Shruti Gogna
Monday, January 15, 2024.
ICS4U1-1
Mrs. Kapustina

FINAL - Classes and Methods Description

Classes

`BackgroundPanel`

The BackgroundPanel class sets up the background image for various screens of the game by importing the images from files.

`Card`

The Card class creates a Card object and stores all information about it (i.e. the suit and value of the card). This class does not include the GUI portion of the object.

`CardImage`

The CardImage class creates a JPanel that generates what is meant to be an image of a card based on the Card, width, and height of the desired image. This is the GUI portion of the Card object.

`Deck`

The Deck class stores the attributes (i.e. the number of cards currently in the deck) and behaviour (ex. resetting the deck) for a deck of 52 cards.

`GamePanel`

The GamePanel class sets the screen for the playable game by setting up the board with all the players and their variables. The class implements ItemListener and uses ActionListener to complete tasks (ex. Picking up and dropping cards). It also includes the computer players' strategy to determine what cards to drop.

`GameWindow`

The GameWindow class sets the frame for the game. It contains a main method to be the first class that is run when the program starts. It sets the default screen to be the MenuPanel.

`Hand`

The Hand class creates a Hand object and stores all information about an ArrayList of cards. It allows many actions to be performed with a hand of cards (ex. Sorting a hand, removing a card, etc.).

`InstructionsPanel`

The InstructionsPanel sets the screen to display the instructions of the game.

`LeaderboardPanel`

The LeaderboardPanel sets the screen to display the end game information (including who won, ranking of players, etc.)

`MenuPanel`

The MenuPanel sets the home screen of the game. The user is allowed to view the InstructionPanel or GamePanel from actions done in this panel.

`PlayerCardPanel`

The PlayerCardPanel stores the CardImages for the hand of each player. This object will get updated for every change done to a player's hand.

Methods

BackgroundPanel

`paintComponent`

Update the screen to add the background image

Card

`getSuit`

Returns the suit of a card or returns the suit at the index in an array of suits (two overloading methods)

`getSuitRank`

Returns the rank of a suit based on the suit.

`getValue`

Returns the value of a card or returns the value at the index in an array of value (two overloading methods)

`toString`

Returns a String including all instance variables of a Card

`equals`

Checks if two Cards are equal. Returns a boolean.

CardImage

`loadImage`

Load an image (suit and/or rank) based on file name, width and height. Returns an ImageIcon

Deck

`reset`

Reset a deck with 52 new cards

`replace`

Replace the current deck with a new deck (ArrayList) of Cards

`isEmpty`

Checks if a deck is empty. Returns a boolean.

`shuffle`

Shuffle the current deck.

`draw`

Draw a Card from the Deck. Returns the Card drawn.

GamePanel

`turn`

Play one turn of the game (i.e. one player gets to complete their turn)

`compTurn`

Control the computer player's turn (i.e. what card(s) they draw or place)

Strategy involved: the computer will place all other cards except for special cards. The computer will stack as many cards as possible (except for special cards) as opposed to dropping only 1 card at a time. This requires the player's Hand and their position in the array of players.

`updateChoices`
Update the user's choices (i.e. the checkbox with their cards). Requires a boolean that stores whether the player has already dropped one card in this turn.

`itemStateChanged`
Implement itemStateChanged method for the user's Cards checkboxes. This requires an ItemEvent.

`start`
Start the game and initialize the board. This method requires a GamePanel

`replaceDeck`
Replace the Deck when empty by taking the discardPile cards as the new Deck.

`updateTopCard`
Update the topCard image, the validSuit, and the validValue. This requires the Card that was just dropped, and a boolean to determine if the card was dropped to set up the board or not.

`getTopCard`
Returns the Card that is at the top of the discardPile.

`gameOver`
Returns a true boolean if the game has ended.

`validCardPlay`
Return a boolean if the Card parameter is a playable card.

`loadImageBack`
Load the back face of a card based on the width and height parameters. Returns an ImageIcon.

GameWindow
`main`
Sets up the frame for the game.

Hand
`sortBySuit`
Sort the Hand by Suit and return the sorted Hand

`sortByValue`
Sort the Hand by Value and return the sorted Hand

`getCardCount`
Return an int for how many cards are in this Hand

`addCard`
Add the Card parameter to this Hand. Returns the new Hand.

`removeCard`
Remove the Card parameter from this Hand. Returns the new Hand.

`get`
Return the Card at the specific index (int parameter).

InstructionsPanel
No methods; just constructor

LeaderboardPanel
`beforeRanking`
Set characteristics of the panel before the ranking of players takes place (set the background panel; meaning the different constructor parameters do not impact the tasks in the method).

`afterRanking`
Set the characteristics of the panel after the ranking of players takes place (set messages for the user and add images; meaning the different constructor parameters do not impact the tasks in the method)/

MenuPanel
No methods; just constructor

PlayerCardPanel
`loadImageBack`
Load back face of a card based on width and height parameters. Returns an ImageIcon

`loadRotatedImageBack`
Load back face of card based on width and height parameters. Is a rotated image of the back for players 2 and 4. Returns an ImageIcon.

`updatePanel`
Update the panel and show it on CardLayout using the int position of the player and the player's Hand.