IUT de Paris - Rives de Seine
Université Paris Cité

Rapport de Projet

MONTCULIER Alexis

Table des matières

Introduction

Règles du 6 qui prend

Le 6 qui prend est un jeu assez méconnu mais qui nous a été très intéressant de découvrir et de comprendre afin d'optimiser au mieux notre code, il se joue de 2 à 10 joueurs et se déroulera toujours en 10 tours avec un jeu de 104 cartes, numérotés de 1 à 104 et possédant chacune un nombre de tête de bœuf, qui change en fonction de son nombre. Chaque joueur possède 10 cartes dans sa main, il doit poser sur une des 4 séries sur la table, une carte de sa main avec un nombre supérieur à celui au bout d'une de ces 4 séries. Si aucune de ses cartes ne l'est, il devra alors ramasser une série qu'il choisit et mettra ces cartes de côté. Le joueur possédant le plus de tête de Bœuf à la fin du jeu perd la partie.

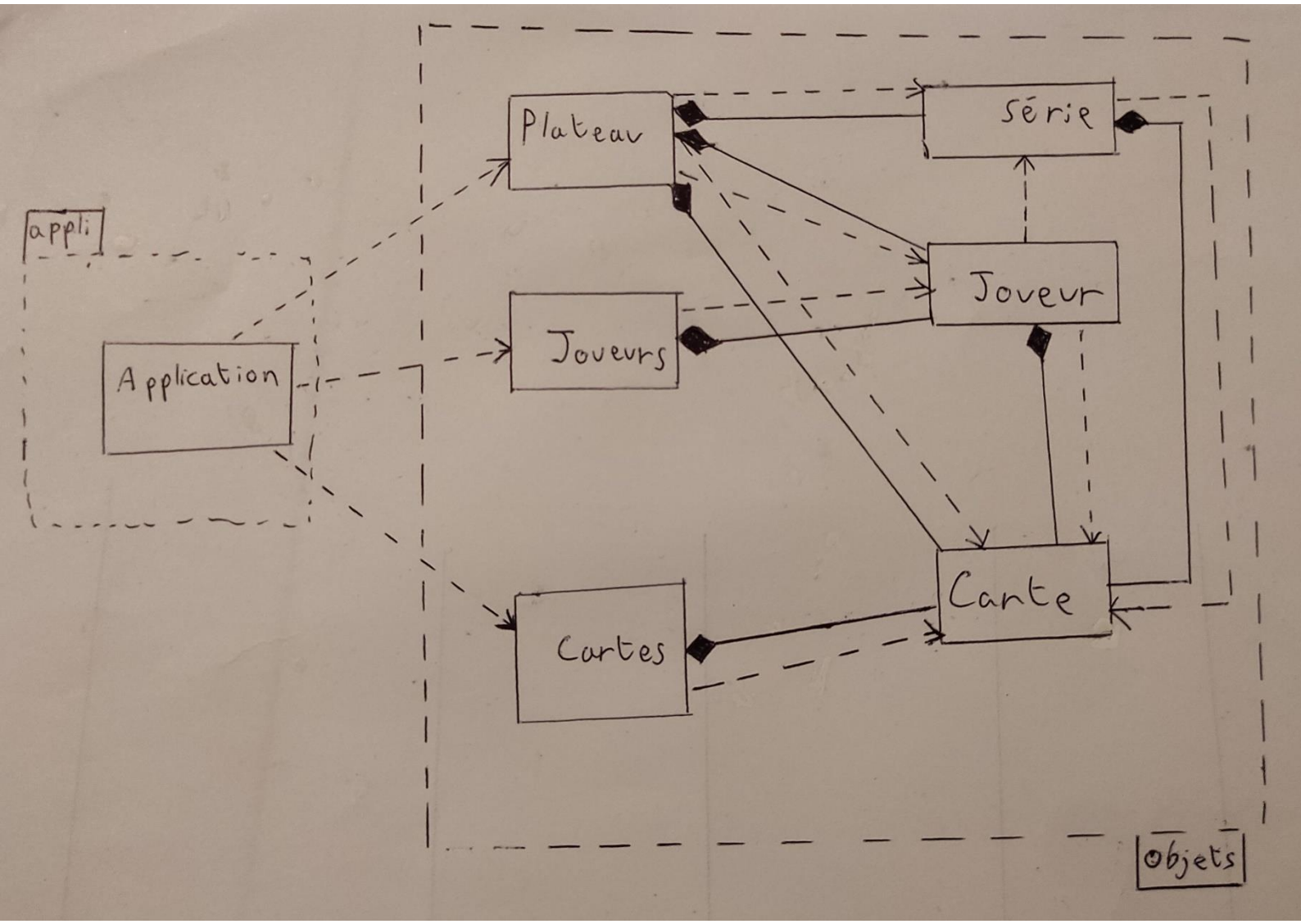
Objectifs

Les objectifs de ce projet sont multiples. Il représente tout d'abord un réel défi, celui de créer en Java, un langage que nous avons découvert il y a 7 semaines, notre propre version du jeu du « 6 qui prend », en utilisant la méthode de développement Objet. Notre projet ne comprend pas d'interfaces graphiques mais est composé d'une suite d'exécution de commandes afin de suivre le bon déroulement du jeu.

Outils de développement

Pour réaliser ce projet nous avons utilisé deux IDE distincts. Axel a choisi Eclipse et Alexis a préféré IntelliJ. Ces deux logiciels présentent leurs avantages et leurs inconvénients. Nous avons aussi utilisé GitHub afin d'avancer chacun de notre côté sur ce projet, de pouvoir récupérer la dernière version de ce dernier et de conserver toutes les traces de notre progression. Le seul inconvénient de GitHub est l'impossibilité de nous laisser travailler simultanément sur le projet, cette contrainte fut cependant très marginale.

Diagramme UML



Tests Unitaires

```java
package Tests;

import static org.junit.Assert.assertEquals;
import org.junit.Test;
import objets.*;


public class TestCartes {


	public void testTetes() {

		Carte carte1 = new Carte(1);
		Carte carte2 = new Carte(22);
		Carte carte3 = new Carte(10);
		Carte carte4 = new Carte(15);
		Carte carte5 = new Carte(55);

		assertEquals(1, carte1.getTete());
		assertEquals(5, carte2.getTete());
		assertEquals(3, carte3.getTete());
		assertEquals(2, carte4.getTete());
		assertEquals(7, carte5.getTete());
	}

	public void testNombres() {

		Carte carte1 = new Carte(1);
		Carte carte2 = new Carte(22);
		Carte carte3 = new Carte(10);
		Carte carte4 = new Carte(15);
		Carte carte5 = new Carte(55);

		assertEquals(1, carte1.getNum());
		assertEquals(22, carte2.getNum());
		assertEquals(10, carte3.getNum());
		assertEquals(15, carte4.getNum());
		assertEquals(55, carte5.getNum());
	}
}
```

```java
package Tests;

import static org.junit.Assert.assertEquals;
import org.junit.Test;
import objets.*;

public class TestJoueurs {


	public void testCarteRestante() {

		Joueur joueur = new Joueur("Joueur Test");
		Carte carte1 = new Carte(84);
		Carte carte2 = new Carte(72);

		joueur.prendreCarte(carte1);
		joueur.prendreCarte(carte2);

		assertEquals(2, joueur.carteRestantes());
	}

	public void testNom() {

		Joueur joueur = new Joueur("Joueur Test");

		assertEquals("Joueur Test", joueur.getNom());
	}

	public void testCarteJoueur() {

		Joueur joueur = new Joueur("Joueur Test");
		Carte carte1 = new Carte(84);
		Carte carte2 = new Carte(72);
		joueur.prendreCarte(carte1);
		joueur.prendreCarte(carte2);

		assertEquals(84, joueur.getCarte(0));
		assertEquals(72, joueur.getCarte(1));
	}

	public void testTetesJoueur() {

		Joueur joueur = new Joueur("Joueur Test");
		Carte carte1 = new Carte(84);
		Carte carte2 = new Carte(72);
		joueur.prendreCarte(carte1);
		joueur.prendreCarte(carte2);

		assertEquals(2, joueur.getNbTetes());
	}
}
```

```java
package Tests;

import static org.junit.Assert.assertEquals;
import org.junit.Test;
import objets.*;

public class TestSerie {


	public void testCarteSerie() {
		Carte carte1 = new Carte(15);
		Carte carte2 = new Carte(84);

		Serie serie = new Serie();
		serie.poser(carte1);
		serie.poser(carte2);

		assertEquals(carte1, serie.getCarte(0));
		assertEquals(carte2, serie.getCarte(1));
	}



	public void testDerniereCarte() {
		Carte carte1 = new Carte(15);
		Carte carte2 = new Carte(84);

		Serie serie = new Serie();
		serie.poser(carte1);
		serie.poser(carte2);


		assertEquals(carte2, serie.getLast());
	}

	public void testNbCartes() {
		Carte carte1 = new Carte(15);
		Carte carte2 = new Carte(84);

		Serie serie = new Serie();
		serie.poser(carte1);
		serie.poser(carte2);

		assertEquals(2, serie.getNbCartes());

	}

}
```

Code Java du Projet

```
package appli;
import objets.*;
import util.Console;
import java.util.Scanner;

public class Application {
    public static Scanner scan = new Scanner(System.in);

    //Fonction principale du jeu
    public static void main(String[] args) {
        Joueurs listeJ = new Joueurs();
        Cartes deck = new Cartes();
        Plateau table = new Plateau();

        //Distribution cartes
        for (int i = 0; i < listeJ.getNbJoueurs(); ++i)
            for (int j = 0; j < deck.INIT_CARTES; ++j)
                listeJ.getJoueur(i).prendreCarte(deck.prendreCarte());
        //Pose des cartes sur le plateau
        for (int i = 0; i < table.NB_SERIES; ++i)
            table.getSerie(i).poser(deck.prendreCarte());

        //Affichage initial
        System.out.print("Les " + listeJ.getNbJoueurs() + " joueurs sont ");
        for (int i = 0; i < listeJ.getNbJoueurs(); ++i) {
            System.out.print(listeJ.getJoueur(i).getNom());
            if (i == listeJ.getNbJoueurs() - 1)
                System.out.print(". Merci de jouer à 6 qui prend !\n");
            else
                if (i == listeJ.getNbJoueurs() - 2)
                    System.out.print(" et ");
                else
                    System.out.print(", ");
        }

        //Début du jeu
        for (int c = 0; c < deck.INIT_CARTES; ++c) {
            for (int i = 0; i < listeJ.getNbJoueurs(); ++i) {
                System.out.println("A " + listeJ.getJoueur(i).getNom() + " de
jouer.");
                Console.pause();
                System.out.println(table);
                System.out.println(listeJ.getJoueur(i));
```

```
                table.poseCarte(listeJ.getJoueur(i),
listeJ.getJoueur(i).choixCarte());
                Console.clearScreen();
            }
            table.placerCartes();
        }
        System.out.println("** Score final");
        for (int i = 0; i < listeJ.getNbJoueurs(); ++i) {
            System.out.print(listeJ.getJoueur(i).getNom() + " a ramassé " +
listeJ.getJoueur(i).getNbTetes());
            if(listeJ.getJoueur(i).getNbTetes() > 1)
                System.out.println(" têtes de boeufs");
            else
                System.out.println(" tête de boeufs");
        }
        scan.close();
    }
}
```

```java
package objets;

public class Carte {
    private int num;
    private int nbTete = 1;

    //Initialisation de la valeur et calcul du nombre de tête de boeuf
    public Carte(int numCarte) {
        this.num = numCarte;

        //Définition nombre de têtes de boeuf
        if (numCarte % 10 == 0)
            this.nbTete += 2;
        if (numCarte % 10 == 5)
            this.nbTete++;
        if (numCarte % 11 == 0) {
            this.nbTete += 4;
            if (numCarte == 55)
                this.nbTete++;
        }
    }

    //Getteurs
    public int getNum() {
        return this.num;
    }
    public int getTete() {
        return this.nbTete;
    }

    //Fonction toString
    public String toString() {
        String s = Integer.toString(getNum());
        if (this.getTete() > 1)
            s = s + " (" + this.getTete() + ")";
        return s;
    }
}
```

```java
package objets;
import java.util.ArrayList;
import java.util.Collections;

public class Cartes {
    public final int NB_CARTES = 104;
    public final int INIT_CARTES = 10;
```

```
    //Création d'une ArrayList représentant le paquet de cartes
    private ArrayList<Carte> paquet = new ArrayList<Carte>();

    //Initialisation et mélange du paquet de cartes
    public Cartes() {
        for (int i = 1; i <= NB_CARTES; ++i) {
            Carte c = new Carte(i);
            paquet.add(c);
        }
        Collections.shuffle(paquet);
    }

    //Fonction permettant de réinitialiser le paquet de cartes
    public void reinitialiser() {
        paquet.clear();
        for (int i = 1; i <= NB_CARTES; ++i) {
            Carte c = new Carte(i);
            paquet.add(c);
        }
        Collections.shuffle(paquet);
    }
    //Fonction retirant et retournant la carte au sommet du paquet
    public Carte prendreCarte() {
        assert(!paquet.isEmpty());
        Carte aux = paquet.get(0);
        paquet.remove(0);
        return aux;
    }

    //Getteur
    public int getTaille() {
        return paquet.size();
    }

    //Fonction toString
    public String toString() {
        String s = "";
        for (Carte c: paquet)
            s = s + c.toString() + " ";
        return s;
    }
}
```

```java
package objets;
import java.util.ArrayList;
import appli.*;


public class Joueur {
    private String nom;
    private int teteBoeuf;
    private ArrayList<Carte> main = new ArrayList<Carte>();
    private int aRamasse;

    //Initialisation du nom du joueur
    public Joueur(String nomJoueur) {
        this.nom = nomJoueur;
        this.teteBoeuf = 0;
        this.aRamasse = 0;
    }

    //Fonction permettant d'ajouter une carte à la main du joueur
    public void prendreCarte(Carte c) {
        this.main.add(c);
    }
    //Fonction permettant au joueur de poser une carte
    public Carte poserCarte(Carte c) {
        assert(main.contains(c));
        this.main.remove(c);
        return c;
    }
    //Fonction permettant de faire choisir une carte à un joueur dans sa main
    public Carte choixCarte() {
        boolean cartePresente = false;
        Carte aux = this.getCarte(0);
        System.out.println("Saisissez votre choix : ");
        do {
            String choix = Application.scan.nextLine();
            for (int j = 0; j < this.carteRestantes(); ++j)
                if (this.getCarte(j).getNum() == Integer.parseInt(choix)) {
                    cartePresente = true;
                    aux = this.getCarte(j);
                }
            if (!cartePresente)
                System.out.print("\nVous n'avez pas cette carte, saisissez
votre choix : ");
        }while (!cartePresente);
        return aux;
    }
    //Fonction permettant à un joueur de ramasser une série donnée
    public void ramasseSerie(Serie s) {
        for (int i = 0; i < s.getNbCartes(); ++i)
```

```
            this.teteBoeuf += s.getCarte(i).getTete();
        s.reinitialiser();
    }

    //Setteur
    public void setTetesRamasse(int i) {
        this.aRamasse = i;
    }

    //Getteurs
    public int carteRestantes() {
        return main.size();
    }
    public String getNom() {
        return this.nom;
    }
    public Carte getCarte(int i) {
        return main.get(i);
    }
    public int getNbTetes() {
        return this.teteBoeuf;
    }
    public int getTetesRamasses() {
        return this.aRamasse;
    }

    //Fonction toString
    public String toString() {
        String s = "- Vos cartes : ";
        for (int i = 0; i < main.size(); ++i) {
            s = s + main.get(i).toString();
            if (i < main.size() - 1)
                s = s + ", ";
        }
        return s;
    }
}
```

```java
package objets;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

public class Joueurs {
    private int nbJoueurs;
    public static final int MAX_JOUEURS = 10;

    //Création d'une ArrayList représentant l'ensemble des joueurs
    private ArrayList<Joueur> listeJoueurs = new ArrayList<Joueur>();

  //Initialisation de la liste de joueurs à partir d'un fichier texte
    public Joueurs() {
        try {
            String fichier ="config.txt";
            InputStream input=new FileInputStream(fichier);
            InputStreamReader reader=new InputStreamReader(input);
            BufferedReader buffer=new BufferedReader(reader);
            String ligne;
            while ((ligne=buffer.readLine())!=null) {
                Joueur j = new Joueur(ligne);
                listeJoueurs.add(j);
                nbJoueurs = listeJoueurs.size();
            }
        buffer.close();
        }
        catch (Exception e){
        System.out.println(e.toString());
        }
    }

    //Getteurs
    public Joueur getJoueur(int i) {
        return listeJoueurs.get(i);
    }
    public int getNbJoueurs() {
        return nbJoueurs;
    }

    //Fonction toString
    public String toString() {
        String s = "Nombre de joueurs : "  + this.nbJoueurs + "\n";
        s = s + "Liste joueurs : \n";
        for (Joueur j : listeJoueurs)
            s = s + j.getNom() + System.lineSeparator();
        return s;
```

```
        }
    }
```

```java
package objets;
import java.util.ArrayList;
import java.util.HashMap;
import appli.*;

public class Plateau {
    public final int NB_SERIES = 4;

    //Création d'une ArrayList représentant l'ensemble des séries sur la table
    private ArrayList<Serie> listeS = new ArrayList<Serie>(NB_SERIES);
    //Création d'une HashMap représentant les cartes posées par chaque joueur
à chaque tour
    private HashMap<Joueur, Carte> carteJouees = new HashMap<Joueur, Carte>();

    //Initialisation d'un plateau avec 4 séries
    public Plateau() {
        for (int i = 0; i < NB_SERIES; ++i)
            listeS.add(new Serie());
    }

    //Fonction permettant de placer les cartes jouées par les joueurs sur les
séries
    public void placerCartes() {
        boolean placee;
        int serieChoisie;
        int ecart;
        Serie aux;

        //Affichage initial
        String str = "Les cartes ";
        ecart = 0;
        for (Joueur player: carteJouees.keySet()) {
            ecart++;
            if (ecart < carteJouees.size() - 1)
                str = str + carteJouees.get(player).getNum() + " (" +
player.getNom()+ "), ";
            else
                if(ecart == carteJouees.size() - 1)
                    str = str + carteJouees.get(player).getNum() + " (" +
player.getNom()+ ") et ";
                else
                    str = str + carteJouees.get(player).getNum() + " (" +
player.getNom() + ") ";
        }
        str = str + "vont être posées.";
        System.out.println(str);

        for (Joueur player: carteJouees.keySet()) {
            placee = false;
```

```
            aux = null;
            ecart = 104;
            player.setTetesRamasse(0);
            for (Serie s: listeS)
                if (carteJouees.get(player).getNum() - s.getLast().getNum() <
ecart && carteJouees.get(player).getNum() - s.getLast().getNum() > 0) {
                    aux = s;
                    ecart = carteJouees.get(player).getNum() -
s.getLast().getNum();
                }
            if (aux != null) {
                placee = true;
                if (aux.getNbCartes() == aux.MAX_SERIE) {
                    ecart = player.getNbTetes();
                    player.ramasseSerie(aux);
                    player.setTetesRamasse(player.getNbTetes() - ecart);
                    aux.poser(carteJouees.get(player));
                }else
                    aux.poser(carteJouees.get(player));
            }
            //Impossible de placer la carte, le joueur choisit une série à
ramasser
            if (!placee) {
                ecart = player.getNbTetes();
                System.out.println("Pour poser la carte " +
carteJouees.get(player).getNum() + ", " + player.getNom() + " doit choisir la
série qu'il va ramasser.");
                System.out.println(this.toString());
                System.out.println("Saisissez votre choix : ");
                do {
                    serieChoisie =
Integer.parseInt(Application.scan.nextLine());
                    if (serieChoisie > 4 || serieChoisie < 1)
                        System.out.print("\nCe n'est pas une série valide,
saisissez votre choix : ");
                }while(serieChoisie > 4 || serieChoisie < 1);
                player.ramasseSerie(this.getSerie(serieChoisie - 1));
                this.getSerie(serieChoisie -
1).poser(carteJouees.get(player));
                player.setTetesRamasse(player.getNbTetes() - ecart);
            }
        }
        str = "Les cartes ";
        ecart = 0;
        for (Joueur player: carteJouees.keySet()) {
            ecart++;
            if (ecart < carteJouees.size() - 1)
                str = str + carteJouees.get(player).getNum() + " (" +
player.getNom()+ "), ";
```

```
            else
                if(ecart == carteJouees.size() - 1)
                    str = str + carteJouees.get(player).getNum() + " (" +
player.getNom()+ ") et ";
                else
                    str = str + carteJouees.get(player).getNum() + " (" +
player.getNom() + ") ";
        }
        str = str + "ont été posées.";
        System.out.println(str);
        boolean tetesRamassees = false;
        for (Joueur player : carteJouees.keySet()) {
            if (player.getTetesRamasses() > 0) {
                tetesRamassees = true;
                System.out.print(player.getNom() + " a rammassé "+
(player.getTetesRamasses()));
                if (player.getTetesRamasses()> 1)
                    System.out.println(" têtes de boeufs.");
                else
                    System.out.println(" tête de boeufs.");
            }
        }
        if (!tetesRamassees)
            System.out.println("Aucun joueur ne ramasse de tête de boeufs.");
        carteJouees.clear();
    }
    //Fonction servant à réinitialiser la partie
    public void reinitialiser() {
        for (int i = 0; i < NB_SERIES; ++i)
            listeS.get(i).reinitialiser();
    }

    //Setteur
    public void poseCarte(Joueur j, Carte c) {
        this.carteJouees.put(j, c);
        j.poserCarte(c);
    }

    //Getteurs
    public Serie getSerie(int i) {
        return listeS.get(i);
    }
    public Carte getCarteJouee(Joueur j) {
        return carteJouees.get(j);
    }

    //Fonction toString
    public String toString() {
        String s = "";
```

```
        for (int i = 0; i < NB_SERIES; ++i) {
            s = s + listeS.get(i).toString();
            if (i < NB_SERIES - 1)
                s = s + "\n";
        }
        return s;
    }
}
```

```java
package objets;
import java.util.ArrayList;

public class Serie {
    private int indice;
    private static int compteur = 0;
    private int nbCartes;
    private ArrayList<Carte> ligne = new ArrayList<Carte>();
    public final int MAX_SERIE = 5;

    //Initialisation d'une série
    public Serie() {
        this.indice = compteur++;
        this.nbCartes = 0;
    }

    //Méthode permettant de poser une nouvelle carte sur une série
    public void poser(Carte c) {
        ligne.add(c);
        nbCartes = ligne.size();
    }
    //Méthode permettant de réinitialiser une série
    public void reinitialiser() {
        this.ligne.clear();
        this.nbCartes = 0;
    }

    //Getteur
    public Carte getLast() {
        return ligne.get(nbCartes - 1);
    }
    public Carte getCarte(int i) {
        return ligne.get(i);
    }
    public int getNbCartes() {
        return this.nbCartes;
    }

    //Fonction toString
    public String toString() {
        String s = "- série n° " + ((this.indice % 4) + 1) + " : ";
        for (int i = 0; i < nbCartes; ++i)
            if (i < nbCartes-1)
                s = s + ligne.get(i) + ", ";
            else
                s = s + ligne.get(i);
        return s.trim();
    }
}
```

Bilan du Projet

Le défi qu'a représenté ce Projet

Ce Projet fut une véritable épreuve, nous avons passé beaucoup de temps à analyser le sujet afin de trouver le moyen le plus efficace pour développer et optimiser au mieux notre projet final. Nous sommes très fiers de notre résultat car il est fonctionnel et Nous permet de jouer une partie entière.

Les difficultés

La plus grande difficulté que nous avons rencontrée sont les petits détails d'affichage et les exceptions que nous n'avions pas forcément prévu. La gestion du temps et l'évaluation de la charge de travail en début de projet fut aussi une petite épreuve, nous ne pensions pas que ce projet prendra autant de temps à réaliser, surtout au niveau des tests, en effet, le fait de devoir rejouer une partie en entier et manuellement pour corriger les petites erreurs fut assez contraignant mais nous avons réussi a nous en sortir.

Nos points forts

Nous avons porté un point d'honneur sur la structuration et l'optimisation de notre code. Le Java nous permet d'organiser notre travail et de le rendre très lisible et compréhensible, c'est pourquoi nous avons passé du temps pour qu'il le soit au mieux.

Pour conclure

Pour conclure, nous avons pris beaucoup de plaisir à réaliser ce projet, il a représenté un défi que nous avons surmonter et nous en sommes très fier. Nous avons hâte de continuer de nous améliorer.