PPA Assignment 6

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

November 20, 2016

1 Introduction

In this assignment we create a text based adventure game. The player will choose between two rooms, *blueDoorRoom* and *redDoorRoom* where one leads to monster room and another one to next room. If player can reach the final room without losing all lives the player wins. (Topics from week 1,...,7).

2 Pseudocode

2.1 Class Coordinates

Pseudocode 1: This class models player characteristics.

```
Initialise private String name
Initialise private integer lives
Initialise private Room currentRoom
Define Player
    Set name
    Set lives
    set currentRoom
Define move
    Set currentRoom
Define getCurrentRoom
    Return currentRoom
Define getLives
    Return lives
Define decreasePlayerLives
    Subtract 1 from lives
Define printStatus
    Print name
    Print lives
    Print name of currentRoom
```

2.2 Class Room

Pseudocode 2: This class has a characteristics of a maze room.

```
Initialise private String $name$
Initialise private Room $blueDoorRoom$
Initialise private Room $redDoorRoom$
Initialise private boolean $isFinalRoom$
Initialise private boolean $containsMonster$
Define Room
    Set $name$
    Set $containsMonster$
    Set $isFinalRoom$
Define Room
    Set $name$
    Set $blueDoorRoom$
    Set $redDoorRoom$
Define getName
    Return $name$
Define isFinalRoom
    Return $isFinalRoom$
Define doesContainMonster
    Return $containsMonster$
Define getBlueDoorRoom
    Return $blueDoorRoom$
Define getRedDoorRoom
    Return $redDoorRoom$
```

2.3 DoorMazeGame

Pseudocode 3: This class is going to drive our program

```
Initialise Scanner in
Initialise Room monsterRoom with $”TheMonsterRoom”, true, false$
Initialise Room $room6$ with $”TheGreatHall”, false, true$
Initialise Room $room5$ with $”TheFifthHall”, room6, monsterRoom$
Initialise Room $room4$ with $”TheFourthHall”, monsterRoom, room5$
Initialise Room $room3$ with $”TheThirdHall”, room4, monsterRoom$
Initialise Room $room2$ with $”TheSecondHall”, room3, monsterRoom$
Initialise Room $room1$ with $”TheFirstHall”, monsterRoom, room2$
Print message that asks the user to type his name
Initialise Player player with input name, $2, room1$
while player lives is greater than 0 AND current room isFinalRoom is false do
    Print Current player status
    Initialise String nextRoom to take next user input
    if String value of nextRoom and "blue" are equal AND blue door room contains monster
      then
        Subtract 1 from player lives
    else if String value of nextRoom and "blue" are equal then
        move player to blue door room
    else if String value of nextRoom and "red" are equal AND red door room contains
      monster then
        Subtract 1 from player lives
    else if String value of nextRoom and "red" are equal then
        move player to red door room
    else
        Print messages to explain the user number of available commands
    end
end
if current room’s flag $isFinalRoom$ is true then
    Print Victory messages
else
    Print Game Over messages
end
Close scanner in
```

3 Class Diagram

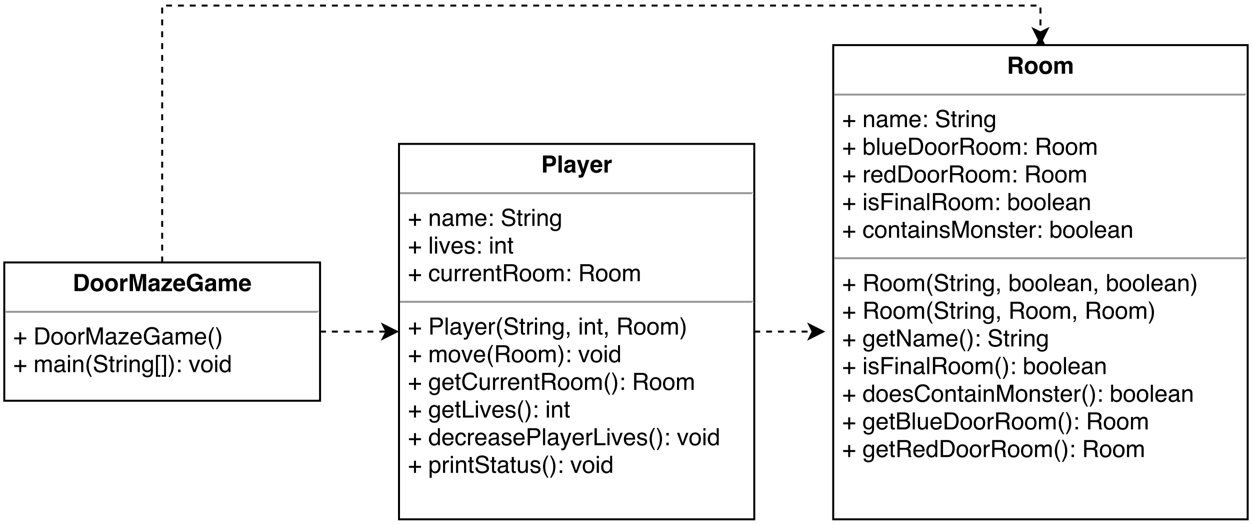

Figure 1: Maze game Class Diagram.

4 Description

1. Class Player
   Class Player stores characteristics of a game player: name, lives and current room the player is located. The method decreasePlayerLives deduct player lives by 1 each time it is called, instead of using original setter method for the lives. This is because player lives are always reduced by 1 in game mechanics.

2. Class Room
   This class represents game rooms. Two most interesting features in this class are two boolean flags: isFinalRoom and containsMonster. These represent whether the current room contains monster or the final room. Note for this assignment alone these two flags are not necessary because comparing two rooms objects directly is possible. i.e. if (player.getCurrentRoom() == room6) is sufficient to check whether the player is in the final room, as each of these objects will reference to the same object in memory, or even better we can use equals() method. However, this method isn't under the scope of current topics being tested and in terms of scalability and better code management having two flags will be better in practice. For example, we might have multiple rooms contain monsters instead of having single dedicated monster room.

3. Class DoorMazeGame
   Our driver class. In this assignment, we handled all adventure part of the game here, but having separate class Map or FloorLevel would have been better to manage more complex multiple room structures. Ideally, as a maze we would have up to 4 doors per room: each room door corresponding north, east, south, west. Depending on which part of the map the room is, it may not have the door to west (at the far left side of the dungeon) or may even only have 1 doors available, the one player used to enter. We could have simulated the different available number of doors by checking the contained room object is not null. Back to our current code, I believe most students utilised breaks to end the current game but I tried to avoid using it as it is a better programming practice to form more careful conditions and avoid using breaks as much as possible. Therefore, the while loop checks for whether two end conditions are not met and run the rest of the loop. Furthermore, the player never actually move inside the monster room. Rather, the code 'peeks' into the chosen room and

if it is a monster room, instead of actually moving the player it only displays some print statements that tricks the player that he is inside the monster room. This way, having a temporary room object to hold previous room to return is no longer needed.