IFT2255 – Génie logiciel
Devoir 3

Vincent Antaki
Guillaume Poirier-Morency
Émile Trottier

8 décembre 2014

Résumé

L'objectif du travail est d'analyser et conçevoir un système confronté à une série de changements.

Le projet a été séparé en trois paquetages Java. Pour minimiser l'implantation, les classes non-modifiées sont réutilisés depuis les paquetages précédents.

Le projet est accompagné d'une série de tests JUnit au lieu de spécifier une classe Main. C'est plus simple et cela nous permet de soumettre notre implantation à des cas d'utilisation plus réalistes.

1 Diagramme de classes initial

Les différents types d'observateurs ont été découpé en plusieurs interfaces au lieu d'être combinés en une seule afin de laisser la liberté aux éléments de décider à quels types d'observateurs ils désirent s'attacher.

Par exemple, l'interface ActivateObserver est utile que pour la classe Dossier, ce qui évite d'avoir des fichiers activables.

La classe Element a été séparée de l'interface Observable afin de mieux découper le programme en composantes réutilisables.

2 Diagramme de classe modifié

Le patron décorateur a été implanté simplement par un agrégat et un héritage de la classe Element. Il s'agit du seul changement introduit pour répondre à l'énoncé.

Le changement ne pose absolument aucun impact sur les autres classes déjà existantes.

3 Diagramme de classe avec Client

L'introduction d'un Client qui peut déterminer la taille d'un dossier se fait à l'aide d'un patron visiteur.

Le visiteur abstrait est spécialisé pour être accepté par un élément abstrait. Ce qui nous permet de définir n'importe quel type de visiteur.

Le visiteur de taille est une réalisation du visiteur abstrait

4 Introduction de changements

4.1 Première requête de changement

Pour effectuer le changement, nous introduisons la classe Raccourci et ElementRaccourciable. Un raccourci possède un agrégat vers une instance d'ElementRaccourciable.

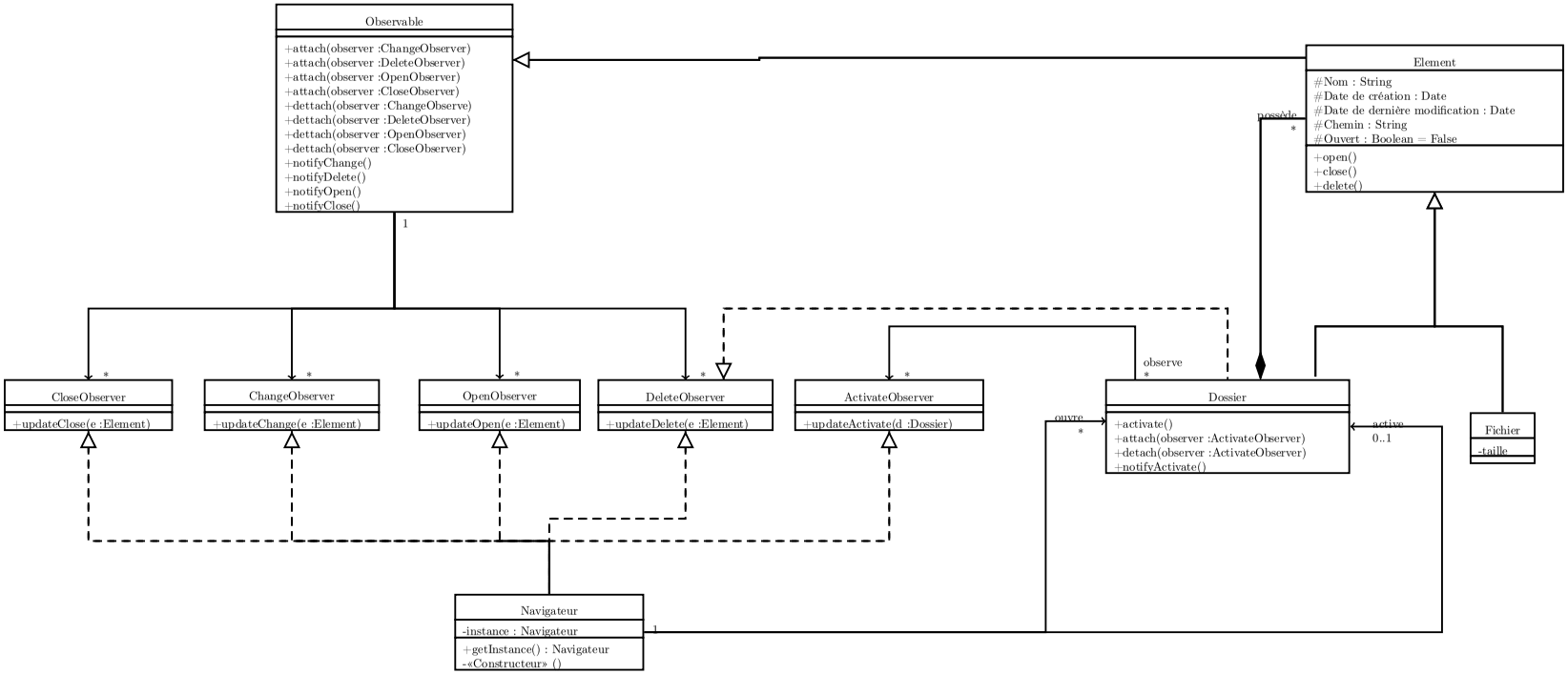

FIGURE 1 – Diagramme de classes initial

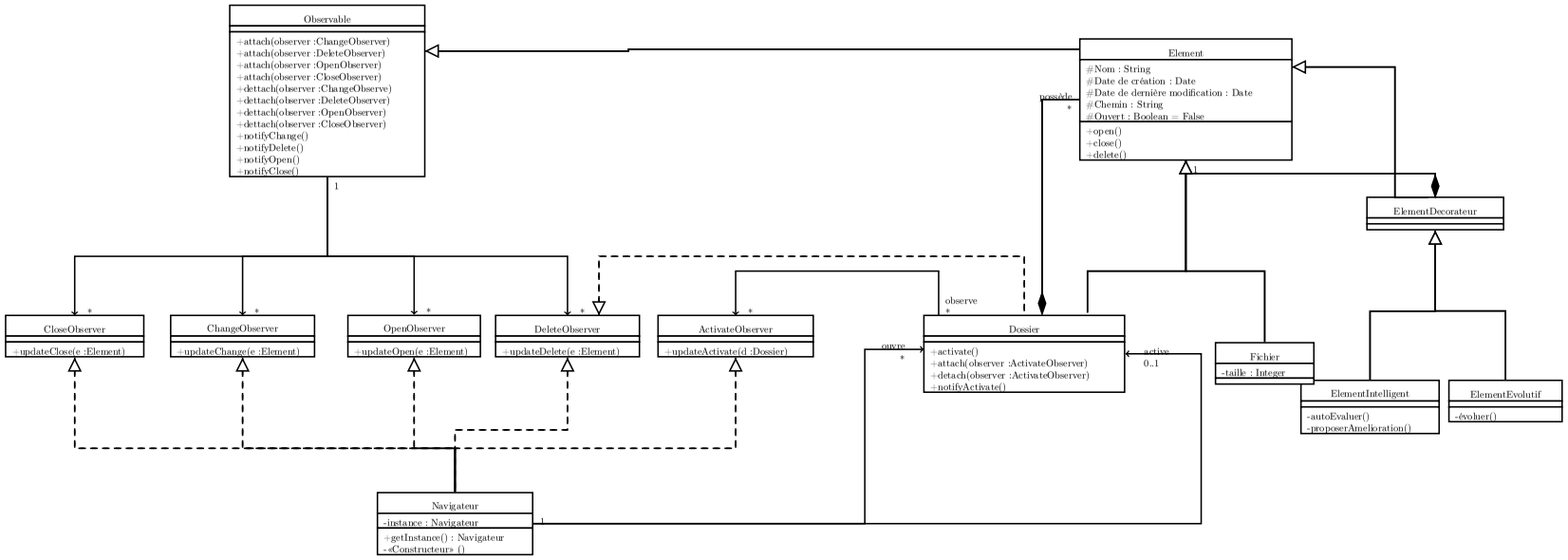

FIGURE 2 – Diagramme de classes

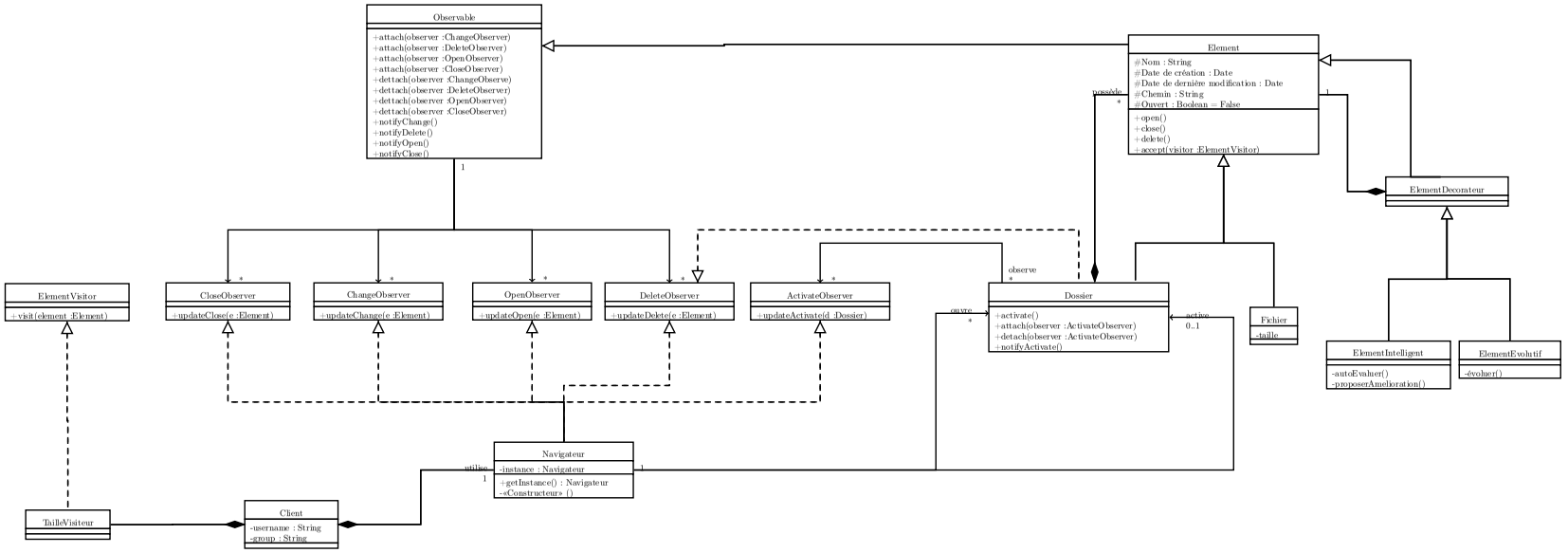

FIGURE 3 – Diagramme de classes

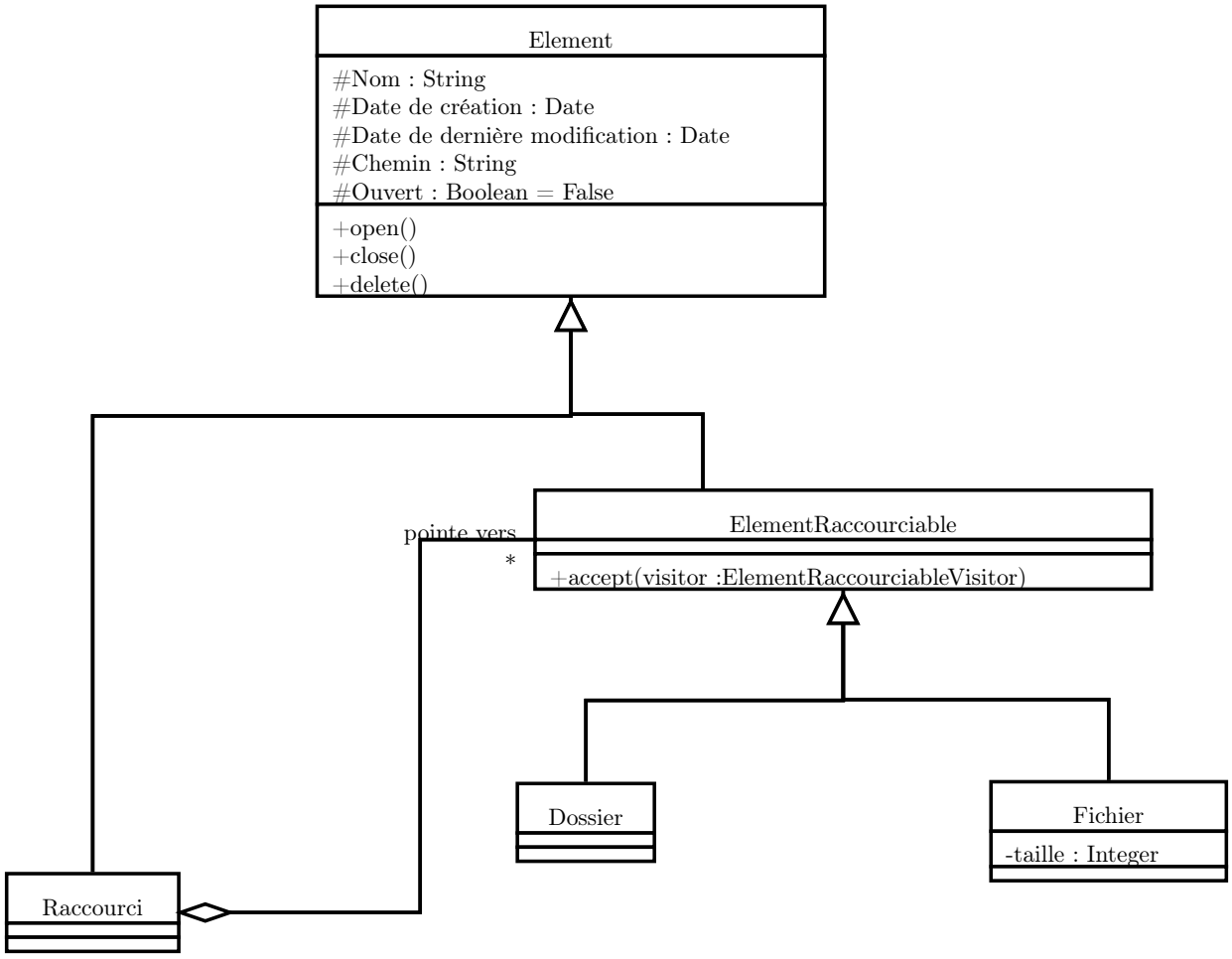

FIGURE 4 – Diagramme de classes pour les raccourcis

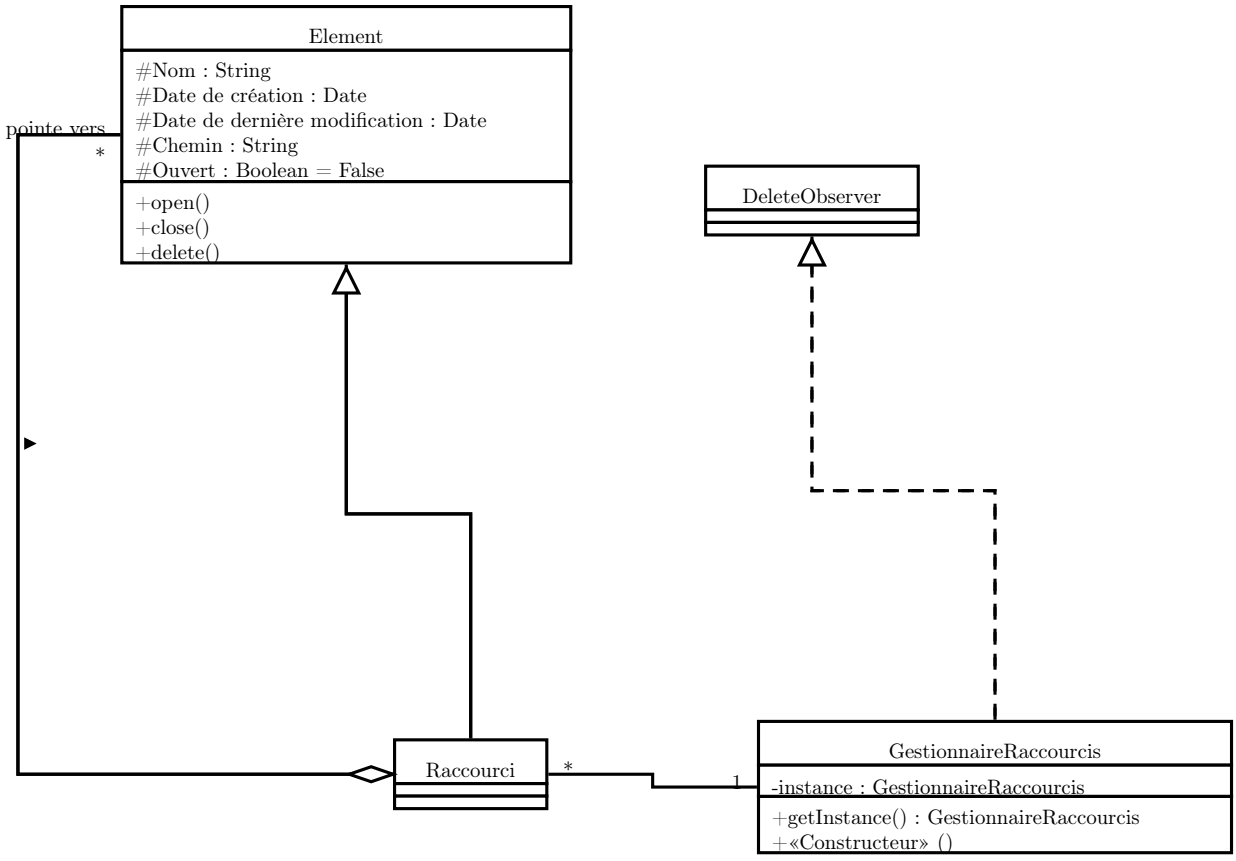

FIGURE 5 – Diagramme de classes pour le gestionnaire de raccourcis

Il s'agit d'une version modifiée du partron décorateur utilisé pour ElementDecorator, car un agrégat le lie à l'élément qu'il pointe.

Les classes Fichier et Dossier seront impactés, car elles devront désormais hériter de la classe ElementRaccourciable.

Il faut aussi déplacer la fonction taille de la classe Element vers la nouvelle classe ElementRaccourciable, car un raccourci ne possède pas de taille à proprement parler. On évite notamment un problème cycle pour le calcul de la taille d'un dossier dans le cas ou un dossier contiendrait un raccourcis qui pointerait vers ce même dossier.

L'impact n'est pas localisé, car il affecte des parties existantes du programme. Notamment les classes Fichier et Dossier qui doivent hériter d'un nouvelle classe.

4.2 Deuxième requête de changement

Pour implanter ce changement, nous introduisons le singleton GestionnaireRaccourcis et nous lui faisons implanter l'interface DeleteObserver.

Lorsqu'un un raccourcis est créé, il attache le gestionnaire en tant que DeleteObserver de l'élément qu'il pointe. Lorsque l'élément pointé est supprimé, le gestionnaire capture l'événement et est en mesure de supprimer le raccourcis approprié.

Le changement est localisé, car il n'affecte pas le code existant. Seule la classe Raccourci est modifiée pour s'enregistrer auprès du gestionnaire lorsqu'il est créé.