BBM203: Software Laboratory 1
Assignment 1 Report

Süleyman Ekmekçi, 21985921

November 17, 2020

1 Software Design Notes

1.1 Problem

Solitaire is a game that is played with 52 cards. This assignment is based on a classic version of the game, Klondike Solitaire. We are expected to read commands and deck from txt file. According to these commands, we have to update the game table by the rules.

1.2 Solution

1.2.1 Approach

As you can see in the UML diagram, I created different classes for each area on the game table. I created Card class so methods and operations could be done easily thanks to the features of this class. I used suit feature of card to remove card and if necessary, to put dummy to array so that arrays can be filled. To be more clear, if the suit of the card is 'n' it means that card is not valid, it is dummy. Moreover, it has is opened value which is true or false. Each class is responsible for its own commands. For example, Waste and Stock have their own arrays to hold cards, methods to transfer, remove and initialize first state of the game. In order to stick to the rules, I created validation functions so before each operation, my program checks if this operation is valid.

1.2.2 Class Diagram

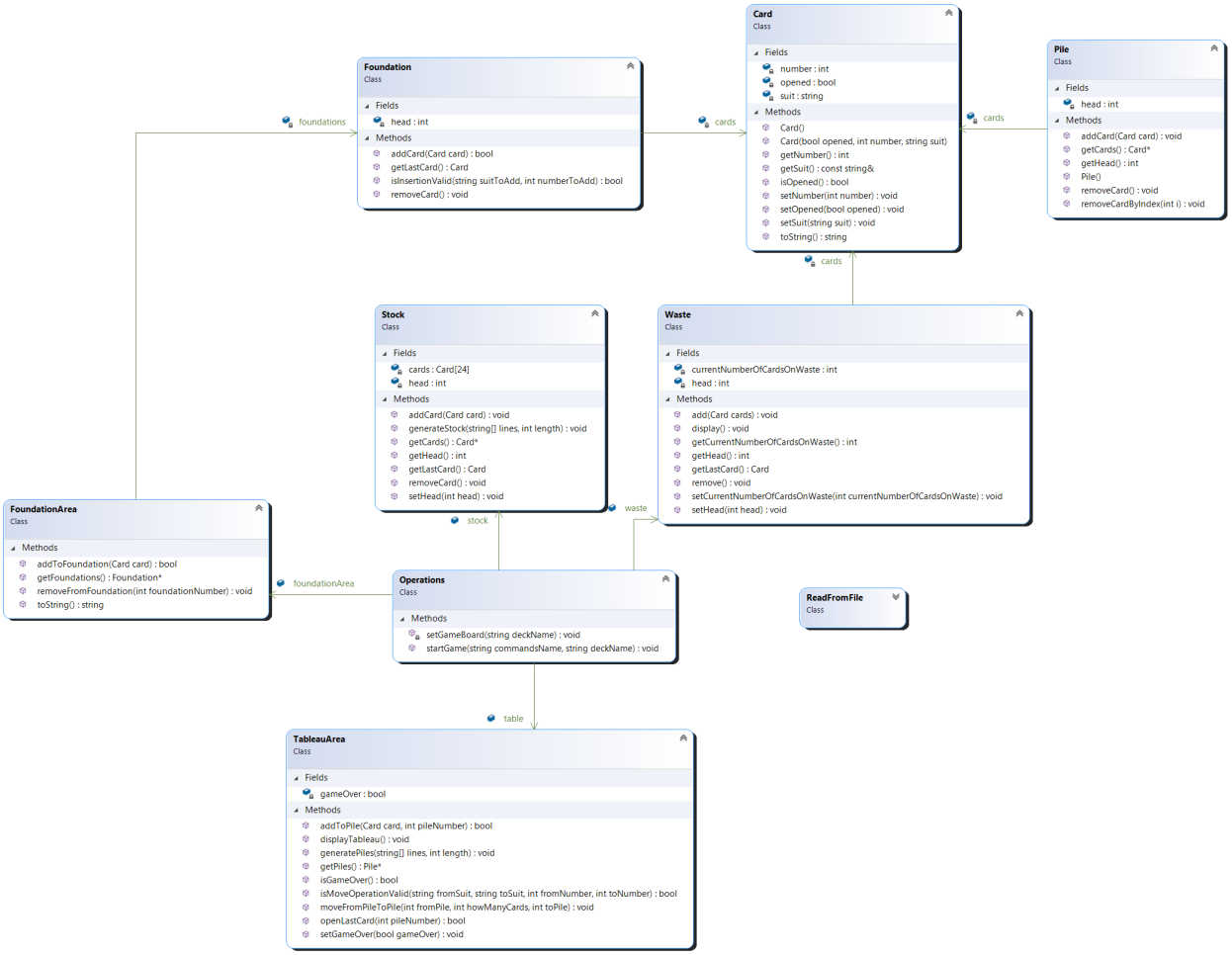

Figure 1: Class Diagram

Card

Card has number for its rank, suit for its type, opened for its current situation (opened or closed) and toString() method so that it can be easily printed.

Foundation

Foundation holds card array, and has addCard(), removeCard() and validation operations. Validation is basicly checks if card to add is appropriate for the foundation.

Foundation Area

Foundation Area holds foundation arrays, and has addToFoundation(), removeFromFoundation() and toString() methods. It adds and removes convenient cards based on rules.

Operations

Operations holds table, stock, waste and foundation area objects. It sets the board by calling these objects functions and starts game.

Pile

Pile holds card array and has addCard(),removeCard() and removeCardByIndex() (for multiple move operations) functions.

Tableau Area

This class holds array of piles. It controls flow of the piles and make validation operations to keep the rules from being broken.

Stock

This class holds array of cards. It communicates with waste and make add, remove and generate operations. Generate basicly initializes the first state of the game.

Waste

This class holds array of cards which come from stock. It communicates with stock and make add, remove and transfer operations. Transfer, transfers back all the cards from waste to the stock.

ReadFromFile

This class reads file and returns lines as an array of strings.

1.2.3 Where were the arrays used

I used arrays for almost every class. Simple summary of the class is like that, Waste and Stock have their own arrays of cards, Foundation Area has four Foundation arrays, which hold different arrays of cards. Last but not least, Tableau area has seven piles and each pile has arrays of cards. To maintain status of cards, protect order of cards and make operations easily, almost every class has its own array.