TD patterns de création

Nous proposons de réaliser un jeu de labyrinthe. Dans un premier temps, nous nous intéressons à la construction du labyrinthe, afin de se familiariser avec les différents patterns de création.

Labyrinthe

Un labyrinthe est composé de salles carrées de même taille, juxtaposées les unes aux autres. Les côtés d'une salle peuvent être une autre salle, un mur ou une porte donnant sur une autre salle. Chaque salle a un numéro unique et une porte peut être verrouillée ou non. La figure 1 illustre le diagramme de classe de notre labyrinthe. L'interface `MapSite` représente les différents composants d'un labyrinthe (*i.e.*, salle, mur, porte) et définit une opération `Enter` permettant d'entrer dans le composant. La classe `Room` définit les relations clé entre les composants du labyrinthe. En particulier, elle gère des références à d'autres objets `MapSite` (ses côtés). Enfin, la classe `Maze` représente une collection de salles et fournit des opérations pour ajouter une salle au labyrinthe et pour trouver une salle étant donné son numéro. Cette classe ne garde toutefois aucune référence à d'autres objets `MapSite`.

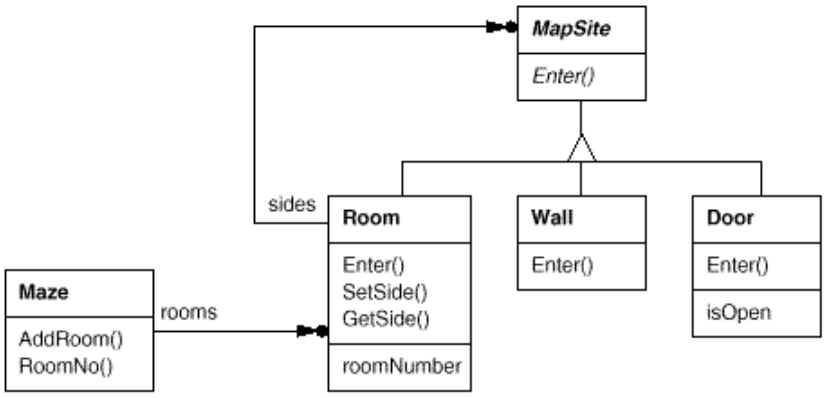

FIGURE 1 – Diagramme de classes d'un labyrinthe

1. Quel type de relation existe-t-il entre les classes :
   (a) `Maze` et `Room` ?
   (b) `Room` et `MapSite` ?
   (c) `Door` et `Room` ?
2. Complétez le diagramme de classes.
3. Écrivez une implémentation des classes du diagramme [1].
4. Créez une classe `MazeGame` comportant une méthode `createMaze` qui permet de construire un labyrinthe avec deux salles séparées par une porte.

1. Vous pouvez utiliser un type énuméré pour représenter les côtés Nord, Sud, Est et Ouest d'une salle.

5. Ajoutez de nouveaux types de composant au jeu : salle enchantée, salle minée, mur fissuré, porte magique, *etc.* afin de pouvoir créer de nouveaux types de labyrinthe.
6. Écrivez un programme de test qui crée un labyrinthe, plus une version enchantée et une version minée de ce labyrinthe, sans modifier la manière dont le jeu crée des labyrinthes.

Méthode de fabrique

1. Proposez un diagramme de classes utilisant le pattern Méthode de fabrique pour la création de labyrinthe.
2. Écrivez une implémentation des nouvelles classes du diagramme.
3. Écrivez un programme de test qui crée un labyrinthe, plus une version enchantée et une version minée de ce labyrinthe, en utilisant des méthodes de fabrique.
4. Quelles sont les propriétés de cette solution pour notre problème ?

Fabrique abstraite

1. Proposez un diagramme de classes utilisant le pattern Fabrique abstraite pour la création de labyrinthe.
2. Écrivez une implémentation des nouvelles classes du diagramme.
3. Écrivez un programme de test qui crée un labyrinthe, plus une version enchantée et une version minée de ce labyrinthe, en utilisant des fabriques.
4. Où est la différence avec le pattern Méthode de fabrique ?

Singleton

1. Modifiez le diagramme de classes en utilisant le pattern Singleton, afin qu'il ne puisse exister qu'une seule instance de fabrique pour créer des labyrinthes.
2. Modifiez l'implémentation des fabriques en conséquence.

Prototype

1. Modifiez le diagramme de classes afin d'utiliser le pattern Prototype pour la création de labyrinthe.
2. Écrivez une fabrique qui utilise des prototypes pour créer les salles, les murs et les portes d'un labyrinthe.
3. Écrivez un programme de test qui crée un labyrinthe, plus une version enchantée et une version minée de ce labyrinthe, en utilisant cette nouvelle fabrique.

Monteur

1. Proposez un diagramme de classes utilisant le pattern Monteur pour la création de labyrinthe.
2. Écrivez une implémentation des nouvelles classes du diagramme.
3. Écrivez un programme de test qui crée un labyrinthe, plus une version enchantée et une version minée de ce labyrinthe, en utilisant des monteurs.
4. Écrivez un monteur qui utilise une fabrique pour créer les salles, les murs et les portes d'un labyrinthe.
5. Utilisez ce nouveau monteur pour créer des labyrinthes.