Advanced Programming Assignment 1
Exercise 1 - The 8 puzzle

Francesco Benocci 602495

May 2024

1 Introduction

The assignment involves implementing the 8 puzzle game using Java Beans, providing a graphical user interface with interactive features. This document outlines the design decisions made to fulfill the assignment requirements.

The final UI at the game start is showed in the Figure 1. In the Figure 2 there is the solved puzzle.

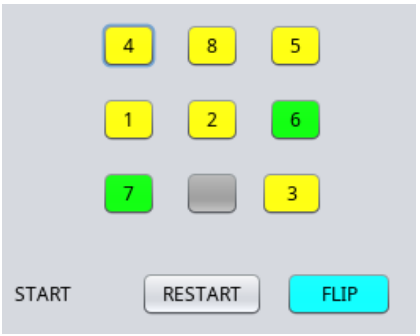

Figure 1: Not ordered puzzle.

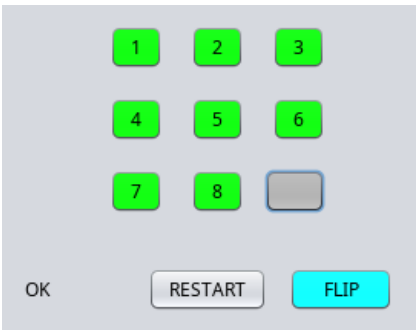

Figure 2: Solved puzzle.

The key components of the application are:

- **EightController**: that handles the game logic, including tile movements and state validation.
- **EightTile**: that represents each individual tile in the game, managing its appearance and behavior.
- **EightBoard**: that would typically manage the overall layout of the tiles and interact with the controller, though its implementation is inferred rather than provided.

2 EightController bean

The `EightController` class extends `JLabel` and implements `Serializable`, `PropertyChangeListener`, and `VetoableChangeListener`. It manages the state of the game board and enforces the game rules.

The board is initialized with an `ArrayList` of integers representing the tiles. The empty space is denoted by the number 9. The class have a method `isAdjacent(int label)` that checks if a tile is adjacent to the empty space, allowing for a valid move, a method `updateBoard(int firstLabel, int secondLabel)` that updates the board after a successful move.

The `vetoableChange(PropertyChangeEvent pce)` method handles swap and flip events, ensuring moves are valid before updating the board. The class uses `PropertyChangeSupport` to notify listeners about state changes.

3 EightTile bean

The `EightTile` class extends `JButton` and implements `Serializable` and `PropertyChangeListener`. It represents individual tiles on the game board. It has two private properties: `Position` and `Label`. The methods `changeColor()` and `changeText()` manage the tile's background color and text based on its state. The `propertyChange(PropertyChangeEvent pce)` method updates the tile's label when a restart event occurs.

When a tile is clicked, the `clickedTile()` method is invoked. This method attempts to swap the tile with the empty space by firing a **vetoable change event**. If the swap is valid, the tile's label is updated to reflect the new state. If the swap is invalid, the tile flashes red to indicate an error. Both `EightController` and `EightTile` listen for property change events to update the game state and user interface.

4 EightBoard dashboard

The `EightBoard` extends `JFrame` and implements `PropertyChangeListener`. It sets up the game interface with a 3x3 grid of `EightTile` beans, an `EightController` label, a RESTART button and a `flip` button.

The board initializes tiles with their positions and assigns random labels via a Restart event. Both the tiles and controller register as listeners to this event.

5 Flip button

The `Flip` button is designed to handle flipping the labels of the first and second tiles when clicked, ensuring that an unsolvable puzzle configuration can be made solvable.

The `Flip` class extends `JButton` and implements `Serializable` and `PropertyChangeListener`. It's initialized with default text and color. When clicked, it fires a vetoable change event to notify listeners (e.g., the EightController). If not vetoed, it swaps the labels of the specified tiles. The propertyChange method updates the labels based on a restart event to ensure the flip can operate on the correct tiles after a restart.