Assignment 6

Samuel Godwin, Computer Science BSc, K1630575

November 16, 2016

1 Introduction

The purpose of this assignment is to create an interactive game in which a user makes choices in an attempt to reach an end goal. This uses a driver class 'DoorMazeGame' as well as a 'Player' class and 'Room' class. 'Room' acts as a complex data type to manage information about the current room a player is in as well as the rooms a player may or may not enter. A player may choose to try and enter through either a blue or red door from a current room, resulting in either entry to another room, or the loss of 1 life from a set number of lives a player has left. If a player reaches the final room, they win.

(Pseudocode on p2).

2 Pseudocode

Room Class

```
CLASS 'Room':
    Create String field 'name' = ""
    Create Room field 'blueDoorRoom'
    Create Room field 'redDoorRoom'
    Create Room field 'containsMonster' = 'false'
    Create Room field 'isFinalRoom' = 'false'
    METHOD 'Room' <name>:
        Set field 'name' to <name>
    END METHOD.
    METHOD 'setBlueDoorRoom' <blueDoorRoom>:
        Set field 'blueDoorRoom' to <blueDoorRoom>
    END METHOD.
    METHOD 'setRedDoorRoom' <redDoorRoom>:
        Set field 'redDoorRoom' to <redDoorRoom>
    END METHOD.
    METHOD 'setContainsMonster' <containsMonster>:
        Set field 'containsMonster' to <containsMonster>
    END METHOD.
    METHOD 'setIsFinalRoom' <isFinalRoom>:
        Set field 'isFinalRoom to <isFinalRoom>
    END METHOD.
    METHOD 'getIsFinalRoom':
        Return isFinalRoom
    END METHOD.
    METHOD 'getBlueDoorRoom':
        Return blueDoorRoom
    END METHOD.
```

```
        METHOD ‘getRedDoorRoom’:
                Return redDoorRoom
        END METHOD.
        METHOD ‘getContainsMonster’:
                Return containsMonster
        END METHOD.
        METHOD ‘getName’:
                Return name
        END METHOD.
END CLASS.
```

Player Class

CLASS ‘Player’:

Create String field ‘name’

Create int field ‘lives’

Create Room field ‘currentRoom’

Create boolean field ‘outcome’

METHOD ‘Player’ <lives, currentRoom>:

Set field ‘lives’ to <lives>

Set field ‘currentRoom’ to <currentRoom>

END METHOD.

METHOD ‘move’ <nextRoom>:

IF result of ‘getContainsMonster’ through <nextRoom> == true THEN:

Set field ‘lives’ = ‘lives’ – 1

Set field ‘outcome’ = false

ELSE:

Set field ‘currentRoom’ to <nextRoom>

Set field ‘outcome’ = true

ENDIF.

Return outcome

END METHOD.

METHOD ‘setCurrentRoom’ <currentRoom>:

Set field ‘currentRoom’ to <currentRoom>

END METHOD.

METHOD ‘setName’ <name>:

Set field ‘name’ to <name>

END METHOD.

METHOD ‘getCurrentRoom’:

Return currentRoom

END METHOD.

```
        METHOD ‘getLives’:
                Return lives
        END METHOD.
        METHOD ‘getName’:
                Return name
        END METHOD.
END CLASS.
```

DoorMazeGame Class

CLASS ‘DoorMazeGame’:

Import ‘Scanner’

METHOD ‘main’:

Create instance of ‘Scanner’ called ‘in’

Create object of ‘Room’ called ‘monsterRoom’<name>,
set name = “The Monster Room”

Go to ‘setContainsMonster’ in ‘monsterRoom’, set containsMonster = true

Create object of ‘Room’ called ‘room1’<name>, set name = “Cave One”

Create object of ‘Room’ called ‘room2’<name>, set name = “Cave Two”

Create object of ‘Room’ called ‘room3’<name>, set name = “Cave Three”

Create object of ‘Room’ called ‘room4’<name>, set name = “Cave Four”

Create object of ‘Room’ called ‘room5’<name>, set name = “Cave Five”

Create object of ‘Room’ called ‘room6’<name>, set name = “Cave Six”

Go to ‘setBlueDoorRoom’ in ‘room1’, set blueDoorRoom = room2

Go to ‘setRedDoorRoom’ in ‘room1’, set redDoorRoom = monsterRoom

Go to ‘setBlueDoorRoom’ in ‘room2’, set blueDoorRoom = monsterRoom

Go to ‘setRedDoorRoom’ in ‘room2’, set redDoorRoom = room3

Go to ‘setBlueDoorRoom’ in ‘room3’, set blueDoorRoom = room4

Go to ‘setRedDoorRoom’ in ‘room3’, set redDoorRoom = monsterRoom

Go to ‘setBlueDoorRoom’ in ‘room4’, set blueDoorRoom = monsterRoom

Go to ‘setRedDoorRoom’ in ‘room4’, set redDoorRoom = room5

Go to ‘setBlueDoorRoom’ in ‘room5’, set blueDoorRoom = room6

Go to ‘setRedDoorRoom’ in ‘room5’, set redDoorRoom = monsterRoom

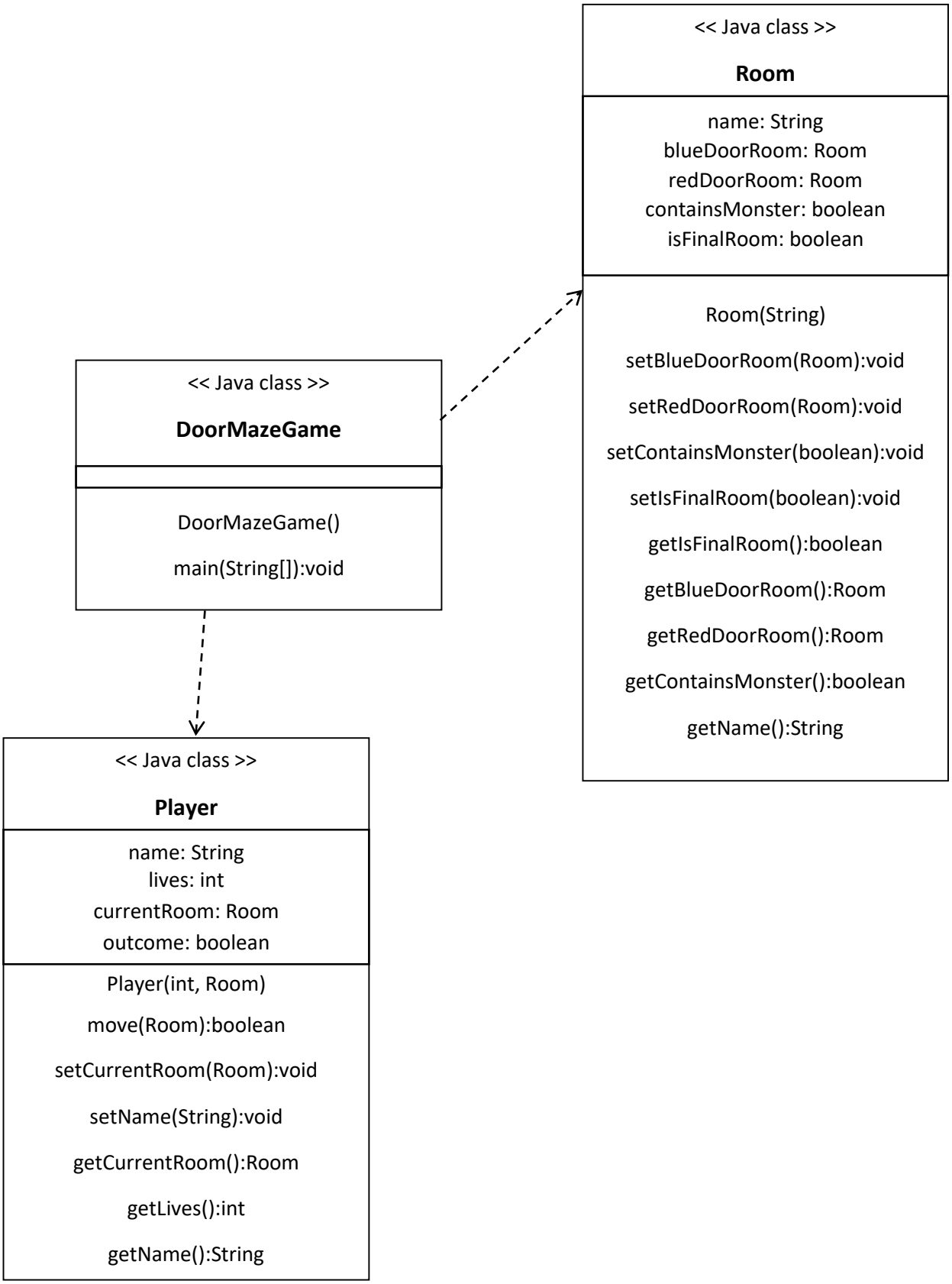
Go to ‘setIsFinalRoom’ in ‘room6’, set isFinalRoom = true

Create object of ‘Player’ called ‘player’<lives, currentRoom>,
set lives = 2
set currentRoom = room1

```
Print “DOOR MAZE GAME. CHOOSE EITHER THE RED OR BLUE DOOR IN EACH ROOM
      TO ATTEMPT TO ENTER THE NEXT ROOM. AVOID MONSTER!”
Print “Enter name: ”
Go to ‘setName’ in ‘player’, set name = String input from user
DO:
      Print result of ‘getName’ and ‘getLives’ in ‘player’
      Print result of ‘getName’ through ‘getCurrentRoom’ in ‘player’
      Print “Please input your choice of door, either 'blue' or 'red':”
      Create String variable ‘nextRoom’, set ‘nextRoom’ = String input from user
      IF ‘nextRoom’ == “red” THEN:
            IF result of ‘getRedDoorRoom’ through ‘getCurrentRoom’
              in ‘player’ == true THEN:
                  Print “Correct choice”
            ELSE:
                  Print “Not correct choice”
            ENDIF.
      ELSE IF ‘nextRoom’ == “blue” THEN:
            IF result of ‘getBlueDoorRoom’ through ‘getCurrentRoom’
              in ‘player’ == true THEN:
                  Print “Correct choice”
            ELSE:
                  Print “Not correct choice”
            ENDIF.
      ELSE:
            Print “You must enter either ‘red’ or ‘blue’ for your choice!”
      ENDIF.
```

```
            WHILE result of 'getIsFinalRoom' through 'getCurrentRoom' in 'player' == false
                AND result of 'getLives' in 'player' != 0.
            ENDWHILE.
            IF result of 'getLives' in 'player' == 0 THEN:
                Print "You have lost all your lives - YOU LOSE."
            ELSE:
                Print "You find yourself in the final room! YOU WIN!"
            ENDIF.
            Call the 'close' method on my 'Scanner' instance
        END METHOD.
END CLASS.
```

3 Class Diagram



4 Description

My solution to this assignment involves the creation of a program in which my driver class 'DoorMazeGame', which contains my 'main' method, creates instances of my two other methods, 'Player' and 'Room'. Instances of my 'Room' class are used in modelling the current status of the door maze game. The current room, and the rooms which are through each of the blue door and the red door in the current room are 'Room' objects. These instances are also used in the making of decisions, using fields. For example, whether or not the room through the door the user chooses contains a monster. This involves the 'containsMonster' field in the 'Room' class. This will result in the loss of 1 life – where 'life' is a field in the 'Player' class.

In my driver class, 'DoorMazeGame' I first import 'Scanner', a class from Java's class library. Inside of my 'main' method, I create an instance of this class, 'in', for use in taking input from the user later in my program. After this, I construct seven instances of 'Room'. I use the 'setContainsMonster' method within 'Room' through the first of these instances, 'monsterRoom', and set the Boolean field 'containsMonster' for this instance to true. The 'name' field of this room is appropriately set to "The Monster Room", a String which I pass as a parameter as the object is constructed. My other six 'Room' instances are 'room1', 'room2', 'room3', 'room4', 'room5' and 'room6', the last of which has its 'isFinalRoom' field, which is otherwise false by default, set to true through the 'setIsFinalRoom' method. The Strings which I pass as parameters for the 'name' fields of these instances are "Cave One", "Cave Two", "Cave Three", "Cave Four", "Cave Five" and "Cave Six" respectively.

After this, I call the two methods 'setBlueDoor' and 'setRedDoor' through each of my six room objects, passing as a parameter the room which is next through that door (from each room). The last object I create of my own classes in my program is the object 'player' of the type 'Player'. As I construct this object I pass the values '2' and 'room1' as parameters, for the maximum lives the player starts with and also the room the player starts in. At this point I print a message to the screen asking the user to enter their name and I use the 'nextLine' method through my 'in' object to take user input of the 'String' data type. I pass this as a parameter to the 'setName' method through the 'player' object. This method subsequently sets the 'name' field of the player object to the name input by the user.

My 'main' method contains a do while loop. I choose this loop as the contents of my loop always run at least once. This loop repeatedly asks the user to input their choice of door, after first presenting the game status to the user - their name, number of lives and the room they are currently in. I again use the 'nextLine' method through my 'in' object to take some user input, however this time I do so for the initiation of a local variable, 'nextRoom'. The 'String' input entered by the user is set as the value of this local variable. An if statement follows, with the condition of whether or not the 'String' input entered by the user (i.e. the door they chose) is equal to "red". My code for this is `nextRoom.equals("red")`. Within this if statement is a nested if statement. This simply uses the value of 'outcome' returned at the end of the 'move' method in the 'Player' class to determine whether or not a player made the correct choice in choosing a door – 'outcome' will equal true if they made the correct decision and 'false' if not. A corresponding message is printed for each outcome. The 'else if' which follows has the same content as the 'if' above, except it concerns the scenario of the blue door being chosen instead of red. If neither of these conditions are true, a message is printed to the screen telling the user that they must enter either "red" or "blue" for the 'String' value when choosing a door. This message is only displayed in the scenario that the user tried to enter input which was neither of these options.

The above paragraph is the contents of my do while loop, and the while condition follows. This condition is first whether the result of the 'getIsFinalRoom' method through the 'currentRoom' for 'player' is false, as well as the result of the 'getLives' accessor method for 'player' being a value which is not 0. The two are connected by a logic OR operator (&&). What this condition essentially denotes is while it is not the end of the game, i.e. the player has not yet reached the final room and also still has lives remaining. This ensures repetition of the loop until the game ends, i.e. until the player either reaches the final room, or has 0 lives. This mechanic plays a part in ensuring that when the player's live reach 0, they lose the game as after the end of the do while loop, I use an if statement to confirm the cause of the end of the game. If the 'player' object's 'lives' field is 0, then the player ran out of lives and they did not reach the final room – thus, they lose the game. Otherwise, the player wins as these two occurrences are the only ways in which a game can end. To confirm to the user that they have either lost or won the game, a print message appears, notifying them. Compiling this code will result in a warning unless I call the close method on my Scanner instance, so I do so.

My 'Room' class contains information about a room in fields, as discussed earlier. It also contains a series of accessor and mutator methods, for relevant fields. My 'Player' class is similar, for each of its fields, however contains an additional method 'move' which, as mentioned earlier, returns a Boolean value 'outcome' which is used for conditions in my driver class. This method simply removes 1 life if the room corresponding to the door the user chose contains a monster, or updates the room the player is in if not. I do this using an if statement, first for the condition of whether the result of 'getContainsMonster' is true through 'nextRoom' - which is a parameter passed from the driver class and contains the contents of either 'redDoorRoom' or 'blueDoorRoom' for a 'Room' object. Since 'nextRoom' corresponds to the door a user chose, this method ultimately determines the consequence of a user's actions. If the room the user tried to enter contains a monster, they are informed and the 'lives' field through the player object is decreased by 1. On top of this, the 'outcome' field is set to false. Otherwise, the room the user tried to enter does not contain a monster, so was the correct door to enter. The user is informed via a message and the 'currentRoom' field for 'player' is updated to equal the 'Room' object through the door the user tried to enter (and succeeded in doing so). Accordingly, the value of 'outcome' is set to true for use in my driver class. At the end of this method, 'outcome' is returned.

Initially, the 'move' method in my 'Player' class used the labelling of 'currentRoom' for its parameter but, since the context of this parameter is of the *next* room the user wishes to enter, it did not seem appropriate. Subsequently, I used 'nextRoom' instead. Other changes I made during the development of my program were similarly minor. The initial implementation of my 'Player' class in my code included an accessor method for 'outcome' – 'getOutcome'. However, upon realising that this method was not necessary in my code, since the value of 'outcome' is returned at the end of the 'move' method, which I use in my driver class instead, I removed this method.

Something which I struggled to implement during the development of my program was the placement of my use of mutator methods through objects, in my driver class (such as 'setRedDoorRoom' through 'room1'). Initially, each of these methods were called immediately after creation of a room object, for that room object. However, since the parameters passed would at this point consist of objects not created yet, i.e. 'room2', this meant that my code would not compile. To solve this problem, I placed each of my uses of mutator methods through 'Room' objects *after* the creation of all the 'Room' objects, collectively.