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`.

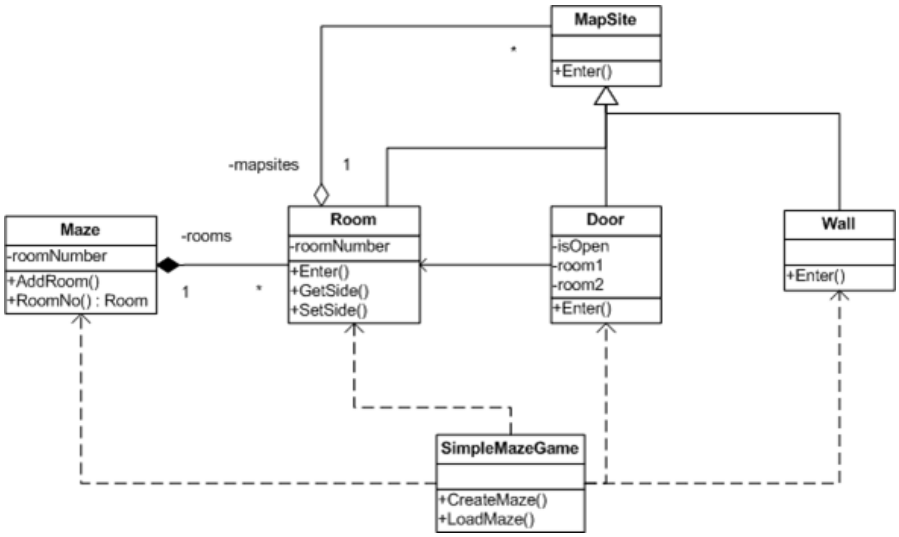


FIGURE 1 – Diagramme de classes d'un labyrinthe

1. Récupérez l'archive `maze.jar`, puis importez-la dans Eclipse.
2. Créez une classe `SimpleMazeGame` comportant une méthode `createMaze` qui permet de construire un labyrinthe avec deux salles séparées par une porte.
3. 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.
4. 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.

5. Quels sont les inconvénients de votre solution ? **Corrigé** ▶ *Deux solutions :*

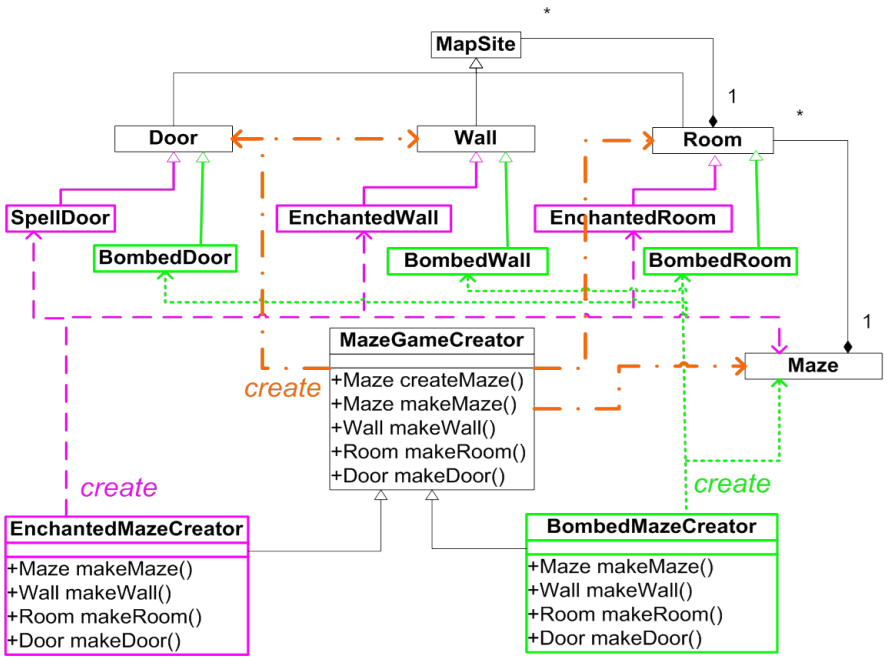
   (a) Dupliquer le code de `createMaze()` :
   - *`createEnchantedMaze()`*
   - *`createBombedMaze()`*
   - *etc.*

   (b) Ajouter des instructions `switch/case` chaque fois qu'un constructeur d'un composant du labyrinthe est invoqué
   - *Utilisation de flags*

   La solution (a) requiert également de changer le client appelant la méthode `createMaze()`.
   La solution (b) doit être récrite chaque fois qu'une nouvelle extension sort, ce qui est sujet à erreurs.
   ◀

Méthode de fabrique

1. Proposez un diagramme de classes utilisant le pattern Méthode de fabrique pour la création de labyrinthe. **Corrigé** ▶



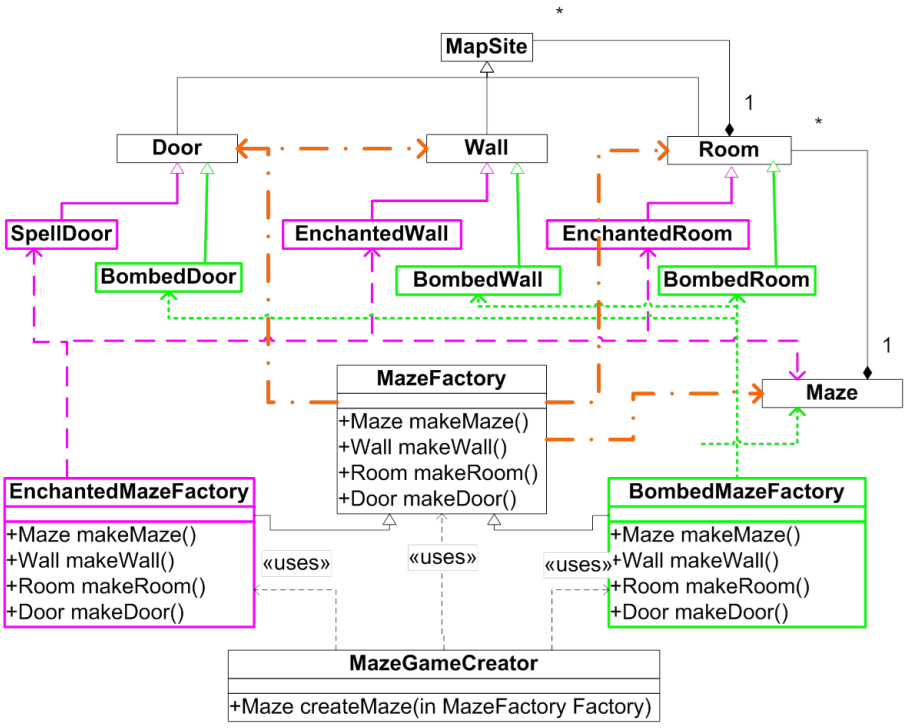
   La méthode `createMaze()` peut toujours être appelée pour créer des labyrinthes ordinaires ou des labyrinthes enchantés sans modification.◀

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 ? **Corrigé** ▶
   - *Le client qui invoque la création de labyrinthes n'a pas besoin de changer.*
   - *Il interagit avec les différentes classes de création de labyrinthes*
     - *Selon l'extension sélectionnée par le joueur ,*
     - *Exactement de la même manière que dans le jeu original.*
   - *Attention :*
     - *Un flag global est nécessaire pour déterminer quelle sous-classe de MazeCreator doit être instanciée dans chaque jeu.*

   ◀

Fabrique abstraite

1. Proposez un diagramme de classes utilisant le pattern Fabrique abstraite pour la création de labyrinthe. Corrigé ▶



La méthode `createMaze()` prend désormais en paramètre une référence à un objet de type `MazeFactory`.◀

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 ? Corrigé ▶*Dans le pattern Fabrique abstraite, une classe délègue la responsabilité de l'instanciation des objets à une autre classe via la composition.*
   Le pattern Méthode de fabrique utilise l'héritage pour gérer l'instanciation des objets désirés.
   Les patterns de création montrent comment rendre la conception plus flexible, mais pas forcément plsu petite. En particulier, ils rendront plus facile le changement des classes qui définissent les composants d'un labyrinthe. ◀