Solitaire

Session: 2023 – 2027

Submitted by:

Muhammad Tayyab 2023-CS-101

Supervised by:

Sir. Nazeef-ul-Haq

Course:

CSC-200 Data Structures and Algorithms

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Introduction

This is a simple Card game named Klondike Solitaire game. For this game you need only one player and a set of regular playing cards. A regular playing cards consists of 52 cards. Out of which 26 are red and 26 are black cards. The 52 cards are divided into four suits i.e. **Hearts**, **Clubs**, **Spades**, **Diamonds**. Each suit has 13 cards, 3 Face cards i.e. King, Queen and Jack and 1 Ace Card and 9 Cards numbered from 2-10.

The game consists of four sections: Tableau, Foundations, Stock, Waste.

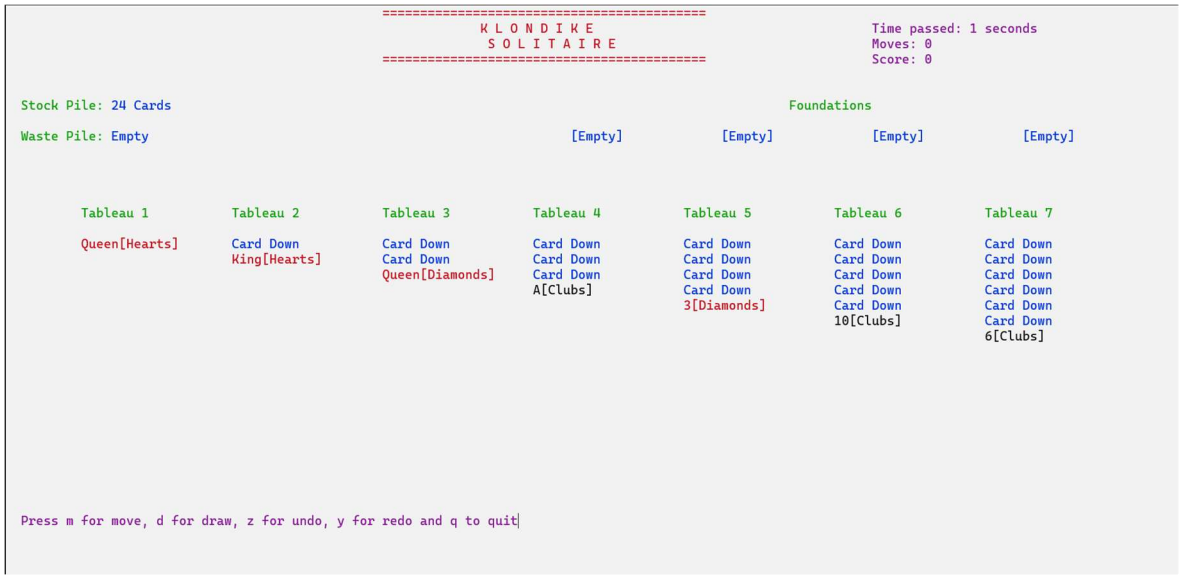
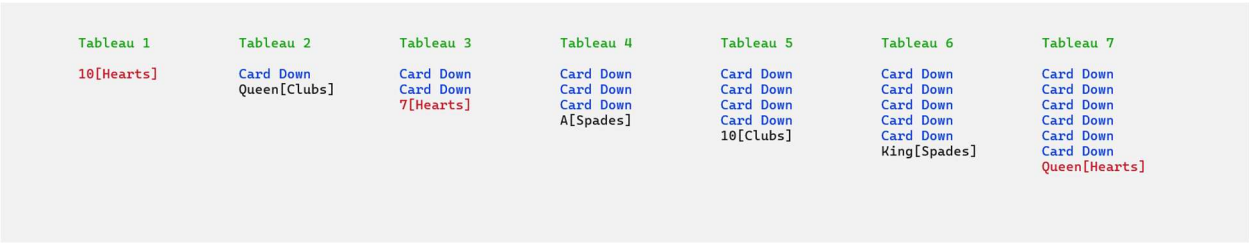


Tableau:

The Tableau is the seven piles of cards on the table where 28 cards are placed. The first column has one card, the second has two, and so on up to seven cards in the last column. Only the last card in each pile is face up while the other cards are face down.



Foundation:

The Foundation is the four piles of cards on the cards which are initially empty. In this, cards are placed in ascending order by suit.

```
                    Foundations
[Empty]          [Empty]          [Empty]          [Empty]
```

Stock:

After placing all the cards in the tableaus, we are left with 24 cards. These cards are placed into stock. We can draw a card from it and use it in tableau or foundation pile. They all are face down.

Waste:

When we draw a card from stock, the card comes into a waste pile. When the stock is empty, all the cards from the waste pile move into the stock.

```
Stock Pile: 24 Cards

Waste Pile: Empty
```

```
Stock Pile: 23 Cards

Waste Pile: King[Hearts]
```

Objective:

Our goal is to move all the cards in the foundation piles by moving the cards between tableau piles or from stock to tableau.

```
                    Foundations
King[Hearts]     King[Spades]     King[Diamonds]     King[Clubs]
```

Technology Used:

Programming Language: C++
The game is fully implemented in C++, using OOP concepts to manage game components like cards, tableaus, foundations, and stock.

IDE: VS Community
Used for writing, compiling, and debugging the code.

Data Structures: Data Structures used in the game are Linked List, Stack, Queue.

Class Structure and Functionality:

Card Class

Purpose: Represents a playing card in the game.

Methods:

- Card (int rank, char suit): Constructor to initialize a card with a rank and suit.
- void display (): Displays the rank and suit of the card.

Move Class

Purpose: Represents a move in the game. Useful in implementing undo/redo feature.

Methods:

- Move (string cmd, int from, int fromIdx, int to, int toIdx, int noOfCards, bool face): Constructor to initialize a move with source and destination details.

Node Class

Purpose: Represents a node in a linked list.

Methods:

- Node (T data): Constructor to initialize a node with data.

LinkedList Class

Purpose: A template class that manages a list of nodes.

Methods:

- void insertAtHead(T val): Inserts a node at the head.
- void insertAtEnd(T val): Inserts a node at the end.
- Node<T>* findNode(T x): Finds a node with a specific value.
- void deleteFromStart(): Deletes a node from the start.
- void deleteFromEnd(): Deletes a node from the end.
- bool isEmpty():s Checks if the list is empty.
- int size (): Returns the size of the list.

Queue Class

Purpose: Implements a queue using linked lists.

Methods:

- void enqueue (T val): Adds an element to the queue.
- T dequeue (): Removes an element from the queue.
- T peek (): Returns the front element of the queue.
- bool empty (): Checks if the queue is empty.
- int size (): Returns the size of the queue.

Stack Class

Purpose: Implements a stack using linked lists.

Methods:

- void push (T val): Adds an element to the stack.
- T pop (): Removes an element from the stack.
- T top (): Returns the top element of the stack.
- bool empty (): Checks if the stack is empty.
- int size (): Returns the size of the stack.

SolitaireGame Class:

Purpose: Manages the overall game logic and state.

Methods:

- bool isValidMove(Card& c, Card& topCard): Checks if a move is valid between tableaus.
- bool isValidMoveForF(Card& c, Card& topCard): Checks if a move to the foundation is valid.
- void drawCard(): Draws a card from the stock pile to the waste pile.
- bool win(): Checks if the game is won.
- void moveCardBetweenTableaus(LinkedList<Card>& from, LinkedList<Card>& to, int noOfCards): Moves cards between tableaus.
- void moveCardFromWtoF(Queue<Card>& w, Stack<Card>& f): Moves a card from the waste pile to a foundation.
- void moveCardFromTtoF(LinkedList<Card>& t, Stack<Card>& f): Moves a card from a tableau to a foundation.
- void moveCardFromWtoT(Queue<Card>& w, LinkedList<Card>& t): Moves a card from the waste pile to a tableau.
- int noofFaceupCards(LinkedList<Card>& LL): Counts the number of face-up cards in a linked list.
- void play (): Starts the game loop.

ConsoleUtility Class

Purpose: Provides common functions for console operations.

Methods:

- void setConsoleColors(int textColor, int backgroundColor): Set the text color and background color.
- void gotoxy(int x, int y): Moves the cursor to specific position.

UI Class

Purpose: Handles the user interface.

Methods:

- void displayInstructions(): Displays the instructions to the user.
- void displayHeader(): Displays the header of the game.
- void displayFrontPage(): Displays the menu page to the user.
- void displayScore(): Displays the scoring criteria of the game.
- void displayShuffleOptions(): Asks the users about the difficulty of game.
- void displayGame(LinkedList<Card> tableaus[], Stack<Card> foundations[], Queue<Card>& stockPile, Queue<Card>& wastePile, int& score, int& moves): Displays all the things on the screen at their positions.

Rules:

- We can move a top card of the tableau to another tableau card if it's the opposite color and one lower rank.
- Only a King or a pile with King as top card can be placed in empty tableau.

Challenges:

The challenges I faced while making the project were:

- Manipulating linked lists while moving the cards between tableaus.
- The undo feature was quite challenging.
- Adding a timer was also very difficult.

Future Improvements:

I chose a console-based Solitaire game as GUI wasn't a requirement. In future, I would like to develop a GUI version to increase user attraction and interaction. Currently, my easy and hard levels use the same card sequence each time, and I would like to find a way for implementing the levels correctly.

Test Cases:

Test Case Number	Test Case	Description	Expected Result
1	Initial Setup	Verify that the game initializes correctly and displays the initial layout of the cards.	The game should display the initial layout with cards arranged in the tableau, stock, and waste piles.
2	Move Card Between Tableaus	Confirm that the cards can be moved between tableaus.	The cards should be moved according to the valid move.
3	Move Card from Tableau to Foundation	Ensure that a card can move from the tableau to foundation.	A card should be moved to foundation if it is valid.
4	Move Card from Waste to Tableau	Confirm that a card can move from the waste to tableau.	A card should be moved from waste pile to tableau if it is valid.
5	Move Card from Waste to Foundation	Ensure that a card can move from waste pile to foundation.	A card should be moved from waste pile to foundation if it is valid.
6	Draw a Card	Verify that a card can draw from the stockpile to the waste pile using ‘draw’ command.	A card should be drawn.
7	Undo a Move	Ensure that the last move can be undone using ‘undo’ command.	The last move should be undone and the cards should return to their previous positions.
8	Redo a Move	Confirm that the last undo move can be redone using ‘redo’ command.	The last undone move should be redone, and the cards should return to their positions after the move.
9	Quit Game	Verify that the game can be quit using the ‘quit’ command.	The game should exit and return to the main screen.
10	Winning the Game	Confirm that the game is won when all cards are moved to the foundation piles in the correct order.	The game should display a winning message when all cards are correctly placed in the foundation piles.