CSC 317: Project 3

Solving Puzzles – I

The Eight Puzzle and a Human Player

Total: 100 points

In this project you will start building a generic puzzle suite, particularly implement the eightpuzzle (a sliding tile puzzle) and allow a human player to solve it. In subsequent projects, you will implement other puzzles and other (automated) solvers/players, with the ability to pair any kind of player with any of the puzzles with minimal code change in the main function. Therefore, the project should be designed, and your code organized, with that **future extension in mind**. Since how you choose to design and organize your code will be a big part of your grade, I will give you less instruction about it, unlike the previous projects. But consider what uniform interfaces an arbitrary puzzle and a solver might need to satisfy, and code to those interfaces. Furthermore, players and puzzles could be separate packages. The main requirements are listed below:

1. You should *not* generate the initial tile configuration randomly, since some configurations have no solution. Instead, your code should be able to read the initial tile configuration from a text file as input. The following is a sample of the input format:

```
-------------
| 1 |   | 5 |
-------------
| 3 | 2 | 8 |
-------------
| 6 | 4 | 7 |
-------------
```

2. In a loop, your code should display the current tile configuration, ask the (human) player for the next move, and execute the player's move, thus changing the tile configuration.
3. A “move” is supposed to slide a tile adjacent to the blank cell, into it. Another way to view a move is to consider the blank cell as being moved (in one of the four cardinal directions: up, left, right, or down, as available) to exchange location with an adjacent tile. So, rather than sliding the tile 2 up, think of the move as sliding the blank cell down (thus moving 2 up). You may find this view of a “move” easier to code. This view is also used in the sample run shown later.
4. The game ends when the tile configuration satisfies the following criteria:

   a) The tiles are in the correct numeric order

b) The blank cell is either at the top-left corner, or at the bottom right corner. That is,

```
-------------                              -------------
|   | 1 | 2 |                              | 1 | 2 | 3 |
-------------                              -------------
| 3 | 4 | 5 |                              | 4 | 5 | 6 |
-------------                              -------------
| 6 | 7 | 8 |                              | 7 | 8 |   |
-------------                              -------------
```

OR

As an example, the player can reach the goal configuration on the left, starting from the initial configuration as shown above, by executing the following sequence of moves on the blank cell: 'down', 'down', 'right', 'up', 'up', 'left', 'left'.

5. In this project, your code does not have to figure out the solution move sequence. Instead, just let a human player figure it and play it out. However, your code must recognize a goal configuration when reached, and terminate the game.
6. (**5 points extra credit**) Make a more general (n^2-1)-puzzle, i.e., 8-, or 15-, or 24-puzzle, etc., where *n* is also read as an input from the user at the beginning.
7. (**20 points extra credit**) Build a GUI where the human player can interact via the mouse, instead of the text-based interaction as outlined below. Then, clicking on a tile adjacent to the blank cell will exchange its location with the blank cell. This may be a good opportunity to learn awt, swing, javafx. However, if you are just starting to learn Java in this course, then this may be overambitious.

A Sample Run

```
Enter the file name for the initial configuration: eightpuzzle_1.txt
Current configuration:
-------------
| 3 | 1 | 2 |
-------------
| 4 | 7 | 5 |
-------------
| 6 |   | 8 |
-------------
Please select your next move from the following choices:
   1. Up
   2. Left
   3. Right
Your selection: 1

Current configuration:
```

```
-------------
| 3 | 1 | 2 |
-------------
| 4 |   | 5 |
-------------
| 6 | 7 | 8 |
-------------
Please select your next move from the following choices:
   1. Up
   2. Left
   3. Right
   4. Down
Your selection: 2

Current configuration:
-------------
| 3 | 1 | 2 |
-------------
|   | 4 | 5 |
-------------
| 6 | 7 | 8 |
-------------
Please select your next move from the following choices:
   1. Up
   2. Right
   3. Down
Your selection: 1

Current configuration:
-------------
|   | 1 | 2 |
-------------
| 3 | 4 | 5 |
-------------
| 6 | 7 | 8 |
-------------
Great! You have reached the goal in 3 moves. Bye.
```

What to submit

Submit the following 3 files (**all required separately**) on Canvas:

1. A .pdf file showing the UML class diagram for this project.
2. A .zip file containing the source code (.java files, possibly organized in subdirectory structure).
3. Runnable .jar file of the project. I should be able to run the .jar from command line like so:
   `java –jar P3.jar`