Obligatorisk uppgift 3 - 1DL028

Alhassan Jawad

May 14, 2024

Information

Name: Alhassan Jawad
Contact: Alhassan.Jawad.9989@student.uu.se

1 Introduction

In this report, I will explain and summarize how I solved the 4th Assignment of the the course *Object Oriented Programming with Java*. To start the solitaire game, run the *RunMe.java* code after following the instructions written within *HowToStartTheGame.txt* inside the zipped source code folder.

2 Organisation & Structure of the Code

Using event-driven programming, standard libraries, and object-oriented programming concepts, the program shows how to create a well-structured and executed solution. The intended functionality is achieved and the design choices are driven by sound reasoning.

The program is organized into 11 different classes:

- RunMe.java: Initializes the game window and its components.
- Solitaire.java: Manages the game and renders it
- StartTheGameActions.java: Handles game initialization
- Window.java: Provides the foundation for the game window
- Card.java: Represents an individual playing card
- Arrangement.java: Represents a position on the game board where a card can be placed
- Button.java: Deals with the button panel
- GameRules.java: Contains the core game logic for validating player actions based on given game rules
- DrawStack.java: Contains code for the deck of cards that haven't been dealt yet
- DiscardedStack.java: Contains code for the stack where players discard cards during game play
- CardStack.java: Contains code for a stack of cards on the table.

2.1 Quick overview of the code

It is a little difficult to point out specific methods from each class as the most important one as all methods work with each other so that the solitaire game can run smoothly while following the rules given in the assignment information. But if I were to give a quick overview of how the code would react upon running the *RunMe.java* script (which starts the game):

1. The RunMe.java code creates a window with the help of the Window.java.
2. The RunMe.java code adds the solitaire panel and button panel where the solitaire panel implements the games rules, creates all the card objects and initializes all starting values with the help of the StartTheGameAction.java script. The button panel contains the "Start anew" and "Exit Game" buttons.
3. Each card object belongs to a stack that belongs to either the DiscardedStack.java or DrawStack.java codes (subclasses of the CardStack.java code).
4. Finally, all stacks are positioned where they should be using the Arrangement class.

The primary functions shared by all of these classes are essentially running the game smoothly while at the same time reacting to mouse events in response to user input/action.

3 Initial Approach

My initial approach to solving this problem involved learning in depths how the game works as I don't play such games. Afterwards I started by first creating the layout of the game where I have a space in the center and to the sides for the different stacks. After making sure that the layout looks good, I started implementing event responses to the mouse clicks that a user makes. In an attempt to make the event-driven part of the assignment more easy to program, I created the GameRules.java script (or something similar to it) as soon as possible so that I only needed to take into account the moves that are allowed to be made according to the game rules and the additional rules stated in the assignment instructions.

This game, Napoleon's Grave or commonly known as Solitaire have most of its moves automated in a general situation. But as the assignment wanted us to give near complete freedom to the player, more code was needed to discard the automated moves & logic behind it.

4 Essential libraries & usage of given code

This solitaire game uses the following fundamental standard libraries:

- Swing (JFrame, JPanel): The graphical user interface (GUI) of the game is built upon the usage of these libraries. The primary window is created by a JFrame, while other interactive elements inside the window are contained by a JPanel.
- AWT.event (MouseListener, ActionListener): These libraries let the application react to user input/action, including button presses and mouse clicks. Clicks, drags, and other mouse events are tracked by MouseListener, and user actions initiated by buttons or other interactive elements are handled by ActionListener.
- AWT.Graphics: This library gives you the basic tools you need to create and work with graphics on the screen..

The given code snippets helped me mostly understanding how to draw the cards (more appropriate to say render) on the GUI as I had the most difficulties with understanding that you can basically make use of external images.

5 Conclusion

I am mostly satisfied with my solution as I wasn't sure if I could make this game as I haven't played it before.

One improvement opportunity would be to decrease the number of classes as it can be a little difficult for other programmers to understand what each script is responsible for.