IUT de Paris - Rives de Seine
Université Paris Cité

Quentin
Wendling

Groupe 108

Département Informatique

Développement orienté objets

Projet 6-qui-prend !

Table des matières

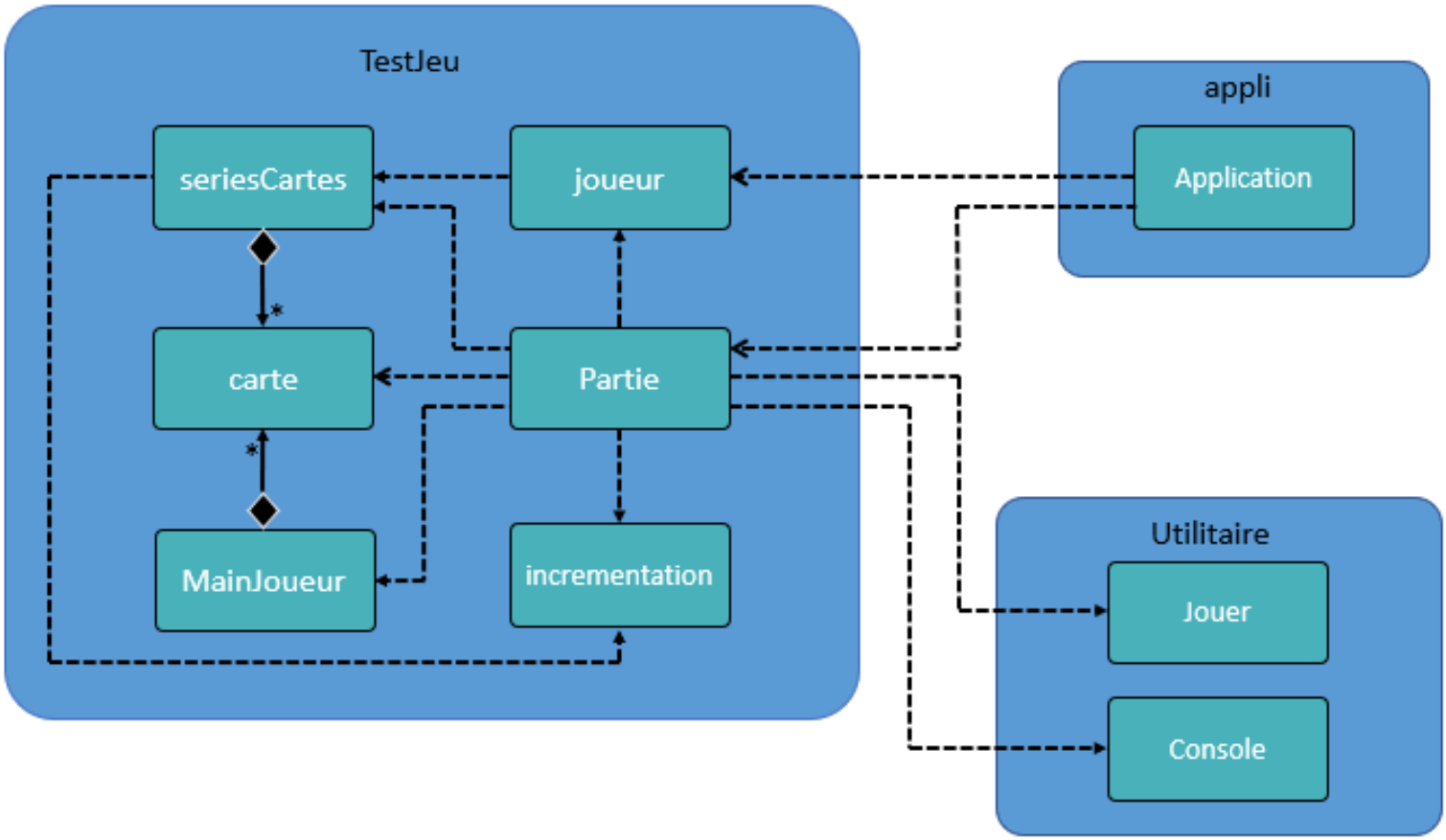
1. Introduction

L'objectif de ce projet était de modéliser un jeu classique (le 6 qui prend !) à partir d'une implémentation via un langage orienté objet (Java).

Pour ceci, j'ai été amené à utiliser différentes structures de classes, de listes et de données. Je me suis essentiellement appuyé sur des *ArrayList* pour gérer les différents objets nécessaires au bon fonctionnement du jeu.

Les principaux éléments qui ont été traités sont les joueurs, les cartes du jeu et celles distribuées par joueur ainsi que la comptabilisation des « têtes de bœufs » par joueur durant une manche (10 cartes jouées par personne).

2. Diagramme UML



3. Code Java des tests unitaires

3.1. carteTest

```
package TestJeu;

import static org.junit.Assert.*;

import org.junit.Test;

public class carteTest {


	@Test
	public void testCarte() {
	for (int i = 1; i < 105;i++) {
		carte c = new carte(i, 0);
		 if(i % 11 == 0) {
			if(i == 55) {
				assertTrue(c.getVCarte() == 7);
			}
			else {
				assertTrue(c.getVCarte() == 5);
			}
		 }
		 if(i % 5 == 0 && i != 55) {
			if(i % 10 == 0) {
				assertTrue(c.getVCarte() == 3);
			}
			else {
				assertTrue(c.getVCarte() == 2);
			}
		 }
		 else if(i % 11 != 0 && i % 55 != 0 && i % 5 != 0 && i % 10 !=
0) {
				assertTrue(c.getVCarte() == 1);
			}
		 assertTrue(c.getNCarte() == i);
		}
	}
}
```

3.2. joueurTest

```
package TestJeu;

import static org.junit.Assert.*;

import org.junit.Test;

public class joueurTest {


	@Test
	public void testGetNomJoueur() {
		joueur.addJoueur("alpha");
		joueur.addJoueur("beta");
```

```
            joueur.addJoueur("charlie");
        String s1 = "alpha";
        String s2 = "beta";
        String s3 = "charlie";
        assertEquals(joueur.GetNomJoueur(0), s1);
        assertEquals(joueur.GetNomJoueur(1), s2);
        assertEquals(joueur.GetNomJoueur(2), s3);
    }

      @Test
      public void testStringBuilder() {
            joueur.addJoueur("alpha");
            joueur.addJoueur("beta");
            joueur.addJoueur("charlie");

        StringBuilder s1 = new StringBuilder(", beta et charlie. Merci de
jouer à 6 qui prend !");
        System.out.println(joueur.affichageNom() + s1.toString());
        assertEquals(joueur.affichageNom().toString(), s1.toString());
      }
}
```

3.3. MainJoueurTest

```
package TestJeu;

import static org.junit.Assert.*;

import java.util.Collections;

import org.junit.Test;

public class MainJoueurTest {

      @Test
      public void testCarteMainJoueur() {
            for(int a = 0; a < 104 ; a++) {
                  carte.addCarte(a, 0);
            }
            Collections.shuffle(carte.cartesListe);
            int nb = 0;
            for(int joueur = 0; joueur < 2;joueur++) {
                  MainJoueur.addCarteDansMain(joueur, (nb));
                  nb = nb + 10;
            }
            for(int c = 0; c< 10 ; c++) {
                  for(int c2 = 0;c2 < 10;c2++) {
                        assertTrue(MainJoueur.GetNbrMainJoueur(0, c) !=
MainJoueur.GetNbrMainJoueur(1, (c2)));
                  }
            }
      }
}
```

4. Code Java complet

4.1. Package

4.1.1. appli

4.1.1.1. Application

```java
package appli;

import ClassCode.Partie;
import ClassCode.joueur;

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

public class Application {

    public static void main(String[] args) {
        try {
            File F = new File("config.txt");
            Scanner scanner = new Scanner(F);

            String s;
            while (scanner.hasNextLine()) {
                s = scanner.nextLine();
                joueur.addJoueur(s);
            }
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        joueur.verifJoueur();
        joueur.affJoueur();
        Partie.CreationCarte();
        Partie.CreationMain();
        Partie.CreationSeries();
        Partie.Jeu();
    }
}
```

4.1.2. ClassCode

4.1.2.1. Partie

```
package TestJeu;

import Utilitaire.Jouer;

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

import static Utilitaire.Console.clearScreen;
import static Utilitaire.Console.pause;

public class Partie {
    private static final int nbrDeCarte = 104;
    private static final int nbrSeries = 4;
    private static int DistribCartes = 0;

    private static Scanner sc = new Scanner(System.in);

    public Partie(int DistribCartes){
        Partie.DistribCartes = DistribCartes;
    }

    public static void CreationCarte(){
        for(int tmpCarte = 1; tmpCarte < (nbrDeCarte + 1); tmpCarte++){
            carte.addCarte(tmpCarte,tmpCarte);
        }
        Collections.shuffle(carte.cartesListe);
    }

    public static void CreationMain() {
        for (int i = 0; i < joueur.nbPlayers(); i++) {
            MainJoueur.addCarteDansMain(i, DistribCartes);
            DistribCartes = 10 * (i + 1);
        }
    }

    public static void CreationSeries(){
        for(int i = 0; i < nbrSeries; i++){
            seriesCartes.addSerie(i, DistribCartes);
            DistribCartes++;
        }
    }

    public static void Jeu() {
        while (MainJoueur.cartesMain.get(MainJoueur.cartesMain.size() -
1).size() > 0) {
            for (int joueurActuelle = 0; joueurActuelle <
joueur.nbPlayers(); joueurActuelle++) {
                System.out.println("A " +
joueur.GetNomJoueur(joueurActuelle) + " de jouer.");
                pause();
                seriesCartes.affSerie();

                System.out.print("- Vos cartes : " +
MainJoueur.GetNbrMainJoueur(joueurActuelle, 0));
                if (MainJoueur.GetValMainJoueur(joueurActuelle, 0) != 1)
```

```
                    System.out.print(" (" +
MainJoueur.GetValMainJoueur(joueurActuelle,0) + ")");

                for (int i = 1; i <
MainJoueur.cartesMain.get(joueurActuelle).size(); i++) {
                    System.out.print(", " +
MainJoueur.GetNbrMainJoueur(joueurActuelle, i));
                    if (MainJoueur.GetValMainJoueur(joueurActuelle, i) !=
1)
                        System.out.print(" (" +
MainJoueur.GetValMainJoueur(joueurActuelle,i) + ")");
                }
                System.out.println();

                System.out.print("Saisissez votre choix : ");

                // verification si la carte existe
                String tmpjouer;
                boolean exist = false;
                int valueCard = 0;
                while (!exist) {
                    tmpjouer = sc.next();
                    while (Jouer.isStringInteger(tmpjouer) == false) { //
verification si pas nombre (caractere)
                        System.out.print("Vous n'avez pas cette carte,
saisissez votre choix : ");
                        tmpjouer = sc.next();
                    }
                    valueCard = Integer.parseInt(tmpjouer);
                    for (int avoir = 0; avoir <
MainJoueur.cartesMain.get(joueurActuelle).size(); avoir++) {
                        if (valueCard ==
MainJoueur.GetNbrMainJoueur(joueurActuelle, avoir)) { // verification
valeur existe
                            exist = true;
                            incrementation.addIncrementation(avoir,
joueurActuelle);

MainJoueur.cartesMain.get(joueurActuelle).remove(avoir);
                            break;
                        }
                    }

                    if (!exist) {
                        System.out.print("Vous n'avez pas cette carte,
saisissez votre choix : ");
                    }
                }
                clearScreen();
            }

            Collections.sort(incrementation.incrementationCarte);

            // incrementation des cartes
            for (int i = 0; i < joueur.nbPlayers(); i++) {
                int ValserieTmp = 0;
                int NumserieTmp = -1; // -1 en dehors des series
                boolean existS = false;
                for (int j = 0; j < nbrSeries; j++) {
                    if
(seriesCartes.ListSerieCarte.get(j).get(seriesCartes.ListSerieCarte.get(j).
```

```
size() - 1).getNCarte() < incrementation.GetNbrSurCarteIncrem(i)) {
                        if
(seriesCartes.ListSerieCarte.get(j).get(seriesCartes.ListSerieCarte.get(j).
size() - 1).getNCarte() > ValserieTmp) {
                            NumserieTmp = j;
                            ValserieTmp =
seriesCartes.ListSerieCarte.get(j).get(seriesCartes.ListSerieCarte.get(j).s
ize() - 1).getNCarte();
                            existS = true;
                        }
                    }
                }

                if (existS == true) {
                    if (seriesCartes.ListSerieCarte.get(NumserieTmp).size()
== 5) { // si le joueur pose la 6eme carte
                        for (int sum = 0; sum < 5; sum++) {
                            joueur.getAjoutPointJoueur((NumserieTmp+1),
sum,incrementation.GetNumJoueurIncrem(i));
                            int tmp =
seriesCartes.GetValSurCarteSerie(NumserieTmp, sum);
                            incrementation.getPointIncrem(i, tmp);
                        }

                        for (int rmv =
seriesCartes.ListSerieCarte.get(NumserieTmp).size() - 1; rmv > 0; rmv--) {

seriesCartes.ListSerieCarte.get(NumserieTmp).remove(rmv);
                        }

seriesCartes.ListSerieCarte.get(NumserieTmp).remove(0);
                    }
                    seriesCartes.addDansSerie(NumserieTmp, i);
                }

                else {
                    SelectCart(); // voir en bas
                    System.out.println(" et " +
incrementation.GetNbrSurCarteIncrem(joueur.nbPlayers() - 1) + " (" +
joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(joueur.nbPlayers() -
1)) + ")" + " vont être posées.");
                    System.out.println("Pour poser la carte " +
incrementation.GetNbrSurCarteIncrem(i) + ", " +
joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(0)) + " doit choisir
la série qu'il va ramasser.");
                    seriesCartes.affSerie();
                    System.out.print("Saisissez votre choix : ");
                    int choix = -1; // pas valeur index d'une serie
                    String tmpChoix;

                    boolean exist = false;
                    while (!exist) {
                        tmpChoix = sc.next();
                        while (Jouer.isStringInteger(tmpChoix) == false) {
// verif carac
                            System.out.print("Ce n'est pas une série
valide, saisissez votre choix : ");
                            tmpChoix = sc.next();
                        }
                        choix = Integer.parseInt(tmpChoix);
```

```
                        if (choix > 0 && choix < 5) {
                                exist = true;
                        }
                        if (!exist) {
                            System.out.print("Ce n'est pas une série
valide, saisissez votre choix : ");
                        }
                    }

                    for (int sum = 0; sum <
seriesCartes.ListSerieCarte.get(choix - 1).size(); sum++) {
                        joueur.getAjoutPointJoueur(choix, sum,
incrementation.GetNumJoueurIncrem(0));
                    }

                    for (int sum = 0; sum <
seriesCartes.ListSerieCarte.get(choix - 1).size(); sum++) {
                        int tmp = seriesCartes.GetValSurCarteSerie(choix -
1, sum);
                        incrementation.getPointIncrem(0, tmp);
                    }

                    if (seriesCartes.ListSerieCarte.get(choix - 1).size() >
1) {
                        seriesCartes.ListSerieCarte.get(choix -
1).subList(1, seriesCartes.ListSerieCarte.get(choix - 1).size()).clear();
                    }
                    seriesCartes.addDansSerie(choix - 1, 0);
                    seriesCartes.ListSerieCarte.get(choix - 1).remove(0);
                }
            }

            Collections.sort(incrementation.incrementationCarte);

            SelectCart(); // voir en bas
            System.out.println(" et " +
incrementation.GetNbrSurCarteIncrem(joueur.nbPlayers() - 1) + " (" +
joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(joueur.nbPlayers() -
1)) + ")" + " ont été posées.");
            seriesCartes.affSerie();
            boolean ramasseOk = false;
            for (int ramasser = 0; ramasser < joueur.nbPlayers();
ramasser++) {
                if (incrementation.GetRamIncrem(ramasser) != 0) {

System.out.println(joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(ra
masser)) + " a ramassé " + incrementation.GetRamIncrem(ramasser) + " têtes
de boeufs");
                    ramasseOk = true;
                }
            }
            if (!ramasseOk) {
                System.out.println("Aucun joueur ne ramasse de tête de
boeufs.");
            }

incrementation.incrementationCarte.removeAll(incrementation.incrementationC
arte);
        }
        // test point ect
        joueur.JoueurTest.sort(joueur::comparerJoueurs);
```

```
        System.out.println("** Score final");
        for (int i = 0; i < joueur.nbPlayers(); i++) {
            System.out.println(joueur.GetNomJoueur(i) + " a ramassé " +
joueur.getPointJoueur(i) + " têtes de boeufs");
        }
    }

    private static void SelectCart() {
        System.out.print("Les cartes " +
incrementation.GetNbrSurCarteIncrem(0) + " (" +
joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(0)) + ")");
        for (int affJ = 1; affJ < (joueur.nbPlayers() - 1); affJ++) {
            System.out.print(", " +
incrementation.GetNbrSurCarteIncrem(affJ) + " (" +
joueur.GetNomJoueur(incrementation.GetNumJoueurIncrem(affJ)) + ")");
        }
    }
}
```

4.1.2.2. Joueur

```
package TestJeu;


import java.util.ArrayList;

public class joueur {

    private String NomJoueur;
    private int pointJoueur;

    private static int nbPlayers = 0;
    private static final int JOUEUR_MAX = 10;
    private static final int JOUEUR_MIN = 2;

    static ArrayList<joueur> JoueurTest = new ArrayList<>();

    public joueur(String NomJ, int pointJ){
        this.NomJoueur = NomJ;
        this.pointJoueur = pointJ;
    }


    public static void addJoueur(String s){
        JoueurTest.add(new joueur(s, 0));
        nbPlayers++;
    }

    public static void verifJoueur(){
        if (nbPlayers > JOUEUR_MAX) {
            System.out.println("il y a trop de joueurs !");
            System.exit(0);
        }
        if (nbPlayers < JOUEUR_MIN) {
            System.out.println("il y a pas assez de joueurs !");
            System.exit(0);
        }
    }
```

```
    public static StringBuilder affichageNom(){
        int joueurDeb = 1;
        StringBuilder sb = new StringBuilder();
        while(joueurDeb < (nbPlayers - 1)){
            sb.append(", "+ JoueurTest.get(joueurDeb).NomJoueur);
            joueurDeb++;
        }
        sb.append(" et " + JoueurTest.get(joueurDeb).NomJoueur + ". Merci
de jouer à 6 qui prend !");
        return sb;
    }

    public static int nbPlayers(){
        return nbPlayers;
    }

    public static String GetNomJoueur(int i){
        return JoueurTest.get(i).NomJoueur;
    }


    public static void getAjoutPointJoueur(int choix, int sum, int idJ) {
        JoueurTest.get(idJ).pointJoueur +=
seriesCartes.GetValSurCarteSerie(choix - 1, sum);
    }
    public static int getPointJoueur(int i) {
        return JoueurTest.get(i).pointJoueur;
    }

    public static void affJoueur(){
        String s =  "Les " + nbPlayers + " joueurs sont " +
JoueurTest.get(0).NomJoueur;
        s += affichageNom();
        System.out.println(s);
    }

    public int comparerJoueurs(joueur j){
        if(this.pointJoueur == j.pointJoueur) return
this.NomJoueur.compareTo(j.NomJoueur);
        return this.pointJoueur - j.pointJoueur;
    }
}
```

4.1.2.3. seriesCartes

```
package TestJeu;

import java.util.ArrayList;

public class seriesCartes {

    static ArrayList<ArrayList<carte>> ListSerieCarte = new ArrayList<>();

    public static void addSerie(int i, int DistribCartes) {
        ListSerieCarte.add(new ArrayList<carte>(i));
        ListSerieCarte.get(i).add(new
```

```
carte(carte.cartesListe.get(DistribCartes).getNCarte(),carte.cartesListe.ge
t(DistribCartes).getVCarte()));
    }
    public static void affSerie(){
        for(int SerieCAff = 0; SerieCAff < 4;SerieCAff++){
            System.out.print("- série n° " + (SerieCAff + 1) + " :");
            System.out.print(" " +
ListSerieCarte.get(SerieCAff).get(0).getNCarte());
            if(ListSerieCarte.get(SerieCAff).get(0).getVCarte() != 1)
                System.out.print(" (" +
ListSerieCarte.get(SerieCAff).get(0).getVCarte() + ")");
            for (int j = 1; j < ListSerieCarte.get(SerieCAff).size(); j++)
{
                System.out.print(", " +
ListSerieCarte.get(SerieCAff).get(j).getNCarte());
                if(ListSerieCarte.get(SerieCAff).get(j).getVCarte() != 1)
                    System.out.print(" (" +
ListSerieCarte.get(SerieCAff).get(j).getVCarte() + ")");
            }
            System.out.println();
        }
    }
    public static void addDansSerie(int NumSerie ,int i){
        ListSerieCarte.get(NumSerie).add(new
carte(incrementation.GetNbrSurCarteIncrem(i),
incrementation.GetValSurCarteIncrem(i)));
    }

    public static int GetValSurCarteSerie (int Serie, int i){
        return ListSerieCarte.get(Serie).get(i).getVCarte();
    }
}
```

4.1.2.4. MainJoueur

```
package TestJeu;

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

public class MainJoueur{
    static ArrayList<ArrayList<carte>> cartesMain = new ArrayList<>();
    private int nbrSurCartes;

public static void addCarteDansMain(int idJoueur, int DistribCartes){
    cartesMain.add(new ArrayList<>(idJoueur));
    for (int j = 0; j < 10; j++) {
        cartesMain.get(idJoueur).add(new
carte(carte.cartesListe.get(DistribCartes).getNCarte(),carte.cartesListe.ge
t(DistribCartes).getVCarte()));
        DistribCartes++;
    }
}


    public static int GetNbrMainJoueur (int joueur,int indexC){
        return cartesMain.get(joueur).get(indexC).getNCarte();
    }
```

```
    public static int GetValMainJoueur (int joueur,int indexC){
        return cartesMain.get(joueur).get(indexC).getVCarte();
    }
}
```

4.1.2.5. Incrementation

```
package TestJeu;


import java.util.ArrayList;

import static TestJeu.MainJoueur.*;

public class incrementation implements Comparable<incrementation> {
    static ArrayList<incrementation> incrementationCarte = new
ArrayList<>();


    private int nbrSurCartes;
    private int ValeurCartes;
    private int NumJoueur;
    private int ramasser;


    public static void addIncrementation(int avoir, int joueurActuelle){
        incrementationCarte.add(new
incrementation(GetNbrMainJoueur(joueurActuelle, avoir),
GetValMainJoueur(joueurActuelle, avoir),joueurActuelle, 0));
    }

    public incrementation(int nbrSurCartes, int ValeurCartes, int
NumJoueur,int ramasser) {
        this.nbrSurCartes = nbrSurCartes;
        this.ValeurCartes = ValeurCartes;
        this.NumJoueur = NumJoueur;
        this.ramasser = ramasser;
    }

    public static int GetNbrSurCarteIncrem (int nbr){
        return incrementationCarte.get(nbr).nbrSurCartes;
    }
    public static int GetRamIncrem (int nbr){
        return incrementationCarte.get(nbr).ramasser;
    }

    public static int GetValSurCarteIncrem (int Val){
        return incrementationCarte.get(Val).ValeurCartes;
    }

    public static int GetNumJoueurIncrem (int joueur){
        return incrementationCarte.get(joueur).NumJoueur;
    }

    public static void getPointIncrem(int i, int tmp) {
        incrementationCarte.get(i).ramasser += tmp;
    }
```

```
    @Override
    public int compareTo(incrementation o) {
        return (this.nbrSurCartes - o.nbrSurCartes);
    }
}
```

4.1.2.6. Carte

```
package TestJeu;

import java.util.ArrayList;

public class carte {

    private int numCarte;
    private int valeurCarte;

    static ArrayList<carte> cartesListe = new ArrayList<>();

    public carte(int numCarte, int valeurCarte){
        this.numCarte = numCarte;
        boolean tmpAucunChangement = false;
        this.valeurCarte = 0;
        if (numCarte % 5 == 0) { // si c'est divisible par 5 ajoute a la
valeur temporaire 2
            this.valeurCarte = this.valeurCarte + 2;
            tmpAucunChangement = true;
        }
        if (numCarte % 10 == 0) { // si c'est divisible par 10 remplace sa
valeur temporaire par 3
            this.valeurCarte = 3;
        }
        if (numCarte % 11 == 0) { //si c'est divisible par 11 ajoute a la
valeur temporaire 5
            this.valeurCarte = this.valeurCarte + 5;
            tmpAucunChangement = true;
        }
        if (!tmpAucunChangement) // si aucune des instructions au dessus
n'est realise remplace la valeur temporaire par 1
            this.valeurCarte = 1;
    }

    public static void addCarte(int numCarte, int valeurCarte){
        cartesListe.add(new carte(numCarte, valeurCarte));
    }

    public int getNCarte(){
        return numCarte;
    }
    public int getVCarte(){
        return valeurCarte;
    }

}
```

4.1.3. Utilitaire

4.1.3.1. Console

```
package Utilitaire;

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 {
        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() {
    }
}
```

4.1.3.2. Jouer

```
package Utilitaire;

import java.util.Scanner;

public class Jouer {
    public static boolean isStringInteger(String stringToCheck) {
        Scanner sc = new Scanner(stringToCheck.trim());
        if(!sc.hasNextInt()) return false;
        sc.nextInt();
        return !sc.hasNext();
    }
}
```

5. Bilan

5.1. Difficultés rencontrées

Les principales difficultés que j'ai rencontrées provenaient de la structure de données à modéliser. Au départ j'ai voulu programmer ce projet, en utilisant des tableaux dynamiques, mais il me manquait des éléments sur la programmation objet à partir du langage java. Ceux-ci ont été corrigés au fur et à mesure des cours et TD/TP. Du coup je me suis plus consacré, au départ, sur la partie algorithmique (analyse et conception) pour bien modéliser les différentes parties du problème et leur enchainement.

5.2. Réussi

Dans l'ensemble je pense avoir réussi avoir une implémentation correcte du jeu. J'ai rajouté de nombreux tests comme : l'absence de participant, nombre de participant de devant pas être supérieur à 10 et inférieur à 2 pour vérifier le bon déroulement du jeu.

5.3. Amélioration possible

Différentes améliorations sont possibles comme l'utilisation d'une stratégie intelligente pour jouer contre la machine. La réalisation d'une interface graphique (carte, joueur) qui entrainerait plus de tests unitaires. Enfin une pourrait le jeu originel comporte normalement plusieurs manches (jusqu'à 66 « tête de bœufs » pour le perdant). Ceci pourrait être facilement intégré grâce à une boucle (avec vérification et sortie éventuelle du jeu).