Uno Game Engine

Atypon / waily

Hadeel Abunassar

1 – Card class:

In the Uno game, I've created an abstract card class that represents the superclass for all cards. I've then implemented three subclasses: NormalCard, ActionCard, and an additional abstract subclass named Wildcard. This structure allows developers to create their own wildcard classes tailored to their Uno variations.

The Wild and Wild Draw Four cards are common in all Uno variations. To represent these, I've created two specific classes: Wild and WildDrawFour. These classes offer a standardized way to handle these cards, ensuring consistency across different Uno versions.

In my own Variation, I have added addition SWAP class that extends the Wild class.

The Card class Contains some main methods:

returnedValue() : That represent the value of the card , this method is used to calcite the scores in the final step of the game.

cardAction(): This method is invoked when the player chose a valid card , so the method will apply the action of that card.

canBePlayed() : This method determine if the card is allowed to be played or not – compared with the topCard

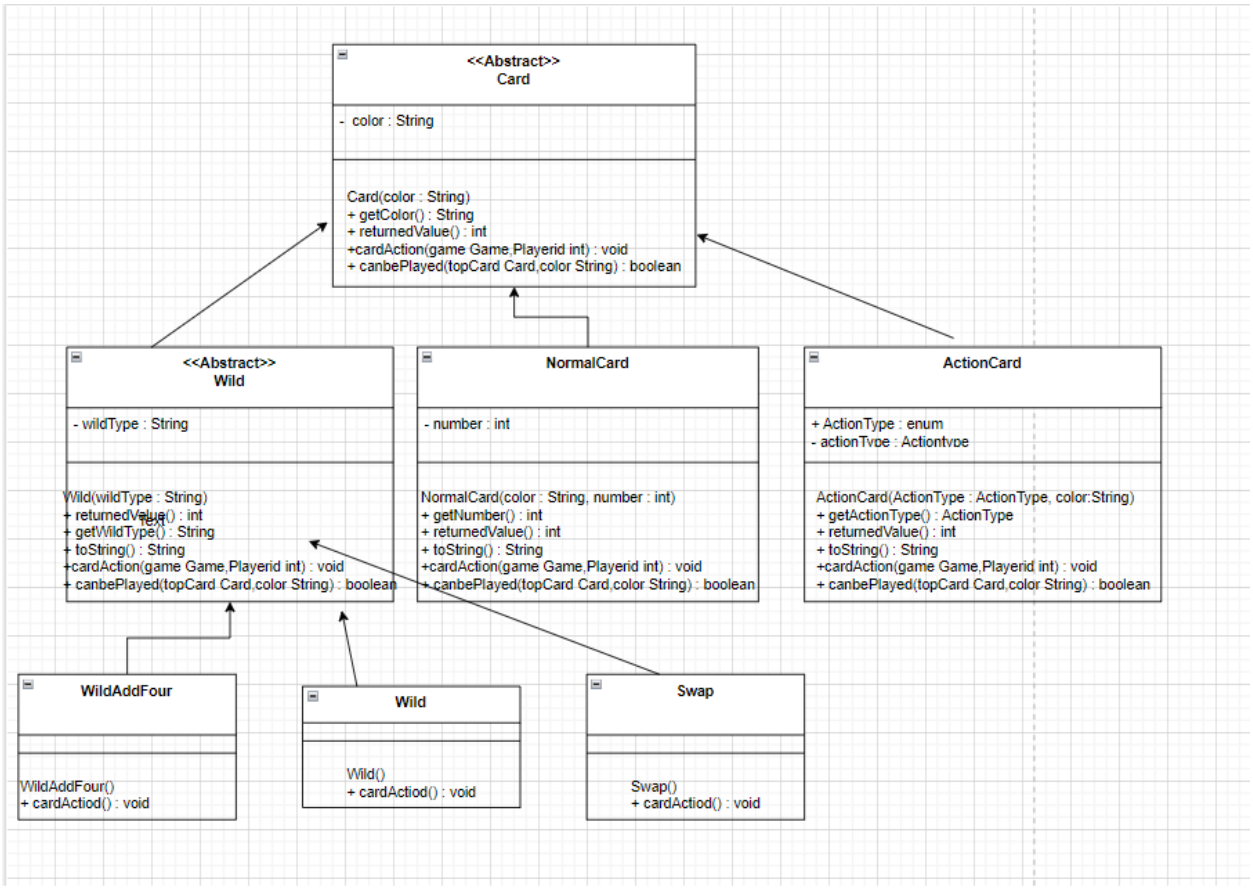

Figure 1 – UML Diagram of the Card Class.

2 – Deck class:

The main component of the Uno Game is the deck, and its mainly a List of Card.

This class contains the following methods:

- Shuffle() : to shuffle the cards.
- dealCards() : this is invoked in the game start , so each player will get a set of cards.
- Draw() : this is invoked each time a player want to draw a card.
- topCard() : this will represent the topCard , its also add the topCard to the discardedCards set.
- firstTopCard() : this method is invoked in the game start so the initial card will be drawn – it must be a normal card-.

Since the deck initialization depends on the Cards , I have created an interface called DeckInitializer so each developer will initialize the deck depending on the cards he have.

3 – Player class:

An uno game is not a game without the players , so there is a separate class the represent the players , this class contains instance variables that represent the player id , Player cards , and some variable for “uno” String that the user will call if he have one remaining card , also it contains the following methods :

-chooseCard() : that allow the player to choose the card he wants to play.

- Score(): to calculate the player score at the game end.

-displayCards() : to display the player cards.

Also , I have implemented two other classes , Humanplayer and ComputerPlayer that extends the Player class but I have not used them in my code.

4 – Game class:

game class is an abstract class that the developer will use to create his/her own game class , this class contains the following methods:

1- play() , which is an abstract method
2- winner() to determine the winner
3- howToWin() , another abstract method , to determine the winning rules of the game
4- nextPlayerDraw()
5- initializePlayers() , it takes the number of players as input , and return a set of players where each player have his own card list.
6- reverseDirection ()
7- chooseColorIfWild() , since the wild card make the player choose the color of the next player.

I Have created my own game in the MyGame class , that represent the standard game in addition to a swap wild card, the I have created a subclass for it , then I created some object from it in the MyGameDeckInitializer class that implements the DeckInitializer interface.

Defend your code against SOLID principles:

- All the previous classes have one only responsibility , each class represent one single unit in the uno game.
- Regarding the Open/Close principle , we can see that card/game/player/deck class is open for extension depending on the game variation , all these classes have the basic functionality that all the variations need so no need for modifications.
- Regarding the liskov's substitution principle , the methods functionality is clear and it depends on the subclasses.
- No duplicates codes , the reusable methods are used continuously
- The only long method is the play() method in MyGame class , but that is what the method need!

Explain design patterns used in your code:

- What I have done is a -ready to use- template of any uno game.
- Regarding the Mediator pattern , the game class is a centralized point for all other classes.
- All classes are totally encapsulated .
- The DeckInitializer interface, used for initializing the deck based on the game, suggests the Strategy Pattern. Different strategies for initializing the deck can be implemented by different concrete classes.
- The abstract card class and its subclasses (NormalCard, ActionCard, Wildcard, Wild, WildDrawFour,Swap) suggest the use of the Abstract Factory Pattern Each card type is a product created by a corresponding concrete factory.

Youtube link : https://youtu.be/XPLfOrkO494