IUT de Paris Descartes – Base de la Programmation Objet

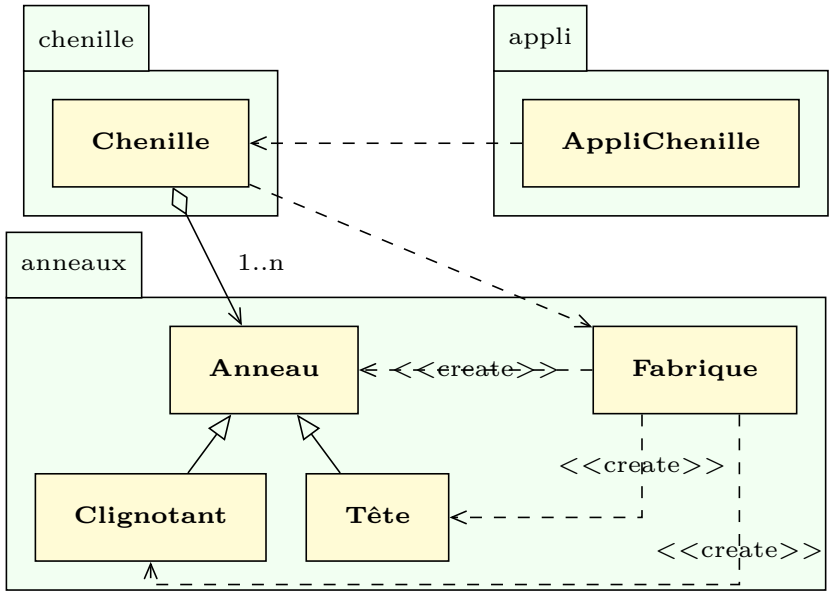
Travaux Pratiques – Sujet n°11

Objectifs

- Savoir relâcher les contraintes de l'héritage en se basant sur le sous-typage (les interfaces).
- Savoir étudier les dépendances entre paquetages en fonction de leur stabilité.
- Savoir inverser une dépendance néfaste en se basant sur le sous-typage.
- Améliorer l'indépendance des classes entre-elles en ajoutant du sous-typage dans une application basée sur l'héritage

Les chenilles revisitées à nouveau

Nous continuons à améliorer l'application permettant l'animation des déplacements de chenille. L'architecture de l'application est à présent la suivante :



Elle présente deux défauts (qui partagent la même solution) :

1. Si nous souhaitons définir un nouveau type d'anneau accepté par la chenille, nous sommes obligés d'hériter de la classe `Anneau` (et de ses attributs). C'est un inconvénient. Par exemple, un anneau emprisonné dans la case de coordonnée (0, 0) n'aurait pas besoin d'hériter des attributs `x` et `y` de la classe `Anneau`.
2. La classe `Chenille` a été restructurée lors de la séance précédente pour la rendre indépendante des sous-classes d'anneau. Cela veut dire que son code est suffisamment générique pour supporter tout

type d'anneau. Le paquetage `chenille` est *stable* et ne devrait plus évoluer. Au contraire, il est souhaitable que de nouveaux types d'anneau soient développés. Le paquetage `anneaux` est *instable* et devrait être enrichi régulièrement.
Il n'est pas souhaitable qu'un paquetage stable dépende d'un paquetage moins stable. Une chenille est composée d'anneaux et elle emploie la fabrique d'anneau. Ces deux dépendances doivent être supprimées.

1. Le code résultant de la séance précédente est fournie sur le serveur commun. Créez un projet et importez les fichiers sources nécessaires.
2. Le premier problème est facile à régler. Il suffit de faire en sorte que la classe `Chenille` ne repose plus sur la classe `Anneau` mais sur une abstraction de celle-ci. La forme la plus abstraite d'une classe est une interface. Déterminez les méthodes que doit comporter cette interface en repérant les méthodes de la classe `Anneau` invoquées par la classe `Chenille`. Pour savoir où est invoquée une méthode donnée de la classe `Anneau`, sélectionnez le nom de la méthode et demandez à Eclipse d'afficher les appelants (menu `Navigate` ⇒ `Open Call Hierarchy`).
3. Créez une interface nommé `IAnneau` comportant les méthodes repérées à la question précédente. Eclipse peut faire le travail pour vous. Sélectionnez la classe `Anneau` et demandez lui d'en extraire une interface (menu `Refactor` ⇒ `Extract Interface`).
4. Cette interface n'a pas vocation à évoluer. Déplacez la vers le paquetage stable `chenille` (menu `Refactor` ⇒ `Move`). Étudiez les nouvelles dépendances créées par ce déplacement. L'interface `IAnneau` ne doit pas dépendre de la classe `Anneau`. Corrigez ce point. Notez que les corrections automatiques proposées par Eclipse sont souvent les bonnes.
5. Faites en sorte que la classe `Chenille` emploie uniquement des objets de type `IAnneau` et qu'elle ne dépende plus de la classe `Anneau`.
6. La dernière dépendance allant du paquetage `chenille` vers le paquetage `anneaux` est l'emploi de la fabrique d'anneaux. Commencez par faire en sorte que la fabrique retourne des objets de type `IAnneau`.
7. Le constructeur de la classe `Chenille` invoque une méthode statique de la classe `FabriqueAnneau`. Pour casser cette dépendance, nous devons employer un objet sachant fabriquer des anneaux (de type `IAnneau`). Transformez la méthode de classe `getAnneau` en une méthode d'instance et faites en sorte que le constructeur de la classe `Chenille` reçoive une fabrique en paramètre et s'en serve pour la fabrication des objets. Corrigez en conséquence le programme principal.
8. Il s'agit à présent de rendre la classe `Chenille` indépendante de la fabrique concrète. Nous procédons de manière similaire à ce que nous avons fait pour les anneaux. Définissez une fabrique abstraite (i.e. une interface) au sein du paquetage `chenille` et faites en sorte que la chenille ne dépende que de celle-ci. Pour définir l'interface, vous pouvez l'extraire de la fabrique concrète actuelle.
9. En vous basant sur la nouvelle architecture, transformez le programme principal de manière à ce qu'il anime une chenille composée uniquement d'anneaux clignotant. Vous devez travailler uniquement dans le paquetage `appli` (le moins stable).