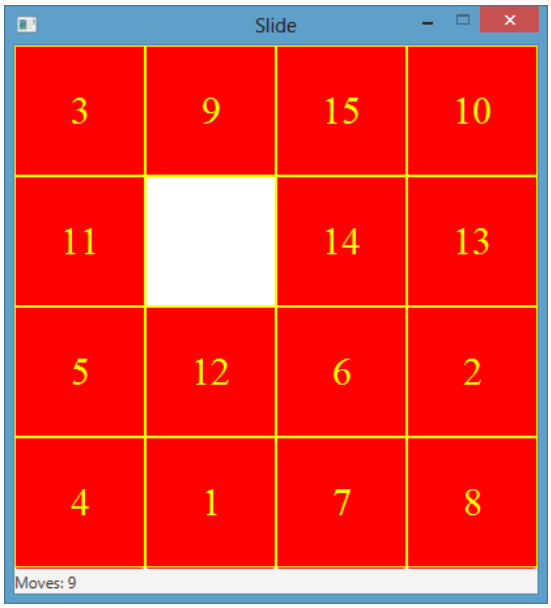
W. P. CAREY SCHOOL of BUSINESS
ARIZONA STATE UNIVERSITY

CIS 494 – Business Systems Development with Java – Spring 2015

Project: Sliding Puzzle

Due on Blackboard: Tuesday, May 5, by 11:59 PM

Game Demo: Final Exam Day, Wednesday, May 6, 12:10 - 2:00 PM



There are various kinds of sliding puzzles that aim to present users with a scrambled pattern to be solved by sliding pieces in a specified direction. The aim in these games is to rearrange the pieces into the target pattern in the fewest moves possible.

For the project, create a basic two-dimensional sliding number puzzle, which follows these rules:

- A rectangular block of size 4 * 4 is needed as the puzzle board.
- The puzzle board has a total of (n*n – 1) tiles.
- Tiles that are adjacent to the blank can move into the blank's space.
- Tiles can move only horizontally or vertically.
- The tiles are numbered sequentially from 1 to (n*n-1).
- The tile faces or values are randomized at game startup.

Monitor and display the total number of moves the player makes!

Sample Working Application

Play a game of 15 Puzzle at http://mypuzzle.org/sliding to see how your game should work. Select a 4x4 15 Puzzle from the combo box on the right.

Solving the Implementation Problem

This puzzle application requires basic JavaFX understanding. It would probably be a good idea to write the JavaFX code yourself rather than use FXML documents since writing code gives you flexibility to create a custom-sized puzzle. There is nothing very sophisticated in the JavaFX portion of this application – you can use any control to represent the tiles. The application in screenshot on the first page uses Buttons.

There are three primary challenges in this application.

First, you need to develop a model of the logical information that you need to maintain. You could, of course, maintain a two-dimensional array of Buttons, but do you need to store more information to play the game? If so, you will need to create objects to correspond with those Buttons. *You will have the GUI elements been seen by the user but you should also have a logical model that stores information about the state of the game.*

Second, the biggest challenge is developing the logic that determines when a tile can be moved. If you know which tile is blank and which tile has been clicked, you can visually and mentally decide if that tile can be moved or not. You need to transfer that logic into code.

Third: how will you "move" the tiles? In my sample application, the tiles don't really move. Only their values do! Feel free to go with whichever option suits you!

Deliverables

- Working JavaFX application uploaded to Blackboard as a ZIP file
- Demo of working game on Final Exam Day.

Optional Enhancements

If you wish, enhance your gaming application by implementing one or more of the following:

- Find square pictures split into 16 squares and put them on the face of the tile (as backgrounds) – instead of a number puzzle, make it into a picture puzzle.
- Let the user choose if she would like to play a number puzzle or a picture puzzle.
- Allow the user to reset or restart a new game.
- Allow the user to select the size of the puzzle, e.g., 3x3 or 4x4.
- Build in a timer.

Getting Started and Getting Help

Developing game logic requires a serious and detailed study of the rules of the game. This requires time as it is not something that has one set solution that can be discovered. Sometimes it does need trial and error.

Get started quickly on first creating the GUI with the Buttons. Then try to enhance it so that you can get randomized numbers to show up. Then try to implement the logic to simulate and/or process moves.

Put in early effort to create the GUI and feel free to come talk to me about your ideas in modelling your game as well as implementing move logic. Do not delay this project! It requires creative energy that will be hard to pull off under stress near finals! Make sure you are leaving enough time to get help.

Developing games ought to be just as fun as playing them – don’t plan to do this late under stress!

PROJECT ORIGINALITY

Any work submitted must be original and represent your own effort. Original code means that it was conceptualized, designed, and written by you. You must not submit code that has not been written by you or your team members or that you cannot together explain conceptually and cannot replicate again. As per the syllabus, you can be asked to explain any portion of the code you submit for credit.

You may be asked to explain code briefly during the final exam session and may be asked to come in later and explain/rewrite/fix code in detail in the presence of a W. P. Carey staff or faculty member. If you are not available to come in, you will receive a grade of Incomplete (I), which will be changed only after you come in.

Project originality will also be checked using Artificial Intelligence software pattern detection software. This software is able to detect duplicate code despite name changes, variable changes, dead code, blank lines, new lines, or any other device that makes the code look different but still behave in the same manner. If your project is flagged, you may be called in to explain/rewrite/fix code in the presence of a W. P. Carey staff or faculty member as detailed above. Your project will also be forwarded to the W. P. Carey Academic Affairs office.

If you are not able to sufficiently explain or replicate the code that you submitted as your own work, a grade of zero can be given for the project and academic dishonesty proceedings may be initiated resulting in harsher penalties, such as a grade of E or XE, as stated in the syllabus and ASU policy documents. XE grades are assigned for failure due to academic dishonesty and may remain on student transcripts permanently.

All duplicate projects will be considered as instances of cheating – both “original” and “copied” projects may receive the same penalty.

Academic Integrity violations will be documented with W. P. Carey’s Academic Affairs Office. The College will pursue additional actions and penalties above and beyond class penalties such as removal from the major and from the W. P. Carey School of Business. After you finish this class, your project will still be run against projects submitted in the future. If a match is found, it will be reported and penalties will still apply.

Summary: Don’t submit code you haven’t thought of and written on your own. Don’t give your code to others. Writing code thought of by and dictated by somebody else does not comprise producing original work.