Advanced Programming: Main design choices of 8 Puzzle

Domenico Ferraro
d.ferraro7@studenti.unipi.it
559813

February 14, 2024

Contents

1 EightTile

The EightTile class encapsulates the functionality of individual tiles within the 8 Puzzle game. It ensures that tiles respond correctly to user interactions, prevent illegal moves, and provide visual feedback to enhance the gaming experience. Moreover, the chosen design focuses on ensuring that the EightTile's bean can be correctly shared and used in other projects.

The EightTile class provides two constructors, one without parameters and one with a position parameter. Since the position is constant, the one without parameters is only available to accommodate Java Beans' requirements. When a tile is built, it sets up an action listener to handle the tile clicks event. This design is very important since the click behaviour must be preserved. Having the click behaviour must not be the responsibility of the bean's user, and the same is true for the tile's text: since the tile's text must change according to specific behaviours the bean's user must not be allowed to remove that behaviour. This design allows the EighTile bean to be used in other projects preserving its main behaviours and characteristics and allowing the bean's user to use the EightTile without the need for any specific configuration.

This class extends JButton and implements custom UI rendering using MetalButtonUI to enhance the appearance of tiles. It refreshes the UI based on the tile's label and position, for example by updating the background color, text, and cursor based on the tile's state.

It implements the PropertyChangeListener interface to handle property change events, particularly for restart events triggered by the EightBoard class. When that happens, it updates the tile label when the board layout changes, which in turn triggers a UI refresh.

When a tile is clicked, it fires vetoable change events, preventing illegal moves. It updates the tile label and UI when a move is valid or invalid, in particular, it flashes the tile background red briefly to indicate an invalid move by using a Timer object. The object is reused to ensure that only one concurrent object applies the flash behaviour.

2 EightController

The EightController class effectively manages the game logic, handles user interactions, and ensures that tile movements and operations are performed according to the rules of the 8 Puzzle game. It maintains a clear separation of concerns, making it easy to understand and maintain.

The layout of tiles is stored in this class as a map that maps each tile label into the position of the tile that has that label. Any change in the layout made by this class always fires layout change events attaching a matrix of changes. In particular, the matrix has one row for each tile that changed and each row is a pair consisting of the position of the changed tile followed by the new tile label. This design was chosen to ensure that only the changed data is passed through event-based communication. The responsibility of creating the layout is left to the EightBoard class.

To validate whether moves are legal or not, the EightController class computes the Manhattan Distance from the hole tile to the clicked tile, ensuring validation in *O(1)* time. In particular, the Manhattan Distance from the hole tile to itself is zero. The Manhattan Distance from the tiles above, below and on the sides of the hole tile to the hole tile is one, while the Manhattan Distance from the hole tile to any other tile is higher but not equal to one. When a tile becomes the new hole it will change its label since the event is not vetoed by the Controller. However, the controller will apply the changes to its internal layout (the labels of the hole and the clicked tile are swapped) and it will fire a layout change event. The EightBoard, which listens for layout change events, will have the responsibility to change the old hole tile's label.

The EightController class listens for restart events since it stores the layout and restarting the game produces a new layout. When a restart occurs, the new layout is received as attached to the event and then the internal layout is updated.

2.1 Implementation of flipping

The flip behaviour is implemented by this class: when the Flip button is clicked by the user, EightController's `flip()` method is called. If the hole is in position 9, EightController will change the layout by swapping the tiles in the first and second positions. Because the layout is changed, the changes are propagated through event-based communication by triggering a layout change event. The EightBoard, which listens for layout change events, will have the responsibility of changing the tiles' labels.

3 EightBoard

The EightBoard class extends JFrame, and the GUI consists of a 3x3 grid layout representing the game board, with each cell containing an EightTile object. There are also buttons for restarting the game and flipping tiles.

The class implements the PropertyChangeListener interface to handle layout change events. It listens for changes in the layout of the tiles, updating their labels accordingly by calling the tile's `setTileLabel()` method.

The constructor initializes the GUI components, registers event listeners, and initiates the game by calling the `restart()` method. It registers the EightController as a vetoable change listener for each tile and as a listener for restart events. It registers each tile as a listener for restart events.

Restarting the game generates a new layout (random permutation of numbers 1 to 9) and fires a property change event to notify listeners (the EightController) about the restart.

The GUI provides buttons for restarting the game and flipping tiles. Clicking the "RESTART" button calls the `restart()` method. Clicking the "FLIP" button invokes the `flip()` method of the EightController. Last but not least, this class specifies the `main()` method which sets the look and feel of the GUI (the preferred one is the "*Metal*" look and feel).