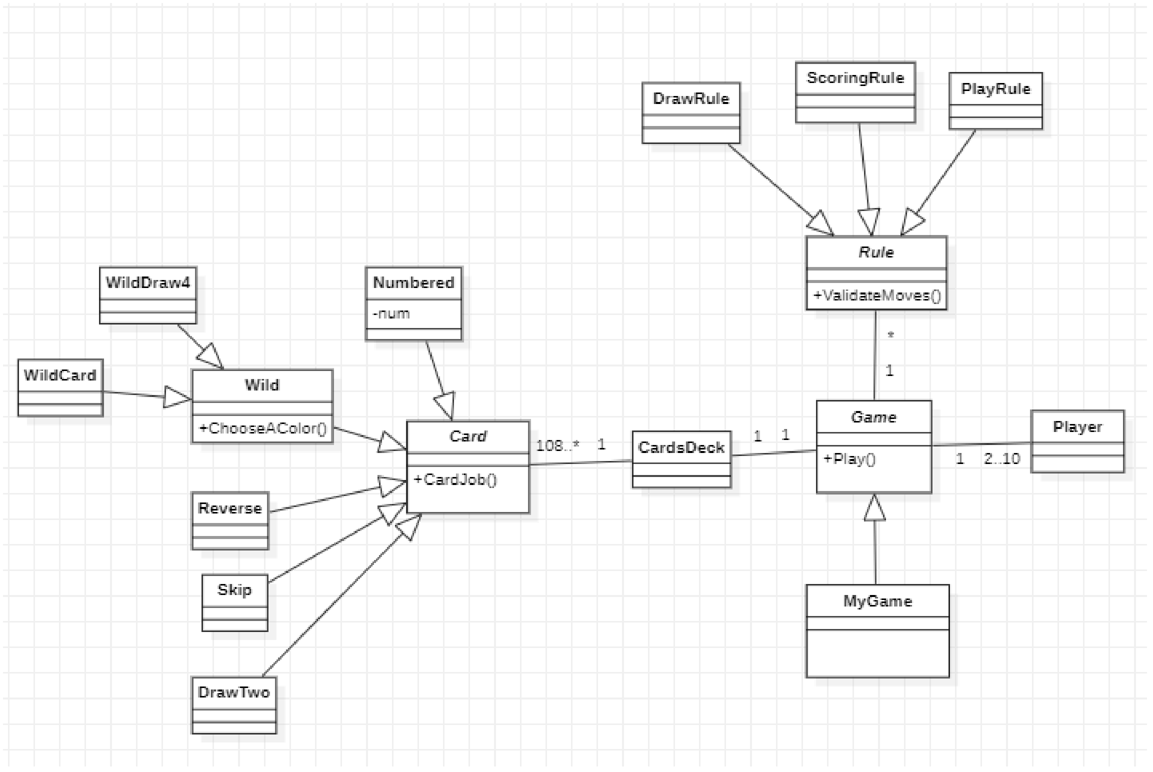
Rayyan Al-Hourani

UNO game engine

Atypon assignment

A-OOP design



So first we 6 main classes:

1- Player class that contain the details of player and his cards.

2- Rule class that contain the rules of play and draw cards and scoring rules.

3- Card class the is general type of cards type.

4- CardsDeck class that have at least 108 cards ,and discard pile and draw pile that players draw from it.

5- Game class , the game class contains 2-10 players and contains many rules and has one card deck that contains many cards.

6-MyGame class : this class extend the game class and implement the play method and set own game variations by some methods.

B-design patterns

I use for my engine 2 Design patterns

1-singleton Design patter : I use this pattern in Game class and CardsDeck class because we only have one game and one cardsDeck to draw from it and one discard deck

2 -visitor design pattern : I use this pattern in Rule class because we have many types rules and in Card class because we have many type of cards

C-Clean code

to make my code clean I do these things:

1- variables name is a meaning full name and avoid single letter variables

2- each function his name explain what is do

3- add a comments inside functions when the function name need more explain

4- make a packages and but in side it each class with his types to organize the files

5- format a code to be more readable

6-use SOLID principles

7- use OOP four principles

8-avoiding negative conditions

9-don’t having methods with more than 2 parameters

D-Effective java

There are many items that use in effective java book

1- Enforce the singleton property with a private constructor

2- Minimize the accessibility of classes and members

3- Minimize the scope of local variables

4- avoid return null

5- Prefer lists to arrays

6- Prefer interfaces to abstract classes

7- Refer to objects by their interfaces

8-Use interfaces only to define types

E-SOLID Principles

SPR- each class is use for one reason , the card class only for create a a cards, cardsDeck class only for but all cards in one place and manage them, the player class only for create player and manage them , the rule class is for create the rules of game , finally the game class the create a game with all it's elements

OCP-All classes is closed to modification and open for extends and do some edits using some functions and the developer can do his own rules and cards and add them to game.

LSP- most of inherited classes can replaced with parents without without disrupting the behavior of the program.

LSP-Card interface is small so I don't need to split it , but this class only have necessary method named CardJob that the developer have to implement it

DIP- Rule class and Card class are abstract class so they don't have many details inside them so all details is the child class , and the game class depends on them so they have an instance object inside the game class

For developer

First developer will extend the game class and but this code inside the play method

```
public void play() {
    //init the game
    CreatePlayers();
    initGame();

    //while max score < 500
    while (true) {
        LastCardPlayed();
        ConfirmUser();
        playCards();
    }
}
```

after that there many ways to adding rules and cards:

for adding cards so he extend the Card class and create the card and implement the CardJob() method , like change color or turn or any thing else , then he in the class then extended from game class he add the card on cardsDeck using ,

for add rule he extends the Rule class then implement the validate() method and then use setPlayRule() to make the class that you implement the rule of the game.

Also the developer can use any of implemented methods to build his own version and also he can override the methods.