POLYTECH ANGERS

TP 5A - Génie Logiciel
Patrons de conception: Observer

Nicolas Delanoue

(1 séance de 1h20)

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

1 Cahier des charges et prototype

L'objectif est de concevoir un programme permettant
- de rentrer des chaînes de caractères une à une,
- d'afficher (au moyen d'une interface graphique), à chaque entrée d'une nouvelle chaîne de caractère, l'historique des chaînes entrées.

Le projet `TPObserver_eclipse_sujet` est un exemple d'implémentation possible (voir Figure 1 pour la modélisation UML).

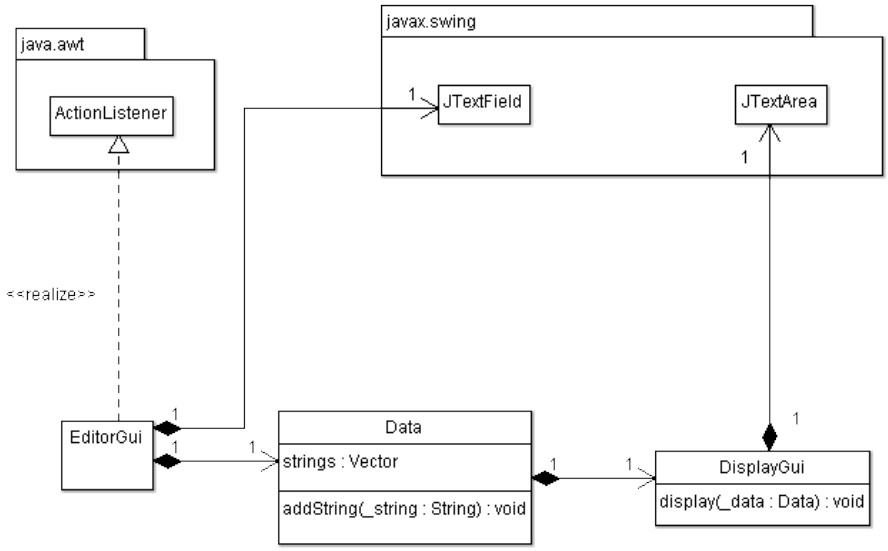

FIGURE 1 – Diagramme de classe UML du projet `TPObserver_eclipse_sujet`.

2 Travail à réaliser

On souhaite que ce programme soit suffisamment modulaire pour pouvoir facilement changer le mode d'affichage (e.g. Console ou Graphique), voire de combiner aisément plusieurs

modes d'affichage qui se rafraîchissent automatiquement lors des ajouts de nouvelles chaînes de caractères. On souhaite également éliminer la dépendance de la classe principale du programme (i.e. `Data`) vis à vis des packages graphiques `java.awt` et `javax.swing`, cette dépendance résultant de la dépendance à la classe `DisplayGui`. La configuration du programme doit pouvoir se faire depuis le lanceur (`Main`) sans modifier la classe `Data`. Pour cela, on propose d'intégrer le patron de conception *observer* (voir Figure 2) au code initial.

Le travail à réaliser consiste à :

— proposer un diagramme de classe UML adapté à notre contexte.

— ajuster le code de la solution précédent en conséquence, afin d'obtenir le même comportement.

On étendra ensuite les fonctionnalités à :

— un affichage simultané sur plusieurs fenêtres (mode graphique),

— un affichage simultané en mode graphique et en mode console (on implémentera une classe spécifique dans ce dernier cas).

On modélisera, au moyen d'un diagramme de séquence, le scénario d'ajout d'une chaîne de caractères depuis l'éditeur graphique, jusqu'à l'affichage.

Remarque 2.1 *On prendra garde de renommer la méthode `Notify` du design pattern (e.g. en `fire()`) pour éviter tout conflit avec la classe java.lang.object qui propose une méthode `notify()` (implémentation `java` du design pattern observer).*

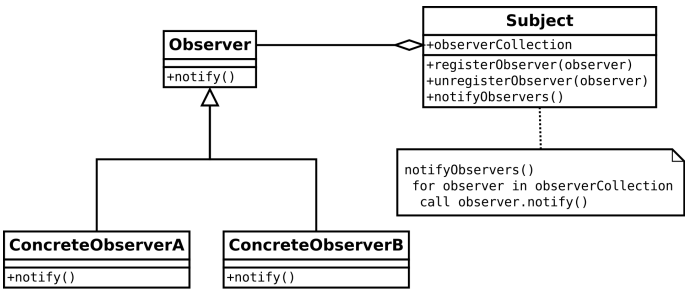

FIGURE 2 – Modèle UML du patron de conception *Observer*.