**COMP41670 Software Engineering (EEEN 40310)**

Individual project

**Due by: As indicated on Brightspace**

Name:\_Yevhenii Mormul

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Student number: | 1 | 9 | 2 | 0 | 3 | 9 | 7 | 7 |
|  | |  |  |  |  |  |  |  |  |

Declaration of Authorship

I declare that all material in this assessment is my own work.



Signed\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_ 3/10/2022



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

# Self Assessment of code

|  |  |  |
| --- | --- | --- |
| Feature | Points | Comment |
| Display cards and labels | 4 | Fully working with score and number of commands displayed  Diagram, schematic  Description automatically generated |
| Prompt user for input | 1 | Fully working: |
| Accept user input | 1 | Fully working with lower and upper case commands |
| Error message for invalid commands | 2 | Fully working:   * Text    Description automatically generated with medium confidence * Text    Description automatically generated with medium confidence * Text    Description automatically generated with medium confidence   etc |
| Error message for invalid play | 2 | Fully working:          etc |
| Draw card from pile working | 4 | Fully working:   * Can move from deck pile to column * Can move from deck pile to suit pile |
| Move card working | 6 | Fully working:   * Implemented an automatic move of card piles between columns if a play is legal * Can move from column to suit pile * Can move from suit pile to column |
| Quit working | 2 | Fully working:  Text  Description automatically generated with medium confidenceA picture containing text  Description automatically generated |
| Total | 24 | All the features are fully working |

SCRIPT:

My name is Yevhenii mormul and today I will show you my solitaire implementation. As you will be able to see in the video code is able to display cards, prompt user input, show error messages, fetch cards from deck, move cards from deck into columns and suit piles, move cards between columns and put cards into suit piles from columns. In addition It can display Victory, once all the cards are sorted in suit piles, or allow user to exit the game.

The code was designed using OOP approach.

Firstly there is a Card class representing the card, with its value and suits implemented as enum objects, and isVisible Boolean responsible for revealing the card. Card can be displayed by 3 main means: simple verbose, top part of the card with suit and value, and the full card.



Shape

Description automatically generated

A picture containing antenna, clock

Description automatically generated

It also can display the card face down and creating placeholders.

Secondly there is an abstract class Pile, which is responsible for storing an ArrayList of cards, and managing it: appending other piles, adding, removing, popping cards etc.

Pile class has 3 children Deck (upside down pile) Column (lane) and a suit pile.

Deck is implemented as a virtual cyclic array, revealing only 1 card at a time, using an extra integer currentCard. Its done so that user can only draw cards from it and see 1 card at a time.

Column is implemented as a virtual stack, with an extra integer variable identifying the number of the column. any cards can be at the head as long as they are invisble, and if one wants to append more cards to it, its important to verify that the suit has an opposite color and value, which is less by 1 unit than the preceding. If column is empty can only start appending from king.

Suit pile is also a stack, with an extra variable of class Cardenum suits, which defines the base suit of the suit pile. One can only append cards of the same suit, starting with the smallest value (ACE) and up to the largest one (KING).

The game is managed by the class table, which contains 7 indexed columns, 4 suit piles with different suits and a deck. This class can shuffle a normal 52 card deck, distribute them among columns and deck in the solitaire fashion.

It is also responsible for printing out the game state displaying the current state of all the piles, current score and number of turns.

It also takes a safe user input and performs a move or warns user If it is invalid.

Finally it allows to determine if victorious state is reached

Solitaire game is responsible for instantiating the table and keeping the game in loop as long as user doesn’t quit it. It also ensures that the input is safe and warns user if invalid command is entered.

Diagram

Description automatically generated

The code was tested according to specification, thus initially all the valid moves were tested (moving cards between columns, from deck to suit pile and columns, and from columns to suit pile) using lowercase and upper case inputs. then invalid commands and moves were tested to make sure that the warnings work as expected and there are no unexpected outcomes, for example when trying to draw from an empty pile etc.

As per specification, the scoring and move number were tested as well. Thus indeed code awards 5 points for each card moved between columns, 10 points for moving card from deck into suit pile and 20 points for moving a card from column to the pile. Also it was verified that only valid moves contribute to the incrementing the count of moves.

In the end of the game, game indeed congratulates the user.

Finally force quitting with Q or q was tested