N-Puzzle

Write a Java program that simulates the famous game N-Puzzle If you do not know what a it is, search it on google.

The rules are simple. A field of numbers 1-8 is generated by the computer and it leaves the ninth spot as blank. The user has to move the numbers around to arrange the numbers in the correct order. For this program, you do not need to know how to use the computer to calculate the best move, but you just need to implement the game for user to enjoy.

Copy the code that I sent to you into the N-Puzzle file that is already in your APCS project(the one we used in class)

Now, I will explain each method you need to implement and the method I have already implemented.

NPuzzle.java

This file will contains a few constants that you should use to help you finish the game.

REMEMBER TO CHECK THE SAMPLE RUN AT THE END.

The constants that I implemented for you are

BOARD_SIZE

This is the size of the board. In the method `initialize()`, you will initialize the board with this size.

finalBoard

This is the correct arrangement for the board. The user should try to manipulate the board to make the board exactly the same as this one.

board

The board the user will manipulate. Remember to initialize it in the initialize() method.

kb

This is the scanner that you will use to prompt the user input. If you forgot how to use it, check the previous project. If you have further questions about his, ask me. Note that I have already initialized it, so you will just use it directly.

UP DOWN LEFT RIGHT

These are the constants that corresponds to w/s/a/d. Use these properly to help you implement the methods.

Here are the ones you need to implement:

public static void initialize()

This method will initialize the board with the size specified. Then it fills the number 1-8(as String). One of the spots should be blank. You can do whatever you want to randomize the board. Make sure you don't have duplicate numbers in different stop. One thing to note is that the blank could be anywhere on the board. Your implementation **should NOT** fix the blank spot at, for example, the bottom right corner.

public static boolean checkInput(String input)

This method checks if the input is one character, if so, it checks if the character is w or a or s or d. Then it needs to check whether that movement is valid. For example, for a board like this, "s" or "a" should be invalid since there is no number on top or right side of the blank spot to move down or left. Regarding the movement, if you are confused, you can check the sample output or ask me.

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

public static String getInput()

This method will get the user input. Once it passes the checkInput method, you should return the input.

public static void processInput(String input)

Use the value of the input String to manipulate the board. You can assume that by this point, the input is valid, and you just have to move the number according to its value. A sample movement is like this:

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

Please enter your move(w/a/s/d): w

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

Please enter your move(w/a/s/d):
```

public static boolean isOver()

In this method, you will need to check if the board is the same as the finalBoard constant. If they are identical, you should return true as the user has completed the game.

Here are the ones I have already implemented. Do not modify them!

public static void main(String args[])

This is the main loop of the game. It calls the methods that you wrote to run the game.

public static void displayBoard()

This method will display the board in a good format.

Sample Run

(Yours may not look exactly the same since the numbers are at different positions each run. Again, user inputs are underlined)

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

Please enter your move(w/a/s/d): w

Invalid input! Try again
Please enter your move(w/a/s/d): s

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

Please enter your move(w/a/s/d): d

+---+---+---+
| 1 | 5 | 8 |
+---+---+---+
|   | 4 | 3 |
+---+---+---+
| 6 | 7 | 2 |
+---+---+---+
Please enter your move(w/a/s/d): a

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

Please enter your move(w/a/s/d): a

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

Please enter your move(w/a/s/d): a

Invalid input! Try again
Please enter your move(w/a/s/d): w

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

Please enter your move(w/a/s/d): w

Invalid input! Try again
Please enter your move(w/a/s/d): w

Invalid input! Try again
Please enter your move(w/a/s/d): a

Invalid input! Try again
```

```
Please enter your move(w/a/s/d):

.
. (Infinite Trials Later… TT)
.

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

****************************************
*  Congratulations! You have solved it! *
****************************************
    It took you 9999999999 steps.
```