OOP PROJECT 8 UNO GAME Varun Sahni 2020A7PS0144P  
 Akshat Gupta 2020A7PS0144P

Documentation

* Card Interface

Card is an interface that is implemented by the NormalCard Class. It contains the methods that are applicable to all the cards.

It implements the following functions

* + restart function is used to create the deck of cards. This method makes all 108 cards in the deck. it makes one 0 card for each suit and two card for every other number in the suit. It Also makes two draw two cards and two skip cards and two reverse cards and four wild and four Wild Draw Four.
  + subsituteDeckWith is used to replace the current deck with the some other deck of cards.
  + rearrange is used to rearrange the cards in the deck.
  + isEmpty is used to check if the deck is empty.
  + DrawCard is used to draw a card from the deck.
  + DrawImageIcon is used to draw the image of the card.
* Main Class

This class is the main class of the program. It contains the main method which is the entry point of the program. It runs the game. It passes the command to MainScreen class.

* MainScreen Class

This class is the MainScreen class. It is the first screen that the user sees when the game starts. This class ask the user to either begin the game or exit. It passes the command to the Player class.

* Normal Class

This class implements the interface card interface. It performs the functions as stated in the card interface

* Game class

It initialises the deck so as to start the game. The other methods present in this class are as follows.

* + start: this methods is used for drawing the first card from the deck. If the card is wild or drawtwo or wildFour then this again calls itself to draw another card. If the card drawns is skip or reverse it skip and reverses the direction of the game respectively. Finally it adds the card to the discardPile.
  + getTopCard() : return the information about the top card present in the discard pile.
  + getTopcardImage: return the image of the top card present in the discard pile.
  + IsGameOver() : this method check if the players hand is empty or not. If it is empty then it declares the player as the winner of the game.
  + getCurrentPlayer: returns the information of the current player.
  + getPreviousPlayer: It returns the information about the player preceding the current player
  + getPlayers: This method returns the information about all the players.
  + getPlayerHand: this method return the cards a player is having in the form of an arrayList.
  + getPlayerHandsSize: this method return the number of cards a player is having.
  + getPlayerCard: This method is used the get the details of the card player has chosen.
  + HasEmptyHand: Check whether the player’s cards has ended or not
  + validCardPlay: it matches weather the chosen card by the player is valid or not.
  + submitDraw: if the Draw pile becomes empty the this method replaces the draw pile with the draw pile
  + setCardColor: This method is used to set the value of the current card color with a color value.
  + submitPlayerCard: If the chosen card by the player is valid this method removes it from the array list. It also contains the working of the special cards.
* PlayArea

This class implements all the methods cardInfo, NormalCard and GameClass. The game is played with all the rules under this class

* ChoseWildColor

It allows the user to set the color after selecting the wild or wildFour card.

* CardConfirmation

Whenever a player clicks on a particular card it asks the player to make the move or to choose another card. If the card doesn’t match the required criteria then an appropriate message is printed on the screen.

Application of the OOP Principle

1. Encapsulates but varies: The entire program in divided into seprates modules. Each class is separated into different folders according to their functions. The project also implements enumeration.
2. Favour Composition over inheritance: All the classes are present at the same level hence by just making an object of the class one can access the methods of those classes, this thing is shown in the different classes like the game class, NormalCard, etc.
3. Class should be open for extension and closed for modification; Each class contains the getter and setter methods which can make the open for extension, while some class contain only the getter method which makes it closed for further extension.
4. Program to an interface not implementation : the normal card class implements the card interface.