FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Algoritmos y Programación 3
Trabajo Práctico - Etapa 1
Solitario

Cátedra Essaya - Corsi

Alumnos :
Lara Daniela Converso - 107632
Lucas Ezequiel Villarrubia - 108230

Etapa 1:

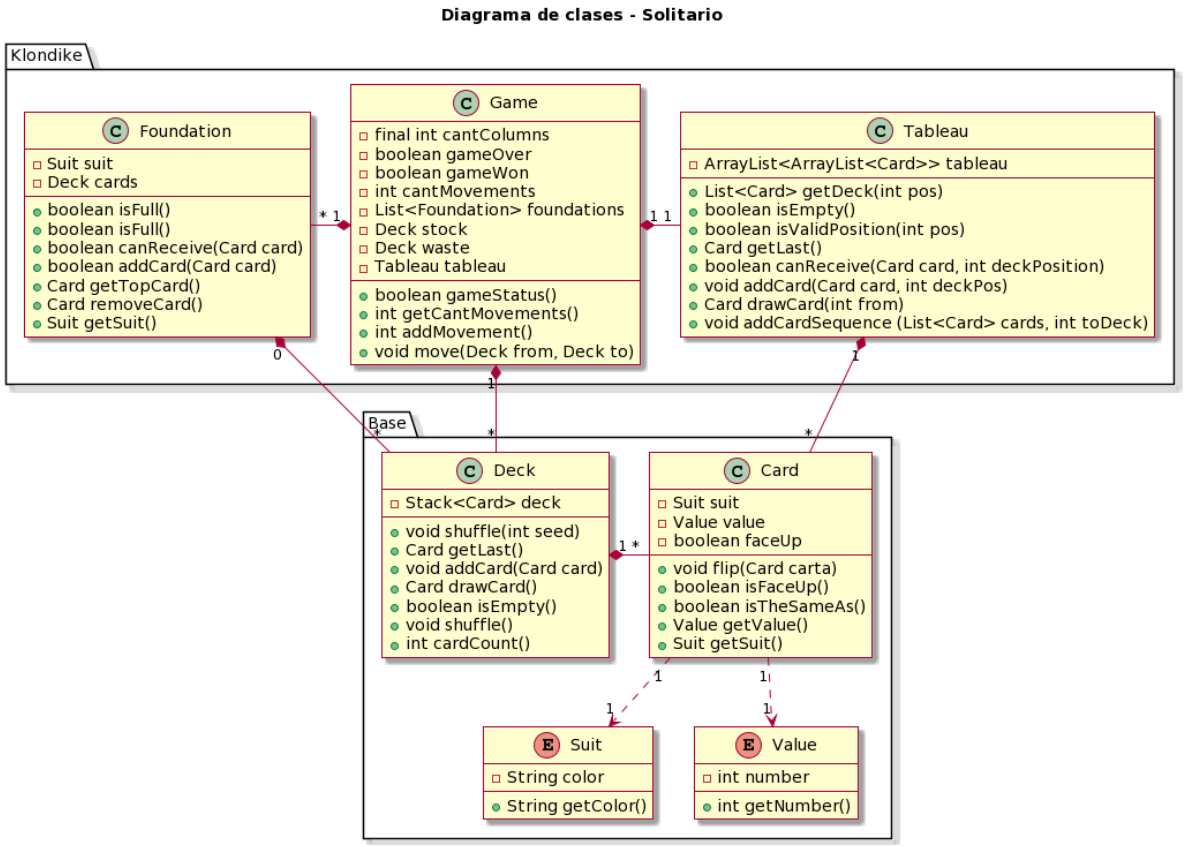
El repositorio:

Trabajamos en el repositorio https://github.com/lucasvillarrubia/algo3_tp para la entrega de esta etapa utilizamos una rama llamada ‘wip-etapa1’ (work in progress etapa 1).

Programa:

El programa contiene dos paquetes, el paquete de la Base del juego donde se encuentra la lógica base de un solitario. Y el paquete de Klondike donde se encuentra la lógica correspondiente al juego del tipo Klondike particularmente.

Diagrama:



Enums:

- Suit: define los palos que puede tener una carta. Estructura: NOMBRE(color)
- Value: están definidos los valores posibles de una carta. Estructura: NOMBRE(valor)

Clases:

Card: representa las cartas del juego.

- Atributos: suit, value, faceUp (booleano que define si la carta esta de frente o no)
- Métodos:
    - flip: se encarga de dar vuelta la carta, transformando el estado actual del atributo faceUp.
    - isFaceUp : consulta si la carta está de frente.
    - isTheSameAs: evalúa si las cartas tienen el mismo valor y el mismo número.

- getValue : obtiene el valor de la carta.
- getSuit: obtiene el palo de la carta.

Deck: representa los mazos de cartas

- Atributos: deck de tipo Stack.
- Métodos:
  - addCard: agrega una carta al deck.
  - getLast: muestra la última carta del deck y la pone de frente, si el deck está vacío retorna null.
  - drawCard: saca la última carta del deck y la pone de frente, si el deck está vacío retorna null.
  - isEmpty: chequea si el deck está vacío.
  - cardCount: retorna la cantidad de cartas agregadas.
  - shuffle: a partir de una semilla recibida por parámetro mezcla el deck con la función shuffle de Collections del paquete java util.

Foundation: representa la pila de cartas en donde se van acumulando de menor a mayor según su valor dependiendo de su palo.

- Atributos: suit y cards de tipo Deck.
- Métodos:
  - isFull: chequea si la foundation está completa.
  - canReceive: evalúa si la carta pasada por parámetro puede ser agregada a la foundation, con las reglas del Klondike.
  - addCard: agrega una carta a la foundaiton.
  - getTopCard: retorna la última carta agregada.
  - removeCard: elimina la última carta del deck.
  - getSuit: retorna el palo de la foundation.

Tableau: representa las disposición de cartas iniciales del juego. En el Klondike, el Tableau está compuesto por siete columnas de cartas, con la primera con su valor visible y el resto hacia abajo.

- Atributos: tableau que es de tipo lista y dentro tiene otras listas con cartas.
- Métodos:
  - getDeck: retorna la lista de cartas.
  - isEmpty: recorre todas las columnas del tableau y se fija si están vacías.
  - isValidPosition: evalúa si la posición es válida según la cantidad de columnas en el tableau.
  - getLast: retina la última carta.
  - canReceive: evalúa si la columna recibida por parámetro del tableau puede recibir la carta también recibida por parámetro según las reglas del Klondike. Se fija si el color es distinto y el número es menor al de la carta anterior y si se encuentra vacía solo recibe el valor King.
  - addCard: agrega la carta a la columna indicada.
  - drawCard: saca la última carta de la columna pasada por parámetro.
  - addCardSequence: agrega una secuencia de cartas a la columna indicada.

Game: clase abstracta que contiene la inicialización y métodos básicos del juego.

- Atributos: cantColumns (cantidad de columnas) en este caso = 7. waste de tipo deck, gameOver, gameWon, cantMovement, foundations que es una lista de Foundation, stock de tipo Deck y tableau.

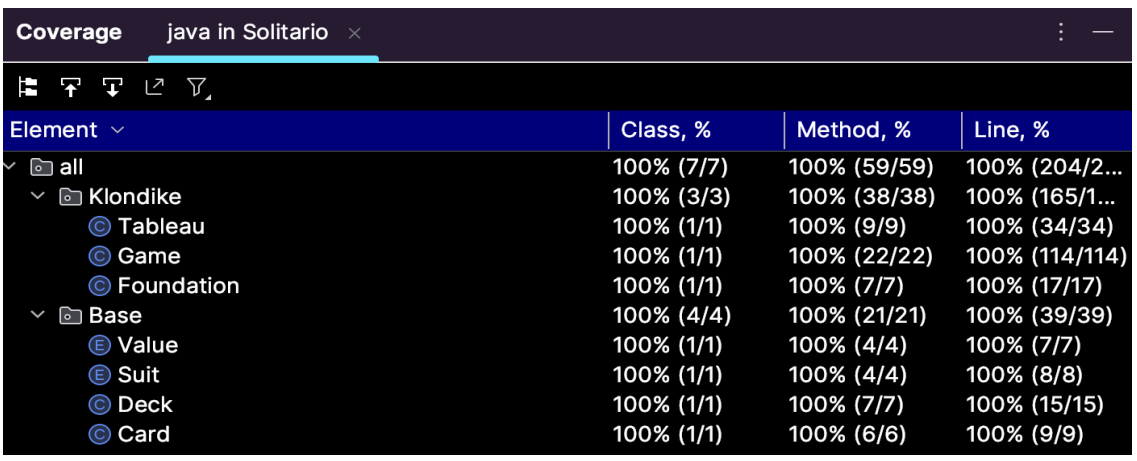
- Métodos:
    - isGameWon: retorna el valor del atributo gameWon.
    - winGame: modifica el valor gameWon si cumple con las condiciones de juego ganado para klondike (el tableau está vacío, las cuatro foundations llenas y el mazo de cartas vacío) y termina el juego.
    - gameStatus: retorna el valor del atributo gameOver.
    - getCantMovements: devuelve la cantidad de movimientos..
    - addMovements: suma un movimiento.
    - initStock: inicializa el stock con las 52 cartas, 13 de cada palo.
    - initTableau: inicializa el tableau con las cartas en cada columna correspondiente, y da vuelta la primera.
    - resetStock: resetea el mazo una vez que se terminan las cartas.
    - showStockCard: muestra la carta del mazo.
    - getFoundationBySuit: busca la foundation según el suit (palo) pasado por parámetro.
    - getTableau: retorna el tableau del game.
    - getStock: retorna el stock del game.
    - getWaste: retorna el waste del game.
    - areAllFoundationsFull: recorre la lista de las foundations y chequea si están completas.

  A continuación están los métodos para realizar movimientos en el juego, cada una evalúa si el movimiento a realizar es válido, en el caso de hacerse se suma un movimiento y chequea si el juego está ganado.
    - moveFromStockToFoundation
    - moveBetweenTableauDecks
    - moveSequenceInTableau
    - moveFromTableauToFoundation
    - moveFromStockToTableauDeck
    - moveFromFoundationToTableauDeck

  Esta clase tiene dos constructores, en uno donde es requerida la semilla para mezclar el mazo, y otro donde permite inicializar el juego en un estado particular.

Pruebas:

Las pruebas se encuentran en la carpeta : …src/test/java…
Dentro de esta carpeta se replica la misma estructura del programa. Y cada clase tiene su test en donde se prueba el funcionamiento de sus métodos y el funcionamiento de las clases en conjunto.
Cobertura obtenida:



Element	Class, %	Method, %	Line, %
all	100% (7/7)	100% (59/59)	100% (204/2...
Klondike	100% (3/3)	100% (38/38)	100% (165/1...
Tableau	100% (1/1)	100% (9/9)	100% (34/34)
Game	100% (1/1)	100% (22/22)	100% (114/114)
Foundation	100% (1/1)	100% (7/7)	100% (17/17)
Base	100% (4/4)	100% (21/21)	100% (39/39)
Value	100% (1/1)	100% (4/4)	100% (7/7)
Suit	100% (1/1)	100% (4/4)	100% (8/8)
Deck	100% (1/1)	100% (7/7)	100% (15/15)
Card	100% (1/1)	100% (6/6)	100% (9/9)

Post corrección:

Corregimos la declaración de la constante `AMOUNT_COLUMNS`.
Agregamos una clase Column para que Tableau sea una lista de Columnas.
Agregamos el Enum referente a Color, para asignarle a cada palo.