Assignment 2 (Worth 4%) Fall 2012

Given: Wednesday, October 3, 2012
Electronic Copy Due: Monday, October 15, 2012 (Midnight)

Objectives:
- Java and object oriented design.
- Junit
- Working with Stacks
- Using the built in java stack class

The Problem

You will be building a program that uses stacks to play the game of solitaire with the additional feature of an undo action. This feature will undo/reverse the previous actions the user has executed (one at a time as requested). The undo action can be done at any time and applied as many times as the user wants. So theoretically the user could play the game and undo every action they have done until they are back at the original starting point of the game.

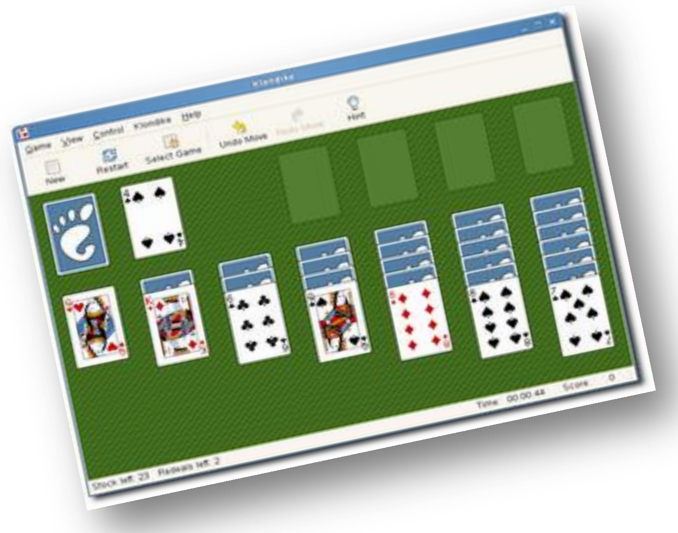
For the whole program you must use the built in java.util.Stack class and the provided code that handles the representation of the cards and the deck of cards. You are not allowed to make any changes to the Deck or Card class.

You should set up a proper object oriented design for the program, failure to do so will result in the loss of marks. You will also be required to implement Junit tests for specific features of your program. So you should think about separating your code for displaying the game and executing actions for the user.

The user should be allowed to make the following actions:
- Ask a card to be dealt
    - If the previous dealt card was not used (i.e. moved) then it should be added back into the deck.
- Move a dealt card to a pile or to a foundation
- Move a card between piles (just the top card)
- Move a card from a pile to a foundation (just the top card)
- Undo previous action (it can be called multiple times to continually undo a sequence of previous actions).
- End/Exit game

The Game of Solitaire

http://en.wikipedia.org/wiki/Klondike_solitaire



You will be implementing variant of Klondike Solitaire. Below are the rules as fond on the wiki page:

Taking a shuffled standard 52-card deck of playing cards (without Jokers), one upturned card is dealt on the left of the playing area, then six downturned cards (from left to right). On top of the downturned cards, an upturned card is dealt on the left-most downturned pile, and downturned cards on the rest until all piles have an upturned card. The piles can look like the figure to the right. For your program you will only have to display the top upturned card and a number to indicate how many face down cards are in the pile. (eg 4H (2) would mean the top card is a 4 of hearts and there are 2 hidden cards)

The four foundations (light rectangles in the upper right of the figure) are built up by suit from Ace (low in this game) to King, and the tableau piles can be built down by alternate colors/suit and smaller values (e.g. [6H] [5C] [4D][3S] H = heart, C = club, D=diamond, S = spade).

In this game you will only be able to either flip up a face down card or move a upturned card between piles or foundations as long as the move follows the proper rules of the game.

Any empty piles can be filled with a King and no other cards to start with. The aim of the game is to build up a stack of cards of a foundation starting with an ACE and ending with King, all of the same suit. If all foundations are filled the user wins the game.

For this game the deck will only deal one card at a time. The card can be used or placed back into the deck if not used. So the user can initiate a draw card action. If they then make a move action from the deck the card is used. If the user instead decides to draw another card the original should be placed back into the deck.

Design & Implementation

The program must use an object oriented design. The comments for each class should include a short description of why you created this class and where it fits into your design.

Junit Testing

http://www.junit.org/ (main site)

http://code.google.com/p/t2framework/wiki/JUnitQuickTutorial (learning about)

You must write j unit tests that verify the move action

- moving a dealt card to pile
- moving a card between piles
- moving a card between the pile and a foundation

Coding Style and Documentation

You must document your code. This documentation **must** include your name in every file you submit! In addition, basic documentation for methods and at the top of your main file is essential. Finally, poor style (e.g. incorrect indentation) will be taken into account in the grading.

Submission

Submit everything to the I Drive in a folder with your name and assignment 2 (eg JordanKidney_Asst2). Incorrect submissions will result in the loss of marks.