**Lab 4 : Patrons, patrons, patrons...**

**But du laboratoire**

Ce laboratoire vous permettra d'acquérir les connaissances suivantes :

* Intégration de plusieurs patrons de conception pour réaliser une application simple d'affichage d'images
* Séparation des classes en groupes : modèle, vue, contrôleur
* Sérialisation d'objet en Java

Les patrons suivants seront utilisés (les **patrons en gras sont obligatoires**) :

* **Command** [GoF95]
* Immutable [Grand98] (optionnel)
* Factory [GoF95] (optionnel)
* Mediator [GoF95] (optionnel)
* Memento[GoF95] ou Snapshot [Grand98] (optionnel)
* **Observer** [GoF95]
* Null Object [Woolf97] (optionnel)
* **Singleton** [GoF95]
* Strategy[GoF95] (optionnel)

Pour ce laboratoire, vous devrez faire preuve de plus de créativité que pour les autres. Ne vous limitez donc pas à cette liste de patrons. Utilisez un patron plus d'une fois si c'est nécessaire. Le but de ce laboratoire est de vérifier si vous avez bien compris les patrons et leur utilisation dans une application. Vous devrez donc voir à bien expliquer et justifier l'utilisation de chaque patron.

Il est fortement conseillé de lire aussitôt que possible la description de chacun des patrons. Votre compréhension individuelle aidera grandement l'équipe dans ses choix de conception qui sont la base de ce laboratoire.

**Description du laboratoire :**

Le laboratoire consiste à concevoir et à réaliser une application qui permet d'afficher une image avec *plusieurs perspectives*. L'utilisateur pourra appliquer sur la perspective d'une image les opérations de :

* changement du facteur d'échelle (zoom)
* changement de la région affichée (translation).

Les opérations doivent être contrôlées par la souris et **l'utilisateur aura toujours la possibilité de défaire (undo) l'opération.**

L'utilisateur pourra aussi, grâce à un article "Sauvegarder" d'un menu "Fichier", sauvegarder l'état (l'image et les perspectives actuelles) sur disque dans un fichier. Consultez ce [lien](http://java.sun.com/j2se/1.5.0/docs/guide/serialization/index.html) pour obtenir plus d'informations sur la sérialisation d'objet.

Vous allez travailler en équipe (minimum trois personnes) et ces équipes seront déterminées par le professeur. Vous devrez décider entre vous du rôle de chaque coéquipier.

**Contraintes de développement :**

L'implantation des différents patrons doit être en concordance avec les caractéristiques de ces patrons. De légères modifications devront être justifiées. Pour chaque patron, vous devez être certain de son application *dans le contexte de votre application.* Donc, il faut être capable de montrer son application dans un tableau ainsi :

|  |  |
| --- | --- |
| **Nom de l'élément du patron (classe, méthode, etc.)** | **Nom dans le laboratoire** |
| Subject | Observable |
| attach(Observer) | addObserver(Observer) |
| notify() | notifyObservers() |
| ConcreteSubject | Perspective |
| ... | ... |

Il faut séparer des classes selon le patron **Modèle/Vue/Contrôleur** (MVC).

* Les classes du modèle doivent fournir les opérations permettant d'avoir accès aux informations. Vous devez avoir au moins deux classes pour le modèle : l'image et la perspective. Chaque classe du modèle doit être un *sujet distinct*dans le patron **Observer**.
* Pour modifier les différents paramètres du modèle, il faut définir des contrôles qui agissent sur celui-ci et qui peuvent être interchangeables en tout temps.
* Utilisez le patron **Command** pour permettre de défaire (undo) les opérations.

Un gestionnaire de commande doit être utilisé afin de gérer la mémorisation des différentes commandes effectuées et il doit être une instance unique (patron **Singleton**). Ce gestionnaire aura la responsabilité aussi de défaire les commandes.

Vous aurez besoin de plusieurs vues pour bien prouver l'utilisation de MVC et des patrons :

* Une première vue de l'image réduite (vignette, "thumbnail");
* Une seconde et une troisième vues changeront au gré des commandes effectuées sur chacune;
* L'opération de défaire une commande devrait s'effectuer **sur la bonne vue.**

**Points bonis**

Plusieurs options s'offrent à ceux qui voudraient aller un peu plus loin. Il est fortement recommandé de s'attaquer aux demandes minimales du laboratoire avant d'en faire plus. Vous pouvez gagner jusqu'à 20 % de plus pour les fonctionnalités additionnelles, à condition d'avoir satisfait des demandes minimales du laboratoire.

L'application des patrons permet l'intégration de nouvelles fonctions sans modifier toute l'architecture du programme. Voici quelques suggestions :

**Fonction Défaire, variantes**

Pour défaire une opération, vous pouvez mémoriser l'état du modèle avant d'effectuer l'opération avec le patron **Memento** ou **Snapshot**.

**Fonction Copier-Coller**

Ceci permet de copier certains paramètres d'un modèle d'une image et de les appliquer (coller) à une autre image. Il ne s'agit pas de copier l'image, mais plutôt les paramètres d'affichage comme :

* le facteur d'échelle seulement
* la translation seulement
* les deux
* rien

Le patron **Strategy** permet de déterminer la stratégie à adopter lorsque "copier" est effectué.

L'intégration de cette opération s'effectue en utilisant le patron **Mediator** qui permet de faire la gestion entre le moment où la copie est faite et le moment du collage et aussi de gérer la possibilité de faire plusieurs fois la fonction Copier et de ne coller que les paramètres de la dernière copie.

**Démonstrations**

**A) Revue de conception (4/10)**

Les **diagrammes UML**doivent être remis après le déroulement de la moitié de l'activité. **Cette étape n'est pas facultative!**Vérifiez cette date avec votre chargé de laboratoire. **Si vous n'avez pas eu une revue de votre conception, la note de ce laboratoire sera 0.**

Les diagrammes requis sont :

* Diagramme(s) de classes de toutes les classes de votre application.
* Diagrammes de séquence démontrant visuellement le flot entre le(s) contrôleur(s), les classes du modèle et la (les) vue(s) :
  1. Effectuer une certaine commande
  2. Défaire une certaine commande
* Tableaux d'application du patron ([voir ci-haut](https://cours.etsmtl.ca/log121/private/pboucher/laboratoires/labo4/labo4.htm#TableauApplicationPatron)) pour chaque patron appliqué

Afin d'avoir suffisamment de temps pour coder, vous pourrez conserver votre design original pour l'implémentation en discutant soigneusement dans le rapport des lacunes de votre implémentation, des améliorations que vous devriez lui apporter et des raisons pour lesquelles ces améliorations seraient nécessaires. Expliquez le design amélioré

Vous pouvez présenter, expliquer et discuter un design amélioré même s'il n'est pas implémenté. Ce design étendu peut également couvrir les points boni. Dans le rapport, regroupez ces améliorations non implémentées dans une section bien identifiée.

Ces diagrammes pourront **légèrement** être modifiés pour la remise du rapport final.

**B) évaluation finale (6/10)**

La dernière séance de laboratoire sera une évaluation fonctionnelle de votre laboratoire, tout comme les évaluations des autres laboratoires.

**Documents, fichiers et indices utiles**

* Vous n'êtes pas obligés de réutiliser le code des autres laboratoires. Par contre, si vous le faites, il faut, bien sûr, citer la référence.
* La partie conception de ce laboratoire est primordiale. Mettez assez de ressources (toute l'équipe) là-dessus le plus tôt possible. Une bonne conception vous épargnera du temps plus tard.
* N'hésitez pas à consulter les sites Web sur les patrons. Votre compréhension individuelle des patrons utilisés définira en grande partie la qualité de vos discussions, de vos choix et, ultimement, de votre laboratoire.
* Révision du [Modèle-vue-contrôleur](https://cours.etsmtl.ca/log121/private/pboucher/laboratoires/mvc/mvc.htm)

**Foire aux questions (FAQ)**

1. La translation s'effectue-t-elle à l'aide de la souris ou à l'aide d'un menu ?

*La souris principalement, mais un menu est aussi pratique. Si vous avez bien appliqué le patron Command, ça sera facile de faire les deux. Pour le menu, les arguments de la translation doivent passer par un dialogue. Pour la souris, il faut les obtenir à partir du geste. Mais, cela devrait être la même opération qui modifie la perspective quelle que soit la source des arguments.*

1. À quoi sert la sauvegarde ?

*On sauvegarde afin de pouvoir récupérer plus tard la perspective sur une image. Si on avait une image énorme (image satellite, image médicale) avec une très haute résolution, ça serait utile de pouvoir sauvegarder des perspectives. On sauvegarde aussi pour apprendre à utiliser la sérialisation d'objet pour la persistance, quelque chose qui est souvent utile dans les applications.*

1. J'ai des problèmes avec les mises à jour dans l'interface. Par exemple, lorsque j'effectue une translation, l'image vignette (thumbnail) scintille. Comment éviter que toutes les parties de mes vues soient mises à jour pour toutes les actions?

*Il peut y avoir plusieurs raisons pour ce problème. Voici une explication possible, si vous avez une seule classe "Modèle" contenant des références aux objets concrets, par exemple, les images, les perspectives, etc.*

*Idéalement, vous devriez faire en sorte que chaque objet concret (image, perspective, etc.) soit un sujet****indépendant****dans le patron Observateur. Autrement dit, l'objet de la classe "Modèle" ne doit pas être le seul sujet (ou un sujet du tout!) dans le patron Observateur. Bien qu'il soit plus simple d'implémenter le patron Observateur de cette façon, cette conception n'est pas idéale, car chaque modification du modèle nécessite une mise à jour de toutes les vues. Par exemple, une opération de translation n'affecte qu'une partie du modèle (une perspective), elle ne devrait jamais affecter l'objet représentant l'image et donc il est inutile d'afficher de nouveau la vignette à la suite d'une translation.*

*Si vous n'avez qu'une classe Modèle qui est le seul sujet du patron Observateur, alors quelle que soit la modification du modèle, tous les observateurs vont essayer de se mettre à jour. Donc, vous devriez plutôt décomposer le modèle, ainsi que la (les) vue(s), en plusieurs classes, chacune étant un sujet ou un observateur respectivement. Par exemple, l'élément dans la vue qui affiche la vignette de l'image devrait s'abonner (addObserver()) seulement au sujet représentant l'image dans le modèle et non pas au sujet représentant une perspective.*

*Une mauvaise solution à ce problème est de faire une méthode spécialisée dans le patron Observateur de ce genre : "notifyTranslation()" avec l'intention d'avoir une notification pour chaque opération affectant le modèle. Pourquoi est-ce mauvais ? Parce que votre modèle doit ainsi connaître ces méthodes "notify" spécialisées de la vue. Donc, il dépend de la vue! Cela ne respecte plus le principe MVC car le modèle est censé ignorer les détails de ses contrôleurs et de ses vues.*

*Finalement, le fait de décomposer le modèle en plusieurs sujets ayant leurs propres abonnés indépendants dans la vue est la solution fonctionnellement équivalente à un notifyX() spécialisé. Pourtant, il est plus propre en termes du patron Observateur puisque chaque observateur s'abonne seulement à la partie du modèle qui lui intéresse.*

1. Pourquoi la perspective est-elle dans le modèle et non dans la vue ?

*La perspective fait partie des propriétés du modèle à sauvegarder et à récupérer en passant d'une vue à une autre.*

1. Les images peuvent-elles être modifiées lors de l'exécution ?

*En théorie, oui, mais il n'est pas spécifié précisément dans l'énoncé (voir la question suivante). L'application d'un filtre à une image (à la Photoshop) serait un bon exemple d'une modification de l'image.*

1. A-t-on un chemin d'accès sur l'image dans notre modèle ?

*Si tu veux dire plutôt, "Doit-on pouvoir spécifier le fichier à charger pour l'application" alors la réponse est "oui" dans la solution idéale.****Tu peux considérer ça comme une exigence.***

1. Pourquoi la totalité des exigences n'est-elle pas spécifiée par l'énoncé ?

*Dans la réalité, qui va spécifier toutes les exigences ? Il est presque impossible de les connaître toutes, a priori. Les exigences sont parfois très floues au début d'un projet. L'ingénieur en logiciel doit accepter ce fait et faire de son mieux pour améliorer la situation. Vous avez plus de responsabilités dans ce laboratoire à découvrir les détails que dans les laboratoires précédents.*

1. Pourquoi les liens sur les technologies à utiliser dans le laboratoire ne sont-ils pas sur la page de l'énoncé ?

*Il y a deux raisons: (1) Même si on pouvait donner toutes les réponses pour les technologies de pointe, les technologies évoluent tellement vite que ça risque de ne plus être valable dans une année ou deux. (2) On aurait pu vous donner des liens. On aurait pu vous donner le code source pour la solution du laboratoire aussi ! Le but est de vous apprendre plutôt à utiliser les ressources (Java Tutorial, API, livres sur les patrons, Google) pour trouver les réponses à vos questions.*

**Références :**

* [Grand98] [Grand, Mark, *Patterns in Java,* 2eme édition, J. Wiley and Sons , c2002](http://library.books24x7.com/toc.asp?bookid=5397)
* [GoF95] Gamma, E., Helm, R., Johnson, R & Vlissides, J, *Design Patterns* , Addison Wesley, c1995