Name: Kristy Carpenter

Project name: Solitaire

Date: May 4, 2015

## Functional Overview

My project will be a solitaire game, very similar to the application that comes on most computers, and following the exact rules of traditional solitaire. It will run as a Java program, probably in jGrasp. If I can make it work as an applet or some sort of exportable program, I will do that.

This program is intended to be usable for anyone who knows how to play solitaire (although it will ideally include a help page which goes over the basic rules of the game).

## Design Overview

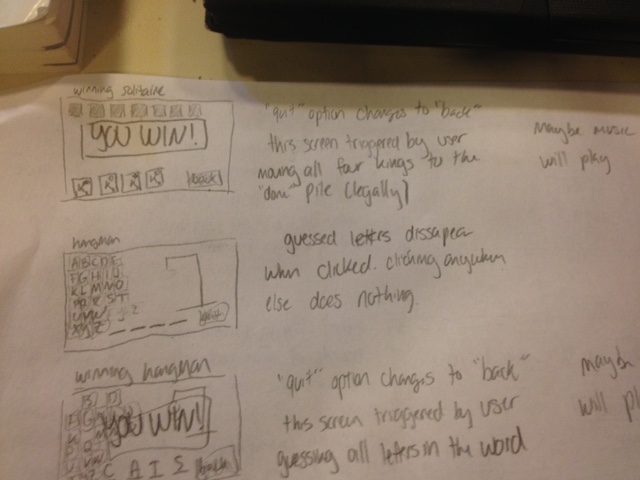
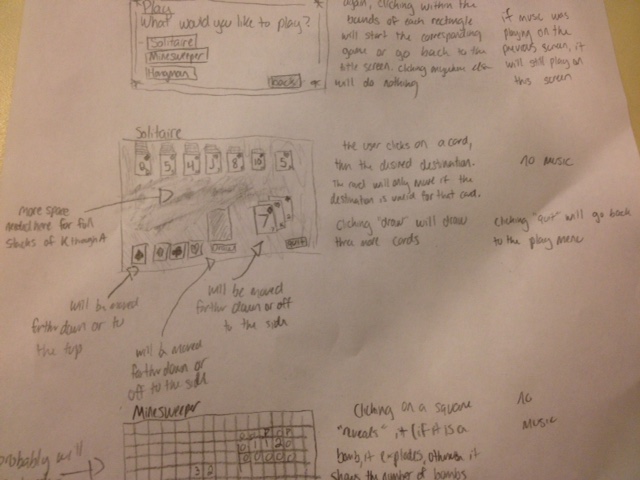
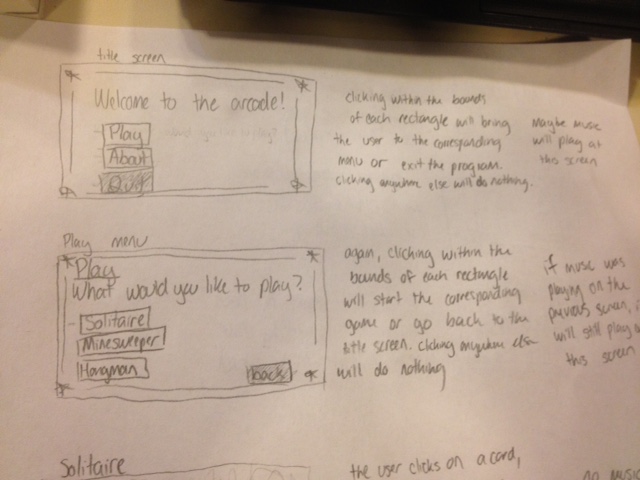
The user will interact with the program by clicking boxes. Each of the cards will be implemented with a class that is modified from Tile (from the Tiles assignment), and will be able to tell (with a private boolean) if a point within their boundaries has been clicked by the user. After clicking the desired card, the user will click the location to which the card will be moved. The program will check to see if the designated location is valid for the selected card; if so, the card will be moved there, and if not, the card will be de-selected.

The game begins with seven piles which have one to seven cards in them, with only the first card face up. The user can move any of these cards on top of another card on top of a pile if it has a value that is one higher and is of the opposite color. Beginning with ace and going up to king, the user can also move revealed cards on the tops of piles to the “done piles” (better name to be determined). The user can also deal out three cards at a time from the “deck” (better name to be determined), and may move the top card of these three revealed cards to any of the piles (if legal). The game is won when all four kings have been moved to the “done piles.” If the user has run out of possible moves to make, the game will not put up a lose screen, and it will be up to the user to terminate the game via a quit button which will either close the program or go back to some sort of title screen.

## Design Details

I am planning on using Java; the major challenging aspect of this project will be learning how to implement a GUI.

Below are some sketches of screens that will appear in the program, along with notes on what will specifically happen at each screen. Some of the included drawings will not be implemented (the original idea was an arcade with minesweeper and hangman in addition to solitaire).



Pseudocode design (not complete, but has most stuff):

**Clickable (interface)**

Methods:

* boolean isHitIint x, int y)

**Card implements Clickable**

Variables:

* boolean isClicked
  + self-explanatory
* Suit suit
  + Use an enum for this
* Int value
  + 1-13. 1 is Ace, 11 is Jack, 12 is Queen, 13 is King, 2-10 are those 2-10
* Point location
  + Top left corner coordinates
* Color color
  + Can only either be red or black

Methods:

* Card(Suit s, int v, Point l)
  + Set suit to s
  + If suit is club or spade
    - Color is black
  + Else
    - Color is red
  + Set value to v
  + Set location to l
  + Set isClicked to false
* boolean isHit(int x, int y)
* Color getColor()
  + Return color
* Suit getSuit()
  + Return suit
* Int getValue()
  + Return value
* //also needs to draw self, I don’t know if I want this implemented here or in Pile and Deck

**Pile implements Clickable**

Variables:

* Point location
  + Coordinates of top left corner
* Stack<Card> cardList
  + This list of cards in the Pile

Methods:

* Pile(Point l)
  + Sets location to l
  + Sets isEmpty to true
  + Initialize cardList to a stack
* boolean isHit(int x, int y)
  + if x and y are within the bounds of the pile
    - return true
  + return false
* boolean addCard(Card c)
  + if cardList is empty and c is a King
    - add c to the stack
    - return true
  + else if the last card in cardList is red and c is black or vice versa and the last card in cardList has a value that is one greater than c’s value
    - add c to the stack
    - return true
  + return false
* boolean isEmpty()
  + if cardList size is zero
    - return true
  + return false

**DonePile extends Pile**

Variables:

* Suit suit
  + Use an enum to designate suits

Methods:

* DonePile(Suit suit, Location l)
  + Call the parent class with l as a parameter
  + Set the suit instance variable to the suit parameter
* boolean isComplete
  + if the last card is a King
    - return true
  + return false
  + //this method assumes that the Cards are being correctly added to the cardList, i.e. that if a King is added then all the other cards have been added correctly
* boolean addCard(Card c)
  + if c’s suit matches suit and (c’s value is one greater than the last card in cardList or cardList is empty and c’s value is 1)
    - add c to the stack
    - return true
  + return false
* void removeCard()
  + if not empty
    - return last card in stack and remove that card from the stack
  + else
    - throw some exception

**Deck**

Variables:

* ArrayList<Card> fullDeck
  + All of the cards remaining in the deck, regardless of whether they are visible to the user and regardless of whether the user has seen them in this run through of the deck. Cards are only removed from this ArrayList if they have been successfully moved to a pile.
* ArrayList<Card> holding
  + Contains a maximum of three cards, minimum of zero. All of the cards in this ArrayList are visible to the user, but only the topmost card can be moved. When three more cards are dealt, the remaining cards in this ArrayList go to dealtUnseen, and are replaced by the dealt cards.
* ArrayList<Card> undealt
  + These cards cannot be seen by the user. They are in the deck, but have not yet been dealt in this run through. The top three cards from this ArrayList get moved to the holding ArrayList each time the deal button is pushed.
* ArrayList<Card> dealtUnseen
  + These cards cannot be seen by the user. When cards are dealt from undealt to holding, the cards that were in holding go here. The only way to get cards out of dealtUnseen is to empty holding (then the top card comes out of dealtUnseen and goes to holding) or to empty undealt (then the entirety of dealtUnseen goes back to undealt)
* boolean isEmpty

Methods:

* Deck(ArrayList<Card> deck)
  + //assumes the deck parameter is already shuffled
  + Set fullDeck to deck
  + Initialize holding to an empty ArrayList<Card> with size three
  + Initialize undealt to an empty ArrayList<Card>
  + Initialize dealtUnseen to an empty ArrayList<Card>
  + Initialize isEmpty to false
* void deal()
  + if it is not empty
    - if there cards in undealt
      * move contents of holding to dealtUnseen
      * if undealt has more than two cards
        + move top three cards in undealt to holding
      * else
        + move all cards in undealt to holding
    - else
      * move all cards in dealtUnseen to undealt
* boolean holdingHasStuff()
  + if holding size is zero
    - Return false
  + Return true
* Void drawDeck
  + If there are cards in undealt
    - Draw generic card back in the undealt area
  + If there are cards in holding
    - Draw the bottom card at bottom place
    - Draw the next card (if there is one) slightly offset
    - Draw the top card (if there is one) slightly offset

**Solitaire**

Methods:

* main
  + Boolean hasWon is false
  + Boolean quit is false
  + create new arraylist unshuffledDeck
  + for each suit in enum suit
    - for 1 to 13
      * create card with suit and value and add to unshuffledDeck
  + shuffle unshuffledDeck
  + create 7 piles and fill them with 1-7 cards
  + put the rest of the deck in deck
  + draw the screen
  + while hasWon is false and quit is false
    - GUI stuff… clicking things and checking if things can go places…
    - This part is not fleshed out at all
    - If quit is pressed, then quit is true
    - If all four donepiles are complete
      * hasWon is true
  + If hasWon is true
    - Display win text

## Testing

I know that I will find more edge cases to test as I develop this program, but here are the things that I plan to test at this time:

* Clicking on parts of the screen that are not covered by cards or buttons (if buttons are implemented), and making sure that nothing happens
* Clicking on a card in each of the positions where it can be legally moved. Does it change to its clicked appearance?
* Clicking on a card in each of the positions where it is visible as a card, but cannot be legally moved (in piles facedown, underneath another card in the deck). Does it change to its clicked appearance (it shouldn’t)? Can a clickable card be successfully clicked?
* What happens when a card is clicked twice in a row? (It should become selected and then de-selected)
* Click a card, then click a place where it cannot be legally moved (empty space, button, deck, wrong done pile, wrong pile). It should become deselected and nothing should move. Make sure to test cases where the numbers work (nine on top of ten) but the suits don’t (heart on diamond) and vice versa.
* Click a card, then click a place where it can be legally moved (correct pile, correct done pile). Does it work?
* Does the game set itself up correctly? Does the game set itself up correctly after a previous game ends? Do the cards actually get shuffled? Do the cards actually go to the correct places?
* Attempt to move multiple cards that are legally stacked to a new legal position. Do they all move?
* Does the deal button work? Click it a lot. Does it still work? Move a card to a pile. Click the button more. Does it still work? Do the cards stay in order? Do the cards get consistently dealt in threes? What happens when the number of cards remaining in the deck is not divisible by three? What happens when all three cards in the holding area are legally moved to piles? Does something weird happen when the holding area is clicked? Does the button still work?
* Does the quit function work?
* Can cards be successfully moved to the done pile? What about off the done pile?
* Do cards reveal themselves when all the visible cards of a pile are moved elsewhere?
* Can Kings be moved to blank spaces? Can other cards be moved to blank spaces?
* Can Aces be moved to empty done piles? Can other cards be moved to empty done piles?
* Make sure cards can only go into the done piles of their own suit. Make sure they can only go into the done pile if the correct preceding card is already there.
* What happens when one king gets moved to the done pile? Two? Three? Four? When does it trigger the win screen, if ever?
* Can cards be selected once the win screen is triggered? (they shouldn’t be)
* What happens to the quit button once the win screen is triggered? Does it still work?
* Make a complete pile, from king to ace. Does it fit on the screen? Can the entire pile be moved? Does the whole pile fit on the screen in every possible pile location?
* What happens with a right click? (nothing should happen) What about literally any input other than a left click (spacebar, typing, shift click, etc.)? (nothing should happen)

I will also hand off prototypes and the finished product to my dad over the course of this project, as he does QA for a living and will be more than willing to try to break everything possible in my program.

## Grading Rubric

Style/Documentation: 6 points

* 2 pts: Coding standards are adhered to
* 2 pts: Code is readable, no magic numbers
* 2 pts: Meaningful Javadoc is present

External Correctness: 24 pts

* 4 pts: Game sets itself up correctly
* 2 pts: Cards have consistent appearance
* 5 pts: Cards move if and only if the selected location is valid
* 1 pts: Cards change to their selected appearance if and only if clicked when nothing is selected
* 2 pts: Win screen appears when game has been legally won
* 6 pts: The deal function works fully
* 2 pts: The quit button works
* 2 pts: Cards can move from one pile to another if legal

Internal Correctness: 10 pts

* 2 pts: Reasonably efficient
* 3 pts: Code compiles
* 5 pts: No exceptions get thrown

## Proposed Implementation Schedule

5/8-5/9: Finalize terminology for each location in the game. Look up if there are actual solitaire terms for all the different piles. Create a labeled diagram with the terms so as not to confuse yourself in naming variables, methods, conducting tests, and using this document. Create each of the classes to be used in the program and put in all variable names and method headers. Include JavaDoc/pseudocode descriptions of each one.

5/9-5/11: Read Building Java Programs chapter 14. Learn exactly what a GUI is and if it needs to be implemented. Figure out how the Tiles program worked, figure out how the GUI relates to the DrawingPanel (are they two entirely different things? Can a GUI be used within a DrawingPanel? Can I forego the GUI and somehow incorporate user interaction into a DrawingPanel?). Begin to conceptualize how the program will be put together.

5/12-5/14: Modify the Tile class and turn it into a Card class. Finalize design of Cards and figure out how to make each card’s appearance be encapsulated in instance variables (suit, color, value). Figure out how to draw images of suit symbols and put them on the card. Figure out the back design of each card.

5/14-5/15: Test basic functions of the Card class. Put some Cards in an otherwise empty DrawingPanel/GUI and ensure that they have the right appearance. Finalize layout of game screen—where will the piles go, where will the deck go? Make sure it looks nice and that the longest possible pile can fit on the screen without covering up any other part.

5/15-5/16: Create Pile.java and DonePile.java (or whatever their final names will be). Test their basic functions without any Cards. Do they go to the right place? Create the win screen and hardcode it into Solitaire.java, make sure it works when called. Make sure it disables the ability to select Cards.

5/17-5/20: Begin implementing user interaction. Do it a little bit at a time: start with getting Cards to recognize when they are clicked, then get Cards to move to empty Piles, then get only Kings to move to empty Piles, then get Cards to move to DonePile, then get only Aces to move to empty DonePile, then get all Cards to move to the correct DonePile at the correct time, then get Cards to move on top of other Cards with the correct conditions. Do all this testing by instantiating new Cards in the DrawingPanel/GUI and forcing them to be convenient values for each test case.

5/20-5/21: Implement the deck and the deal function. Get it to deal out three cards at a time, and then start over when the deck runs out of cards.

5/21-5/22: Get the deck to work with the piles. Enable Cards from the holding area to be moved to legal locations on top of other Cards in Piles. Enable Cards from the holding area to be moved to a legal DonePile. Ensure that Cards from the holding area cannot be moved to illegal Piles or DonePiles. When all four kings are in their respective DonePiles, make sure the win screen is triggered.

5/23-5/24: Get the game to set up correctly. Implement shuffle method and enable function that deals cards out to the Piles and then put the rest in the Deck. Ensure that all the moving functions still work when the game is in its actual set up. Test specific cases by rigging the deck instead of shuffling it and making sure you can test all the possible situations. If not implemented already, put in the quit option and make sure it works.

5/24-5/26: Make it so that stacks of cards can be moved from Pile to Pile legally. If the facedown/faceup capability hasn’t been attended to properly, do it now. Make sure the Cards know if they are facedown or faceup and when they should change from one to the other. Identify all problems with the program and decide how and when to fix them.

5/26-5/27: Last minute testing and fixes. JavaDoc if this hasn’t happened yet. If time, learn how to export program as an applet and try to do that.

Other things to implement if time/if I end up ahead of schedule (in order of feasibility): title screen, help screen, music/sound effects, undo function, lose screen (and knowledge of when the user has lost)

## Potential Showstoppers

* Learning how to implement a GUI is going to be the most critical part of this project, as well as the most difficult. I think this is the one part of the project that I have absolutely no knowledge of right now.
* One important part of solitaire that will be difficult to implement is moving stacks of cards around (for example, the seven of spades and the six of hearts). If I cannot figure out how to make this work, the game won’t truly be solitaire

## Open Questions

* This has been mentioned previously in this design but: are GUIs completely different from DrawingPanels, or are they implemented in DrawingPanels? How exactly did the Tile program work—that seemed to work with a DrawingPanel, did it also use a GUI? Should I use a DrawingPanel instead of a GUI?