POLYTECH ANGERS

TP 5A - Génie Logiciel
Patrons de conception: FactoryMethod

Nicolas Delanoue

(1 séance de 1h20)

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

1 Cahier des charges et prototype

Il s'agit d'étendre le programme de gestion d'un stock de pièces, intégrant le patron de conception *composite* (le code du prototype est presque celui de la solution du TP dédié à ce patron de conception auquel on a rajouté une interface graphique).

Le contenu de ce projet est disponible dans le fichier `TPFactory_Method_eclipse_sujet`.

L'inconvénient de ce programme est que le détail des pièces créées est géré par le lanceur (e.g. détail de la composition d'un `velo`). Il se trouve que ces pièces sont régulièrement créées et enregistrées dans le stock : ceci pourrait par exemple être réalisé au moyen d'une IHM dédiée permettant d'ajouter la pièce dont on aurait saisi le nom sous forme de chaîne de caractères : « Compteur », « Garde-boue » ou « Vélo ».

Prenons l'exemple du code de création d'un vélo :

```
1  PieceComposite velo = new PieceComposite("Velo",20);
2  velo.add(new PieceDeBase("Pedalier",10));
3  velo.add(new PieceDeBase("Derailleur",15));
4
5  PieceComposite roue_avant_1 = new PieceComposite("Roue Avant",20);
6  roue_avant_1.add(new PieceDeBase("Pneu",10));
7  roue_avant_1.add(new PieceDeBase("Jante",15));
8
9  PieceComposite roue_arriere_1 = new PieceComposite("Roue Arriere",20);
10 roue_arriere_1.add(new PieceDeBase("Pneu",10));
11 roue_arriere_1.add(new PieceDeBase("Jante",15));
12
13 velo.add(roue_avant_1);
14 velo.add(roue_arriere_1);
```

Dans l'état actuel du programme, ce code de création d'un vélo devrait être dupliqué dans le code du `main` et dans le code de l'IHM. Une modification d'un paramètre du vélo implique alors d'éditer ces deux codes. Si le programme se complexifie, le développeur risque d'oublier une mise à jour ou de générer des erreurs.

On souhaite déléguer le détail de leur création et configuration à une entité dédiée, cachant la complexité de leur construction (comme par exemple c'est le cas d'un vélo). Cette entité sera

distincte du lanceur et du gestionnaire de stock. Cette entité sera invoquée par le lanceur pour construire ces pièces.

2 Travail à réaliser

Pour résoudre ce problème, on souhaite gérer la construction des pièces en s'inspirant du design pattern `FactoryMethod` (voir Figure 1) :

- Donner une représentation arborescente d'un vélo.
- Créer une classe abstraite `Creator`, et créer trois fabriques concrètes qui réaliseront les vélos, les compteurs et les gardes-boues.
- Proposer un diagramme de classe (UML) adapté à notre contexte,
- ajuster le code de la solution précédent en conséquence, sachant que le programme modifié doit se comporter exactement de la même manière que le programme non modifié,
- proposer un diagramme de séquence illustrant le processus de création de pièces et d'ajout au stock (i.e. déroulement du `Main`)

A quelles classes correspondent les éléments `Product` et `ConcreteProduct` du patron de conception générique ?

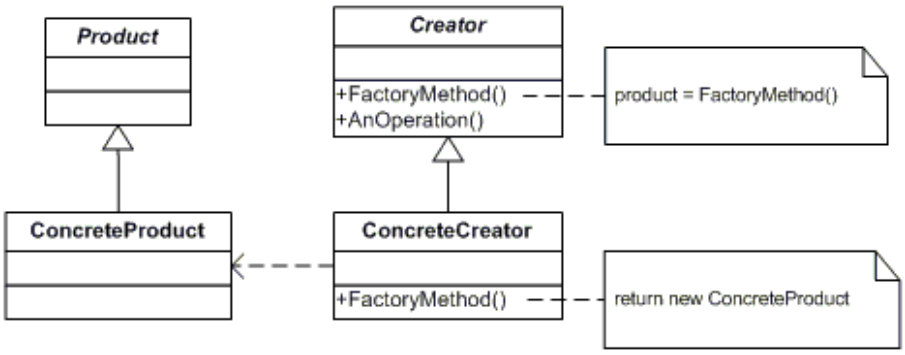

FIGURE 1 – Modèle UML du patron de conception *FactoryMethod*.