POLYTECH ANGERS

TP 5A - Génie Logiciel
Patrons de conception: Command

Nicolas Delanoue

(1 séance de 1h20)

Objectif : Se familiariser avec le patron de conception (Design Pattern) *Command*. Améliorer un code existant (notion de *refactorying*).

1 Cahier des charges et prototype

On considère une structure de données simple, réduite à un entier `x` et un nom `name`. On souhaite disposer d'une interface graphique permettant d'effectuer des opérations élémentaires : addition et soustraction sur cet entier `x` :

```
1     public void add(int _operande)
2     {x = x + _operande;};
3
4     public void sub(int _operande)
5     {x = x - _operande;};
```

où `operande` est un paramètre de l'opération choisie par l'utilisateur via l'interface graphique.

Il doit également être possible de conserver l'historique des opérations et de les défaire/refaire (`undo/redo`).

L'archive `TPCommand_code_sujet.zip` donne un exemple de programme respectant ce cahier des charges (voir la figure 1 pour la modélisation UML).

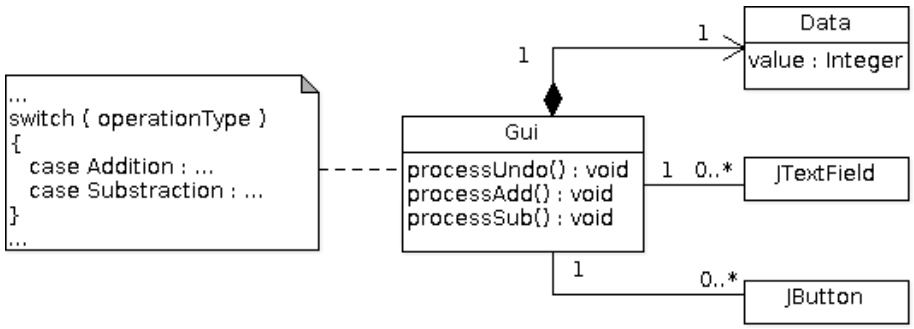

FIGURE 1 – Diagramme de classe UML du projet `TPCommand_eclipse_sujet`.

De manière à gérer l'historique, on trouvera les propriétés suivantes dans la classe `Gui`.

```
1 private List< Integer > m_parameters ;
```

```
2 private List< Integer > m_operations ;
3 private Integer m_historyIndex;
```

La liste `m_parameters` stocke l'historique des paramètres α, et `m_operations` l'historique des opérations effectuées avec le codage suivant : addition codée par 0 et soustraction codée par 1.

2 Travail à réaliser

Un défaut de ce programme est que la classe `Gui` gère le détail de l'invocation des opérations élémentaires, de la gestion de l'historique ainsi que celui des `undo/redo`. Cette architecture n'est pas satisfaisante. En effet, la taille et la complexité de cette classe risquent d'augmenter avec le nombre des opérations élémentaires (e.g. ajout de la multiplication, division, racine carrée,...).

On souhaite améliorer la conception du programme par rapport à ce point. Pour cela, on propose d'intégrer le patron de conception `Command` pour gérer l'exécution des opérations d'ajout et de soustraction. On conservera une trace des opérations jouées au moyen d'une liste (historique) : ceci permettra de pouvoir `annuler` (`undo`) les opérations et de pouvoir les `refaire` (`redo`). La notion de `undo` est associé à l'inverse des commandes exécutées (e.g. la soustraction est l'inverse de l'addition). La figure 2 donne la modélisation générique de ce patron de conception.

On souhaite enfin généraliser la possibilité du `undo/redo` au changement du nom : l'interface graphique initiale permet de changer le nom de l'objet sans pouvoir revenir en arrière. On souhaite que cela devienne possible et que cette opération soit gérée de manière analogique (généricité) aux deux opérations arithmétiques addition et soustraction.

Le travail à réaliser consiste à proposer une nouvelle modélisation du programme et à adapter le programme initial en conséquence, en intégrant ce modèle de conception.

Le pattern Commande : le diagramme de classes

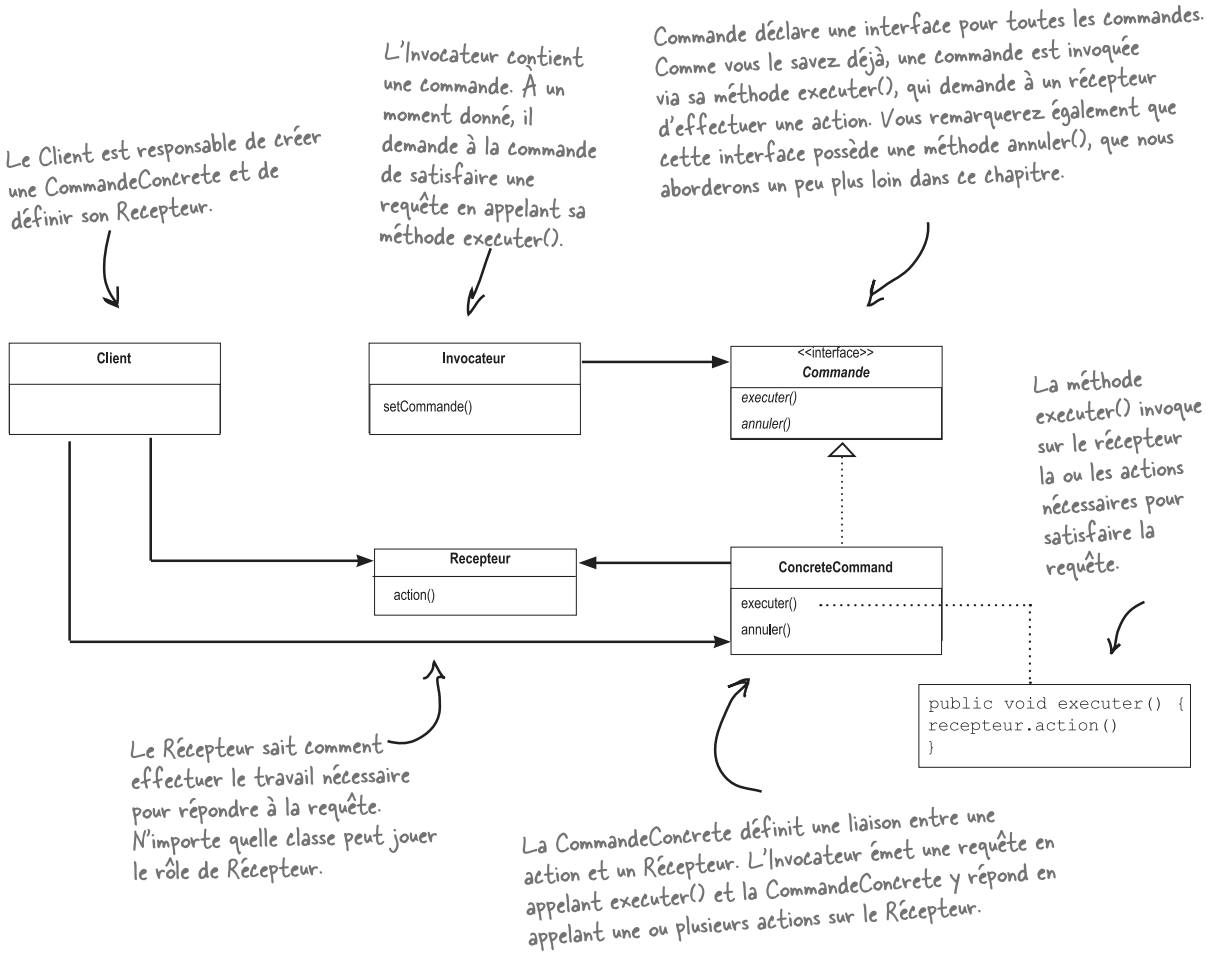

FIGURE 2 – Modèle UML du patron de conception *Command* issue du livre Design patterns : tête la première, Freeman, E. and Freeman, E. and Baland, M.C. and Sierra, K., O'Reilly.