RAPPORT DE PROJET JAVA (R2.01) : JEU DU 6 QUI PREND

Imane BENYETTOU

GROUPE 105

Table des matières

Introduction du projet

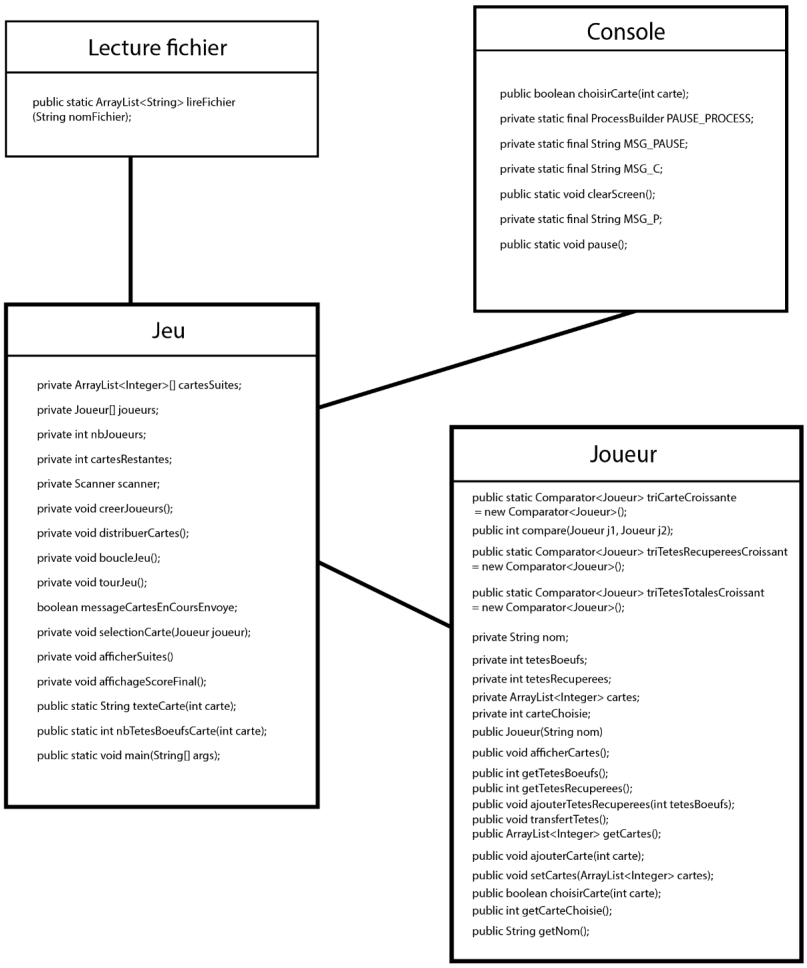
L'objectif de ce projet est de programmer le jeu du 6 qui prend afin de pouvoir permettre aux joueurs de réaliser une partie de bout en bout.
Les fonctionnalités devant être implémentées sont les suivantes : mélanger les cartes, les distribuer, et placer les 4 cartes représentant les 4 séries initiales au début de la partie. Il faudra ensuite programmer l'appel des joueurs via un message d'invitation, le choix de sa carte, et sa pose automatique en fonction des règles du jeu.

A la fin de chaque tour, le nombre de têtes de boeufs est affiché pour chaque joueur, ainsi que leur cumul.

Le format de saisie et d'affichage suivant était à respecter : la carte est désignée par sa valeur, qui sera affichée entre parenthèses. Les messages du programme sont affichés en noir, et les donnés saisies par les joueurs en vert. Les séries seront indiquées en colonne par ordre croissant, numérotées dans ce même ordre. Les cartes que le joueur choisit à son tour seront affichées horizontalement.

Le programme se termine après le dernier affichage, qui est le nombre total de têtes de bœuf ramassées par chaque joueur.

Diagramme UML des classes



Bilan du projet

Ce qui à été réussi

Ayant réalisé ce projet seule, je pense avoir bien géré mon temps ainsi que mon organisation : j'ai pu implémenter une méthode de travail plus organisée que pour mes projets antérieurs, ce qui m'a permis de mieux appréhender la programmation de ce jeu. En ayant commencé par analyser le sujet puis en extraire les informations utiles et les traduire en notions de programmation (classes, méthodes…) j'ai pu avoir une meilleure compréhension de la programmation objet, en implémentant notamment certaines notions apprises en cours, en TD et en TP.

Difficultés

J'ai choisi de réaliser ce projet seule, ce qui était une première pour moi étant donné que tous les projets de programmation et TPs antérieurs étaient en binôme ou en groupe. Avoir une certaine autonomie vis à vis du travail m'à beaucoup plus, mais se révélait difficile car je n'avais personne pour m'aider ou me débloquer à certain moment. La programmation objet et Java était de plus une compétence toute nouvelle, et il m'a fallu du temps pour m'adapter à ce type de programmation que je n'avais encore presque jamais encore vu auparavant. De plus, la charge de travail étant plus conséquente, il à été difficile de gérer la programmation de ce jeu, ainsi que les autres travaux et révisions liés à cette matière ou non.

"Traduire" le jeu de carte en programme complètement digital était aussi un challenge : associer et implémenter une notion de programmation pour tenter de copier une situation réelle, tester le jeu, et réaliser une partie sans contexte…

Ce qui peut être amélioré

Je pense pouvoir améliorer la partie test de mon projet. En effet, je ne pense pas bien avoir compris comment réaliser celle-ci, surtout dans un environnement de programmation nouveau. C'est donc quelque chose que je compte revoir et retravailler.

De plus, je pense mieux structurer mon travail la prochaine fois, et notamment mon organisation vis à vis de la réalisation de celui-ci : même si je pense avoir progressé par rapport aux anciens projets, je pense pouvoir faire encore mieux notamment en terme de gestion de temps et répartition des différentes tâches au cours des semaines.

Enfin, en termes de code, je pense pouvoir optimiser et éliminer les lignes et implémentations superflues, pour pouvoir avoir un programme plus léger, et plus facile à lire.

Annexe

Code source

Console

```
package jeu.utils;

import java.io.IOException;

public class Console {
   private static final ProcessBuilder CLEANER_PROCESS;
   private static final ProcessBuilder PAUSE_PROCESS;

   private static final String MSG_PAUSE = "Appuyez sur une touche pour
continuer..." + System.lineSeparator();

   static {
       System.out.println(System.console());
       if (System.console() != null) {
           String[] cdeClean;
           String[] cdePause;
           if (System.getProperty("os.name").contains("Windows")) {
               cdeClean = new String[] { "cmd", "/c", "cls" };
               cdePause = new String[] { "cmd", "/c", "pause" };
           }
           else {
               cdeClean = new String[] { "clear" };
               cdePause = new String[] { "read", "-n1", "-rsp", MSG_PAUSE };
           }
           CLEANER_PROCESS = new ProcessBuilder(cdeClean).inheritIO();
           PAUSE_PROCESS = new ProcessBuilder(cdePause).inheritIO();
       } else
           CLEANER_PROCESS = PAUSE_PROCESS = null;
   }

   private static final String MSG_C = "<clearScreen>";

   public static void clearScreen() {
       if (CLEANER_PROCESS != null)
           try {
               CLEANER_PROCESS.start().waitFor();
           } catch (InterruptedException e) {
               Thread.currentThread().interrupt();
           } catch (IOException e) {
               System.out.println(MSG_C);
           }
       else
           System.out.println(MSG_C);
   }

   private static final String MSG_P = "<pause>";
```

```
    public static void pause() {
        if (PAUSE_PROCESS != null)
            try {
                PAUSE_PROCESS.start().waitFor();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            } catch (IOException e) {
                System.out.println(MSG_P);
            }
        else
            System.out.println(MSG_P);
    }

    private Console() {
    }
}
```

Lecture de fichier

```
package jeu.utils;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class LectureFichier {
    public static ArrayList<String> lireFichier(String nomFichier) {
        ArrayList<String> lignes = new ArrayList<>();

        try {
            // Ouverture du fichier
            File fichier = new File(nomFichier);
            Scanner scanner = new Scanner(fichier);

            // Lecture des lignes
            while (scanner.hasNextLine()) {
                String ligne = scanner.nextLine();
                lignes.add(ligne);
            }

            // Fermeture du scanner
            scanner.close();
        } catch (FileNotFoundException e) {
            System.out.println("Le fichier des noms de joueurs n'a pas pu ?tre
trouv?");
            e.printStackTrace();
        }

        return lignes;
    }
}
```

Joueur

```
package jeu;
```

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

public class Joueur {
   // Comparateurs pour le tri de joueurs
   public static Comparator<Joueur> triCarteCroissante = new
Comparator<Joueur>() {
       @Override
       public int compare(Joueur j1, Joueur j2) {
           if (j1.getCarteChoisie() > j2.getCarteChoisie()) {
               return 1;
           } else {
               return -1;
           }
       }
   };
   public static Comparator<Joueur> triTetesRecupereesCroissant = new
Comparator<Joueur>() {
       @Override
       public int compare(Joueur j1, Joueur j2) {
           if (j1.getCarteChoisie() > j2.getCarteChoisie()) {
               return 1;
           } else if (j1.getCarteChoisie() < j2.getCarteChoisie()) {
               return -1;
           } else { // En cas d'�galit� tri par nom
               return j1.getNom().compareTo(j2.getNom());
           }
       }
   };
   public static Comparator<Joueur> triTetesTotalesCroissant = new
Comparator<Joueur>() {
       @Override
       public int compare(Joueur j1, Joueur j2) {
           if (j1.getTetesBoeufs() > j2.getTetesBoeufs()) {
               return 1;
           } else if (j1.getCarteChoisie() < j2.getCarteChoisie()) {
               return -1;
           } else { // En cas d'�galit� tri par nom
               return j1.getNom().compareTo(j2.getNom());
           }
       }
   };

   private String nom;

   private int tetesBoeufs = 0; // T�tes totales
   private int tetesRecuperees = 0; // T�tes r�cup�rees pendant ce tour

   private ArrayList<Integer> cartes = new ArrayList<>(10);
   private int carteChoisie = 0;

   public Joueur(String nom) {
       this.nom = nom;
   }
```

```
    // Affiche les cartes du joueur
    public void afficherCartes() {
        System.out.print("- Vos cartes : ");
        for (int i = 0; i < cartes.size(); i++) {
            System.out.print(Jeu.texteCarte(cartes.get(i)));
            if (i < cartes.size() - 1) {
                System.out.print(", ");
            }
        }
        System.out.println(); // Retour ◆ la ligne
    }

    public int getTetesBoeufs() {
        return tetesBoeufs;
    }

    public int getTetesRecuperees() {
        return tetesRecuperees;
    }

    public void ajouterTetesRecuperees(int tetesBoeufs) {
        tetesRecuperees += tetesBoeufs;
    }

    // Ajoute les t◆tes r◆cup◆res pendat la manche au total et remise ◆ 0 des
t◆tes r◆cup◆r◆es pendat la manche
    public void transfertTetes() {
        tetesBoeufs += tetesRecuperees;
        tetesRecuperees = 0;
    }

    public ArrayList<Integer> getCartes() {
        return cartes;
    }

    // Ajoute une carte et trie
    public void ajouterCarte(int carte) {
        cartes.add(carte);
        Collections.sort(cartes);
    }

    public void setCartes(ArrayList<Integer> cartes) {
        this.cartes = cartes;
    }

    /*
     * Choisit une carte dans sa main
     * Renvoie true si la carte peut bien ◆tre choisie
     */
    public boolean choisirCarte(int carte) {
        if (cartes.contains(carte)) {
            carteChoisie = carte;
            // Retire la carte de la main du joueur
            // Integer.valueOf(carte) permet de supprimer la carte dans la liste
au lieu de supprimer la carte ◆ l'index donn◆
            cartes.remove(Integer.valueOf(carte));
            return true;
```

```
        }

        return false;
    }

    public int getCarteChoisie() {
        return carteChoisie;
    }

    public void retirerCarteChoisie() {
        carteChoisie = 0;
    }

    public String getNom() {
        return nom;
    }
}
```

Jeu

```
package jeu;


import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

import jeu.utils.Console;
import jeu.utils.LectureFichier;

public class Jeu {

    private ArrayList<Integer>[] cartesSuites; // Tableau contenant pour chaque
suite une liste de cartes
    private Joueur[] joueurs; // Tableau contenant les joueurs

    private int nbJoueurs;
    private int cartesRestantes = 0; // Nombre de cartes dans la main de chaque
joueur

    private Scanner scanner;

    public Jeu() {
        scanner = new Scanner(System.in);

        creerJoueurs();
        distribuerCartes();

        boucleJeu();

        scanner.close();
    }

    private void creerJoueurs() {
        ArrayList<String> nomJoueurs = LectureFichier.lireFichier("config.txt");
        nbJoueurs = nomJoueurs.size();
```

```
        joueurs = new Joueur[nbJoueurs];
        for (int i = 0; i < nbJoueurs; i++) {
            String nom = nomJoueurs.remove(0);
            joueurs[i] = new Joueur(nom);
        }
    }

    private void distribuerCartes() {
        // Cr◆ation et m◆lange des cartes
        ArrayList<Integer> cartesPioche = new ArrayList<>();
        for (int i = 1; i <= 104; i++) {
            cartesPioche.add(i);
        }
        Collections.shuffle(cartesPioche);

        // Distribue 10 cartes ◆ chaque joueur
        for (Joueur joueur : joueurs) {
            for (int i = 0; i < 10; i++) {
                int carte = cartesPioche.remove(0); // Prend une carte de la
pioche
                joueur.ajouterCarte(carte);
            }
        }
        cartesRestantes = 10;

        cartesSuites = new ArrayList[4];
        for (int i = 0; i < 4; i++) {
            int carte = cartesPioche.remove(0); // Prend une carte de la pioche
            cartesSuites[i] = new ArrayList<>(5);
            cartesSuites[i].add(carte);
        }
    }

    // Boucle qui permet de faire une manche compl◆te
    private void boucleJeu() {
        while (cartesRestantes > 0) {
            tourJeu();
        }

        affichageScoreFinal();
    }

    // Un tour de jeu
    private void tourJeu() {

        // Liste des joueurs tri◆ par ordre croissant de la carte choisie par
chacun
        ArrayList<Joueur> listeJoueurs = new ArrayList<>();
        for (int i = 0; i < nbJoueurs; i++) {
            Joueur joueur = joueurs[i];
            listeJoueurs.add(joueur);
        }

        // S◆lection d'une carte pour chaque joueur
        for (Joueur joueur : joueurs) {
            selectionCarte(joueur);
        }
```

```
        // Tri des joueurs par valeur de la carte choisie croissante
        Collections.sort(listeJoueurs, Joueur.triCarteCroissante);

        // Pr⧫paration du message cartes ⧫ placer (envoy⧫ uniquement si un
joueur doit choisir une suite ⧫ r⧫cup⧫rer)
        String messageCartesJoueurs = "Les cartes ";
        for (int i = 0; i < nbJoueurs; i++) {
            Joueur joueur = listeJoueurs.get(i);
            int carte = joueur.getCarteChoisie();

            messageCartesJoueurs += carte + "(" + joueur.getNom() + ")";
            if (i < nbJoueurs - 2) {
                messageCartesJoueurs += ", ";
            } else if (i == nbJoueurs - 2) {
                messageCartesJoueurs += " et ";
            }
        }
        boolean messageCartesEnCoursEnvoye = false;

        // Placement des cartes
        for (Joueur joueur : listeJoueurs) {
            int carte = joueur.getCarteChoisie();

            // Trouve la suite sur laquelle la carte peut ⧫tre pos⧫e (croissant)
avec la diff⧫rence la plus faible
            int indexSuiteChoisie = -1;
            int carteSuiteActuelle = 0;
            for (int i = 0; i < 4; i++) {
                int carteSuite = cartesSuites[i].get(cartesSuites[i].size() -
1);

                if (carteSuite < carte && carteSuite > carteSuiteActuelle) {
                    indexSuiteChoisie = i;
                    carteSuiteActuelle = carteSuite;
                }
            }

            // Si il est possible de poser la carte dans une suite, on la pause
            if (indexSuiteChoisie >= 0) {
                if (cartesSuites[indexSuiteChoisie].size() < 5) {
                    cartesSuites[indexSuiteChoisie].add(carte);
                    joueur.retirerCarteChoisie();
                } else { // D⧫j⧫ 5 cartes dans la suite
                    // Compter les t⧫tes de boeufs et les ajouter au joueur
                    int tetesRecuperees = 0;
                    for (int carteSuite : cartesSuites[indexSuiteChoisie]) {
                        tetesRecuperees += Jeu.nbTetesBoeufsCarte(carteSuite);
                    }
                    joueur.ajouterTetesRecuperees(tetesRecuperees);

                    // Vider la suite et ajouter la carte du joueur
                    cartesSuites[indexSuiteChoisie].clear();
                    cartesSuites[indexSuiteChoisie].add(carte);
                    joueur.retirerCarteChoisie();
                }
            } else { // Aucune suite ne peut recevoir la carte
```

```
                if (!messageCartesEnCoursEnvoye) {
                    messageCartesEnCoursEnvoye = true;
                    System.out.println(messageCartesJoueurs + " vont ⍰tre
pos⍰es.");
                }

                System.out.println("Pour poser la carte " + carte + ", " +
joueur.getNom() + " doit choisir la s⍰rie qu'il va ramasser.");
                afficherSuites();

                // Choix d'une suite
                System.out.print("Saisissez votre choix : ");
                String choix = scanner.nextLine();
                try {
                    indexSuiteChoisie = Integer.parseInt(choix);
                } catch (NumberFormatException e) {
                }

                // Continue de demmander de choisir une s⍰rie jusqu'⍰ ce qu'une
s⍰rie soit choisie
                while (indexSuiteChoisie <= 0 || indexSuiteChoisie > 4) {
                    System.out.print("Choix invalide, saisissez votre choix :
");
                    choix = scanner.nextLine();
                    try {
                        indexSuiteChoisie = Integer.parseInt(choix);
                    } catch (NumberFormatException e) {
                    }
                }

                indexSuiteChoisie--; // D⍰cr⍰mente de 1 car les suites vont de 1
⍰ 4 pour le joueur

                // Compter les t⍰tes de boeufs et les ajouter au joueur
                int tetesRecuperees = 0;
                for (int carteSuite : cartesSuites[indexSuiteChoisie]) {
                    tetesRecuperees += Jeu.nbTetesBoeufsCarte(carteSuite);
                }
                joueur.ajouterTetesRecuperees(tetesRecuperees);

                // Vider la suite et ajouter la carte du joueur
                cartesSuites[indexSuiteChoisie].clear();
                cartesSuites[indexSuiteChoisie].add(carte);
                joueur.retirerCarteChoisie();
            }
        }

        // Message cartes plac⍰es et affichage des nouvelles suites
        System.out.println(messageCartesJoueurs + " ont ⍰t⍰ pos⍰es.");
        afficherSuites();

        // Tri croissant des joueurs par nombre de t⍰tes de boeufs r⍰cup⍰res
        Collections.sort(listeJoueurs, Joueur.triTetesRecupereesCroissant);

        // Affichage des joueurs qui ont rammass⍰ des t⍰tes de boeufs (par ordre
croissant)
        for (Joueur joueur : listeJoueurs) {
```

```
            int tetesRecuperees = joueur.getTetesRecuperees();
            if (tetesRecuperees > 0) {
                String nom = joueur.getNom();
                System.out.println(nom + " a ramass\u00e9 " + tetesRecuperees + "
t\u00eates de boeufs");
            }
        }

        // Transfert les t\u00eates de boeufs r\u00e9cup\u00e9r\u00e9es par chaque joueurs dans son
total de t\u00eates et remise \u00e0 0 des t\u00eates r\u00e9cup\u00e9r\u00e9es
        for (Joueur joueur : listeJoueurs) {
            joueur.transfertTetes();
        }

        // D\u00e9cr\u00e9mente de 1 le nombre de cartes restantes
        cartesRestantes--;
    }

    private void selectionCarte(Joueur joueur) {
        System.out.println("A " + joueur.getNom() + " de jouer.");

        // Pause
        Console.pause();

        // Affichage du jeu
        afficherSuites();
        joueur.afficherCartes();

        // Choix d'une carte
        int carteChoisie = -1;
//       Scanner scanner = new Scanner(System.in);
        System.out.print("Saisissez votre choix : ");
        String choix = scanner.nextLine(); // Entr\u00e9e du joueur
        try {
            carteChoisie = Integer.parseInt(choix);
        } catch (NumberFormatException e) {
        }

        // Continue de demmander de choisir une carte jusqu'\u00e0 ce qu'une carte
soit choisie
        while (!joueur.choisirCarte(carteChoisie)) {
            System.out.print("Choix invalide, saisissez votre choix : ");
            choix = scanner.nextLine(); // Entr\u00e9e du joueur
            try {
                carteChoisie = Integer.parseInt(choix);
            } catch (NumberFormatException e) {
            }
        }

        Console.clearScreen();
    }

    // Affiche la liste des cartes de chaque suite
    private void afficherSuites() {
        for (int i = 0; i < 4; i++) {
            System.out.print("- S\u00e9rie n\u00b0 " + (i + 1) + " : ");
            for (int c = 0; c < cartesSuites[i].size(); c++) {
```

```
                int carte = cartesSuites[i].get(c);
                System.out.print(texteCarte(carte));
                if (c < cartesSuites[i].size() - 1) {
                    System.out.print(", ");
                }
            }
            System.out.println(); // Retour ◆ la ligne
        }
    }

    private void affichageScoreFinal() {
        // Cr◆ation d'une liste qui contient tous les joueurs
        ArrayList<Joueur> listeJoueurs = new ArrayList<>();
        for (int i = 0; i < nbJoueurs; i++) {
            Joueur joueur = joueurs[i];
            listeJoueurs.add(joueur);
        }

        // Tri de la liste par nombre de t◆tes croissant
        Collections.sort(listeJoueurs, Joueur.triTetesTotalesCroissant);

        // Affichage du score
        System.out.println("** Score final");
        for (Joueur joueur : listeJoueurs) {
            String nom = joueur.getNom();
            int tetesBoeufs = joueur.getTetesBoeufs();
            System.out.println(nom + " a ramass◆ " + tetesBoeufs + " t◆tes de
boeufs");
        }
    }

    // Donne un String qui permet de montrer la carte ex: 42, 15(5) ou 55(7)
    public static String texteCarte(int carte) {
        String texte = Integer.toString(carte);
        int tetesBoeufs = nbTetesBoeufsCarte(carte);

        if (tetesBoeufs > 1) {
            texte += "(" + tetesBoeufs + ")";
        }

        return texte;
    }

    // Donne le nombre de t◆te de boeufs d'une carte
    public static int nbTetesBoeufsCarte(int carte) {
        int tetesBoeufs = 1;

        if (carte == 55) {// 55 vaut 7 t◆tes de boeufs
            tetesBoeufs = 7;
        } else if (carte == 101 || carte % 100 / 10 == carte % 100 % 10) {// La
carte a 2 nombres identiques (dizaines et unit◆s) ou (la carte est 101)
            tetesBoeufs = 5;
        } else if (carte % 10 == 5) {// La carte finit par un 5
            tetesBoeufs = 2;
        } else if (carte % 10 == 0) {// La carte finit par un 0
            tetesBoeufs = 3;
        }
```

```
        return tetesBoeufs;
    }

    public static void main(String[] args) {
        new Jeu();
    }
}
```