

◆ COMPSCI 130 Creative Extension ♥

Starr Zhang

UPI: yzhb172

Table of Contents

1. Beyond Section 1- 4	3
1.1. New Features/Functions	3
1.1.1. GUI and Buttons	3
1.1.2. Moving Consecutive Cards	3
1.1.3. Undo	3
1.1.4. Restart and Re-shuffle	4
1.1.5. Updated Class and Data Structures	4
2. Challenges and Difficulties	5
2.1. Difficulty with Tkinter	5
2.1.1. How to light up buttons when clicked	5
2.1.2. How to reset all buttons back to normal	5
2.1.3. How to update button display when data changes	5
2.2. Challenges working on Game Logic	5
2.2.1. Use of Global Variable	5
2.2.2. Late Binding issue	5
2.2.3. Initial Pile hidden content.	5
3. Reflection/Things to Improve	6
3.1. Bug Fix	6
3.1.1. Initial pile can only move as pile	6
3.2. Nice to Have	6
3.2.1. Utilise Node & Linked List ADT	6
3.2.2. Buttons UI link up (dynamic function)	6
3.2.3. Game AI that allows successful deck draw	6

1. Beyond Section 1- 4

This section talks about new features added to the base code provided and completed from Section 1- 4.

1.1. New Features/Functions

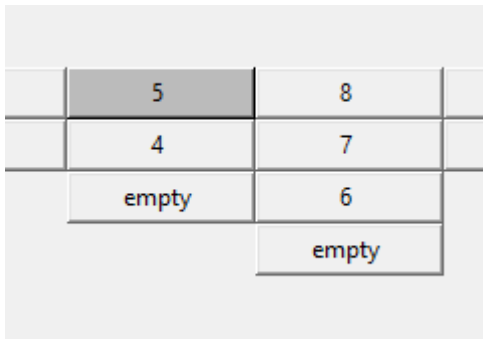
1.1.1. GUI and Buttons

I decided to follow the aesthetic of the original Solitaire game. However, due to limitations in Python Tkinter, the final design is simplified. All buttons in the main game window are clickable and has various features built into them.

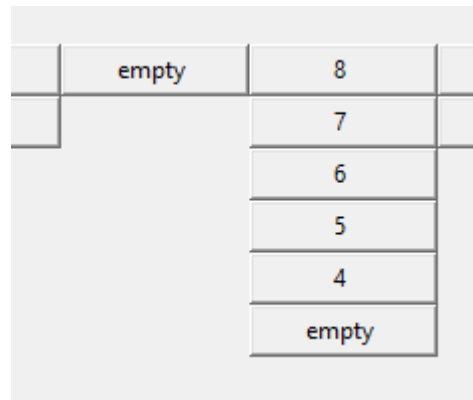
Player can choose the card(s) they wish to move by clicking on them and select an 'empty' slot to insert their desire destination pile. Buttons will change colour as User Interaction feedback, letting the player know that they have selected which button/card.

1.1.2. Moving Consecutive Cards

Carrying the basic from Section 1 – 4, I added a new game feature such that player is allowed to select cards that are consecutively linked together without selecting the whole pile and move the pile of cards to a valid destination.



5	8	
4	7	
empty	6	empty



empty
8
7
6
5
4
empty

New Move Method:

I added some new methods beyond the template provided to implement the above function. This *move()* method calls a *remove_card()* method and an *add_card()* method. The *remove_card()* method deletes and returns a list of consecutive cards in descending order that the player wish to move, and each element in the list is then added to the end of the destination pile.

1.1.3. Undo

I implemented an undo function that exists in majority of the computer games. This undo function utilises the exact move method in the pile class with reversed index.

1.1.4. Restart and Re-shuffle

In case the arrangement of the initial cards fails to successfully arrange into descending order, I made a restart button if the player wants to try the same set of cards again as a new round, or re-shuffle the whole pile.

1.1.5. Updated Class and Data Structures

Card

In addition to code in section 1 - 4, I have added another Node ADT to the creative solution such that each individual card is stored as a Card object.

I have chosen to do this because I have decided to implement the Moving Consecutive Cards (1.1.2.) feature in my solution. I was planning to link nodes which are in descending order and set next value for each of the node if valid. (More in 3.1.1.)

A benefit from defining a separate class for individual card is that I can definition their string representation, this allows me to convert numbers like 13 to K for a poker representation.

SinglePile

This class is the original CardPile class. I have added more methods in addition to the basics. This class is still in use of Stack ADT. Elements can only be added from the bottom of the stack and removed from the bottom of the stack. This contributes to the mechanism of basic solitaire.

Pile

This is also the original Solitaire class as nested list structure but with additional features. I have written a method to test if a pile movement is valid. This is very similar to the test conditions from section 1 – 4, however, since I want to be able to move single cards out of a pile and sometimes breaking from the middle of the pile, I made a new test condition corresponding to this feature. *is_pile_move_valid()* takes in the row index and column index of both the selected card pile to the destination card pile.

2. Challenges and Difficulties

2.1. Difficulty with Tkinter

Learning Python's built-in toolkit for building a graphical user interface is an intriguing process. I started from complete scratch and went through a lot of YouTube videos to get my head around.

2.1.1. How to light up buttons when clicked

I worked around some online materials to figure out how to call a command of a button. The problem I went across was that I had a nested list of buttons, which means that I cannot simply call the `button_clicked()` function in the usual way. The main reason is called a late binding issue. (More details in 2.2.2. *Late Binding issue*)

2.1.2. How to reset all buttons back to normal

For the sake of User Interaction, I decided to de-select / reset colour of selected buttons to inform player that a chosen move is invalid. I went into this problem as in 2.1.1. (More details in 2.2.2. *Late Binding issue*)

2.1.3. How to update button display when data changes

I went into research and tried different solutions to achieve this. Since the data pile is constantly updating with respects to players clicking action, I am also updating the button list which has the

2.2. Challenges working on Game Logic

2.2.1. Use of Global Variable

In order to successfully implement some of the main features in my solution, I realised that I have to be able to access some variables throughout the entire code.

2.2.2. Late Binding issue

With buttons as the main component in my creative solution, I have coded various functions for different clicking events. Since I am calling my functions directly in the command argument of Tkinter's button, python will always give me a return value. This means that by the time I run to the end of me for loop, the only button/data I obtain is the last element of the list.

2.2.3. Initial Pile hidden content.

Since I have implemented an undo method, its really hard to keep track of what cards have been revealed and otherwise, so I had a hard time trying to define and assign a count variable which counts the number of cards moved off of the initial pile.

3. Reflection/Things to Improve

3.1. Bug Fix

3.1.1. Initial pile can only move as pile

Since I was ensuring that the unrevealed cards in the initial piles stays hidden, I realised that I have implemented my game logic to only allow player to select the bottom of the initial pile. Meaning that if I have a consecutive pile in descending order, I can only the pile as a whole instead of moving any card(s) around.

3.2. Nice to Have

3.2.1. Utilise Node & Linked List ADT

I originally planned to link consecutive card by utilising the `set_next()` method in Node ADT, during the process of the writing the game logic, I got drift away and used simple Boolean testing to test if the next card in the pile is the previous card – 1 and appended them into a list to return. My code would look much cleaner if I have utilised the Node ADT properly.

3.2.2. Buttons UI link up (dynamic function)

As stated in 1.1.2., the game allows moving consecutive cards as a whole. However, on the game screen (GUI), only the top selected card/button will give player feedback in confirming selection. The mechanism and rule of my game could be much clear if the cards/buttons that are valid can show player feedback at the same time.

3.2.3. Game AI that allows successful deck draw

The current game is run by importing random in Python. This is not the smartest solution in terms of the winnability of the game. My solution of the game would be much more rewarding if there is an AI ensuring the winnability.