Université Paris Cité

Compléments en Programmation Orientée Objet
TP noté n° 1, groupe Info 4 : Inversion de Dépendance

Exercice 1 : Écriture d'une bibliothèque pratiquant l'inversion de dépendance

Comme alternative au patron Observateur/Observé, on suppose un moteur de jeu ou de simulateur (modèle) agissant directement sur des instances représentant, de façon abstraite, des éléments graphiques. Ainsi, ce moteur de jeu n'ait pas besoin de dépendre de la bibliothèque graphique, de connaître son fonctionnement ou de savoir les détails d'implémentation des éléments graphiques représentant les éléments du jeu.

Plus précisément : le moteur de jeu, doit pouvoir instancier un élément graphique à chaque fois qu'un élément du modèle est créé. Cela veut dire qu'il doit pouvoir créer des instances de classes qu'il ne connaît pas a priori. Pour que ces éléments graphiques soient utilisables, ils doivent implémenter une interface `Sprite` déclarée avec le moteur. Mais cela ne suffit pas, puisqu'une interface n'a pas de constructeur.

Ainsi, en plus de cela, le moteur doit accepter que son client (qui sera en fait la classe principale du lanceur graphique) lui passe les objets permettant de créer des instances d'`Sprite`. Ces objets s'appellent des fabriques (concrètes) ; comme le moteur ne connaît pas non plus leur classe, il déclare une interface que ces fabriques doivent implémenter. Cette interface est la fabrique abstraite. Nommons-la `SpriteFactory`.

Pour cet exercice, nous montrons un petit "jeu" de simulation de trajectoire, ultra-simple, : le simulateur instancie en (0,0) des *flyers* nommés `"Numéro i/j` (où `i` est le numéro d'instance et `j` le nombre de vies total) et les déplace de (+1,+1) à chaque mise à jour de la simulation. Un `flyer` est instancié au départ. Il est détruit dès qu'il sort du plateau. S'il reste des vies, un nouveau flyer est instancié en (0,0) et on recommence, sinon c'est la fin de la simulation.

Exemple d'exécution avec 3 vies, sur une grille de 5 lignes et 8 colonnes (sur la sortie standard) :

```
Numéro 1/3 : je suis en (0, 0).
Numéro 1/3 : je suis en (1, 1).
Numéro 1/3 : je suis en (2, 2).
Numéro 1/3 : je suis en (3, 3).
Numéro 1/3 : je suis en (4, 4).
Numéro 2/3 : je suis en (0, 0).
Numéro 2/3 : je suis en (1, 1).
Numéro 2/3 : je suis en (2, 2).
Numéro 2/3 : je suis en (3, 3).
Numéro 2/3 : je suis en (4, 4).
Numéro 3/3 : je suis en (0, 0).
Numéro 3/3 : je suis en (1, 1).
Numéro 3/3 : je suis en (2, 2).
Numéro 3/3 : je suis en (3, 3).
Numéro 3/3 : je suis en (4, 4).
```

Le programme se divise en 2 packages : le moteur de jeu en lui-même (`model`) et l'application "graphique" (ici en mode texte) qui l'utilise (package `gui`). Ce dernier package vous est fourni (.zip sur Moodle), vous devez programmer l'autre.