SFWRENG 3XA3: Module Interface Specifications
Rummy For Dummies

Lab 2 Group 7, Rummy For Dummies
Joy Xiao, xiaoz18
Benson Hall, hallb8
Smita Singh, sings59

March 18, 2021

Table 1: **Revision History**

Date	**Version**	**Notes**
March 16, 2021	1.0	Started on the MIS
March 18, 2021	1.1	Finished the MIS

1 Module Hierarchy

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Game Operations Module
	Input Module
	Melds Module
Software Decision Module	Computer Module
	Stock Pile Data Structure Module
	Discard Pile Data Structure Module
	Card Data Structure Module
	Hand Data Structure Module
	Player Data Structure Module

Table 2: Module Hierarchy

2 MIS of Card Data Structure Module

2.1 Interface Syntax

2.1.1 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
Card	Suit, integer	-	-
getSuit	-	Suit	-
getRank	-	integer	-
points	-	integer	-
toString	-	String	-
toSymbol	-	String	-

2.2 Interface Semantics

2.2.1 State Variables

rank: int - rank of card
suit: Suit - suit of card

2.2.2 Environment Variables

Not applicable

2.2.3 Assumptions

Values of variables should be set before they are able to be accessed

2.3 Access Program Semantics

getSuit():

- Input: None
- Transition: accesses the suit value
- Output: returns the suit of the card
- Exception: None

getRank():

- Input: None
- Transition: accesses the rank value
- Output: returns the rank of the card
- Exception: None

points():

- Input: None
- Transition: determines the point of the card based on card rank
- Output: returns the point value of the card
- Exception: None

toString():

- Input: None

- Transition: determines the string representation of the card
- Output: returns the string representation of the card
- Exception: None

toSymbol():

- Input: None
- Transition: determines the string representation of the rank and symbol representation of suit
- Output: returns the symbol representation of the card
- Exception: None

3 MIS of Computer Module

3.1 Interface Syntax

3.1.1 Exported Access Programs

Routine Name	In	Out	Exceptions
makeMove	Player, StockPile, DiscardPile	Boolean	EmptyStackException

3.2 Interface Semantics

3.2.1 State Variables

Not applicable

3.2.2 Environment Variables

Not applicable

3.2.3 Assumptions

- The class is a static class with static methods to be accessed without initialization.
- Drawing from the discardPile should never result in any exceptions since there is always at least one card in the discardPile.

3.3 Access Program Semantics

makeMove(player, stockPile, discardPile):

- Input: None
- Transition: If adding the discard pile card into the hand results in a lower deadwood score, draw from the discard pile and remove the card with the highest deadwood score from hand. Or else, draw from the stock pile and remove the card with the highest deadwood score from hand.
- Output: Returns true if deadwood score is 10 or less, otherwise return false
- Exception: Drawing from stock pile with 0 cards will result in a EmptyStackException

4 MIS of Discard Pile Data Structure Module

4.1 Interface Syntax

4.1.1 Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
DiscardPile	-	-	-
displayTopCard	-	-	-

4.2 Interface Semantics

4.2.1 State Variables

Not applicable

4.2.2 Environment Variables

Not applicable

4.2.3 Assumptions

- The DiscardPile() constructor is called for each object instance before any other access routine is called for that object. The constructor can only be called once. It will originally contain 0 cards before the cards are dealt. The class will extend Stack(Card) for functionality.
- The displayTopCard() will return the string representation of the top card in the discard pile. There is no exception because it is assumed that the discardPile will always have a size greater or equal to 1 at the start of the round.
- The DiscardPile class will be extending Stack from java.util.

4.3 Access Program Semantics

DiscardPile():

- Input: None
- Transition: Initializes the DiscardPile object
- Output: None
- Exception: None

displayTopCard():

- Input: None
- Transition: None
- Output: Displays a visual representation of the top card in the pile onto the console
- Exception: None

5 MIS of Game Operations Module

5.1 Interface Syntax

5.1.1 Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
calculateScores	Player, Player	-	-
chooseDiscard	Player	String	-
createStockPile	-	Stock Pile	-
createDiscardPile	-	Discard Pile	-
discardCard	Player, Discard Pile, String	-	-
distributeCards	Player, Player, Stock Pile, Discard Pile	-	-
drawFromStockPile	Player, Stock Pile	-	-
drawFromDiscardPile	Player, Discard Pile	-	-
endGame	-	-	-
playAgain	-	character	-
processDecision	Player, Stock Pile, Discard Pile	boolean	-
resetEverything	Player	-	-
username	-	String	-

5.2 Interface Semantics

5.2.1 State Variables

Not applicable.

5.2.2 Environment Variables

Not applicable.

5.2.3 Assumptions

- All variables and objects have been instantiated

5.3 Access Program Semantics

calculateScores(p1, cpu):

- Input: Two player objects
- Transition: Determines winner of the round and calculates score to add to winner's total score
- Output: None
- Exceptions: None

chooseDiscard(p1):

- Input: Player object
- Transition: Displays cards in hand that can be discarded and prompts user for a response.
- Output: String representation of a card
- Exceptions: None

createStockPile():

- Input: None
- Transition: Creates the initial stock pile
- Output: New Stock Pile
- Exceptions: None

createDiscardPile():

- Input: None
- Transition: Creates the initial discard pile
- Output: New Discard Pile

- Exceptions: None

discardCard(p1, dp, discard):

- Input: Player object, discard pile, and string representation of card to discard
- Transition: Discards the card from the player's hand and places it on top of the discard pile
- Output: None
- Exceptions: None

distributeCards(p1, cpu, sp, dp):

- Input: Two player objects, stock and discard piles
- Transition: Sets up the opening distribution of cards for a new round
- Output: None
- Exception: None

drawFromStockPile(p1, sp):

- Input: Player object, stock pile
- Transition: Takes the top card off the stock pile and adds it to the player's hand
- Output: None
- Exception: None

drawFromDiscardPile(p1, dp):

- Input: Player object, discard pile
- Transition: Takes the top card off the discard pile and adds it to the player's hand
- Output: None
- Exception: None

endGame():

- Input: None
- Transition: Closes all resources
- Output: None

- Exception: None

playAgain():

- Input: None
- Transition: Prompts for user's affirmation or refusal to play a new game of Gin Rummy
- Output: User's affirmation or refusal to play
- Exception: None

processDecision(p1, sp, dp):

- Input: Player object, stock and discard piles
- Transition: Performs the user's desired move for the turn
- Output: True if player knocks, false otherwise
- Exception: None

resetEverything(p1):

- Input: Player object
- Transition: Resets player's hand, melds and deadwood score
- Output: None
- Exception: None

username():

- Input: None
- Transition: Prompts for user's username
- Output: User's username
- Exception: None

6 MIS of Hand Data Structure Module

6.1 Interface Syntax

6.1.1 Exported Access Programs

Routine Name	In	Out	Exceptions
Hand	-	-	-
displayHand	-	-	-
contains	String	Boolean	-
remove	String	Card	-

6.2 Interface Semantics

6.2.1 State Variables

Not applicable

6.2.2 Environment Variables

Not applicable

6.2.3 Assumptions

- The Hand class extends ArrayList from java.util.
- The Hand is initialized to an empty ArrayList with type Card.

6.3 Access Program Semantics

Hand():

- Input: None
- Transition: Initializes a new Hand object
- Output: None
- Exception: None

displayHand():

- Input: None
- Transition: None
- Output: Displays a visual representation of the hand onto the console
- Exception: None

contains(c):

- Input: String representation of a card
- Transition: None
- Output: Returns true if the hand contains the card
- Exception: None

remove():

- Input: String representation of a card
- Transition: Creates new hand with the specific card removed
- Output: Returns the card that is removed from the hand
- Exception: None

7 MIS of Melds Module

7.1 Interface Syntax

7.1.1 Exported Access Programs

Routine Name	In	Out	Exceptions
Meld.checkMelds	Hand	2D ArrayList of Card	-
sortByRank.compare	Card, Card	Integer	-
sortBySR.compare	Card, Card	Integer	-

7.2 Interface Semantics

7.2.1 State Variables

Not applicable

7.2.2 Environment Variables

Not applicable

7.2.3 Assumptions

Hand of cards has 10 cards.

7.3 Access Program Semantics

Meld.checkMelds(h):

- Input: Hand h with 10 cards
- Transition: Computes sequence and group melds and finds the best meld groups leading to least deadwood score
- Output: returns a 2D-ArrayList of Cards representing groups of melds
- Exception: None

sortByRank.compare(c1,c2):

- Input: Two different cards
- Transition: Compares the ranks of c1 and c2
- Output: returns negative integer if c2 is larger than c1 and returns positive integer if c1 is larger than c2
- Exception: None

sortBySR.compare(c1,c2):

- Input: Two different cards
- Transition: Compares the suits of c1 and c2 and if they're the same compares the rank
- Output: returns negative integer if c2 is ranked higher than c1 and returns positive integer if c1 is ranked higher than c2
- Exception: None

8 MIS of Player Data Structure Module

8.1 Interface Syntax

8.1.1 Exported Access Programs

Routine Name	In	Out	Exceptions
Player	String	-	-
getName	-	String	-
getHand	-	Hand	-
getTotalScore	-	integer	-
getDeadwoodScore	-	integer	-
getMelds	-	2D ArrayList of Card	-
addCardToHand	Card	-	-
discardFromHand	String	Card	IllegalArgumentException
addToTotalScore	integer	-	-
extractDeadwood	-	ArrayList of Card	-
recalculateDeadwoodScore	-	-	-
checkMelds	-	-	-
displayHand	-	-	-
resetHand	-	-	-
resetDeadwoodScore	-	-	-
resetMelds	-	-	-

8.2 Interface Semantics

8.2.1 State Variables

name: String - player's name
hand: Hand - data structure representing player's hand
totalScore: int - player's total score in the game
deadwoodScore: int - player's deadwood score for the round
melds: 2D ArrayList of Card - player's current melds from the hand

8.2.2 Environment Variables

Not applicable.

8.2.3 Assumptions

- The user's name is not empty.

8.3 Access Program Semantics

Player(name):

- Input: String representing the player's name
- Transition: Initializes the Player object
- Output: None
- Exception: None

getName():

- Input: None
- Transition: None
- Output: Player's name
- Exception: None

getHand():

- Input: None
- Transition: None
- Output: Player's current hand
- Exception: None

getTotalScore():

- Input: None
- Transition: None
- Output: Player's total score
- Exception: None

getDeadwoodScore():

- Input: None
- Transition: None
- Output: Player's current deadwood score
- Exception: None

getMelds():

- Input: None
- Transition: None
- Output: Player's current melds
- Exception: None

addCardToHand(c):

- Input: A card
- Transition: Adds the card to the player's hand
- Output: None
- Exception: None

discardFromHand(input):

- Input: String representation of a card to discard from the hand
- Transition: Remove the card that represents the input from the hand, if it exists, and return it.
- Output: Card that was discarded
- Exception: IllegalArgumentException if the card does not exist in the hand

addToTotalScore(points):

- Input: Points to add to the total score
- Transition: Points are added to the player's total score
- Output: None
- Exceptions: None

extractDeadwood():

- Input: None
- Transition: Create a list of cards in the hand that are not in a meld.
- Output: ArrayList of deadwood cards
- Exception: None

recalculateDeadwoodScore():

- Input: None
- Transition: Recalculate and update the current deadwood score, given the current hand
- Output: None
- Exceptions: None

checkMelds():

- Input: None
- Transition: Update the current melds in a player's hand
- Output: None
- Exceptions: None

displayHand():

- Input: None
- Transition: Display the player's current hand
- Output: None
- Exceptions: None

resetHand():

- Input: None
- Transition: Reset the player's hand
- Output: None
- Exceptions: None

resetDeadwoodScore():

- Input: None
- Transition: Reset the player's deadwood score
- Output: None
- Exceptions: None

resetMelds():

- Input: None
- Transition: Reset the player's current melds
- Output: None
- Exceptions: None

9 MIS of Stock Pile Data Structure Module

9.1 Interface Syntax

9.1.1 Exported Access Programs

Routine Name	In	Out	Exceptions
StockPile	-	-	-
search	Object	integer	-

9.2 Interface Semantics

9.2.1 State Variables

Not applicable

9.2.2 Environment Variables

Not applicable

9.2.3 Assumptions

- The StockPile class will be extending Stack from java.util.
- The search(c) will override the original method in the API. It will always return -1 since searching for a card in the stock pile should not be done at anytime in the game.

9.3 Access Program Semantics

StockPile():

- Input: None
- Transition: Initializes the StockPile object
- Output: None
- Exception: None

search(obj):

- Input: Any object
- Transition: None
- Output: returns -1
- Exception: None

10 MIS of Input Module

10.1 Interface Syntax

10.1.1 Exported Access Programs

Routine Name	**In**	**Out**	**Exceptions**
chooseDiscard	-	String	-
closeScanner	-	-	-
knock	-	character	-
playAgain	-	character	-
playerDecision	-	integer	-
username	-	String	-

10.2 Interface Semantics

10.2.1 State Variables

scanner: Scanner - Object used to read user keyboard input

10.2.2 Environment Variables

keyboard: external device for user to provide String input when prompted by the game

10.2.3 Assumptions

- User input comes from active keyboard input.
- Input stream is available throughout operation of the program.

10.3 Access Program Semantics

chooseDiscard():

- Input: None
- Transition: Receive user input on card to discard
- Output: User's input of the card to discard
- Exceptions: None

closeScanner():

- Input: None
- Transition: Closes scanner state variable access
- Output: None
- Exceptions: None

knock():

- Input: None
- Transition: Takes the user input on if the player wants to knock, and validates it
- Output: User's affirmation or refusal to knock
- Exception: None

playAgain():

- Input: None
- Transition: Takes the user input on if the player wants to play again, and validates it
- Output: User's affirmation or refusal to play a new game

- Exception: None

playerDecision():

- Input: None
- Transition: After showing the available options for the user to make, takes user input and validates it
- Output: User's decision for the turn
- Exceptions: None

username():

- Input: None
- Transition: Takes user input
- Output: User's input for their username
- Exception: None