

**Equipe 28**

Théo Constant 108

Ethan Brosselard 110

**PROJET**

**PROGRAMMATION**

Ressource

Développement Orientée Objet



**Table des matières**

[Introduction 3](#_Toc98009581)

[Diagramme UML 4](#_Toc98009582)

[Test unitaires 5](#_Toc98009583)

[Code 11](#_Toc98009584)

[Bilan 26](#_Toc98009585)

# **Introduction**

Ce projet est un projet à l’IUT Paris Cité (anciennement Paris Descartes). Le but de ce projet est de recréer le jeu de carte « 6 qui prend » en Java.

Ce jeu de carte est un jeu de société avec des cartes crée par Wolfgang Kramer en 1994. Il est édité chez Amigo et la version française est distribuée par Gigamic. Il se joue entre 2 et 10 joueurs et une partie dure environ 45 minutes.

Dans ce jeu, il y a un 104 cartes, chaque joueur se voit distribué 10 cartes et 4 cartes sont posées afin de constituer 4 rangées. Chaque carte comporte un certain nombre de tête de bœuf. Le but est de poser toutes ces cartes en récupérant le moins de têtes de bœuf à la fin. Le jeu s’arrête quand un des joueurs atteint 66 têtes de bœuf.

Voici les règles du jeu : <https://www.regledujeu.fr/6-qui-prend/>

Dans notre projet, on se contentera de faire le dérouler d’une seule manche, c’est-à-dire, quand tous les joueurs ont posé toutes leurs cartes.

# **Diagramme UML**

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Les nombres désignent le min et le max d’objet qu’utilise la classe. Par exemple, le jeu comporte 2 joueurs minimum, jusqu’à 10 joueurs.

# **Test unitaires**

**CarteTest**

package Tests;  
  
import Cartes.Carte;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
public class CarteTest {  
  
 @Test  
 public void testGetNumero() {  
 for (int i = 1; i <= 104; ++i) {  
 *assertEquals*(i, new Carte().getNumero());  
 }  
 }  
  
 @Test  
 public void testGetTeteDeBoeuf() {  
 *assertEquals*(1, new Carte(23).getTeteDeBoeuf());  
 *assertEquals*(2, new Carte(35).getTeteDeBoeuf());  
 *assertEquals*(3, new Carte(70).getTeteDeBoeuf());  
 *assertEquals*(5, new Carte(99).getTeteDeBoeuf());  
 *assertEquals*(7, new Carte(55).getTeteDeBoeuf());  
 }  
   
 @Test  
 public void testClone() {  
 Carte test = new Carte(11);  
 *assertEquals*(test.getNumero(), test.clone().getNumero());  
 *assertEquals*(test.getTeteDeBoeuf(), test.clone().getTeteDeBoeuf());  
 }  
   
 @Test  
 public void testToString() {  
 Carte test1 = new Carte(15);  
 Carte test2 = new Carte(60);  
 Carte test3 = new Carte(77);  
 Carte test4 = new Carte(55);  
 *assertEquals*(new Carte(37).toString(), String.*valueOf*(new Carte(37).toString()));  
 *assertEquals*(test1.toString(), test1.getNumero() + " (" + test1.getTeteDeBoeuf() + ")");  
 *assertEquals*(test2.toString(), test2.getNumero() + " (" + test2.getTeteDeBoeuf() + ")");  
 *assertEquals*(test3.toString(), test3.getNumero() + " (" + test3.getTeteDeBoeuf() + ")");  
 *assertEquals*(test4.toString(), test4.getNumero() + " (" + test4.getTeteDeBoeuf() + ")");  
 }  
}

**JoueurTest**

package Tests;  
  
import static org.junit.Assert.\*;  
  
import java.util.ArrayList;  
  
import org.junit.Test;  
  
import Cartes.Carte;  
import Joueurs.Joueur;  
  
public class JoueurTest {  
   
 @Test  
 public void testGetNom() {  
 Joueur j = new Joueur("Linki");  
   
 *assertEquals*(j.getNom(), "Linki");  
 }  
   
 @Test  
 public void testGetTeteBoeuf() {  
 Joueur j = new Joueur("Béboo");  
   
 *assertEquals*(j.getTeteBoeuf(), 0);  
 j.ajouterTeteBoeuf(7);  
 *assertEquals*(j.getTeteBoeuf(), 7);  
 }  
   
 @Test  
 public void testGetCartes() {  
 Joueur j = new Joueur("Bebou");  
 ArrayList<Carte> c = new ArrayList<>();  
   
 *assertEquals*(j.getCartes(), c);  
 for (int i = 0; i < 10; i++) {  
 j.getCartes().add(new Carte(42+i));  
 c.add(new Carte(42+i));  
 *assertEquals*(j.getCartes().get(i).getNumero(), c.get(i).getNumero());  
 }  
 }  
   
 @Test  
 public void testEstVide() {  
 Joueur j = new Joueur("Bebow");  
   
 *assertTrue*(j.estVide());  
 j.getCartes().add(new Carte(42));  
 *assertFalse*(j.estVide());  
 }  
   
 @Test  
 public void testClone() {  
 Joueur j1 = new Joueur("UwU");  
 Joueur j2;  
   
 for (int i = 0; i < 5; i++) {  
 j1.getCartes().add(new Carte(87+i));  
 }  
   
 j2 = j1.clone();  
   
 *assertEquals*(j1.getNom(), j2.getNom());  
 *assertEquals*(j1.getCartes().get(3).getNumero(), j1.getCartes().get(3).getNumero());  
 }  
   
 @Test  
 public void testToString() {  
 Joueur j = new Joueur("-\_-");  
   
 *assertEquals*(j.toString(), "-\_-");  
 }  
}

**SerieTest**

package Tests;  
  
import static org.junit.Assert.\*;  
  
import java.util.ArrayList;  
  
import org.junit.Test;  
  
import Cartes.Carte;  
import Cartes.Serie;  
  
public class SerieTest {  
   
 @Test  
 public void testGetSerie() {  
 Serie s = new Serie();  
 ArrayList<Carte> test = new ArrayList<>();  
   
 *assertEquals*(s.getSerie(), new ArrayList<Carte>());  
 for (int i = 0; i < s.getNbCartesMax(); i++) {  
 s.ajouterCarte(new Carte(i));  
 test.add(new Carte(i));  
 *assertEquals*(s.getSerie().get(i).getNumero(), test.get(i).getNumero());  
 }  
 }  
   
 @Test  
 public void testGetCarte() {  
 Serie s = new Serie();  
 ArrayList<Carte> test = new ArrayList<>();  
   
 for (int i = 0; i < s.getNbCartesMax(); i++) {  
 s.ajouterCarte(new Carte(i));  
 test.add(new Carte(i));  
 *assertEquals*(s.getSerie().get(i).getNumero(), test.get(i).getNumero());  
 }  
 }  
   
 @Test  
 public void testGetNbCarteMax() {  
 Serie s = new Serie();  
   
 *assertEquals*(s.getNbCartesMax(), 5);  
 }  
   
 @Test   
 public void testGetDerniereCarte() {  
 Serie s = new Serie();  
 Carte c = new Carte(17);  
   
 for (int i = 0; i < s.getNbCartesMax()-1; i++) {  
 s.ajouterCarte(new Carte(i));  
 }  
 s.ajouterCarte(c);  
 assertEquals(s.getDerniereCarte().getNumero(), c.getNumero());  
 }  
   
 @Test  
 public void testAjouterCarte() {  
 Serie s = new Serie();  
   
 for (int i = 0; i < s.getNbCartesMax(); i++) {  
 s.ajouterCarte(new Carte(i));  
 assertEquals(s.getSerie().get(i).getNumero(), new Carte(i).getNumero());  
 }  
 }  
   
 @Test  
 public void testVider() {  
 Serie s = new Serie();  
   
 for (int i = 0; i < s.getNbCartesMax(); i++) {  
 s.ajouterCarte(new Carte(i));  
 }  
 assertEquals(s.getSerie().size(), s.getNbCartesMax());  
 for (int i = 0; i < s.getNbCartesMax()-1; i++) {  
 s.vider();  
 }  
 assertEquals(s.getSerie().size(), 0);  
 }  
   
 @Test  
 public void testToString() {  
 Serie s = new Serie();  
   
 s.ajouterCarte(new Carte(54));  
 assertEquals(s.toString(), s.getSerie().get(0).toString());  
 s.ajouterCarte(new Carte(77));  
 assertEquals(s.toString(), s.getSerie().get(0).toString() + ", " + s.getSerie().get(1).toString());  
 s.ajouterCarte(new Carte(55));  
 s.ajouterCarte(new Carte(17));  
 s.ajouterCarte(new Carte(20));  
 assertEquals(s.toString(), s.getSerie().get(0).toString() + ", " + s.getSerie().get(1).toString() + ", " +   
 s.getSerie().get(2).toString() + ", " + s.getSerie().get(3).toString() + ", " + s.getSerie().get(4).toString());  
 }  
}

**PaquetTest**

package Tests;  
  
import Cartes.Paquet;  
import org.junit.Test;  
  
import static org.junit.Assert.\*;  
  
public class PaquetTest {  
   
 private Paquet p = new Paquet();  
  
 @Test  
 public void testGetNombreCartes() {  
 *assertEquals*(104, p.getNombreCartes());  
 }  
  
 @Test  
 public void testGetDerniereCarte() {  
 *assertEquals*(104, p.enlever().getNumero());  
 *assertEquals*(103, p.enlever().getNumero());  
 *assertEquals*(102, p.getDerniereCarte().getNumero());  
 }  
  
 @Test  
 public void testEstVide() {  
 while(!p.estVide()) {  
 p.enlever();  
 }  
 *assertEquals*(0, p.getNombreCartes());  
 }  
   
 @Test  
 public void testEnlever() {  
 Paquet p = new Paquet();  
   
 *assertEquals*(104, p.getNombreCartes());  
 for (int i = 0; i < 42; i++) {  
 p.enlever();  
 }  
 *assertEquals*(62, p.getNombreCartes());  
 }  
}

**JeuTest**

package Tests;  
  
import static org.junit.Assert.\*;  
  
import Cartes.Carte;  
import org.junit.Test;  
  
import Jeu.Jeu;  
  
public class JeuTest {  
   
 @Test  
 public void testGetNombreJoueurs() {  
 Jeu jeu = new Jeu();  
 *assertTrue*(jeu.getNombreJoueurs() > 2 && jeu.getNombreJoueurs() <= 10);  
 }  
  
 @Test  
 public void testTriJoueurs() {  
 Jeu jeu = new Jeu();  
 for (int i = 0; i < jeu.getNombreJoueurs(); ++i) {  
 jeu.getJoueur(i).setCarteChoisi(new Carte(104-i));  
 }  
 jeu.triJoueurs();  
 for (int i = 0; i < jeu.getNombreJoueurs()-1; ++i) {  
 *assertTrue*(jeu.getJoueur(i).getCarteChoisi().getNumero() < jeu.getJoueur(i+1).getCarteChoisi().getNumero());  
 }  
 }  
  
}

**Test d’une trace d’exécution**

Une image contenant texte, capture d’écran, moniteur, noir

Description générée automatiquement

maTrace.txt

Les 3 joueurs sont Ethan, Théo et Oliwer. Merci de jouer à 6 qui prend !

A Ethan de jouer.

<pause>

- série n° 1 : 12

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 1, 13, 25 (2), 29, 54, 61, 67, 68, 75 (2), 90 (3)

Saisissez votre choix : 13

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 12

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 3, 17, 33 (5), 36, 58, 64, 85 (2), 98, 99 (5), 104

Saisissez votre choix : 17

<clearScreen>

A Oliwer de jouer.

<pause>

- série n° 1 : 12

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 4, 7, 14, 19, 20 (3), 27, 31, 60 (3), 91, 97

Saisissez votre choix : 14

<clearScreen>

Les cartes 13 (Ethan), 14 (Oliwer) et 17 (Théo) ont été posées.

- série n° 1 : 12, 13, 14, 17

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

Aucun joueur ne ramasse de tête de boeufs.

A Ethan de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 1, 25 (2), 29, 54, 61, 67, 68, 75 (2), 90 (3)

Saisissez votre choix : 75

<clearScreen>

A Oliwer de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 4, 7, 19, 20 (3), 27, 31, 60 (3), 91, 97

Saisissez votre choix : 19

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17

- série n° 2 : 72

- série n° 3 : 48

- série n° 4 : 80 (3)

- Vos cartes : 3, 33 (5), 36, 58, 64, 85 (2), 98, 99 (5), 104

Saisissez votre choix : 85

<clearScreen>

Les cartes 19 (Oliwer), 75 (Ethan) et 85 (Théo) ont été posées.

- série n° 1 : 12, 13, 14, 17, 19

- série n° 2 : 72, 75 (2)

- série n° 3 : 48

- série n° 4 : 80 (3), 85 (2)

Aucun joueur ne ramasse de tête de boeufs.

A Oliwer de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17, 19

- série n° 2 : 72, 75 (2)

- série n° 3 : 48

- série n° 4 : 80 (3), 85 (2)

- Vos cartes : 4, 7, 20 (3), 27, 31, 60 (3), 91, 97

Saisissez votre choix : 20

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17, 19

- série n° 2 : 72, 75 (2)

- série n° 3 : 48

- série n° 4 : 80 (3), 85 (2)

- Vos cartes : 1, 25 (2), 29, 54, 61, 67, 68, 90 (3)

Saisissez votre choix : 90

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 12, 13, 14, 17, 19

- série n° 2 : 72, 75 (2)

- série n° 3 : 48

- série n° 4 : 80 (3), 85 (2)

- Vos cartes : 3, 33 (5), 36, 58, 64, 98, 99 (5), 104

Saisissez votre choix : 58

<clearScreen>

Les cartes 20 (Oliwer), 58 (Théo) et 90 (Ethan) ont été posées.

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58

- série n° 4 : 80 (3), 85 (2), 90 (3)

Oliwer a ramassé 5 têtes de boeufs

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 4, 7, 27, 31, 60 (3), 91, 97

Saisissez votre choix : 60

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 3, 33 (5), 36, 64, 98, 99 (5), 104

Saisissez votre choix : 64

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 1, 25 (2), 29, 54, 61, 67, 68

Saisissez votre choix : 61

<clearScreen>

Les cartes 60 (Oliwer), 61 (Ethan) et 64 (Théo) ont été posées.

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

Aucun joueur ne ramasse de tête de boeufs.

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 4, 7, 27, 31, 91, 97

Saisissez votre choix : 31

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 1, 25 (2), 29, 54, 67, 68

Saisissez votre choix : 25

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 3, 33 (5), 36, 98, 99 (5), 104

Saisissez votre choix : 33

<clearScreen>

Les cartes 25 (Ethan), 31 (Oliwer) et 33 (Théo) ont été posées.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

Aucun joueur ne ramasse de tête de boeufs.

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 1, 29, 54, 67, 68

Saisissez votre choix : 1

<clearScreen>

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 4, 7, 27, 91, 97

Saisissez votre choix : 97

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

- Vos cartes : 3, 36, 98, 99 (5), 104

Saisissez votre choix : 99

<clearScreen>

Les cartes 1 (Ethan), 97 (Oliwer) et 99 (Théo) vont être posées.

Pour poser la carte 1, Ethan doit choisir la série qu'il va ramasser.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 72, 75 (2)

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3)

Saisissez votre choix : 2

Les cartes 1 (Ethan), 97 (Oliwer) et 99 (Théo) ont été posées.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 1

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

Ethan a ramassé 3 têtes de boeufs

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 1

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 29, 54, 67, 68

Saisissez votre choix : 54

<clearScreen>

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 1

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 4, 7, 27, 91

Saisissez votre choix : 4

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5)

- série n° 2 : 1

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 3, 36, 98, 104

Saisissez votre choix : 3

<clearScreen>

Les cartes 3 (Théo), 4 (Oliwer) et 54 (Ethan) ont été posées.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 1, 3, 4

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

Aucun joueur ne ramasse de tête de boeufs.

A Théo de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 1, 3, 4

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 36, 98, 104

Saisissez votre choix : 36

<clearScreen>

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 1, 3, 4

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 7, 27, 91

Saisissez votre choix : 7

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 1, 3, 4

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 29, 67, 68

Saisissez votre choix : 29

<clearScreen>

Les cartes 7 (Oliwer), 29 (Ethan) et 36 (Théo) ont été posées.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 36

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

Théo a ramassé 5 têtes de boeufs

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 36

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 27, 91

Saisissez votre choix : 27

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 36

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 67, 68

Saisissez votre choix : 67

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 36

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 98, 104

Saisissez votre choix : 98

<clearScreen>

Les cartes 27 (Oliwer), 67 (Ethan) et 98 (Théo) vont être posées.

Pour poser la carte 27, Oliwer doit choisir la série qu'il va ramasser.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 36

- série n° 3 : 48, 58, 60 (3), 61, 64

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

Saisissez votre choix : 2

Les cartes 27 (Oliwer), 67 (Ethan) et 98 (Théo) ont été posées.

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 27

- série n° 3 : 67, 98

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

Oliwer a ramassé 1 tête de boeufs

Ethan a ramassé 7 têtes de boeufs

A Oliwer de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 27

- série n° 3 : 67, 98

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 91

Saisissez votre choix : 91

<clearScreen>

A Ethan de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 27

- série n° 3 : 67, 98

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 68

Saisissez votre choix : 68

<clearScreen>

A Théo de jouer.

<pause>

- série n° 1 : 20 (3), 25 (2), 31, 33 (5), 54

- série n° 2 : 27

- série n° 3 : 67, 98

- série n° 4 : 80 (3), 85 (2), 90 (3), 97, 99 (5)

- Vos cartes : 104

Saisissez votre choix : 104

<clearScreen>

Les cartes 68 (Ethan), 91 (Oliwer) et 104 (Théo) ont été posées.

- série n° 1 : 68, 91

- série n° 2 : 27

- série n° 3 : 67, 98

- série n° 4 : 104

Ethan a ramassé 12 têtes de boeufs

Théo a ramassé 14 têtes de boeufs

\*\* Score final

Oliwer a ramassé 6 tête de boeufs

Théo a ramassé 19 tête de boeufs

Ethan a ramassé 22 tête de boeufs

# 

# **Code**

**Console.java**

package appli;  
  
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() {  
 }  
}

**Carte.java**

package Cartes;  
  
public class Carte {  
 private final int numero;  
 private final int teteDeBoeuf;  
 private static int compteurCartes = 1;  
  
 /\*========== CONSTRUCTORS ==========\*/  
 */\*\*  
 \** ***@brief*** *Constructeur qui initialise le numero et le nombre de tete de boeuf automatiquement  
 \*/* public Carte() {  
 // numero auto et unique  
 this(compteurCartes);  
 ++*compteurCartes*;  
 }  
  
 */\*\*  
 \** ***@brief*** *Constructeur qui initialise le numero choisi et le nombre de tete de boeuf  
 \** ***@param*** *numero : Int, numero de la carte  
 \*/* public Carte(int numero) {  
 assert (0 <= numero && numero <= 104);  
 this.numero = numero;  
 // calcule du nombre de tete de boeuf  
 int x = 0;  
 if (numero % 11 == 0) x += 5;  
 if (numero % 10 == 5) x += 2;  
 else if (numero % 10 == 0) x += 3;  
 if (x == 0) x = 1;  
 teteDeBoeuf = x;  
 }  
  
 /\*========== GETTERS ==========\*/  
 */\*\*  
 \** ***@brief*** *Donne le numero de la carte  
 \** ***@return*** *Int, Le numero de la carte  
 \*/* public int getNumero() { return numero; }  
  
 */\*\*  
 \** ***@brief*** *Donne le nombre de tete de boeuf de la carte  
 \** ***@return*** *Int, Le nombre de tete de boeuf de le carte  
 \*/* public int getTeteDeBoeuf() { return teteDeBoeuf; }  
  
 /\* ========== METHODS ==========\*/  
 */\*\*  
 \** ***@brief*** *Permet de copier une carte  
 \** ***@return*** *Cartes.Carte, La copie de la carte  
 \*/* public Carte clone() { return new Carte(numero); }  
   
 */\*\*  
 \** ***@brief*** *To string, "$numero" si = 1, sinon "$numero ($teteDeBoeuf)"  
 \** ***@return*** *String, La chaine de caractere qui represente la carte  
 \*/* public String toString() {  
 if (teteDeBoeuf == 1) return String.valueOf(numero);  
 return numero + " (" + teteDeBoeuf + ")";  
 }  
}

**Joueur.java**

package Joueurs;  
  
import Cartes.Carte;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Joueur {  
 private final String nom;  
 private int nbTeteBoeuf = 0;  
 private Carte carteChoisi;  
 private ArrayList<Carte> cartes;  
  
 */\*\*  
 \** ***@brief*** *Constructeur qui initialise le nom du joueur et ses attributs  
 \** ***@param*** *nom : String, Le nom du joeur  
 \*/* public Joueur(String nom) {  
 this.nom = nom;  
 carteChoisi = null;  
 cartes = new ArrayList<>();  
 }  
  
 /\* ========== GETTERS ========== \*/  
 */\*\*  
 \** ***@brief*** *Donne le nom du joueur  
 \** ***@return*** *String, Le nom du joueur  
 \*/* public String getNom() { return nom; }  
   
 */\*\*  
 \** ***@brief*** *Donne le nombre de tête de boeuf du joueur  
 \** ***@return*** *int, Le nombre de tête de boeuf  
 \*/* public int getTeteBoeuf() { return nbTeteBoeuf; }  
  
 */\*\*  
 \** ***@brief*** *Donne la carte choisi par le joueur  
 \** ***@return*** *Carte, la carte choisi par le joueur  
 \*/* public Carte getCarteChoisi() { return carteChoisi; }  
  
 */\*\*  
 \** ***@brief*** *Donne les cartes du joeur  
 \** ***@return*** *ArrayList*<*Carte*>*, Les cartes du joueur  
 \*/* public ArrayList<Carte> getCartes() { return cartes; }  
  
 */\*\*  
 \** ***@brief*** *Definie la carte choisi par le joueur  
 \** ***@param*** *carte Carte, la carte  
 \*/* public void setCarteChoisi(Carte carte) { carteChoisi = carte; }  
  
 /\* ========== METHODS ==========\*/  
 */\*\*  
 \** ***@brief*** *Ajoute un nombre de tête de boeuf au joueur  
 \** ***@param*** *nbAdd : int, le nombre de tête de boeuf à ajouter  
 \*/* public void ajouterTeteBoeuf(int nbAdd) {  
 nbTeteBoeuf += nbAdd;  
 }  
   
 */\*\*  
 \** ***@brief*** *Regarde si les cartes du joueur sont vides  
 \** ***@return*** *boolean, Si les cartes du joueur sont vides  
 \*/* public boolean estVide() {  
 return !(cartes.size() > 0);  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche les cartes du joueur  
 \*/* public void afficherCartes() {  
 System.out.print("- Vos cartes : ");  
 for (int i = 0; i < cartes.size()-1; ++i) {  
 System.out.print(cartes.get(i) + ", ");  
 }  
 System.out.println(cartes.get(cartes.size()-1));  
 }  
   
 */\*\*  
 \** ***@brief*** *Permet au joueur de choisir une carte qu'il poss�de selon le num�ro  
 \** ***@return*** *Carte, La carte choisi par le joueur  
 \*/* public void choisirCarte() {  
 assert (!estVide());  
 Scanner sc = new Scanner(System.in);  
 Carte carte;  
 boolean carteEstChoisi = false;  
 System.out.print("Saisissez votre choix : ");  
 do {  
 // Le joueur rentre un numero  
 if (sc.hasNextInt()) {  
 int n = sc.nextInt();  
 // On vérifie si le numero correspond a une carte que le joueur possede  
 for (int i = 0; i < cartes.size(); i++) {  
 // C'est bon  
 if (cartes.get(i).getNumero() == n) {  
 carte = cartes.get(i);  
 cartes.remove(i);  
 carteChoisi = carte;  
 carteEstChoisi = true;  
 break;  
 }  
 // Le numero de correspond a aucune carte  
 if (i == cartes.size() - 1)  
 System.out.print("Vous n'avez pas cette carte, saisissez votre choix : ");  
 }  
 }  
 // Le joueur ne rentre pas un nombre  
 else {  
 System.out.print("Vous n'avez pas cette carte, saisissez votre choix : ");  
 sc.nextLine();  
 }  
 } while (!carteEstChoisi);  
 }  
  
 */\*\*  
 \** ***@brief*** *Enleve une carte au joueur  
 \** ***@param*** *carte : Carte, la carte a enlever  
 \*/* public void enleverCarte(Carte carte) {  
 // On veut enlever une carte selon son numero de carte, or le numero d'une carte est de type int  
 // Pour pas que ce soit considérer comme un indice mais bien un élément à enlever  
 // on passe par l'objet Integer  
 cartes.remove(Integer.valueOf(carte.getNumero()));  
 }  
  
 */\*\*  
 \** ***@brief*** *Enleve la carte choisi par le joueur  
 \*/* public void enleverCarteChoisi() {  
 enleverCarte(carteChoisi);  
 }  
  
 */\*\*  
 \** ***@brief*** *Clone le joueur  
 \** ***@return*** *Joueurs.Joueur, Un clone du joueur  
 \*/* public Joueur clone() {  
 // on recopie le nom et les tetes de boeuf du joueur  
 Joueur j = new Joueur(nom);  
 j.nbTeteBoeuf = nbTeteBoeuf;  
 // on recopie le deck  
 for (int i = 0; i < cartes.size(); ++i) j.cartes.add(cartes.get(i).clone());  
 return j;  
 }  
  
 */\*\*  
 \** ***@brief*** *to String, nom  
 \** ***@return*** *String, Le nom du joueur  
 \*/* public String toString() { return nom; }  
  
}

**Paquet.java**

package Cartes;  
  
import java.util.ArrayList;  
import java.util.Collections;  
  
public class Paquet {  
 private ArrayList<Carte> cartes;  
 private static final int *MAX\_CARTES* = 104;  
  
 */\*\*  
 \** ***@brief*** *Constructeur qui initialise automatiquement les 104 cartes du paquets  
 \*/* public Paquet() {  
 cartes = new ArrayList<>();  
 for (int i = 0; i < *MAX\_CARTES*; ++i) cartes.add(new Carte());  
 }  
  
 /\* ========== GETTERS ==========\*/  
 */\*\*  
 \** ***@brief*** *Donne le nombre de carte dans le paquet  
 \** ***@return*** *int, Le nombre de carte dans le paquet  
 \*/* public int getNombreCartes() { return cartes.size(); }  
  
 */\*\*  
 \** ***@brief*** *Donne la derniere carte du paquet  
 \** ***@return*** *Carte, La derniere carte du paquet  
 \*/* public Carte getDerniereCarte() {  
 assert !(estVide());  
 return cartes.get(getNombreCartes()-1);  
 }  
  
 */\*\*  
 \** ***@brief*** *Donne si le paquet est vide ou non  
 \** ***@return*** *boolean, Si le paquet est vide  
 \*/* public boolean estVide() { return !(cartes.size() > 0); }  
  
  
 /\* ========== METHODS ==========\*/  
 */\*\*  
 \** ***@brief*** *Melange le paquet de carte  
 \*/* public void melanger() {  
 Collections.shuffle(cartes);  
 }  
  
 */\*\*  
 \** ***@brief*** *Enleve la derniere carte du paquet et la renvoie  
 \** ***@return*** *Carte, La dernier carte du paquet  
 \*/* public Carte enlever() {  
 Carte c = getDerniereCarte();  
 cartes.remove(cartes.size()-1);  
 return c;  
 }  
}

**Serie.java**

package Cartes;  
  
import Cartes.Carte;  
  
import java.util.ArrayList;  
  
public class Serie {  
 private ArrayList<Carte> serie;  
 private static final int NB\_CARTES\_MAX = 5;  
  
 */\*\*  
 \** ***@brief*** *Constructeur  
 \*/* public Serie() {  
 serie = new ArrayList<>();  
 }  
  
 /\*========== GETTERS ==========\*/  
 */\*\*  
 \** ***@brief*** *Donne la liste de carte  
 \** ***@return*** *ArrayList*<*Carte*>*, La liste de carte  
 \*/* public ArrayList<Carte> getSerie() { return serie; }  
  
 */\*\*  
 \** ***@brief*** *Donne la dernière carte d'une serie  
 \** ***@return*** *Carte, La derniere carte  
 \*/* public Carte getDerniereCarte() { return serie.get(serie.size()-1); }  
  
 */\*\*  
 \** ***@brief*** *Donne le nombre de carte max que peut contenir la serie  
 \** ***@return*** *int, le nombre de carte max  
 \*/* public int getNbCartesMax() { return NB\_CARTES\_MAX; }  
  
 */\*\*  
 \** ***@brief*** *Dis si la série est pleine ou s'il reste de la place pour placer une carte  
 \** ***@return*** *boolean, si la séries est pleine ou non  
 \*/* public boolean estPlein() { return serie.size() >= NB\_CARTES\_MAX; }  
  
 /\*========== METHODS ==========\*/  
 */\*\*  
 \** ***@brief*** *Ajoute une carte dans une serie  
 \** ***@param*** *carte : Carte, La carte a ajouter  
 \*/* public void ajouterCarte(Carte carte) {  
 if (!estPlein())  
 serie.add(carte);  
 }  
  
 */\*\*  
 \** ***@brief*** *Vide la serie et la renvoie  
 \** ***@return*** *ArrayList*<*Carte*>*, La serie  
 \*/* public ArrayList<Carte> vider() {  
 ArrayList<Carte> cartes = new ArrayList<>(serie);  
 serie.clear();  
 return cartes;  
 }  
  
 */\*\*  
 \** ***@brief*** *to String, les cartes de la serie  
 \** ***@return*** *La chaine de caractere qui represente les cartes de la serie  
 \*/* public String toString() {  
 StringBuilder s = new StringBuilder();  
 for (int i = 0; i < serie.size()-1; ++i) {  
 s.append(serie.get(i).toString()).append(", ");  
 }  
 s.append(getDerniereCarte().toString());  
 return s.toString();  
 }  
  
}

**Jeu.java**

package Jeu;  
  
import Cartes.Carte;  
import Cartes.Paquet;  
import Cartes.Serie;  
import Joueurs.Joueur;  
import appli.Console;  
  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.util.\*;  
  
public class Jeu {  
 private Paquet paquet;  
 private Serie[] series;  
 private ArrayList<Joueur> joueurs;  
 private boolean enCours;  
 private final int NB\_SERIES = 4;  
  
 */\*\*  
 \** ***@brief*** *Constructeur qui initialise le paquet, les series et les joueurs  
 \*/* public Jeu() {  
 enCours = true;  
 // Initialise le paquet  
 paquet = new Paquet();  
 paquet.melanger();  
   
 // Initialise les joueurs  
 joueurs = new ArrayList<>();  
 String nomFichier = "src/Joueurs/config.txt";  
 try {  
 Scanner in = new Scanner(new FileInputStream(nomFichier));  
 while (in.hasNextLine()) {  
 joueurs.add(new Joueur(in.nextLine().trim()));  
 }  
 } catch (FileNotFoundException e) {  
 System.*out*.println("Le fichier \"" + nomFichier + "\" est introuvable.");  
 }  
 assert (2 <= joueurs.size() && joueurs.size() <= 10) : "Il y a " + joueurs.size() + " joueurs, le jeu se joue entre 2 et 10 joueurs";  
  
 // Distribution des cartes aux joueurs  
 for (Joueur joueur: joueurs) {  
 for (int i = 0; i < 10; ++i) {  
 joueur.getCartes().add(paquet.enlever());  
 }  
 // Tri croissant des cartes des joueurs  
 Collections.*sort*(joueur.getCartes(), (c1, c2) -> c1.getNumero() - c2.getNumero());  
 }  
  
 // Initialisation des séries  
 series = new Serie[NB\_SERIES];  
 for (int i = 0; i < NB\_SERIES; ++i) {  
 series[i] = new Serie();  
 series[i].ajouterCarte(paquet.enlever());  
 }  
 }  
  
 /\* ========== GETTERS ========== \*/  
  
 */\*\*  
 \** ***@brief*** *Donne le nombre de joeur  
 \** ***@return*** *int, Le nombre de joueur  
 \*/* public int getNombreJoueurs() { return joueurs.size(); }  
  
 */\*\*  
 \* Donne un joueur selon son indice dans la liste  
 \** ***@param*** *index : Int, Indice du joueur  
 \** ***@return*** *Joueur, Le joueur  
 \*/* public Joueur getJoueur(int index) {  
 assert (0 <= index && index < getNombreJoueurs());  
 return joueurs.get(index);  
 }  
  
  
 /\* ========== METHODS ========== \*/  
 */\*\*  
 \** ***@brief*** *Trie les joueurs dans l'ordre croissant des cartes choisi par les joueurs  
 \*/* public void triJoueurs() {  
 // Comparator = fonction lambda  
 Collections.*sort*(joueurs, (j1, j2) -> j1.getCarteChoisi().getNumero() - j2.getCarteChoisi().getNumero());  
 }  
  
 */\*\*  
 \** ***@brief*** *Vérifie si le jeu est terminer et change l'état du jeu en cours si c'est la fin  
 \*/* public void verifJeuEnCours() {  
 for (int i = 0; i < getNombreJoueurs(); i++) {  
 // Si il y a encore des cartes à jouer on continue de jouer  
 if (!getJoueur(i).estVide()) { break; }  
 // Sinon le jeu est finie  
 if (i >= getNombreJoueurs()-1) { enCours = false; }  
 }  
 }  
  
  
 /\* ========== METEHODES AFFICHAGES ========== \*/  
 */\*\*  
 \** ***@brief*** *Affiche les séries  
 \*/* private void afficherSeries() {  
 for (int i = 0; i < NB\_SERIES; ++i) {  
 System.*out*.print("- série n° " + (i+1) +" : ");  
 System.*out*.println(series[i]);  
 }  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche le message de bienvenue au jeu  
 \*/* private void afficherMessageBienvenue() {  
 System.*out*.print("Les " + getNombreJoueurs() + " joueurs sont ");  
 for (int i = 0; i < getNombreJoueurs()-1; ++i) {  
 System.*out*.print(getJoueur(i));  
 if (i < getNombreJoueurs()-2)  
 System.*out*.print(", ");  
 }  
 System.*out*.println(" et " + getJoueur(getNombreJoueurs()-1) + ". Merci de jouer à 6 qui prend !");  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche le message des cartes posées par les joueurs  
 \*/* private void afficherCartesPosees() {  
 System.*out*.print("Les cartes ");  
 for (int i = 0; i < getNombreJoueurs()-1; i++) {  
 System.*out*.print(joueurs.get(i).getCarteChoisi().getNumero() + " (" + joueurs.get(i).getNom() + ")");  
 if (i < getNombreJoueurs()-2)  
 System.*out*.print(", ");  
 }  
 System.*out*.println(" et " + joueurs.get(joueurs.size()-1).getCarteChoisi().getNumero() + " (" + joueurs.get(joueurs.size()-1).getNom() + ") ont été posées.");  
 afficherSeries();  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche le message des têtes de boeuf récupérées par les joueurs  
 \** ***@param*** *compteurTDB : int[], le têtes de boeuf ramassées par les joueurs  
 \*/* private void afficherTDBRecuperees(int[] compteurTDB) {  
 boolean personneRamasseTDB = true;  
 for (int i = 0; i < compteurTDB.length; ++i) {  
 if (compteurTDB[i] > 0) {  
 if (compteurTDB[i] == 1)  
 System.*out*.println(getJoueur(i).getNom() + " a ramassé " + compteurTDB[i] + " tête de boeufs");  
 else  
 System.*out*.println(getJoueur(i).getNom() + " a ramassé " + compteurTDB[i] + " têtes de boeufs");  
 personneRamasseTDB = false;  
 }  
 }  
 if (personneRamasseTDB) {  
 System.*out*.println("Aucun joueur ne ramasse de tête de boeufs.");  
 }  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche le message avant que le premier joueur doit choisir une série  
 \*/* private void afficherMessageChoisirSerie() {  
 // Affichage du message  
 System.*out*.print("Les cartes ");  
 for (int j = 0; j < getNombreJoueurs()-1; ++j) {  
 System.*out*.print(joueurs.get(j).getCarteChoisi().getNumero() + " (" + joueurs.get(j).getNom() + ")");  
 if (j < getNombreJoueurs()-2)  
 System.*out*.print(", ");  
 }  
 System.*out*.println(" et " + joueurs.get(joueurs.size()-1).getCarteChoisi().getNumero() + " (" + joueurs.get(joueurs.size()-1).getNom() + ") vont être posées.");  
 System.*out*.println("Pour poser la carte " + getJoueur(0).getCarteChoisi().getNumero() + ", " + getJoueur(0).getNom() + " doit choisir la série qu'il va ramasser.");  
 afficherSeries();  
 }  
  
 */\*\*  
 \** ***@brief*** *Affiche le score final dans l'ordre croissant des têtes de boeuf des joueurs  
 \*/* private void afficherScore() {  
 System.*out*.println("\*\* Score final");  
 // Trier joueurs selon les têtes de boeuf décroissant  
 Collections.*sort*(joueurs, (j1, j2) -> j1.getTeteBoeuf() - j2.getTeteBoeuf());  
 for (Joueur j: joueurs) {  
 if (j.getTeteBoeuf() > 1)  
 System.*out*.println(j.getNom() + " a ramassé " + j.getTeteBoeuf() + " tête de boeufs");  
 else  
 System.*out*.println(j.getNom() + " a ramassé " + j.getTeteBoeuf() + " têtes de boeufs");  
 }  
 }  
  
  
 /\* ========== METHODS JOUEURS ========== \*/  
 */\*\*  
 \** ***@brief*** *Détermine la série où le joueur doit poser sa carte choisi  
 \** ***@param*** *idxJoueur : int, l'indice du joueur  
 \** ***@return*** *le numero de la serie (-1 si aucune serie est possible)  
 \*/* public int determineSerie(int idxJoueur) {  
 int numSerieDetermine = -1;  
 int ecartMin = 104;  
 int ecart;  
 for (int numSerie = 0; numSerie < series.length; ++numSerie) {  
 ecart = getJoueur(idxJoueur).getCarteChoisi().getNumero() - series[numSerie].getDerniereCarte().getNumero();  
 if (0 < ecart && ecart < ecartMin) {  
 ecartMin = ecart;  
 numSerieDetermine = numSerie;  
 }  
 }  
 return numSerieDetermine;  
 }  
  
 */\*\*  
 \** ***@brief*** *Fais jouer un joueur pour choisir sa carte et affiche les messages  
 \** ***@param*** *j : Joueur, le joueur qui doit jouer son tour  
 \*/* public void joueurChoisiCarte(Joueur j) {  
 System.*out*.println("A " + j.getNom() + " de jouer.");  
 Console.*pause*();  
 afficherSeries();  
 j.afficherCartes();  
 j.choisirCarte();  
 Console.*clearScreen*();  
 }  
  
 */\*\*  
 \** ***@brief*** *Demande de choisir une série à un joueur (entre 1 et 4)  
 \** ***@return*** *int, renvoie la numéro de la série (entre 0 et 3)  
 \*/* public int joueurChoisiSerie() {  
 Scanner sc = new Scanner(System.*in*);  
 boolean numSerieEstChoisi = false;  
 int numSerieChoisi = -1;  
  
 System.*out*.print("Saisissez votre choix : ");  
 do {  
 if (sc.hasNextInt()) {  
 numSerieChoisi = sc.nextInt();  
 if (1 <= numSerieChoisi && numSerieChoisi <= 4) {  
 numSerieEstChoisi = true;  
 }  
 else {  
 System.*out*.print("Ce n'est pas une série valide, saisissez votre choix : ");  
 }  
 }  
 else {  
 System.*out*.print("Ce n'est pas une série valide, saisissez votre choix : ");  
 sc.nextLine();  
 }  
 } while (!numSerieEstChoisi);  
  
 return --numSerieChoisi;  
 }  
  
 */\*\*  
 \** ***@brief*** *Pose la carte du joueur dans la série  
 \** ***@param*** *idxJoueur int, indice du joueur  
 \** ***@param*** *numSerie int, numéro de la série  
 \*/* public void joueurPoseCarte(int idxJoueur, int numSerie) {  
 series[numSerie].ajouterCarte(getJoueur(idxJoueur).getCarteChoisi());  
 getJoueur(idxJoueur).enleverCarteChoisi();  
 }  
  
 */\*\*  
 \** ***@brief*** *Un joueur récupère lees têtes de boeuf d'une série, elle se vide et il pose sa carte  
 \** ***@param*** *idxJoueur int, indice du joueur  
 \** ***@param*** *numSerie int, numéro de la série  
 \** ***@param*** *compteurTDB int[], compteur des têtes de boeuf du tour  
 \*/* public void joueurRecupereSerie(int idxJoueur, int numSerie, int[] compteurTDB) {  
 // On ajoute les têtes de boeuf  
 for (Carte carte: series[numSerie].getSerie()) {  
 compteurTDB[idxJoueur] += carte.getTeteDeBoeuf();  
 }  
 getJoueur(idxJoueur).ajouterTeteBoeuf(compteurTDB[idxJoueur]);  
 // On vide la série et pose la carte du joueur  
 series[numSerie].vider();  
 joueurPoseCarte(idxJoueur, numSerie);  
 }  
  
 */\*\*  
 \** ***@brief*** *Pose la carte du joueur dans la série déterminer et si elle est pleine, il la récupère  
 \** ***@param*** *idxJoueur : int, l'indice du joueur  
 \** ***@param*** *compteurTDB : int, compteur des têtes de boeuf du tour  
 \*/* public void joueurJoueCarte(int idxJoueur, int[] compteurTDB) {  
 int numSerieDetermine = determineSerie(idxJoueur);  
 // Si la serie est pleine  
 if (series[numSerieDetermine].estPlein()) {  
 joueurRecupereSerie(idxJoueur, numSerieDetermine, compteurTDB);  
 }  
 // Si la serie n'est pas pleine  
 else {  
 joueurPoseCarte(idxJoueur, numSerieDetermine);  
 }  
 }  
  
  
 /\* ========== JOUER ========== \*/  
 */\*\*  
 \** ***@brief*** *Pour jouer au jeu du 6 qui prend !  
 \*/* public void jouer() {  
 // Message de bienvenue  
 afficherMessageBienvenue();  
  
 // Jouer tant que le jeu est en cours  
 do {  
 // Compteur des TDB récupéré par les joueurs lors de ce tour  
 int[] compteurTDB = new int[getNombreJoueurs()];  
 for (int i = 0; i < compteurTDB.length; ++i) compteurTDB[i] = 0;  
  
 // Chaque joueur choisi sa carte  
 for (int i = 0; i < getNombreJoueurs(); ++i) {  
 joueurChoisiCarte(getJoueur(i));  
 }  
  
 // On trie l'ordre des joueurs  
 triJoueurs();  
  
 // On place les cartes dans les séries  
 // Le premier joueur est le seul joueur qui peut avoir choisi une carte trop petite pour la poser quelque part  
 // Si c'est le cas, il choisit la série qu'il va récupérer  
 if (determineSerie(0) == -1) {  
 afficherMessageChoisirSerie();  
 joueurRecupereSerie(0, joueurChoisiSerie(), compteurTDB);  
 }  
 // Sinon on joue sa carte normalement  
 else {  
 joueurJoueCarte(0, compteurTDB);  
 }  
  
 // On joue les cartes des autres joueurs  
 for (int j = 1; j < getNombreJoueurs(); ++j) {  
 joueurJoueCarte(j, compteurTDB);  
 }  
  
 // Affichage du message des cartes posées  
 afficherCartesPosees();  
  
 // Affichage des TDB ramasse  
 afficherTDBRecuperees(compteurTDB);  
  
 // On vérifie si le jeu est terminer  
 verifJeuEnCours();  
  
 } while (enCours);  
  
 // FIN DU JEU : affichage du score  
 afficherScore();  
 }  
}

**Application.java**

*/\*\*  
 \** ***@file*** *Appli.Appli.java  
 \** ***@author*** *Théo Constant 108 et Ethan Brossard 110  
 \** ***@brief*** *\* Ce fichier Appli.Appli.java est le fichier qui contient la fonction main().  
 \* C'est ici que le jeu va se lancer.  
 \*/*package appli;  
  
import Jeu.Jeu;  
  
public class Application {  
 public static void main(String[] args) {  
 Jeu jeu = new Jeu();  
 jeu.jouer();  
 }  
}

# **Bilan**

Difficultés

Faire les tests a été compliqué, car il y a beaucoup de fonctions à tester, et qu’elles ne sont pas toutes évidentes à tester. De plus, il y a des fonctions qui n’est pas possible de tester comme les fonctions d’affichage ou certaines fonctions spécifiques, notamment dans la classe Jeu.

L’affichage a aussi été un léger problème, car il fallait faire attention aux détails (à la virgule près) et tout vérifier a pris du temps.

Améliorations possibles

Nous savons qu’il y a des nombres imaginaires dans notre code (comme dans les tests par exemple), cela n’est pas une bonne pratique et c’est quelque chose qui pourrait être améliorer sans trop de difficulté.

Rajouter des getters et des setters pour pouvoir tester plus de fonctionnalités. Cela n’est cependant pas important, car notre jeu marche sans ces getters et setters.

Choix

Pour trier les listes, comme la liste de cartes d’un joueur, ou l’ordre des joueurs selon le numéro de leur carte lors d’un tour, on a utilisé la bibliothèque Collection pour utiliser la méthode sort avec un comparateur qui est une fonction lambda.

Nous avons décidé de faire des fonctions qui ne sert qu’à afficher des messages comme le message de bienvenue au début du jeu, car ça nous parait plus simple, plus lisible, mais c’est aussi notre manière de programmer.