Project 1

Due date: Feb 11, 2020, 9:00am

Objective

- Select suitable linear data structures to implement solutions to implement the undo operation and a search problem.
- Implement a search algorithm, like BFS or DFS.
- Design and implement logic expressions to check for given conditions.

Project Overview

The starter project is an JavaFX project. The given version allows you to play the 8-puzzle. You can click a tile to move the tile to the empty position. Once all tiles have been placed in sorted order, where the lower right corner is empty, the puzzle is solved and you can restart with a new randomly generated 8-puzzle.
The project follows the MVVM design pattern. You do not have to work with the view part of this project. The main tasks require implementing the functionality of the three buttons “undo”, “help”, and “solve” in this project.

Important Requirement

One objective of this project is to identify linear data structures. Therefore, if you declare a collection and the collection is used only as a stack, queue, or deque, you need to declare the collection correspondingly using the available Java type Stack, Queue, and Deque respectively. If you do not declare collections accordingly, I will not be able to assess whether you understand the use of stacks, queues, and deques and you will loose points.

Do not make any changes to the files in the `view` packages.

Getting Started

1. Download the starter project `CS3151Project1Starter` from the course website and import it into Eclipse.
2. Rename the project tot `CS3151Project1`*`FirstNameLastName`*. Where *`FirstNameLastName`* is your first and last name.
3. Since the starter project is a JavaFX project, you need to set the VM Arguments in the Runtime Configurations:
   --module-path "${eclipse_home}/javafx-sdk-11.0.2/lib" --add-modules javafx.controls,javafx.fxml
4. Run the starter project. In the starter project,
   - you can move any time to the empty position.
   - if the tiles are sorted, a message is shown and a new puzzle can be started
   - the button functionality for “Start new puzzle” is implemented only.

Task 1: Limit Tile Movements

Currently, any tile can be clicked and will be moved to the empty position. However, in the 8-puzzle only a tile that is adjacent to the empty position can be moved. In order to fix this issue, update method `Board::moveTile(Position src, Position dest)` in the `model` package to meet its specification.

Task 2: Equal Positions

Implement method `Position::equals` in the `model` package to meet its specification.

Task 3: The Undo Operation

If the undo button is clicked, the message “Replace me by instructions to undo the most recent move” is printed to the console by method `PuzzleViewModel::undo` in the `viewmodel` package. Modify the method so that the most recent operation is undone. It should be possible to keep clicking the undo button to undo all moves one-by-one that have been made since the start of the most recent 8-puzzle game. Make changes to class `PuzzleViewModel` as necessary.

If the undo button is clicked before any move has been made, nothing should happen. If a new puzzle is started, moves made in the previous puzzle are cleared and cannot be undone in the new puzzle. In other words, nothing should happen if the undo button is clicked right after starting a new puzzle.

Task 4: The Help and Solve Buttons

The functionality for the help and move button can be implemented by using the same algorithm and therefore should be tackled together.

Task 4: The Help Button Functionality

If the help button is clicked, the message “Replace me by instructions to set the next tile at the correct position” is printed to the console by method `PuzzleViewModel::help`. You should implement functionality that will determine the shortest sequence of moves to move the next tile that is out of order to the correct position. Then method `PuzzleViewModel::help` should call `PuzzleViewModel::traceMoves`, passing in the corresponding sequence of moves. Method `traceMoves` will execute the moves one-by-one.

Task 4: The Solve Button Functionality

If the solve button is clicked, the message “Replace me by instructions to solve the puzzle” is printed to the console by method `PuzzleViewModel::solve`. You should implement functionality that will determine the shortest sequence of moves to solve the 8-puzzle. Then method `PuzzleViewModel::solve` should call `PuzzleViewModel::traceMoves`, passing in the corresponding sequence of moves.

Task 4: Class PuzzleSolver

Add a new class, called `PuzzleSolver`, to the model package, which should provide the help and solve button functionality. In particular, class `PuzzleSolver` should be responsible for finding the sequence of moves with the least number of moves that solves a given 8-puzzle. It should also allow for finding the sequence of moves with the least number of moves in a given 8-puzzle that moves the next tile that is out of order to its correct position.

Task 4: Method PuzzleViewModel::traceMoves

Method traceMoves will perform the passed in move operations on the puzzle board. The Timeline class is used to perform each move after a 0.5 second delay such that the moves can be observed by the user in the view. It should be possible to undo a move that is made as a result of the help or solve button usage. Add code to the method such that each move performed by the method can be undone. Otherwise, do not modify the method.

Note: You should apply a suitable search algorithm covered in class to solve the puzzle. It is possible that the algorithm may not terminate in reasonable time or may run out of memory. That is okay for this project. For testing purpose, apply the solve functionality to a partially solved puzzle, e.g. the first three tiles are already in correct order.

Submission

Submit a zipped solution to the course website before the due date on Feb 10, 9am.

Grading

Task 1: 8 points

Task 2: 8 points

Task 3: 14 points

Task 4: 70 points