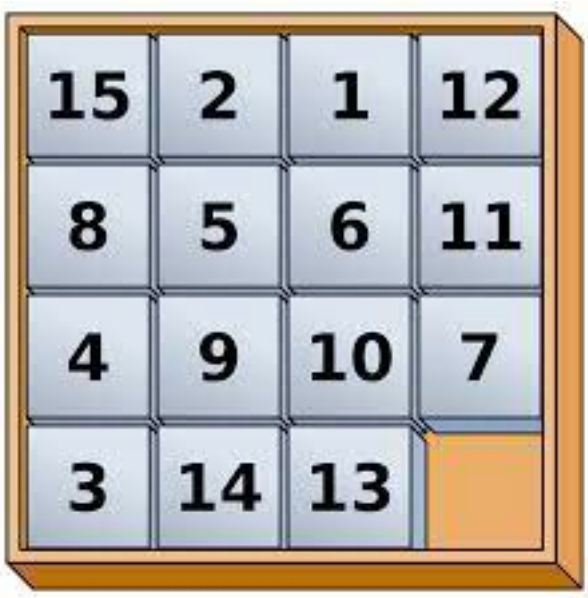
CS 342 **Project#4** **Spring 2021**

Threaded AI 15 Puzzle

Due Date: Sunday, May 2nd @11:59pm

You may submit up to 24 hours late for a 10% penalty
There will be no further extensions or considerations for this project



Description:

In this project, you will create a JavaFX program that allows the user to attempt to solve different 15 puzzles (more on that below). Your program must have a minimum of 10 unique 15 puzzles to solve. If the user can not solve it and wants to see the solution, animated move by move, they can choose to have the AI puzzle solver figure it out with one of two heuristics.
The AI algorithm the program will use is called A* and will be able to use one of two heuristics:

https://en.wikipedia.org/wiki/A*_search_algorithm

You will be given a working version of this algorithm and will have to figure out how to use it in your program.
All threading must be done using the Executors class in Java. This project will be developed as a single Maven project. Use the Maven template and A* java code provided to develop your program.

You may work in groups of two but are not required to.

CS 342 Project#4 Spring 2021

How the program works:

The basic idea of the 15 puzzle is that numbered tiles can be swapped with the blank spot one at a time. After a series of these swaps, a solved puzzle should have all of the tiles in order from lowest number to highest number with the blank spot in the upper left corner. In the algorithm given to you, the zero represents the blank spot.

The game will start with a welcome screen that is displayed for a few seconds and then goes to the game play screen. An unsolved 15 puzzle will be in the center of the graphical interface. There will also be buttons or a menu that allow for a new puzzle to be displayed, solve with AI H1, solve with AI H2, exit the program, and see the solution displayed.

The user can make moves to attempt to solve the puzzle. If they solve it, they will be congratulated and told to either select a new puzzle to solve or exit the program.
If they want one of the AI heuristics to solve the puzzle for them, they can click on AI H1 or AI H2. This will start the DB_solver. When the solution to the puzzle is available, the "see the solution" button will be enabled. When clicked the puzzle will be solved move by move, for the next 10 moves, with short pauses in between so the user can see the moves. If the puzzle is solved in 10 moves or less, the user will be prompted to either select a new puzzle to solve or exit the program. If not solved yet, the user will keep trying to solve it. Why just 10 moves? If you entered in this puzzle:

0 14 13 12 15 9 5 8 11 7 4 1 3 10 6 2

And ran the A* with the second heuristic, there would be 234 moves to the solution with the algorithm examining 85,229 unique configurations of the board.

Implementation Details:

When the user selects to have the puzzle solved and hits one of the two heuristic buttons, the AI solver should run on its own thread and return the solution path to the Application thread to be animated move by move. The Application thread should not be blocked while waiting for the solution path to return from the AI solver. **There will be a 25% penalty for any solution that blocks the Application thread.**

You are free to change, reconfigure, comment out parts of, copy and paste parts into other classes of your creation, the AI code as needed to implement it in your program. However, you must use the files given; you can not discard them and use a different solution found elsewhere.

The A* Algorithm:

When you compile and exec:java the Maven project provided, you will get a blank JavaFX window but also a prompt to “Enter in your puzzle as a string with a space between each number” in the terminal window. In this version of the algorithm, the blank space is the zero. If you entered this string of numbers:

2 6 10 3 1 4 7 11 8 5 9 15 12 13 14 0

You will see this output:

```
This is the puzzle you entered
2 6 10 3 1 4 7 11 8 5 9 15 12 13 14 0


Starting A* Search with heuristic #1....This may take a while


the size of the queue and hash: 889 1718

******Run Time for A* heuristicOne is: 24 milliseconds**********


**************Initial State******************
  2   6  10   3
  1   4   7  11
  8   5   9  15
 12  13  14   0
Next State => 0
```

This means that A* , using the first heuristic, has completed and there is a solution. What you will see next is each move to the solution displayed individually, here is the last few:

```
Next State => 35

  4   1   2   3
  5   0   6   7
  8   9  10  11
 12  13  14  15
Next State => 36
```

```
  4   1   2   3
  0   5   6   7
  8   9  10  11
 12  13  14  15
Next State => 37

  0   1   2   3
  4   5   6   7
  8   9  10  11
 12  13  14  15
**********Goal state**********
```

You will then see a similar print out when A* , with the second heuristic, completes. The solution path is returned as an ArrayList of Nodes from the DB_Solver2 class. Each node contains an Array that represents the whole board after one move.

Testing Code:

You are required to include at least 10 JUnit 5 test cases for your program. You should test the AI code given to you. Add these to the src/test/java directory of your Maven Project.

UI and UX design:

You are required to use the best practices we discussed in lecture for designing your programs user interface. The user should know what is happening and what to do at all times. Your interface should be intuitive and not cluttered. **There will be points deducted if the TA testing your program does not know how to use your program.**

Electronic Submission:

Zip your Maven project and name it with your netid + Project4: for example, I would have a submission called mhalle5Project4.zip, and submit it to the link on Blackboard course website.

Assignment Details:

We will test all projects on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot*

work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between

students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.