qui s’adresse-t-il ?

Vérifiez et complétez les méthodes `incr`, *decr* et *raz* de la classe *Compteur*.

— Votre application doit maintenant être opérationnelle, essayez la.

— **Etendre l’application : ajouter une vue**

On souhaite afficher une seconde vue pour un Compteur, sous la forme d’une barre de progression de type `JProgressBar`. Ainsi le même compteur est visualisé sous 2 formes différentes (voir copie d’écran).

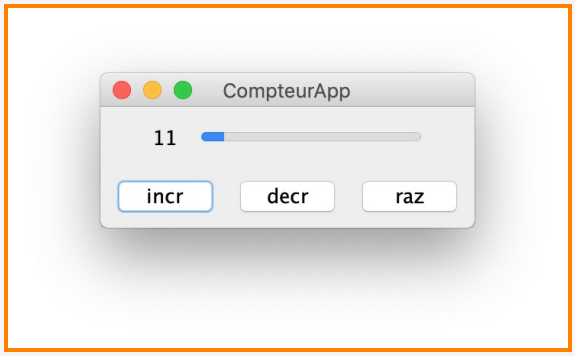

Figure (1) *– Un Compteur avec deux vues et trois boutons*

Question 8.

Etendez l’application pour réaliser cette fonctionnalité. Vous n’avez pas le droit de modifier le code existant (sauf la classe `CompteurApp`).

— **Etendre l’application par d’autres schémas.**

Question 9. Faite que la classe Compteur ne puisse avoir que deux instances, selon le schéma Singleton.