Philip Warton

CS 162

January 25, 2020

Assignment 2 Design Document

Understanding the Problem:

Summary. We are being asked to write a program that allows two users to play the game Crazy Eights. This game works functionally like Uno where the eight can be used as a wild card. To win the game, a player must have an empty hand, or have the least number of cards in their hand when the deck runs out. We must use classes and object-oriented design in our program with classes for cards, decks, hands, players, and games. Users must have the option to place which ever card down they like, and be forced to draw a card if theirs is rejected. If a player plays an eight, they must be allowed to choose which suit they would like it to represent.

Assumptions:

- I am assuming players do not cheat in this game and look at the other players' hand
- I am assuming the players do not intend to play the game with a deck that includes joker
- I am assuming that players input cards that are in their hand

Design:

(see next page)

Program Starts
↓
begin while loop (*)

game object created { creates + shuffles deck obj of 52 cards
creates array of 2 playe

deal cards to both players

Deck	P1 hand	P2 hand
[52] →	[0]	[0] ×7

create stack of cards on table

Deck	Stack
[52-14] →	[0]

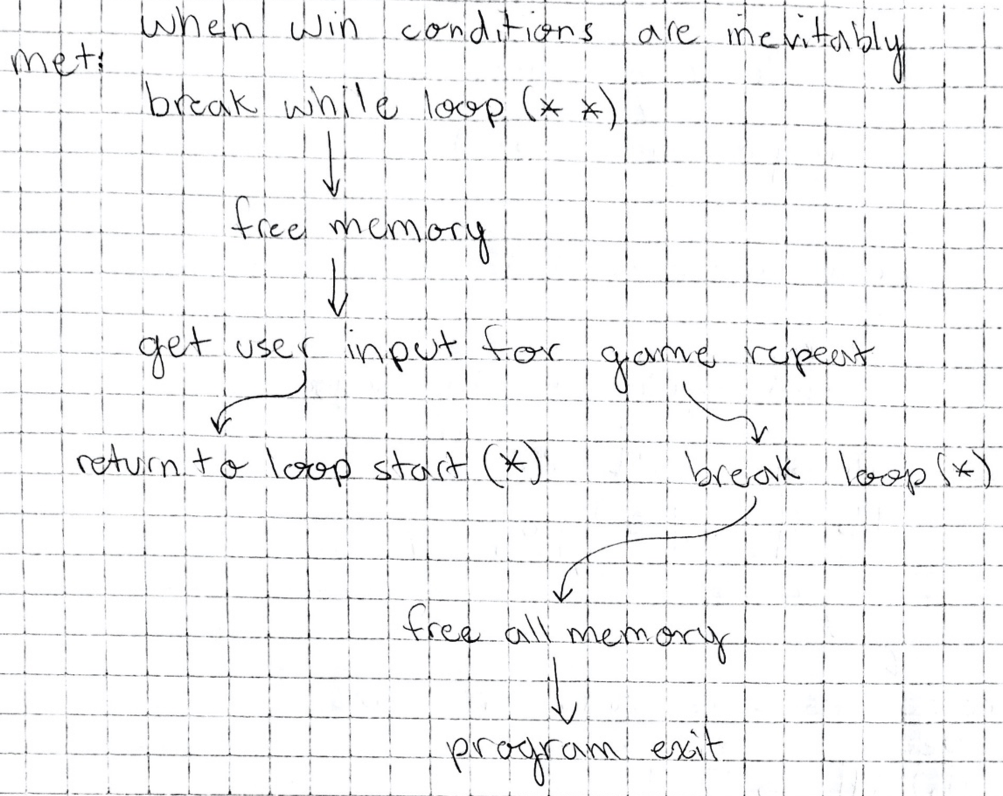
↓
print top of stack

begin while loop (**)

player 1 turn
↓
check win conditions
↓
player 2 turn
↓
check win conditions

Player turn
- (i) print player hand
- (ii) get input
- (iii) place on pile/ give player another card

Win Conditions
- (i) player has empty hand
- (ii) deck is empty



met: when win conditions are inevitably

break while loop (* *)

free memory

get user input for game repeat

return to loop start (*)

break loop (*)

free all memory

program exit

CLASSES

```
card:
  states
    rank
    suit
  methods
    print()

deck
  states
    card[52]
    cards_left
  methods
    take top card()
    is_empty()

hand
  card[ ] (dynamic)
  cards_left
  methods
    add_card()
    take_card()
    is_empty()

player
  states
    hand
    nama
  methods
    print hand()
    add_card()
    take_card()

game
  states
    player[2]
    deck
    stack
  methods
    play()

stack
  states
    cards[52]
    cards_left
  methods
    add_card()
    print_top()
    check_match
```

Testing:

Function	*Case*	*Case Type*	*Outcome*
Check_match	In: Ace of Spades Pile: Queen of hearts	Bad	Player draws a card from the deck
Check_match	In: 4 of hearts Pile: Queen of hearts	Good	Player places card from hand on pile
Check_match	In: Queen of diamonds Pile: Queen of hearts	Good	Player places card from hand on pile
Check_match	In: 0 of clubs Pile: 2 of clubs	Edge	Throw error: card should not exists, exiting program
Get_card_from_hand Print: please which card you would like to try to place	1	Good	Attempts to place first card in hand down
Get_card_from_hand	0	Edge	Error, input must be between 1 and n_cards
Get_card_from_hand	-1	Bad	Error, input must be a positive integer
Get_repeat_input	Input: yes	Bad	Must enter 1 or 0
Get_repeat_input	Input: “ ”	Edge	Error: must input 1 or 0
Get_repeat_input	Input: 010	Edge	Error: must input 1 or 0
Get_repeat_input	Input: 1	Good	Set repeat to true