BBM203: Data Structures Lab
2020 Fall
Assignment #(1) Report

Sümeyye Meryem Taşyürek, 21827871

November 20, 2020

1 Software Design Notes

1.1 Problem

- Our aim was to implement a Klondike Solitaire game according to given representation of the game board in our assignment.

1.2 Solution

My Approach to The Problem

- I have approached this problem by firstly taking into account the game rules. I have done some research about the game. After gaining some information, I have thought about what cannot be done in this game. These were my error possibilities. Then I decided to approach every section in the game board as classes with this I was able to create only one object for each part of the game board and I was able to reach the data of those sections through these objects. I have used methods for every command type. I put this methods into proper classes.

- For me the most important part of the problem was gaining information about the exact rules of our implementation, what should we display, what are invalid moves etc. So examining the given examples for this purpose was the most important part.

1.3 Class Diagram

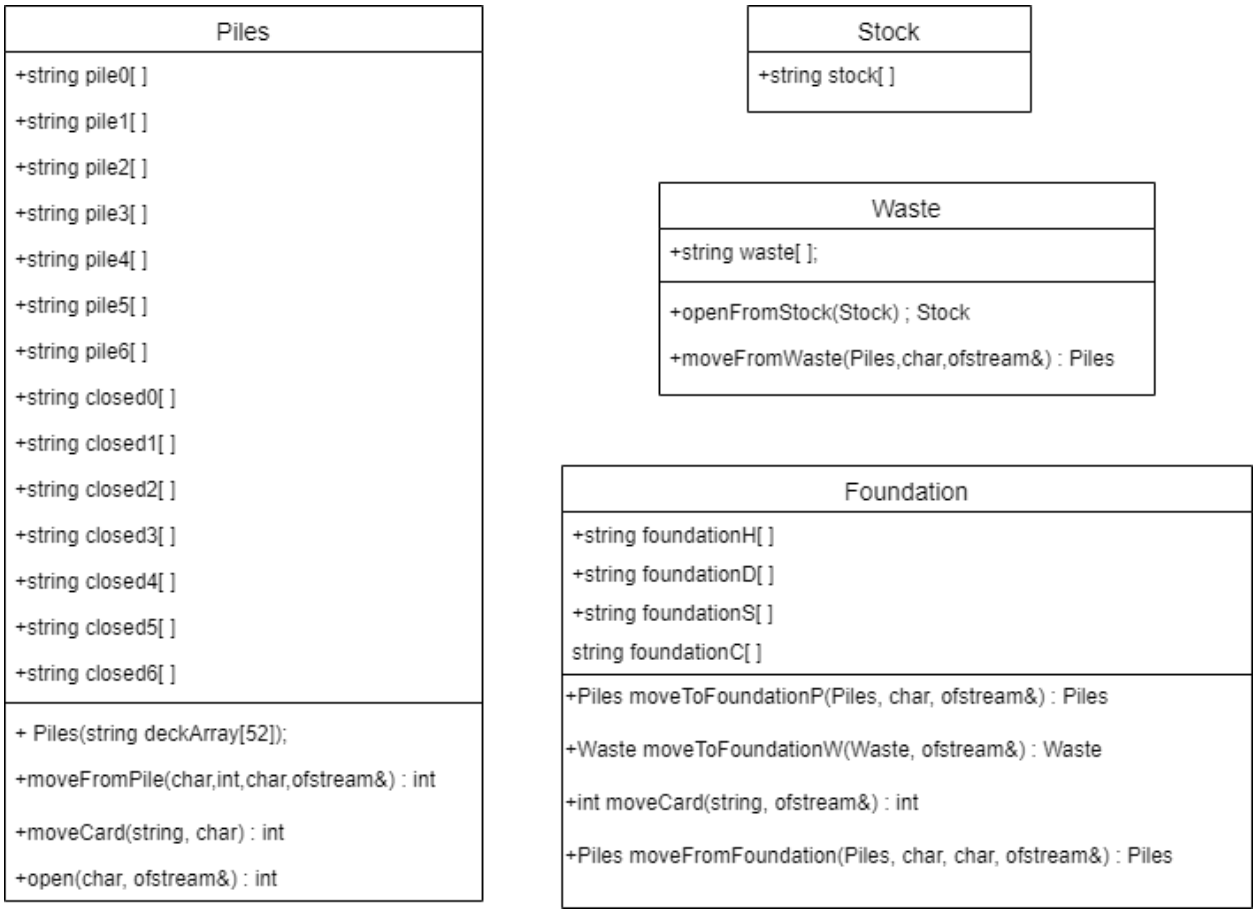

Figure 1: UML

1. Class Piles : This class represents the piles section on the game board. With the object creation all cards in the deck will be dealt into pile arrays and that object will keep the pile information.

2. Class Stock : This class represents the stock section. With the object creation the cards which is left from the piles will be put into stock array and object will keep stock information.

3. Class Waste : This class represents the waste section.With the object creation waste is initially empty. Waste information will be kept with this object.

4. Class Foundation : This class represents the foundation section. With the object creation foundations are initially empty. All information of the foundations will be kept with this object.

1.4 Array Usage

- I have used the array structure to store the card information in sections. Each class has needed array fields for that purpose.

- In class Piles there are 7 open piles array and 7 closed piles array. Open piles are needed for storing real card data for each pile. On the other hand closed piles are depends on these open piles. And closed piles are the printed ones. They are changing with the method calls according to commands.

- In class Stock there is only one array for keeping stock information. Change in the stock happens on that array.

- In class Waste there is again only one array for storing the card data in the waste. Change in the waste happens on that array.

- In Foundation class there are 4 foundation array each for hearts, diamonds, spades and clubs foundations. Every change in the foundations section are stored into these arrays.

- Also in print function I have traversed arrays in different ways to display correct version of the game board.

- My methods takes the objects of sections and changes the arrays in that sections according to given command.

- Because of arrays are fixed size data structures most of the time I had to traverse the arrays to reach all the meaningful data.