**Proyecto del curso Java basico**

**Simulación del juego de cartas UNO**

Reglas del juego

Las cartas de la baraja

19 Cartas azules - un 0 y pares del 1 al 9

19 Cartas verdes - un 0 y pares del 1 al 9

19 Cartas rojas - un 0 y pares del 1 al 9

19 Cartas amarillas - un 0 y pares del 1 al 9

8 cartas especiales ROBA DOS - 2 de cada color

8 cartas especiales CAMBIO DE SENTIDO - 2 de cada color

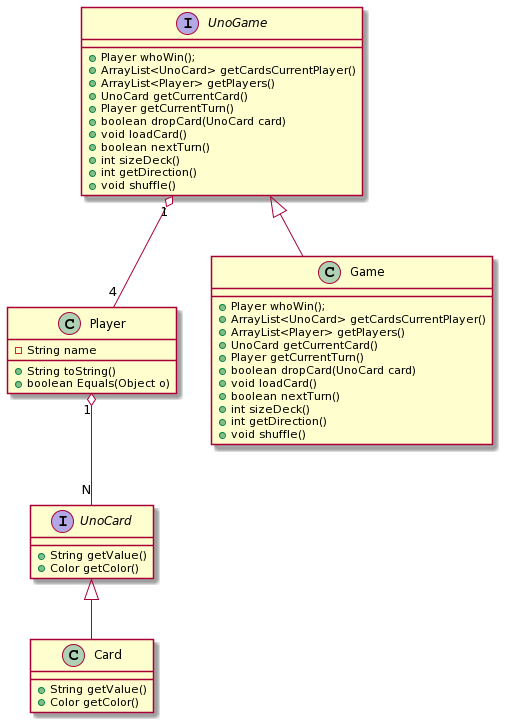
8 cartas especiales PIERDE EL TURNO - 2 de cada color

4 cartas especiales COMODIN CAMBIO DE COLOR

4 cartas especiales COMODIN CAMBIO DE COLOR ROBA CUATRO

Lo que esta resaltado de color rojo son las cartas que omitiremos.

Diagrama de clases inicial



Explicación sobre los métodos a implementar:

**CLASE CARD**

**getValue()**: Devuelve el valor de la carta como un String

para las cartas normales

0 -> "0" 5 -> "5"

1 -> "1" 6 -> "6"

2 -> "2" 7 -> "7"

3 -> "3" 8 -> "8"

4 -> "4" 9 -> "9"

cartas especiales

Reversa -> "R"

Cambia color -> "C"

mas 2 -> "+2"

**getColor():** Devuelveuna instancia de la clase Color

para las cartas normales, Reversa y +2 se usan los colores:

rojo ->Color.RED

verde -> Color.GREEN

azul ->Color.BLUE

amarillio -> Color.YELLOW

para la carta cambia color se utiliza el color:

negro -> Color.BLACK

**CLASE PLAYER**

Hay que sobre escribir estos métodos heredados desde object

**toString():** devuelve un String en nuestro caso el nombre del jugador

**equals(Object o):** devuelve true si el name de los objetos es igual

**CLASE GAME**

La clase Game implementa todos los métodos de UnoGame

**whoWin():** Devuelve un objeto de tipo Player, el cual es el jugador que gano el juego. Si no existe ganador aun, el método devuelve null.

**getCardsCurrentPlayer():** Devuelve la lista de cartas que le pertenecer al jugador actual.

**getPlayers():** Devuelve la lista de jugadores que están en el juego

**getCurrentCard():** Devuelve la carta actual, es decir la última carta que fue descartada.

**getCurrentTurn():** Devuelve el jugador que esta de turno

**dropCard(UnoCard card):** Recibe una carta como parámetro

devuelve true en caso de:

* La carta recibida como parámetro coincide en color o número, respecto a la última carta enviada al basurero.
* La carta es un cambia color

En cualquier otro caso devuelve false

**loadCard():** Carga una carta desde el mazo al jugador de turno, no devuelve nada y solo debería cargarse una carta por turno.

**nextTurn():** Pasa de turno al siguiente jugador

devuelve true en caso de :

El jugador de turno obtuvo una carta del mazo, solo en este caso se permite pasar de turno al siguiente jugador.

Devuelve false en caso de:

El jugador de turno no obtuvo carta del mazo, en este caso el jugador no cambia de turno.

**sizeDeck():** devuelve la cantidad de cartas restantes en el mazo

**getDirection():** devuelve el sentido del juego, los valores pueden ser:

izquierda -> 0

derecha -> 1

**shuffle():** Este método baraja todo el mazo, esto incluye cartas que tenías en el mazo y cartas que tienes en el basurero.

**¿Qué debo hacer y que puedo hacer?**

Deben implementar todos los métodos mencionados anteriormente

No deben cambiar nada en las clases interface

Pueden ver código del paquete view, pero no modificar el mismo para la presentación del proyecto.

Siéntanse libres de preguntar cualquier duda a sus instructores

El constructor de la clase Game nunca debe recibir parámetros

Pueden agregar los atributos y métodos que consideren necesarios en las clases Player, Card, Game

Pueden agregar todas las clases que consideren necesarias al paquete model.

**Presentación del proyecto**

Deben tener su propio diagrama de clases, que seria como una extensión de la base que se les está presentando (en el diagramador de su preferencia).

Subir a su repositorio en github el proyecto.

La fecha se establecerá de manera interna en coordinación con su instructor