Université de Paris

Année universitaire **2020-2021**

Projet BPO

Présenté par **Aimée Diallo et Sofiane Oulad Itto**

Faisant parti respectivement des groups 105 et 108

Projet Java de première année présenté vendredi 08 mars 2021

Table des matières

P3. Présentation du problème

P4. Le code Java complet

P5. Bilan du projet

Présentation du problème :

Nous avons développé en langage Java un programme permettant à deux joueurs, de s’affronter sur un jeu se nommant “Le Duel”. Ce jeu consiste à se débarrasser de ses cartes en suivant certaines règles. Ces règles régissent les conditions de pose et de tirage pour les deux joueurs.

En effet, chaque joueur possède 6 cartes, sur un total disponible de 60 cartes numéroté de 1 à 60 et ont chacun devant eux une file ascendante et descendante. Chaque joueur peut poser autant de cartes qu’il le souhaite mais toujours de valeur plus grande que la valeur de la carte précédente sur la file ascendante, et plus petite sur la file descendante, sauf si la nouvelle carte posée est de la même dizaine que la carte précédente.

Nous pouvons par exemple poser la carte 39 par-dessus la carte 49 même sur la file ascendante.

De plus, il est possible de poser une carte par tour sur la file ascendante et descendante de l’adversaire, en suivant l’inverse des règles énoncées ci-dessus pour les files des joueurs.

Il faut donc gérer les règles de tirage, de pose, les exceptions et les affichages après chaque coup joué. Il faut aussi tenir compte de la gestion d’erreur dans le cas où les entrées données par l’un des joueurs ne correspond pas aux règles du jeu établies.

Le code Java complet :

Nous allons donc vous presenter le code Java, en vous présentant d’abord les classes indépendantes, puis par celles qui dependent d’une autre classe.

Hand.java

package appli.Game.Element;

import java.util.ArrayList;

import java.util.Collections;

public class Hand {

    private ArrayList<Card> cards;

    public Hand() {

        cards = new ArrayList<>();

    }

    public void addCard(Card card) {

        cards.add(card);

    }

    public void removeCard(int nbOfCard) {

        for(int i =0; i < cards.size(); i++) {

            if(cards.get(i).getCardNumber() == nbOfCard) {

                cards.remove(i);

                break;

            }

        }

    }

    public int nbCardsInHand() {

        return cards.size();

    }

    public String displayHand() {

        Collections.sort(cards);

        String cardsListing = "{ ";

        for(int i = 0; i < cards.size(); i++) {

            cardsListing += cards.get(i).toString() + " ";

        }

        return cardsListing + "}";

    }

    public boolean isInHand(int nbToPut) {

        for(int i = 0; i < cards.size(); i++) {

            if(nbToPut == cards.get(i).getCardNumber()) {

                return true;

            }

        }

        return false;

    }

}

Deck.java

package appli.Game.Element;

import java.util.ArrayList;

import java.util.Stack;

public class Deck {

    private Stack<Card> cards;

    public Deck() {

        cards = new Stack<>();

        createDeck();

    }

    private void createDeck() {

        ArrayList<Card> allCards = new ArrayList<>();

        for(int i = 2; i < 60; i++) {

            allCards.add(new Card(i));

        }

        int random;

        for(int i = 0; i < 58; i++) {

            random = (int) (Math.random() \* (58 - i));

            cards.push(allCards.get(random));

            allCards.remove(random);

        }

    }

    public Card drawCard() {

        return cards.pop();

    }

    public int nbCardsInDeck() {

        return cards.size();

    }

}

Hand.java

package appli.Game.Element;

public class Card implements Comparable<Card> {

    private int cardNumber;

    public Card(int cardNumber) {

        this.cardNumber = cardNumber;

    }

    public int getCardNumber() {

        return cardNumber;

    }

    @Override

    public String toString() {

        return cardNumber < 10 ? "0" + cardNumber : Integer.toString(cardNumber);

    }

    @Override

    public int compareTo(Card card) {

        return (Integer.compare(cardNumber, card.getCardNumber()));

    }

}

Player.java qui dépend des trois fichiers ci-dessus:

package appli.Game;

import appli.Game.Element.Card;

import appli.Game.Element.Deck;

import appli.Game.Element.Hand;

import java.util.Stack;

public class Player implements Cloneable {

    private String name;

    private Stack<Card> asc;

    private Stack<Card> desc;

    private Hand hand;

    private Deck deck;

    public Player(String name) {

        this.name = name;

        asc = new Stack<>();

        desc = new Stack<>();

        asc.push(new Card(1));

        desc.push(new Card(60));

        hand = new Hand();

        deck = new Deck();

        drawNumberOfCards(6);

    }

    public void drawNumberOfCards(int nbOfCards) {

        for(int i = 0; i < nbOfCards; i++) {

            hand.addCard(deck.drawCard());

        }

    }

    public String getName() {

        return name;

    }

    public Stack<Card> getAsc() {

        return asc;

    }

    public Stack<Card> getDesc() {

        return desc;

    }

    public Hand getHand() {

        return hand;

    }

    public Deck getDeck() {

        return deck;

    }

    public Object clone() throws CloneNotSupportedException {

        return super.clone();

    }

    public String getPlayerInfos() {

        String playerName = name.equals("SUD") ? name + " " : name;

        String ascStackNumber = asc.peek().toString();

        String descStackNumber = desc.peek().toString();

        String nbCardsInHand = Integer.toString(hand.nbCardsInHand());

        String nbCardsInDeck = Integer.toString(deck.nbCardsInDeck());

        return  playerName

                + " ^[" + ascStackNumber + "]"

                + " v[" + descStackNumber + "]"

                + " (m" + nbCardsInHand

                + "p" + nbCardsInDeck + ")";

    }

    public String displayHand() {

        return "cartes " + name + " " + hand.displayHand();

    }

}

Game.java

package appli.Game;

public class Game {

    private Player nord;

    private Player sud;

    private boolean nordPlayerTurn;

    public Game() {

        nord = new Player("NORD");

        sud = new Player("SUD");

        nordPlayerTurn = true;

    }

    public Player getNord() {

        return nord;

    }

    public Player getSud() {

        return sud;

    }

    public void setNord(Player nord) {

        this.nord = nord;

    }

    public void setSud(Player sud) {

        this.sud = sud;

    }

    public boolean isNordPlayerTurn() {

        return nordPlayerTurn;

    }

    public void changePlayerTurn() {

        nordPlayerTurn = !nordPlayerTurn;

    }

}

Application.java qui contient le main du projet:

package appli;

import appli.Game.Element.Card;

import appli.Game.Game;

import appli.Game.Player;

import java.util.Scanner;

public class Application {

    public static void main(String[] args) throws CloneNotSupportedException {

        Game game = new Game();

        String winner = "";

        boolean regulatoryMove;

        while(true) {

            System.out.println(game.getNord().getPlayerInfos());

            System.out.println(game.getSud().getPlayerInfos());

            if(game.isNordPlayerTurn()) {

                System.out.println(game.getNord().displayHand());

            } else {

                System.out.println(game.getSud().displayHand());

            }

            regulatoryMove = readPlayerInput(game);

            if(game.isNordPlayerTurn() && game.getNord().getHand().nbCardsInHand() == 0 && game.getNord().getDeck().nbCardsInDeck() == 0) {

                winner = game.getNord().getName();

                break;

            } else if(!game.isNordPlayerTurn() && game.getSud().getHand().nbCardsInHand() == 0 && game.getSud().getDeck().nbCardsInDeck() == 0) {

                winner = game.getSud().getName();

                break;

            } else if(game.isNordPlayerTurn() && !regulatoryMove) {

                winner = game.getSud().getName();

                break;

            } else if(!game.isNordPlayerTurn() && !regulatoryMove){

                winner = game.getNord().getName();

                break;

            }

            int nbCardsToDraw = 2;

            if(nbCardPlacedToEnemy > 0) {

                nbCardsToDraw = 6 - game.getNord().getHand().nbCardsInHand();

            }

            if(game.isNordPlayerTurn()) {

                game.getNord().drawNumberOfCards(nbCardsToDraw);

            } else {

                game.getSud().drawNumberOfCards(nbCardsToDraw);

            }

            game.changePlayerTurn();

        }

        System.out.println("partie finie, " + winner + " a gagné");

    }

    private static int nbCardPlaced = 0;

    private static int nbCardPlacedToEnemy = 0;

    private static boolean readPlayerInput(Game game) throws CloneNotSupportedException {

        boolean inputError = false;

        boolean finish = false;

        Scanner scanner = new Scanner(System.in);

        while(!finish) {

            finish = true;

            if(!inputError)

                System.out.print("> ");

            else

                System.out.print("#> ");

            String input = scanner.nextLine();

            String[] tabInput = input.split(" ");

            Player nord = (Player) game.getNord().clone();

            Player sud = (Player) game.getSud().clone();

            nbCardPlaced = 0;

            nbCardPlacedToEnemy = 0;

            boolean asc;

            boolean onCurrentPlayerStack;

            for (String s : tabInput) {

                try {

                    if(s.length() > 4) {

                        inputError = true;

                        finish = false;

                        break;

                    }

                    int nbToPut = Integer.parseInt(s.substring(0, 2));

                    char stackDirection = s.charAt(2);

                    if(stackDirection == '^') {

                        asc = true;

                    } else if(stackDirection == 'v') {

                        asc = false;

                    } else {

                        inputError = true;

                        finish = false;

                        break;

                    }

                    onCurrentPlayerStack = true;

                    if(s.length() == 4) {

                        if(s.charAt(3) == '\'') {

                            onCurrentPlayerStack = false;

                        } else {

                            inputError = true;

                            finish = false;

                            break;

                        }

                    }

                    if(!nbToPutInHand(game, nbToPut)) {

                        inputError = true;

                        finish = false;

                        break;

                    }

                    boolean impossibleMove;

                    if(game.isNordPlayerTurn()) {

                        impossibleMove = isImpossibleMove(nord, sud, asc, onCurrentPlayerStack, nbToPut);

                    } else {

                        impossibleMove = isImpossibleMove(sud, nord, asc, onCurrentPlayerStack, nbToPut);

                    }

                    if(impossibleMove || nbCardPlacedToEnemy > 1) {

                        inputError = true;

                        finish = false;

                        break;

                    }

                } catch (Exception e) {

                    inputError = true;

                    finish = false;

                    break;

                }

            }

        }

        return nbCardPlaced >= 2;

    }

    private static boolean isImpossibleMove(Player nord, Player sud, boolean asc, boolean onCurrentPlayerStack, int nbToPut) {

        boolean impossibleMove = false;

        // TODO : gestion de l'erreur potentielle pour la main et la pile du player

        if(asc) {

            if(onCurrentPlayerStack) {

                if(nord.getAsc().peek().getCardNumber() < nbToPut

                        || nord.getAsc().peek().getCardNumber() - 10 == nbToPut) {

                    nord.getAsc().push(new Card(nbToPut));

                    nord.getHand().removeCard(nbToPut);

                    nbCardPlaced++;

                } else { impossibleMove = true; }

            } else {

                if(sud.getAsc().peek().getCardNumber() > nbToPut) {

                    sud.getAsc().push(new Card(nbToPut));

                    sud.getHand().removeCard(nbToPut);

                    nbCardPlaced++;

                    nbCardPlacedToEnemy++;

                } else { impossibleMove = true; }

            }

        } else {

            if(onCurrentPlayerStack) {

                if(nord.getDesc().peek().getCardNumber() > nbToPut

                        || nord.getDesc().peek().getCardNumber() + 10 == nbToPut) {

                    nord.getDesc().push(new Card(nbToPut));

                    nord.getHand().removeCard(nbToPut);

                    nbCardPlaced++;

                } else { impossibleMove = true; }

            } else {

                if(sud.getDesc().peek().getCardNumber() < nbToPut) {

                    sud.getDesc().push(new Card(nbToPut));

                    sud.getHand().removeCard(nbToPut);

                    nbCardPlaced++;

                    nbCardPlacedToEnemy++;

                } else { impossibleMove = true; }

            }

        }

        return impossibleMove;

    }

    private static boolean nbToPutInHand(Game game, int nbToPut) {

        if(game.isNordPlayerTurn()) {

            return game.getNord().getHand().isInHand(nbToPut);

        }

        return game.getSud().getHand().isInHand(nbToPut);

    }

}

Le Bilan :

L’une des difficultés du projet a été de faire un jeu construit et de diviser le projet en différents composants logiques (classes et applications). En effet, en plus de développer un jeu fonctionnel, le but était aussi de faire un code structuré qui suive une construction logique, au lieu de tout faire dans un seul et même fichier.

De plus, une autre difficulté rencontrée a été de rendre opérationnelle l’une des choses les plus importantes du projet, à savoir les fonctions de poses et de tirages.

La gestion d’erreur a aussi été un élément important qui a pris du temps pour pouvoir prévoir et gérer tous les cas d’erreurs possible.

L’affichage n’a pas été un réel souci du fait de l’exemple proposé dans le sujet.

**Merci**

Par Aimée Diallo (108) et

Sofiane Oulad Itto (105)