SE 310 Software Architecture 1

Homework 1: Basic Maze and File Parsing

Goal:

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

Requirements:

1. Compile the given program to show an empty maze game
2. Modify methods in the given code to add rooms to the maze
3. Compile the modified program to show a simple 2 room maze, each with 3 walls, and connected by a door
4. Modify the program again to read from input files that specify maze structures
5. Compile the modified programs to show the maze using the large.maze file. If you're unable to parse the large.maze file you may use the small.maze file and receive a penalty.

Instructions

There are 3 stages

Stage 1: Make the program run

1. Download **`HW1.zip`** from Blackboard
2. Extract the contents of **`HW1.zip`**
3. Import the folder/directory "**`HW1`**" from the zip file as a new IntelliJ project. Detailed instructions on using IntelliJ and importing zip files can be found on Blackboard. Please follow the instructions.
4. Attempt to run the project. The message "The maze does not have any rooms yet!" will be displayed if you can successfully run the project.
5. Note, that this is a small part of a larger program. You do not need to understand everything that is going on. Just focus on the task at hand.

Stage 2: Build a maze with rooms

1. Fill in the `createMaze` function in the `SimpleMazeGame` class to create a 2 room maze.
    a. You must set the current room number
    b. Room numbers start at `0`

    c. Rooms must be complete with walls and doors
    d. If you receive an error related to UI or 'painter' you did something wrong. ***There is nothing wrong with the supplied jar file.***

2. Compile and run the new program to show a two-room maze

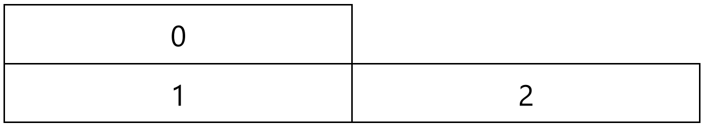
Stage 3: Load a maze from small.maze and large.maze

1. The two maze files are included in the zip. They should be included in your IntelliJ project, inside the project folder but outside of your source folder.

2. The maze file contains the following information:
    a. Object Name (room/door)
    b. Object Identifier (i.e. Room number or Door number)
    c. If it's a room, four entries follow that are either a:
        i. Wall
        ii. Door – Object identifier for a door
        iii. Room Number. – Object identifier for a room

    The four identifiers are in the order North, South, East or West.

    For example, the following maze:



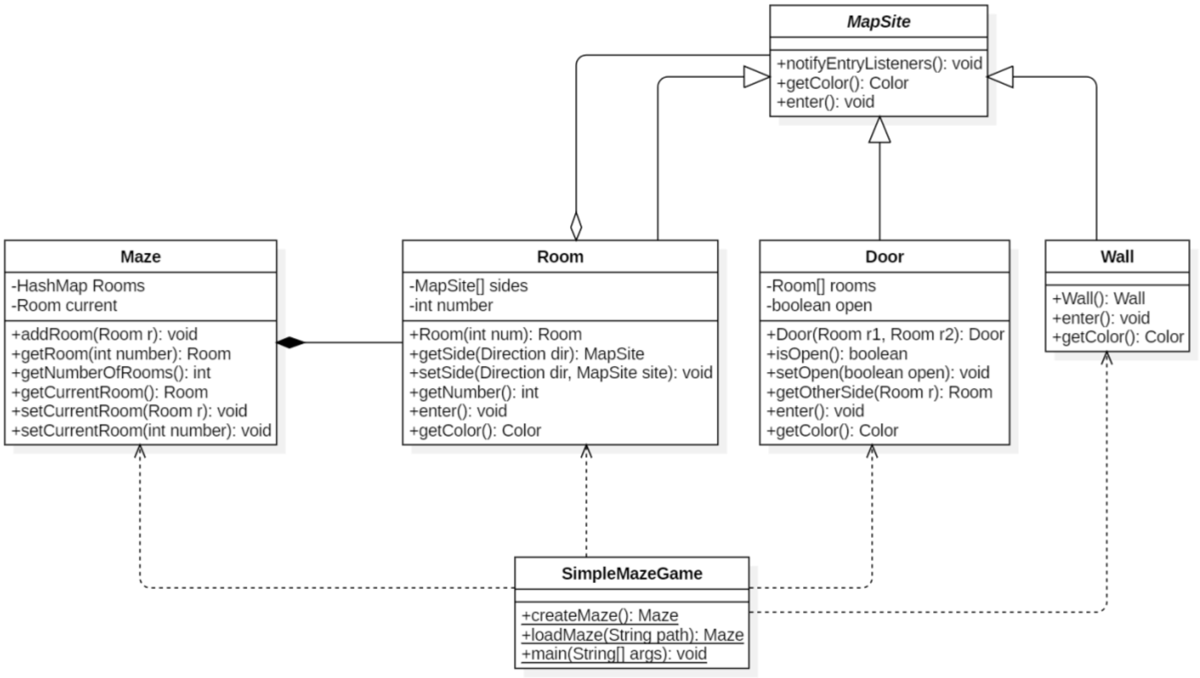
    has the following file format:

```
room 0 wall 1 wall wall
room 1 0 wall 2 wall
room 2 wall wall wall 1
```

3. Initially, to sanity check your loading functionality, test the system with the `small.maze` file and work up to `large.maze`. Parsing code for `small.maze` will need to be modified to handle parsing `large.maze`. One you're able to parse `large.maze` set your `main()` function to automatically use `large.maze` and default to using the `loadMaze()` function instead of `createMaze()`.

Submission:

Please follow the directions outlined in the submission document on Blackboard Learn. Submissions not following this document will receive a 0 (zero). The late policy outlined in the syllabus applies to this assignment.


MapSite
+notifyEntryListeners(): void
+getColor(): Color
+enter(): void
Maze
-HashMap Rooms
-Room current
+addRoom(Room r): void
+getRoom(int number): Room
+getNumberOfRooms(): int
+getCurrentRoom(): Room
+setCurrentRoom(Room r): void
+setCurrentRoom(int number): void
Room
-MapSite[] sides
-int number
+Room(int num): Room
+getSide(Direction dir): MapSite
+setSide(Direction dir, MapSite site): void
+getNumber(): int
+enter(): void
+getColor(): Color
Door
-Room[] rooms
-boolean open
+Door(Room r1, Room r2): Door
+isOpen(): boolean
+setOpen(boolean open): void
+getOtherSide(Room r): Room
+enter(): void
+getColor(): Color
Wall
+Wall(): Wall
+enter(): void
+getColor(): Color
SimpleMazeGame
+createMaze(): Maze
+loadMaze(String path): Maze
+main(String[] args): void