CS350 Software Design/SE310 Software Architecture I

Lab 4: Make an Undoable Maze Game

Goal:

Practice Command Pattern

Requirements:

1. In this lab, you need to apply a Command Pattern to the maze game so that moves in the maze are undoable. This lab is an extension of the maze game with Abstract Factory pattern that you have implemented in lab 3.
2. The new program should accept all the input parameters as in Lab 3, and each movement can be undone by typing "u".

Instructions

Note, a new maze-gui Jar file is provided (in BBLearning). We changed the GUI part slightly to accept and respond to the "u" key press.

1. Copy and paste the prototype maze game you created in lab 3 to a new working directory
2. Clean the bin directory
3. Replace the original maze-ui.jar with the given new jar file.
4. Create the following classes/interfaces according to the Command pattern UML class diagram (you will need to create all the methods and attributes as shown in the UML diagram):
    a. The Command interface with one method.
    b. The UndoableCommand interface that extends the Command interface with an undo() method
    c. The MazeMoveCommand class that implements the UndoableCommand interface.

    Hint: this class wraps a move method into a Command. A *maze* object and a direction are passed to the MazeMoveCommand constructor as parameters.

    You need to:

    (1) Create a **private** move method as follows (we provide this class to avoid the difficulty caused by Java syntax):

```
private void move(Direction dir)
{
        Room curRoom = maze.getCurrentRoom();
        MapSite side = curRoom.getSide(dir);
        side.enter();
        if (side instanceof Room)
                maze.setCurrentRoom((Room)side);
```

```
            else if (side instanceof Door) {
                    maze.setCurrentRoom(((Door)side).getOtherSide(curRoom));
                    maze.getCurrentRoom().enter();
            }
    }
```
(2) implement the execute() method, that is, move to the direction passed to this command

(3) implement the undo() method, that is, move to the opposite of the direction passed to this command

5. Complete the new MazeMover class skeleton.

   Note:

   a. **We provide a MazeMover class skeleton because it interacts with the user interface package by implementing the MazeMoveListener interface. By completing the skeleton, you are supposed to practice how to create Commands, store them into a stack and pop them out when moves need to be undone. The skeleton is provided in BbLearning, please download it to your working directory.**

   b. **DO NOT change the method names within MazeMover, which are used by the UI package. Just complete each method.**

6. Run the program using exactly the same way as required in Lab 3.

Lessons: how design principles are followed:

1. **Open-close principle: without considering the UI part, we completely followed the open-close principle in the Command pattern: nothing needs to be changed in the original Abstract Factory pattern maze game.**
2. **Separation of Concern: the MazeMoveCommand class only concerns about wrapping a moving behavior into a Command, and the MazeMover only cares about managing the stack when a move is issued from the UI.**

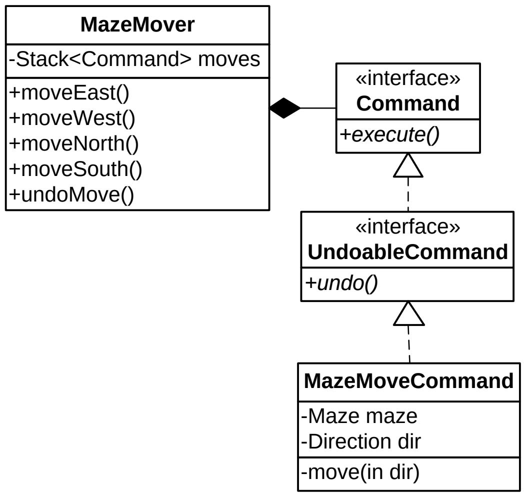

Figure 1: Command Pattern

- Note: you should improve your lab 3 to follow these principles before you start lab 4.
- Write one paragraph to explain what you have done for improvement, if any.