Solitired Documentation

Created by

Pisitpong Kreangkrailipikon 6331331321

Ratchapol Sukonthamarn 6331340021

2110215 Programming Methodology

Semester 2 Year 2020

Chulalongkorn University

Solitired

Introduction

Solitired is inspired by a single-player tabletop game called “solitaire”. The objective of the game is to move all the cards to the "foundation pile".

PS. The rules are a little different from the normal rules.

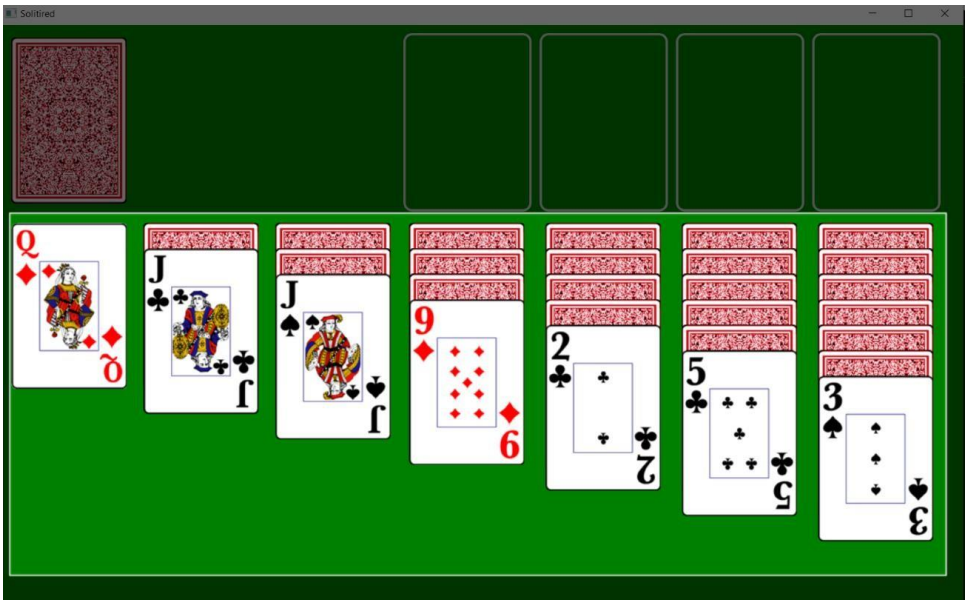
Setting Up

1. Solitired game is played with a standard 52-card deck.

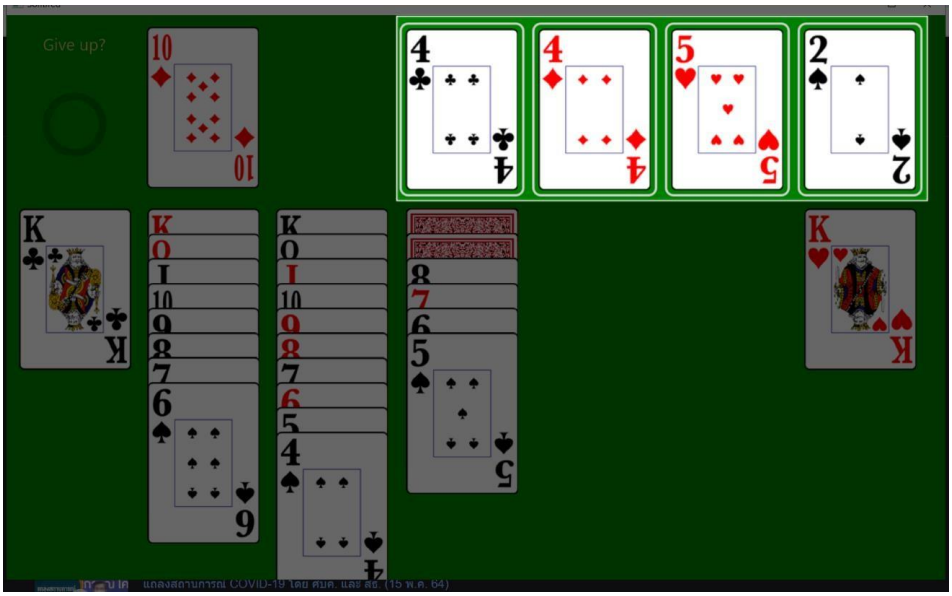
2. The value of cards in Solitaire games is: K (highest), Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2, A (lowest).

3. There are four different piles in Solitired:

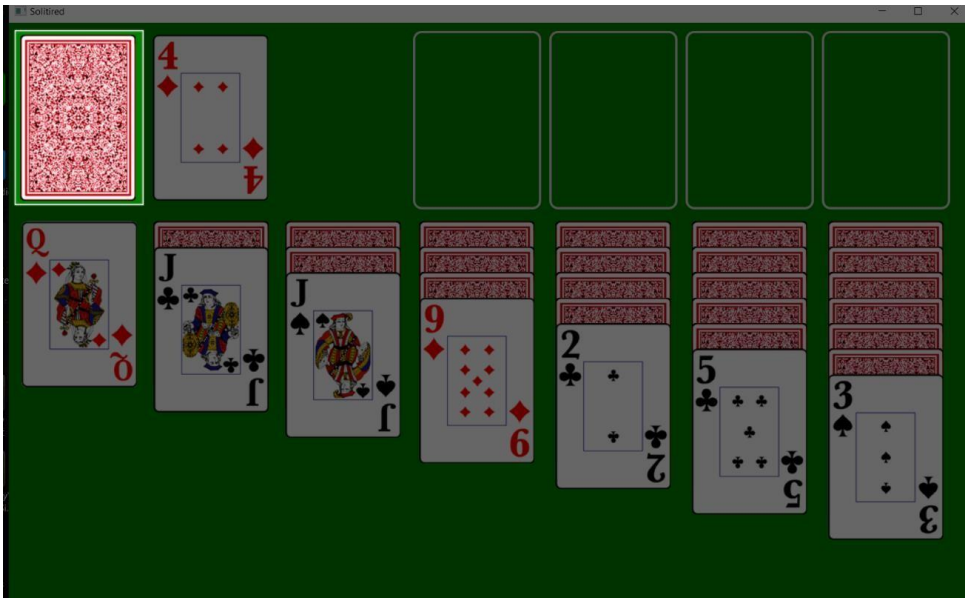
3.1 The Table Pile



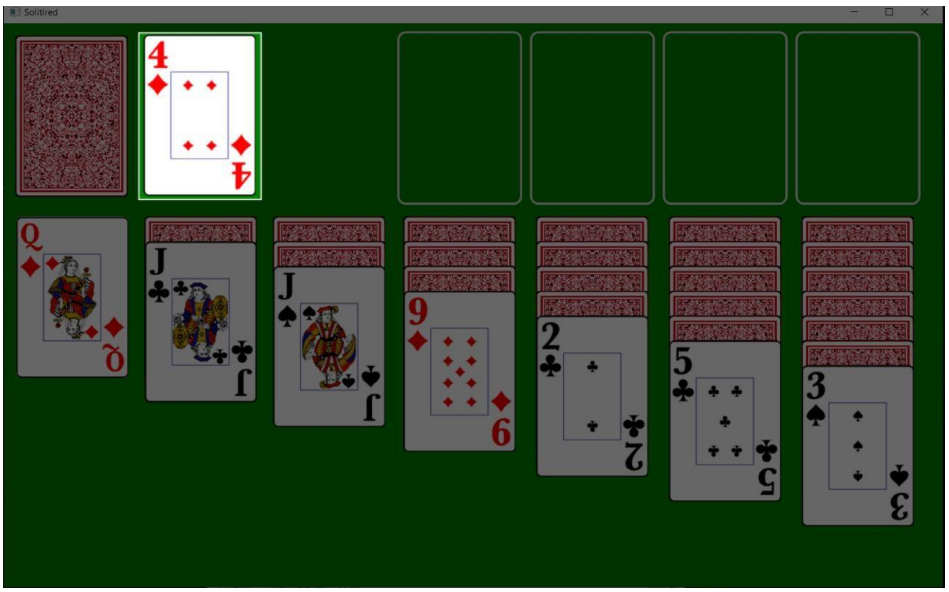
3.2 The Foundation Pile



3.3 The Deck



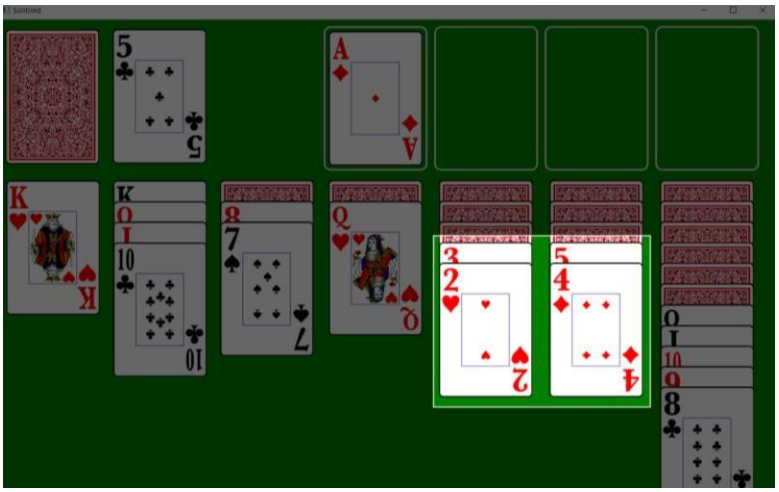
3.4 The Discard Pile



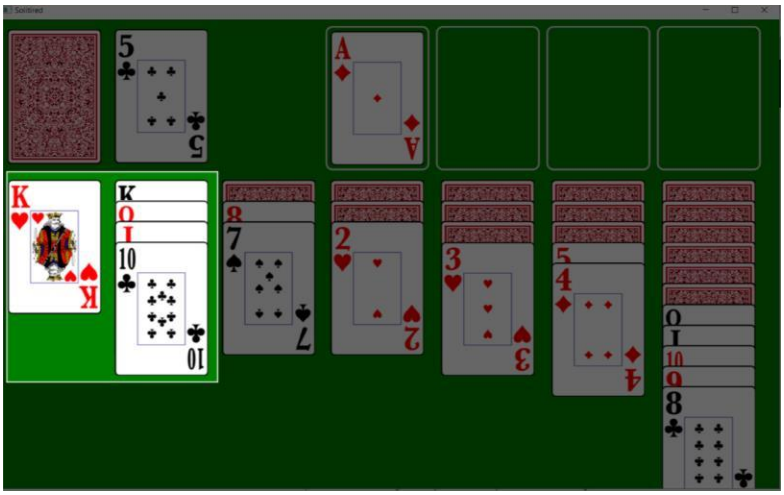
Gameplay

1. Face-up cards can be moved from the discard pile or the table piles to the foundation pile or to other table piles.

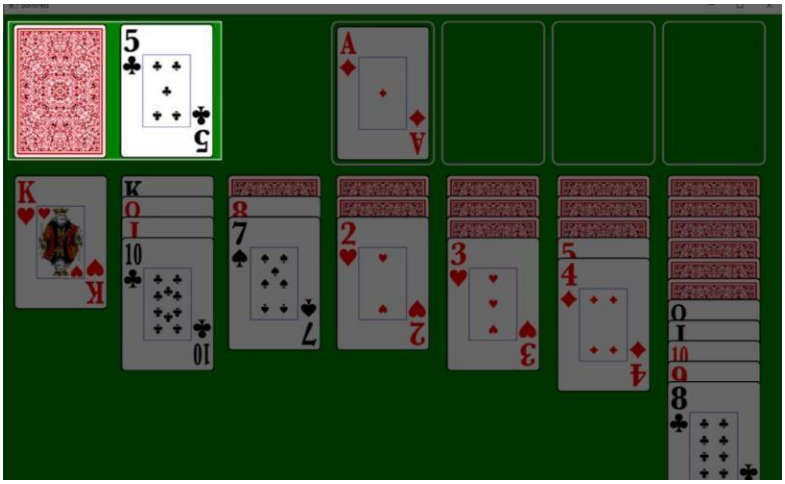
2. To move a card to a table pile, its value must be one less, but unlike the normal solitaire, this game can use all suit. For example, if it was a 9 of hearts, you could put an 8 of spades (♤), hearts (♥), diamonds (♢) or clubs (♧) onto it. Stacks of cards may be moved from one table pile to another as long as they maintain the same order (highest value to lowest value).



3. If you have an empty table pile, you can start a new table pile with a King. Any new table pile must be started with a King (or a stack of cards that starts with a King).

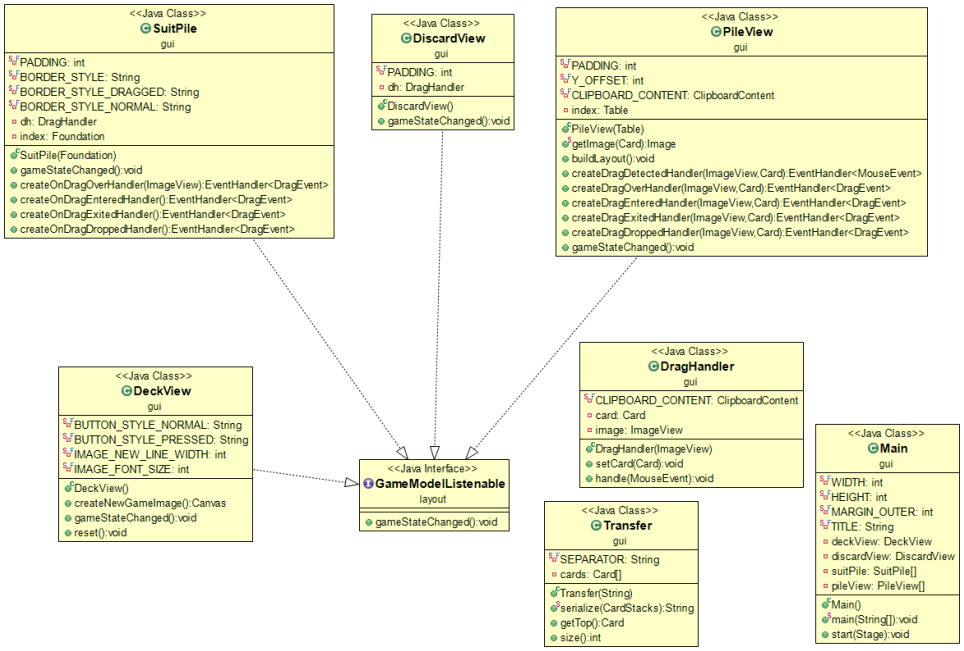


4. To get new cards from the deck, you turn a card at a time face up into the stack next to the deck called the discard pile. When your deck run out of cards and you cannot play anymore, you lose.

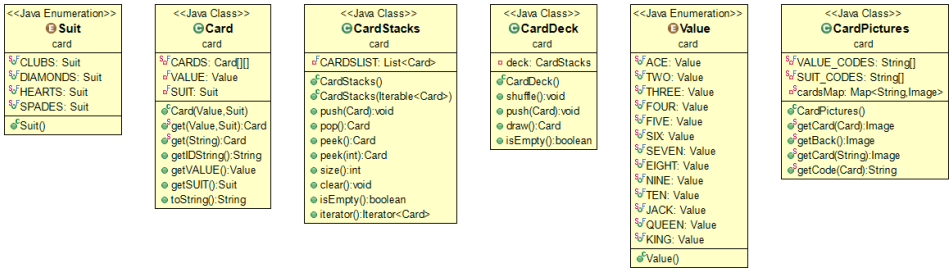


Class Diagram

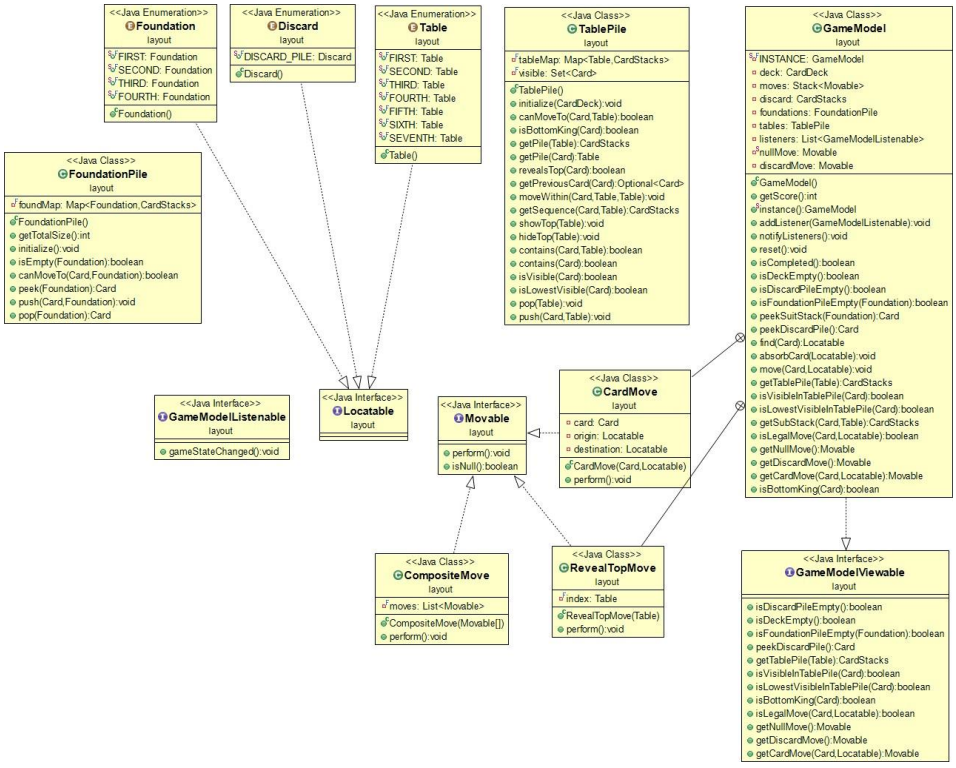
GUI Diagram



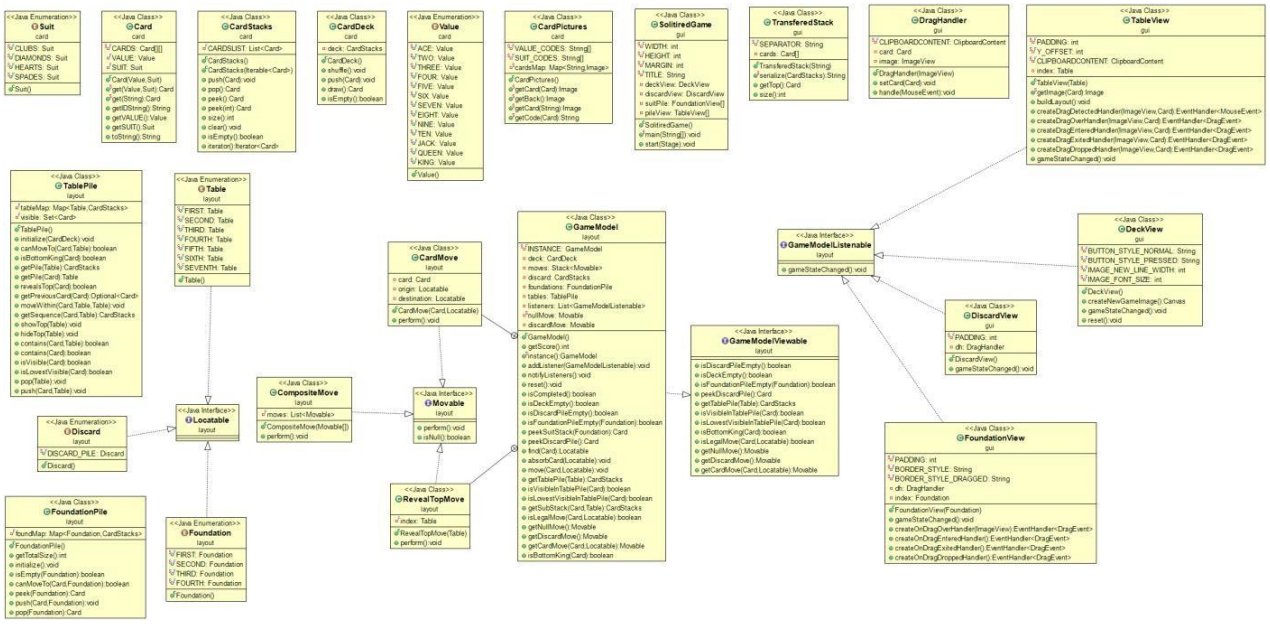
Card Diagram



Layout Diagram



Full Class Diagram



Link to GitHub

https://github.com/SushiRS2002/Solitired

Link to instruction video

https://youtu.be/RsBOpExra5k

1. Package Card

1.1 Enum Suit

1.1.1 Enum

CLUBS, DIAMONDS, HEARTS, SPADES.

1.2 Enum Value

1.2.1 Enum

ACE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, JACK, QUEEN, KING.

1.3 Class Card

1.3.1 Field

- static final Card [][] CARDS	A 2D list which stores 52 card and initializing 2D list by adding all 52 cards
- final Value VALUE	Card's value. Range from 1 (Ace) to 13 (King)
- final Suit SUIT	Card's suit (Club, Diamond, Heart, Spade)

1.3.2 Constructor

+ Card (Value value, Suit suit)	Card's constructor

1.3.3 Method

+ static Card get (Value value, Suit suit)	Returns a card via (value, suit) from 2D list
+ static Card get (String str)	Returns a card via its ID from 2D list
+ String getIDString ()	Returns a card's ID
+ Value getVALUE ()	Returns a card's value
+ Suit getSUIT ()	Returns a card's suit
+ String toString ()	toString() of each card

1.4 Class CardPictures

1.4.1 Field

- static final String [] VALUE_CODES	List for shortened value
- static final String [] SUIT_CODES	List for shortened suit
- static Map<String, Image> cardsMap	Map of card's code and image

1.4.2 Constructor

+ CardPictures ()	Default constructor

1.4.3 Method

+ static Image getCard (Card card)	Return card's image
+ static Image getBack ()	Return card's back side image
+ static Image getCard (String code)	Return card via card's code
+ static String getCode (Card card)	Return card's code

1.5 Class CardStacks

1.5.1 Field

- final List<Card> CARDSLIST	List of cards

1.5.2 Constructor

+ CardStacks ()	CardStacks's constructor
+ CardStacks (Iterable<Card> cards)	CardStacks which contains all card in cards

1.5.3 Method

+ void push (Card card)	Pushes card into the stack
+ Card pop ()	Removes and returns the topmost card on the stack
+ Card peek ()	Returns the topmost card on the stack
+ Card peek (int index)	Returns the card on the stack of that index
+ int size ()	Returns stack's size

+ void clear ()	Removes all card in the stack
+ boolean isEmpty ()	Checks if the stack is empty
+ Iterator<Card> iterator ()	Java iterator for accessing collections

1.6 Class CardDeck

1.6.1 Field

- CardStacks deck	A card deck

1.6.2 Constructor

+ CardDeck ()	Shuffles the deck when initialize it

1.6.3 Method

+ void shuffle ()	Reinitializes the deck
+ void push (Card card)	Pushes a card on top of the deck
+ Card draw ()	Draws and removes a card from the deck
+ boolean isEmpty ()	Checks if the deck is empty

2. Package layout

2.1 Interface Locatable

A location in the game

2.2 Enum Discard implements Locatable

2.2.1 enum

DISCARD_PILE.

2.3 Enum Table implements Locatable

2.3.1 enum

FIRST, SECOND, THIRD, FOURTH, FIFTH, SIXTH, SEVENTH.

2.4 Class TablePile

2.4.1 Field

- final Map<Table, CardStacks> tableMap	A map of tables
- final Set<Card> visible	A visible condition

2.4.2 Constructor

+ TablePile ()	Creates seven empty table piles

2.4.3 Method

+ void initialize (CardDeck deck)	Fill seven table piles with cards
+ boolean canMoveTo (Card card, Table index)	Checks if moving a card to table pile is able to do it
+ boolean isBottomKing (Card card)	Checks if the lowest card is a king

+ CardStacks getPile (Table index)	Returns a copy of that entire pile
+ Table getPile (Card card)	Returns a position which includes that card
+ boolean revealsTop (Card card)	Checks if moving a card away reveals the top of the pile
+ Optional<Card> getPreviousCard (Card card)	Returns a card after moving away from the pile
+ void moveWithin (Card card, Table origin, Table destination)	Moves a stack of cards from A to B
+ CardStacks getSequence (Card card, Table index)	Returns a sequence of cards in that index
+ void showTop (Table index)	Shows a topmost card of that index
+ void hideTop (Table index)	Hides a topmost card of that index
+ boolean contains (Card card, Table index)	Checks if that index has a specified card
+ boolean contains (Card card)	Checks if a specified card in whatever index
+ boolean isVisible (Card card)	Checks if that card is visible
+ boolean isLowestVisible (Card card)	Checks if that card is visible and in the lowest position
+ void pop (Table index)	Removes the top card from that index
+ void push (Card card, Table index)	Adds a visible card on top of that index

2.5 Enum Foundation implements Locatable

2.5.1 enum

FIRST, SECOND, THIRD, FOURTH.

2.6 Class FoundationPile

2.6.1 Field

- final Map<Foundation, CardStacks> foundMap	A map of foundations

2.6.2 Constructor

+ FoundationPile ()	Initializing foundation piles when called

2.6.3 Method

+ int getTotalSize ()	Returns the total number of cards in foundation piles
+ void initialize ()	Initializing four foundation piles
+ boolean isEmpty (Foundation location)	Checks if that foundation pile is empty
+ boolean canMoveTo (Card card, Foundation location)	Checks if we can move a card to that foundation pile
+ Card peek (Foundation location)	Returns a topmost card of that foundation pile
+ void push (Card card, Foundation location)	Places a card on the foundation pile
+ Card pop (Foundation location)	Removes and returns a topmost card of that foundation pile

2.7 Interface Movable

2.7.1 Method

+ void perform ()	Performs a move
+ default boolean isNull ()	Checks if that move advances the game. False by default

2.8 Interface GameModelListenable

2.8.1 Method

+ void gameStateChanged ()	Called when the game state is changed

2.9 Interface GameModelViewable

2.9.1 Method

+ boolean isDiscardPileEmpty ()	Checks if the discard pile is empty
+ boolean isDeckEmpty ()	Checks if the deck is empty
+ boolean isFoundationPileEmpty (Foundation index)	Checks if the foundation pile is empty
+ Card peekDiscardPile ()	Returns the topmost card of the discard pile
+ CardStacks getTablePile (Table index)	Returns a table pile via its index
+ boolean isVisibleInTablePile (Card card)	Checks if the card is visible in that table pile
+ boolean isLowestVisibleInTablePile (Card card)	Checks if the card is visible and in the lowest position
+ boolean isBottomKing (Card card)	Checks if the lowest card is a king
+ boolean isLegalMove (Card card, Locatable destination)	Checks if moving that card is able to do it
+ Movable getNullMove ()	Returns null move
+ Movable getDiscardMove ()	Returns discard move
+ Movable getCardMove (Card card, Locatable destination)	Returns card move

2.10 Class GameModel implements GameModelViewable

2.10.1 Field

- static final GameModel INSTANCE	A model of game
- CardDeck deck	A deck
- Stack<Movable> moves	A stack of moves
- CardStacks discard	A discard pile
- FoundationPile foundations	Four foundation piles
- TablePile tables	Seven table piles
- List<GameModelListenable> listeners	A list of game state listeners
- static Movable nullMove	A null move
- Movable discardMove	A discard move

2.10.2 Method

+ GameModel ()	Reset when called
+ int getScore ()	Return the number of cards in foundation piles
+ static GameModel instance ()	A singleton instance
+ void addListener (GameModelListenable listener)	Add a listener in the list
+ void notifyListeners ()	Called when game state has been changed
+ void reset ()	Reset the game
+ boolean isCompleted ()	Checks if the game is completed
+ boolean isDeckEmpty ()	Checks if the deck is empty

+ boolean isDiscardPileEmpty ()	Checks if the discard pile is empty
+ boolean isFoundationPileEmpty (Foundation index)	Checks if the foundation pile is empty
+ Card peekSuitStack (Foundation index)	Returns the topmost card of that foundation pile
+ Card peekDiscardPile ()	Returns the topmost card of the discard pile
+ Locatable find (Card card)	Returns the card's location
+ void absorbCard (Locatable location)	Removes a card from that location
+ void move (Card card, Locatable destination)	Move a card from A to B
+ CardStacks getTablePile (Table index)	Returns a table pile via its index
+ boolean isVisibleInTablePile (Card card)	Checks if the card is visible in that table pile
+ boolean isLowestVisibleInTablePile (Card card)	Checks if the card is visible and in the lowest position
+ CardStacks getSubStack (Card card, Table index)	Get the stack of the card and the ones below it
+ boolean isLegalMove (Card card, Locatable destination)	Checks if moving that card is able to do it
+ Movable getNullMove ()	Returns null move
+ Movable getDiscardMove ()	Returns discard move
+ Movable getCardMove (Card card, Locatable destination)	Returns card move
+ boolean isBottomKing (Card card)	Checks if the lowest card is a king
+ class CardMove implements Movable	A move of card stacks
+ class RevealTopMove implements Movable	Reveals the top of the stack

2.11 Class CompositeMove implements Movable

2.11.1 Field

- final List<Movable> moves	List of all card moves

2.11.2 Constructor

+ CompositeMove (Movable... move)	Add all move in the list

2.11.3 Method

+ void perform ()	The move’s performance

3. Package gui

3.1 Class DeckView extends HBox implements GameModelListenable

3.1.1 Field

- static final String BUTTON_STYLE_NORMAL	A normal button
- static final String BUTTON_STYLE_PRESSED	A pressed button
- static final int IMAGE_NEW_LINE_WIDTH	Spacing between circle and text at the bottom of the deck
- static final int IMAGE_FONT_SIZE	Text showing at the bottom of the deck

3.1.2 Constructor

+ DeckView ()	DeckView's constructor

3.1.3 Method

+ Canvas createNewGameImage ()	Game's canvas
+ void gameStateChanged ()	Check the game state of the deck
+ void reset ()	Reset the deck when called

3.2 Class DiscardView extends HBox implements GameModelListenable

3.2.1 Field

- static final int PADDING	Discard pile's padding
- DragHandler dh	Drag handler for cards in the discard pile

3.2.2 Constructor

+ DiscardView ()	DiscardView's constructor

3.2.3 Method

+ void gameStateChanged ()	Check the game state of the discard pile

3.3 Class TableView extends StackPane implements GameModelListenable

3.3.1 Field

- static final int PADDING	Table pile's padding
- static final int Y_OFFSET	Offset between each cards in table pile
- static final ClipboardContent CLIPBOARD_CONTENT	Clipboard for CP operations
- Table index	Index of a table pile

3.3.2 Constructor

+ PileView (Table index)	PileView's constructor

3.3.3 Method

+ static Image getImage (Card card)	Returns card's image
+ void buildLayout ()	Table piles' layout
+ EventHandler<MouseEvent> createDragDetectedHandler (ImageView imageView, Card card)	An event when system detect a drag
+ EventHandler<DragEvent> createDragOverHandler (ImageView imageView, Card card)	Call when detected that dragged object is stay over the destination
+ EventHandler<DragEvent> createDragEnteredHandler (ImageView imageView, Card card)	Call when detected that dragged object is enter the destination
+ EventHandler<DragEvent> createDragExitedHandler (ImageView imageView, Card card)	Call when detected that dragged object is exit the origin
+ EventHandler<DragEvent> createDragDroppedHandler (ImageView imageView, Card card)	Call when detected that dragged object is dropped at the destination

+ void gameStateChanged ()	Check the game state of the table pile

3.4 Class FoundationView extends StackPane implements GameModelListenable

3.4.1 Field

- static final int PADDING	Foundation pile's padding
- static final String BORDER_STYLE	Foundation pile's normal border
- static final String BORDER_STYLE_DRAGGED	Foundation pile's dragged border
- DragHandler dh	Drag handler for cards in the foundation pile
- Foundation index	FoundationView's constructor

3.4.2 Constructor

+ SuitPile (Foundation index)	FoundationView's constructor

3.4.3 Method

+ void gameStateChanged ()	Check the game state of the foundation pile
+ EventHandler<DragEvent> createOnDragOverHandler (ImageView image)	Call when detected that dragged object is stay over the destination
+ EventHandler<DragEvent> createOnDragEnteredHandler ()	Call when detected that dragged object is enter the destination
+ EventHandler<DragEvent> createOnDragExitedHandler ()	Call when detected that dragged object is exit the origin
+ EventHandler<DragEvent> createOnDragDroppedHandler ()	When detected that dragged object is dropped at the destination

3.5 Class TransferedStack

3.5.1 Field

- static final String SEPARATOR	Separator for each transfered cards
- Card [] cards	List of transfered cards

3.5.2 Constructor

+ Transfer (String str)	Transfer's constructor

3.5.3 Method

+ static String serialize (CardStacks stacks)	An array of card's ID
+ Card getTop ()	Returns the top card in the transfer
+ int size ()	Returns the transfer's size

3.6 Class DragHandler implements EventHandler<MouseEvent>

3.6.1 Field

- static final ClipboardContent CLIPBOARD_CONTENT	A clipboard content
- Card card	A card
- ImageView image;	A card's image

3.6.2 Constructor

+ DragHandler (ImageView image)	DragHandler's constructor
+ void setCard (Card card)	Set a card

3.6.3 Method

+ void handle (MouseEvent me)	Handle any drag event

3.7 Class Main extends Application

3.7.1 Field

- static final int WIDTH	Application's width
- static final int HEIGHT	Application's height
- static final int MARGIN_OUTER	Application's margin
- static final String TITLE	Application's title
- DeckView deckView	Initializing the deck
- DiscardView discardView	Initializing the discard pile
- FoundationView [] foundationView	Initializing the foundation piles
- TableView [] tableView	Initializing the table piles

3.7.2 Constructor

+ Main ()	Default constructor

3.7.3 Method

+ void start (Stage primaryStage)	launch the application