CS350: Software Design/SE310: Software Architecture

Software Design Lab 2: Build a Simple Maze Game

Goal:

1. Get familiar with basic UML modeling
   a. Understand the given UML diagrams
   b. Implement Java code according to the UML diagram
2. Practice basic OO techniques using Java, including
   a. Using classes and methods
   b. Basic I/O
   c. Java compilation and configuration

Requirements:

1. Modify and compile the given program to show an empty maze game
2. Modify methods in the given code to add rooms in the maze
3. Compile the modified program to show a simple maze with at least two rooms, and the rooms are connected by walls or doors,
4. Modify the program again to read from input files that specify maze structures.
5. Compile the modified program to show maze game as specified in the input files

Instructions

This lab has three stages:

Stage 1: Make the program running.

1. Unzip the given *lab2.zip* into the working directory, and you will see two subdirectories, *src* and *bin*, as well as a *maze-ui.jar* file.
2. Read the .java files within the *src* directory according to the given UML diagrams and understand their relations.
3. Modify and program slightly to make it compliable. Run the program. If successful, you will see an empty small window and a prompt:

   "The maze does not have any rooms yet!"

Stage 2: Build a maze with rooms

1. Fill in the *CreateMaze* function in the *SimpleMazeGame* class to create a maze with at least two rooms, connected with doors.   Note:
   a. please number your room starting from 0!
   b. You need to call setCurrentRoom() to show the small ball moving around the maze.

2. Compile and run the new program to show a maze game with rooms

Stage 3: Load a maze from a given file.

1. Two maze input files, *large.maze* and *small.maze* are provided. The format of each line:

   <Room/Door> <room no./door no.> < North> <South> <East> <West>

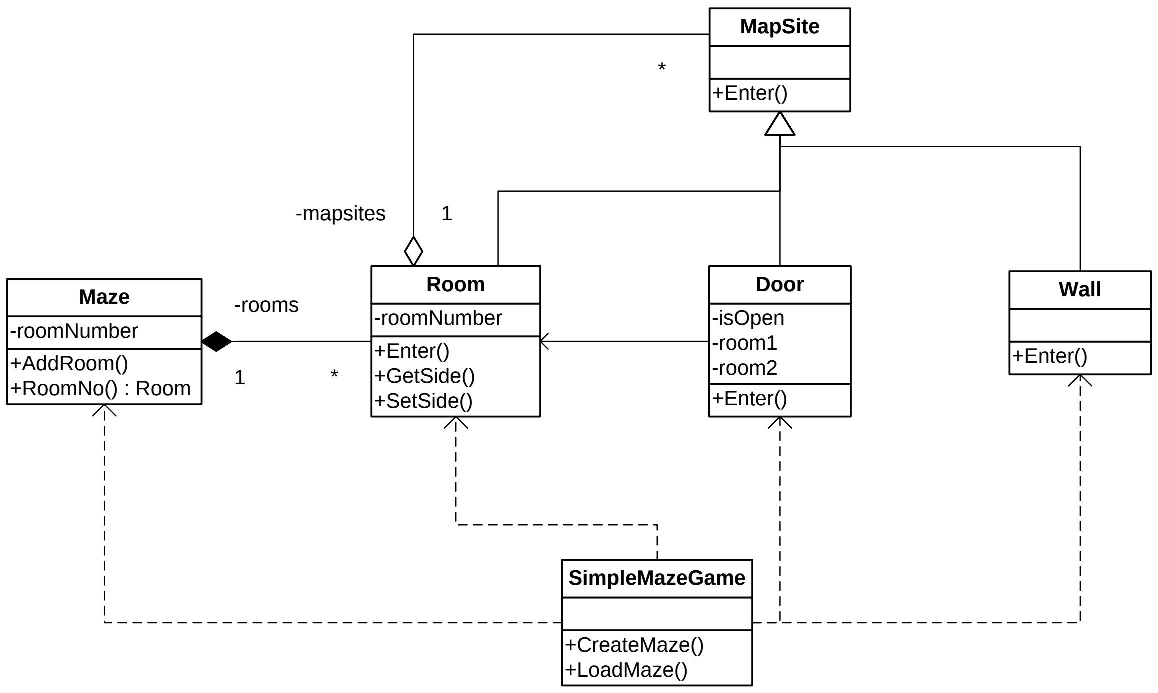
   The order of direction is the same as the enumeration type: Direction.

2. Put these two files into a directory.
3. Fill in the *loadMaze* function in the *SimpleMazeGame* class to read a maze from a given file
4. Change the *main()* function so that
   a. If the input path file is given as a parameter, then a corresponding maze should be produced;
   b. If no parameter is given, then a maze should be created as you did in stage 2.
5. Compile and run the new program.

After Stage 3, pack the final program into a zip file and submit it through BbLearning.

Suggestions: you might want to use "java.util.Scanner" to process file input.

Appendix: Basic Maze UML Class Diagram



Note: There is a *Direction* class in the code distributed. This is an enumeration type that is not shown in the UML class diagram.