CS350 Software Design

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 light red rooms and red walls. The Blue Maze Game will have blue walls, brown doors and green rooms.
2. Modify the maze game you accomplished in lab 2 to use Factory Method pattern. Your program should still load mazes from files.
3. Compile the new program show basic, red or blue maze games.
4. Modify the maze game you accomplished in lab 1 to use Abstract Factory pattern.
5. Compile the new program show basic, red or blue maze games. **Instructions**

This lab has the following two stages:

Stage 1: Using Factory Method Pattern

1. Select a working directory
2. Copy and paste the basic maze game in lab 1 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 new maze game component classes, such as a new room class.
    b. Add a new *RedMazeGameCreator* 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 new maze game component classes, such as a new room class.
    b. Add a new BlueMazeGameCreator class
    c. Change the *main* method to read a “blue” parameter
    d. Run the new program with “blue” option
6. The program should now allow you to create basic, red and blue mazes depending on the input parameters.

7. Save the source code that you have created for part 1 since you will need to submit working source code for both parts.

Stage 2: Using Abstract Factory Pattern

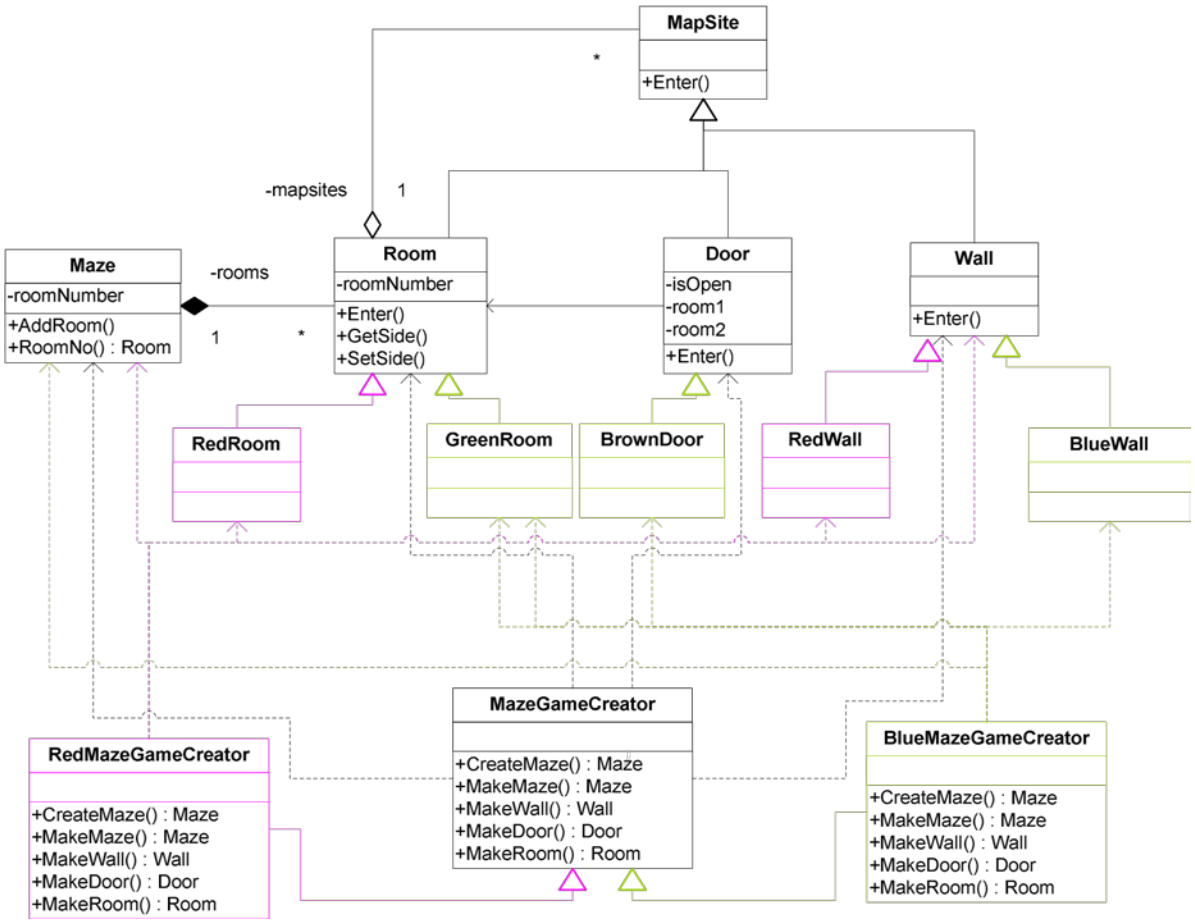
1. Select a working directory
2. Copy and paste the basic maze game in lab 2 to the working directory
3. Modify the basic maze game according to the UML Abstract Factory model:
   a. Modify the *SimpleMazeGame* class into a *MazeGameAbstractFactory* class, add change the *main* method accordingly
   b. Create a *MazeFactory* interface
   c. Run the new program, note it should still load a maze from a file.
4. Add a Red Maze Game Feature
   a. Add necessary new maze game component classes, such as a new room class.
   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 new maze game component classes, such as a new wall class.
   b. Add a new BlueMazeFactory class
   c. Change the *main* method to read a "blue" parameter
6. The program should now allow you to create basic, red and blue mazes depending on the input parameters.

Grading Guidelines

(1) Applying Factory Method pattern
   a. The code conforms to the UML class diagram of FM pattern: 25%
   b. Being able to create the maze game according to given parameters: 25%

(2) Applying Abstract Factory pattern
   a. The code conforms to the UML class diagram of AF pattern: 25%
   b. Being able to create the maze game according to given parameters: 25%

Appendix A: Maze Game Factory Method Pattern



Appendix B: Maze Game Abstract Factory Pattern