TD2 : Modèle de Conception et architecture MVC

En génie de Logiciel, un modèle de conception (*design pattern* en anglais) est un concept destiné à résoudre les problèmes récurrents suivant le paradigme objet. En français on utilise aussi les termes de **motif de conception** ou de patron de conception.

Les modèles de conception décrivent des solutions standard pour répondre à des problèmes d'architecture et de conception des logiciels. On peut considérer un patron de conception comme une formalisation de bonnes pratiques, ce qui signifie qu'on privilégie les solutions éprouvées.

Il ne s'agit pas de fragments de code, mais d'une manière standardisée de résoudre un problème qui s'est déjà posé par le passé. On peut donc considérer les patrons de conception comme un outil de capitalisation de l'expérience appliqué à la conception logicielle.

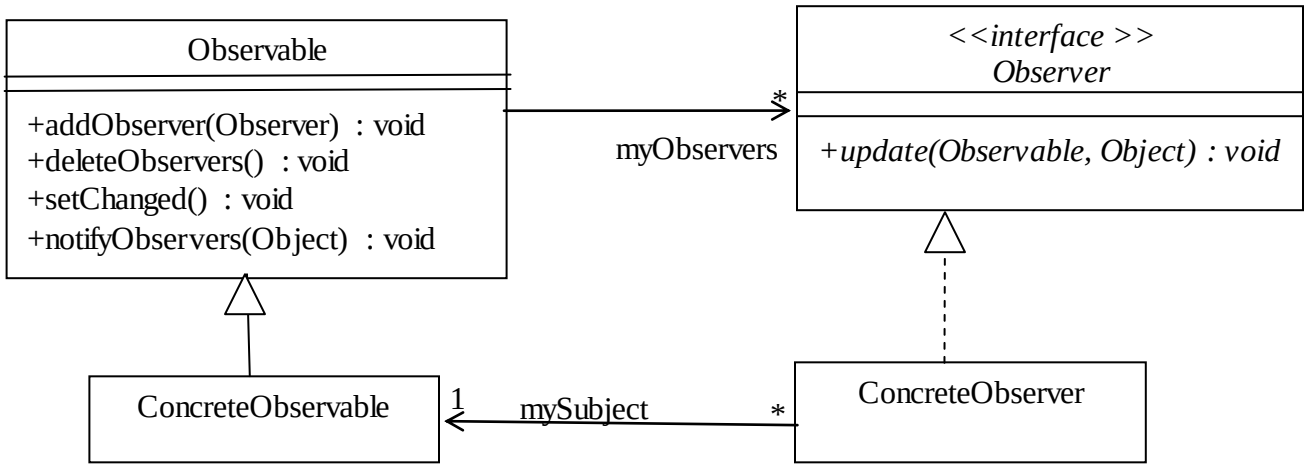
Inconvénients des Modèles de Conception
L'utilisation des Modèles de Conception n'est pas des plus simples et ne convient assurément pas à tout le monde. Les modèles de conceptions requièrent une conception détaillée. Il introduit en outre un niveau approfondi de complexité qui nécessite une attention de tous les instants aux détails.

La charge de travail supplémentaire induite ne doit pas être sous-estimée. Ne l'oubliez pas.

Il en résulte que l'usage des modèles de conception peut être trop complexe pour les petites applications, et même pour bon nombre d'applications moyennes

Exercice 1 En Java, écrivez le code de l'interface `Observer` et le squelette des classes `Observable`, `ConcreteObservable` et `ConcreteObserver` (sans le corps des méthodes).

Modèle de conception « observateur/observable » : Les observables envoient une notification à leurs observateurs en cas de changement de leur état (c'est-à-dire qu'ils exécutent les méthodes setChanged, et notifyObservers). En cas de notification, la méthode update des « ConcreteObserver » concernés est appelé (donc cette méthode est exécutée). Durant l'exécution de cette méthode, les ConcreteObservable peuvent obtenir des informations complémentaires par l'appel d'une ou de plusieurs méthodes de la classe ConcreteObservable.
La classe Observable ainsi que l'interface Observer font parties de l'API Java.



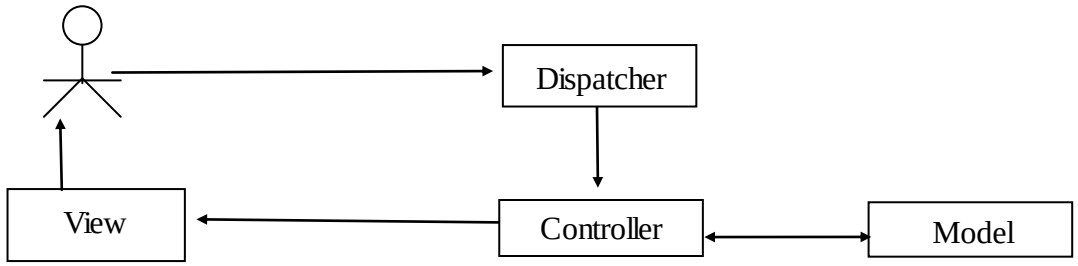
MVC

Le **Modèle-Vue-Contrôleur**, en abrégé MVC, de l'anglais *Model-View-Controller* est une architecture logicielle. Cette architecture a été mise au point en 1979 par Trygve Reenskaug, qui travaillait alors sur Smalltalk dans les laboratoires de recherche Xerox PARC.

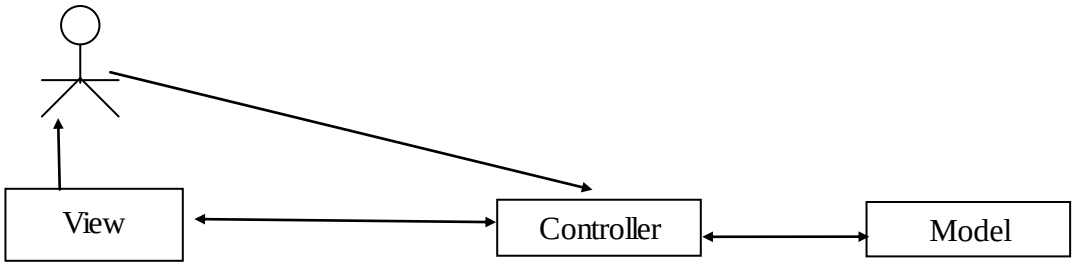
MVC l'une des premières approches pour décrire et mettre en œuvre des architectures basés sur la notion de responsabilité. L'idée était de bien séparer les données, la présentation et les traitements des événements utilisateur. Il en résulte les trois parties : le modèle, la vue et le contrôleur.

Bien que développé à l'origine pour les application "standard", le MVC a été largement adopté comme une architecture dans le monde des applications WEB. Plusieurs "système" Web commerciales et non commerciales ont mises en œuvre ce modèle. L'interprétation de ce modèle, principalement dans la façon dont les responsabilités varie d'un système à un autre.

Par exemple, l'architecture de cakePHP est la suivante :



qui est une adaptation de l'architecture MVC dite passive :



Pour plus d'information

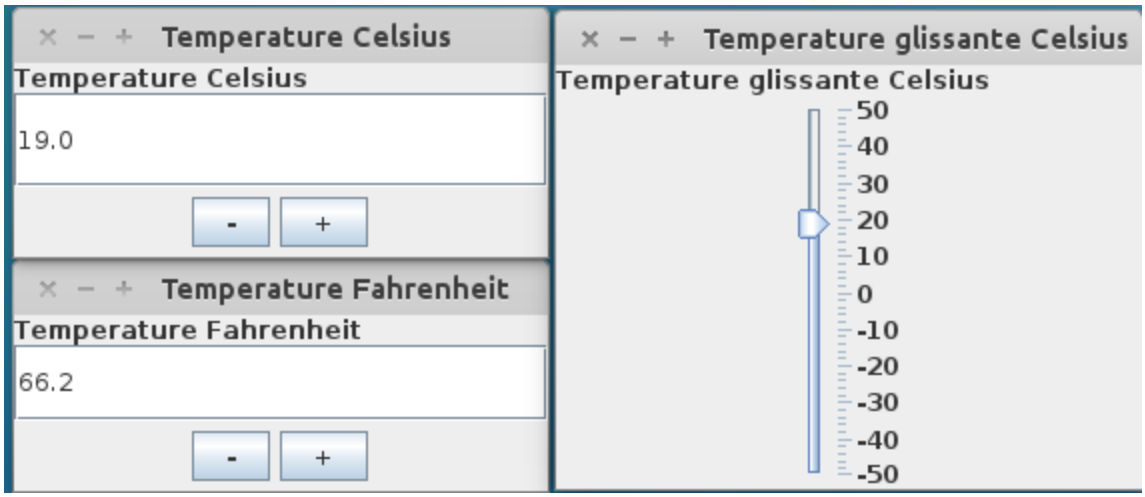
http://www.albertzuurbier.com/index.php/programming/84-mvc-vs-mvp-vs-mvvm
http://forums.codeguru.com/showthread.php?517017-A-MVP-pattern-doubt
http://forums.codeguru.com/showthread.php?517594-Is-MVC-and-MVP-supervising-controller-the-same

Exercice 2 Adapté d'un TP de Master1 – Université de Lille 1 - (resp. Gery Casiez)

L'objectif est de créer une interface permettant le contrôle d'une température en degrés Celsius ou Fahrenheit. L'interface se compose de trois vues représentant la même température sous des formes différentes. La modification d'une des vues doit mettre automatiquement à jour les autres vues.

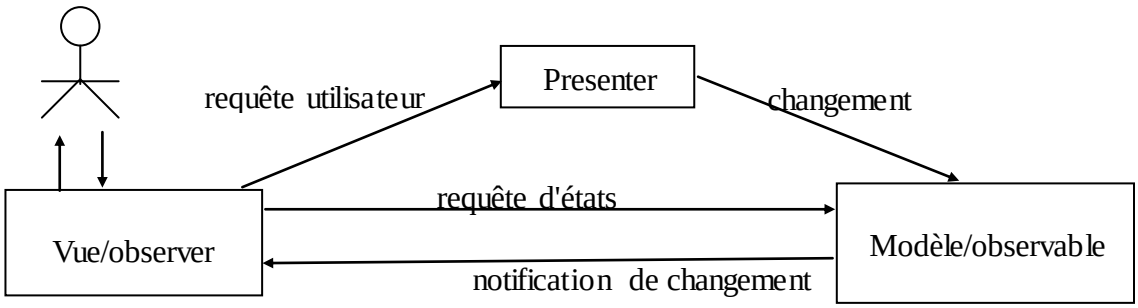
Calcul de la température en Fahrenheit à partir de la température en Celsius :

t-Fahrenheit = (t-Celsius*9.0/5.0) + 32.0



L'interface de contrôle de la gestion de la température

1. Ecrivez le diagramme de classes correspondant à l'application vous devez utilisez le patron de conception Observable/Observateur dans le cadre d'une architecture logicielle variante du MVC, nommé MVP. Plus précisément, vous devez suivre le schéma suivant.



2. Elaborez le diagramme de communication associé au scénario suivant :
   Après que l'application soit lancée, l'utilisateur augmente la température via le bouton « plus » de fenêtre « Temperature Celsius », les affichages sont mis à jour.

Lors d'un autre TP ; vous devrez implémenter en Java l'application que vous avez conçus (diagramme de classes).

Optionnel - Ecrivez le diagramme de classes en suivant le schéma correspondant à la version originale de l'architecture MVC.