CS350: Software Design/ SE310: Software Architecture I

Lab 3: Build Maze Game with Creational Patterns

Goal:

Practice Factory Method Pattern and Abstract Factory Pattern

Requirements:

1. Adding the Red Maze Game and Blue Maze Game feature in addition to the basic function. The Red Maze Game will have **pink rooms**, **red walls** and **green doors**. The Blue Maze Game will have **light gray rooms**, **brown door** and **blue walls**.
2. Modify the maze game you accomplished in lab 2 to use Factory Method pattern.
3. Compile the new program to show basic, red or blue maze games.
4. Modify the maze game you accomplished in lab 2 to use Abstract Factory pattern.
5. Compile the new program to show basic, red or blue maze games.

Instructions

This lab has the following two stages:

Hint: in order to change colors of the doors, rooms, etc, you need to

1. Import Color library using :

   ```
   import java.awt.Color;
   ```

2. Override the getColor() method defined in MapSite.java. For example, if you want a wall to be blue, the code should look like:

```
public Color getColor()    {
        return Color.BLUE;
   }
```

Stage 1: Using Factory Method Pattern

1. Select a working directory
2. Copy and paste the basic maze game in lab 2 to the working directory
3. Modify the basic maze game according to the UML Factory Method model:
   a. Modify the *SimpleMazeGame* class into a *MazeGameCreator* class, add factory methods and change the *main* method accordingly
   b. Run the new program
4. Add a Red Maze Game Feature
   a. Add necessary red maze game classes, such as the pink room class.
   b. Add a new *RedMazeGameCreator* class
   c. Change the *main* method to accept a “red” parameter

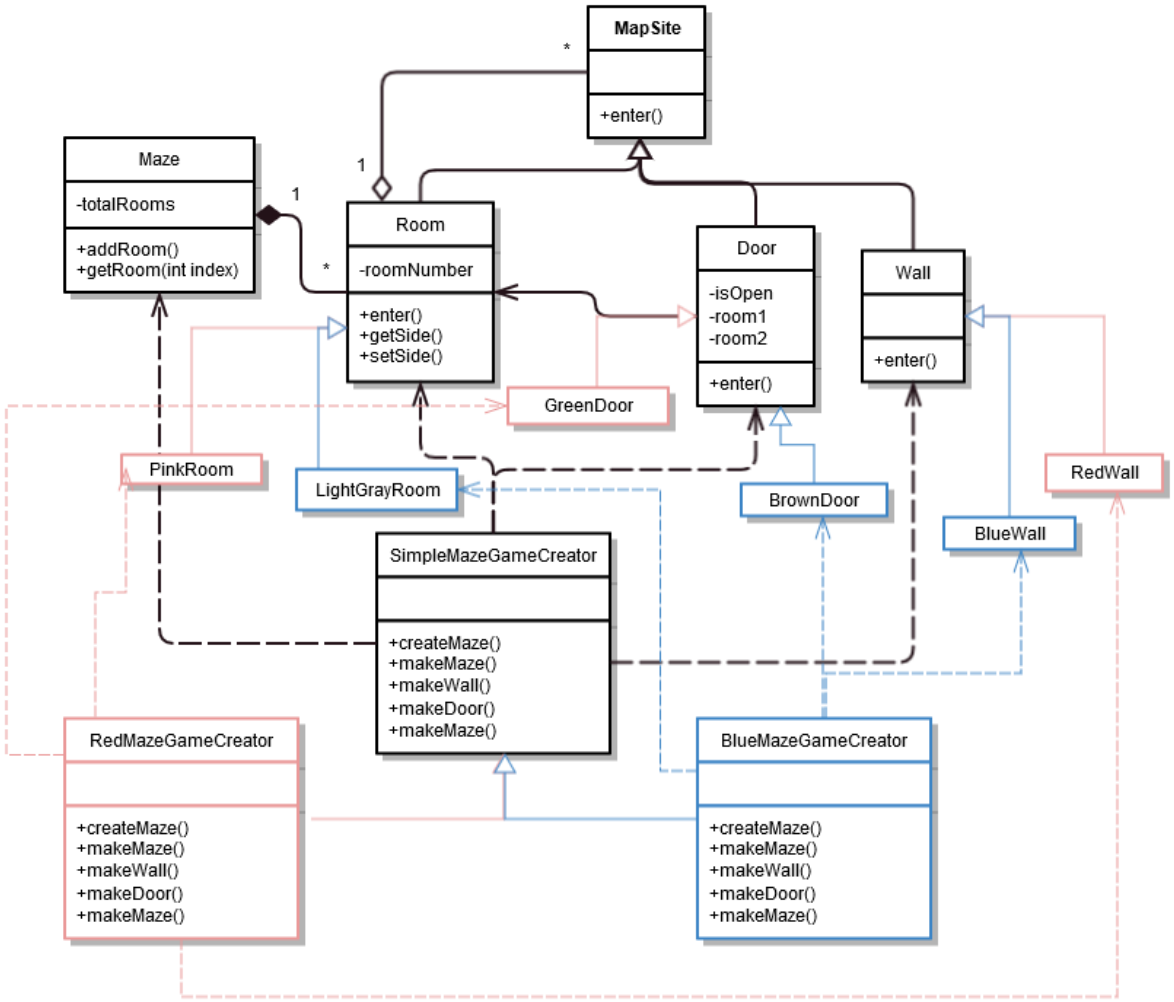
    d. Run the new program with “red” option
5. Add a Blue Maze Game Feature
    a. Add necessary blue maze game classes, such as a new room class.
    b. Add a new BlueMazeGameCreator class
    c. Change the *main* method to read both “red” and “blue” parameters
    d. Run the new program with both “red” and “blue” options

Stage 2: Using Abstract Factory Pattern

1. Select a working directory
2. Copy and paste the basic maze game in lab 2 to the working directory
3. Modify the basic maze game program according to the UML Abstract Factory diagram:
    a. Modify the *SimpleMazeGame* class into a *MazeGameAbstractFactory* class, and change the *main* method accordingly
    b. Create a *MazeFactory* interface
    c. Run the new program
4. Add a Red Maze Game Feature
    a. Add necessary red maze game classes.
    b. Add a new *RedMazeFactory* class
    c. Change the *main* method to read a “red” parameter
    d. Run the new program with “red” option
5. Add a Blue Maze Game Feature
    a. Add necessary blue maze game classes.
    b. Add a new BlueMazeFactory class
    c. Change the *main* method to read both “red” and “blue” parameters

Note: you should improve your lab 2 to follow these principles before you start lab3. If you have improved your code from lab 2 significantly, for example, used better data structure, then write one paragraph to explain what you have done.

Appendix A: Maze Game Factory Method Pattern



Appendix B: Maze Game Abstract Factory Pattern