Vanessa Pham
Xingtong Lin
Groupe 103

Rapport R2.01

Développement orienté objets

Remarque :

13/03/2022

TABLE DES MATIÈRES

Introduction du projet

L’objectif de ce projet est de programmer le jeu “6 qui prend”.
En résumé, c'est un jeu qui consiste à laisser chaque joueur sélectionner une carte de sa main, puis de la placer dans une série de carte de façon croissante et en respectant la limite de cartes qu’une série peut contenir. Selon les situations de jeu des règles et méthodes devront être appliqués.

Pour programmer ce jeu nous devons :
Créer un paquet de cartes, selon certaines contraintes c’est-à-dire 104 cartes avec des informations spécifiques selon un nombre de cartes, car une carte a un certain nombre de têtes de bœuf selon son indice.
Pour ce faire nous avons créé une classe Carte.

On a bien sûr besoin de joueurs, avec un prénom, des cartes en main, des cartes ramassées etc. Nous avons créé une classe Joueur, cette classe aura des méthodes qui représentera dans le monde réel de récupérer une carte qu’on nous distribue, de choisir une carte ou série etc.

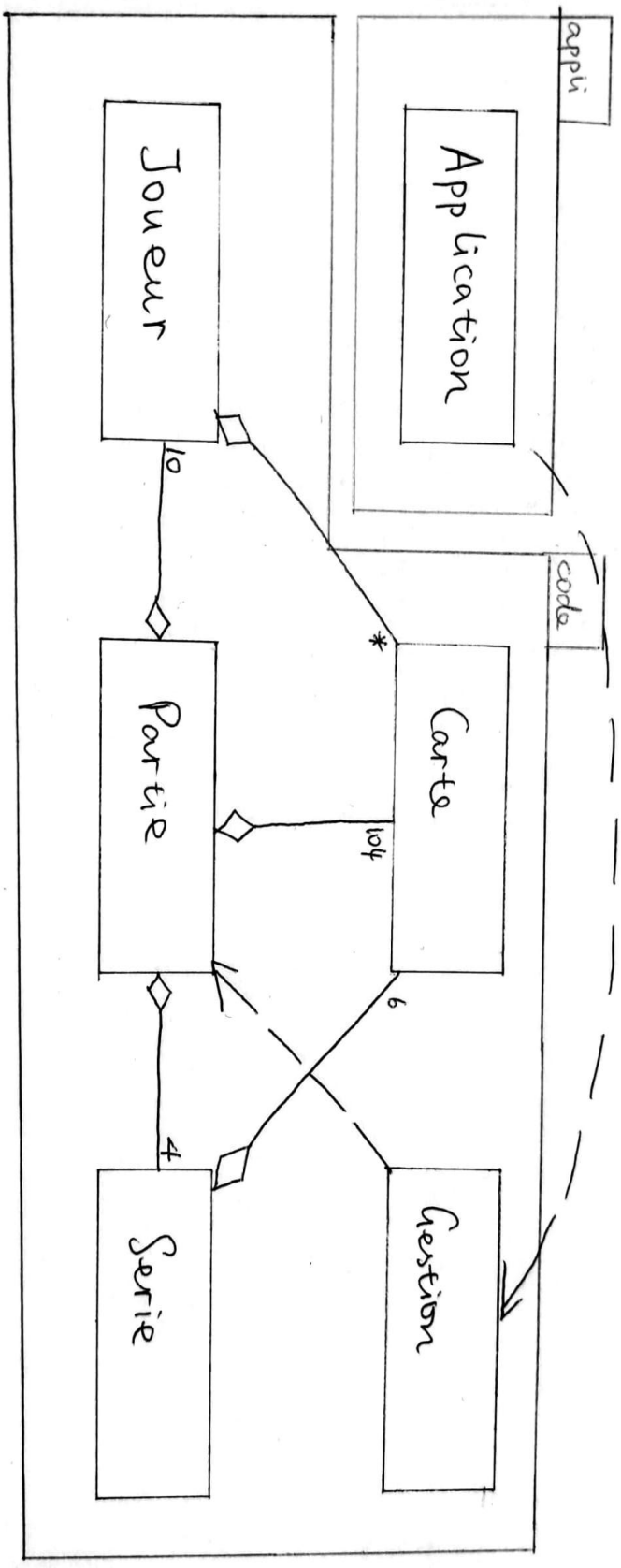
Ensuite le jeu se fera dans un cadre fermé, une partie simple. On crée une classe Partie qui aura besoin de méthode comme créer des cartes, les mélanger, les distribuer, appeler les joueurs à jouer, trier, placer, appliquer des méthodes selon la situation etc.

Pour plus de clarté nous avons créé une classe Série, elle est utilisée dans la classe partie car dans une partie il y a les Séries de cartes ou les cartes devront être placées selon les pivots.

Le programme devra respecter des contraintes de formats et de saisie avec une gestion des erreurs. Elle devra mettre en place une gestion de <pause> et <clearscreen> comme un traitement en console sous un terminal.
Il doit pouvoir lire un fichier texte, dont le nombre et le prénom des joueurs sera indiqué puis lancer une partie en permettant les saisis au clavier.

Nous devons créer une Classe Gestion qui gère la partie mais soit différente de l’Application (Main) afin qu’il y ait une séparation entre le code et le main.

Diagramme UML



Classe Carte		Classe Serie
int carteId; int teteDeBoeuf;		int serieID; ArrayList<Carte> cartesDsSerie; int MAXCarteDsSerie = 6;

public Carte(int carteId);
public int getCarteId();
public int getTeteDeBoeuf;
public String toString();
		public Serie(int serieID);
		public int getSerieId()
		public String toString()

Classe Joueur
String prenom; ArrayList<Carte> cartesRamassees; ArrayList<Carte> hand; int points;
public Joueur(String prenom) public void prendCarte(Carte carte) public void choisirCarte() private boolean checkInDeckCart(int carteACheck) private boolean checkInt(String i) public String toString()

Classe Partie

int nbSerie = 4;
int maxCarteEnMain = 10;
int maxCarte = 104;
ArrayList<Carte> deck = new ArrayList<Carte>();
ArrayList<Joueur> joueurArray = new ArrayList<Joueur>();
Serie[] serieArray = new Serie[nbSerie];

public void lire(String filename) public static int getnbSerie() public String entreeJoueur() public void creerDeck() public void melangerCarte() public void distribuerCarte() public void initialiserSerie() private void function_sort(List<Carte> cartes) public void JoueurSelectionDesCartes()

Code Java des Tests Unitaires

CarteTest	

package test;
import code.Carte;
import static org.junit.Assert.*;
 	
import org.junit.Test;
 	
public class CarteTest {
/*
* On veut tester que créer une carte fonctionne bien
*/
@Test
public void test() {
Carte a=new Carte(5), b = new Carte(15);
assertFalse(a.toString().equals(b.toString()));
}
}
	Succès

JoueurTest	

package test;
import code.Joueur;
import static org.junit.Assert.*;
 	
import org.junit.Test;
 	
public class JoueurTest {
/*
* on veut tester que lire est le prenom d'un joueur fonctionne bien
*/
@Test
public void test() {
String k = "Karl";
String s = "Suzi";
Joueur j = new Joueur(k), b = new Joueur(s);
assertFalse(j.toString().equals(b.toString()));
}
}
	Succès

PartieTest	

package test;
import code.Partie;
import static org.junit.Assert.*;
 	
import org.junit.Test;
 	
import code.Joueur;
public class PartieTest {
 	
/*
* on veut tester les methodes de parties
*/
@Test
public void test() {
Partie partie = new Partie();
partie.creerDeck();
partie.entreeJoueur();
fail("Not yet implemented");
}
 	
}
	Echec

SerieTest	

package test;
import code.Serie;
import static org.junit.Assert.*;
 	
import org.junit.Test;
 	
public class SerieTest {
 	
//on veut tester qu'on attribue bien un indice a une série
@Test
public void test() {
Serie a = new Serie(1), b = new Serie(3);
assertFalse(a.toString().equals(b.toString()));
}
}
	Echec

Bilan du projet

Nous avons rencontré beaucoup de difficultés, tant en algorithmique qu'en logique et en syntaxe.

A travers ce projet nous avons réussi à assimiler les notions de public, private, static, Arraylist, List, les .this, les paramètres, etc.
Nous avons réussi à faire une lecture et récupération d'information d'un fichier, appelé certaines informations stockées dans des listes. Créer une partie en présentant les joueurs, créer des cartes, les mélanger, les distribuer et placer 4 cartes dans un ordre croissant sur 4 séries. Puis nous avons réussi à faire un affichage qui appelle un joueur à saisir une carte et de rentrer au clavier une saisie.

Cependant, nous n'avons pas réussi à avancer d'avantage. Nous n'avons pas réussi à traiter les erreurs dans le cas où on saisit une chaîne de caractère et non spécifiquement un entier. Ce problème nous à fortement ralenti.

Pour améliorer le programme, il aurait fallu avoir des méthodes dans la Classe Partie qui permet de trier les cartes sélectionnées par les joueurs et les placer dans les séries et ainsi de suite pour les autres situations.

Il faudrait également remédier à nos lacunes personnelles vis-à-vis du développement de code. Nous avons des difficultés avec la logique et comment structurer les étapes malgré la visualisation d'un objet réel et ses actions. Il faudra sans doute plus d'aisance en programmation.

Code Java Complet

```
package code;

public class Carte {
	private int carteId;			//numero de la carte
	private int teteDeBoeuf;		//teteDeBoeuf

	/*Constructeur de carte
	  Initialise le nombre de tete de boeuf
	  @param[in] Int carteId
	 */
	public Carte(int carteId) {
		assert ((carteId >= 1) && (carteId <= 104));
		this.carteId = carteId;

		this.teteDeBoeuf = 0;
		if ( carteId % 5 == 0) {		//les cartes terminant par un 5 valent 2
TeteDeBoeuf
			this.teteDeBoeuf += 2;
		} if ( carteId % 10 == 0) {		//les cartes terminant par un 0 en valent 3
TeteDeBoeuf
			this.teteDeBoeuf += 3;
		} if ( carteId % 11 == 0) {		//les cartes formés par deux chiffres égaux
valent 5 TeteDeBoeuf
			this.teteDeBoeuf += 5;
		} if (teteDeBoeuf == 0)			//les autres cartes valent 1
TeteDeBoeuf
			this.teteDeBoeuf = 1;
	}

	//get CarteId, permet de recuperer/retourner l'information "CarteId"
	public int getCarteId() {
		return carteId;
	}

	//get teteDeBoeuf, permet de recuperer/retourner l'information "teteDeBoeuf"
	public int getTeteDeBoeuf() {
		return teteDeBoeuf;
	}

	/* Permet d'avoir une chaine de caractère specifique pour une carte
	 * exemple de format carte 15 avec 2 tetes de boeuf :  15 (2)
	*/
	public String toString() {
		String s = "";
		if (getTeteDeBoeuf()==1)
			s+= getCarteId();
		else
			s+= getCarteId() + " (" + getTeteDeBoeuf() + ")";
		return s;
	}
}
```

```
package code;

import java.util.*;
import java.util.Scanner;


public class Joueur {
	private String prenom;                                    //le prenom du joueur
	private ArrayList<Carte> cartesRamassees; //les cartes ramassees du joueur
	public ArrayList<Carte> hand;                  //les cartes que le joueur possède
	private int points;                                       //points/tetes de boeuf ramassees


	/* Constructeur de joueur
	 * @param[in] String prenom
	 */
	public Joueur(String prenom) {
		this.prenom = prenom;
		this.hand = new ArrayList<Carte>();
		this.cartesRamassees = new ArrayList<Carte>();
		this.points = 0;
	}


	/* Methode : le joueur prend une carte distribue,
	 * le joueur la garde dans sa "main"
	 * @param[in] Carte carte
	 */
	public void prendCarte(Carte carte) {
		hand.add(carte);
	}

	/* Methode : le joueur choisi une carte de sa "main"
	 * il ne peut saisir qu'un carte qu'il possede
	 */
	public void choisirCarte() {//throws java.lang.NumberFormatException
		System.out.println("Saisissez une carte :");
		Scanner sc = new Scanner(System.in);

		try {
			String s = sc.next();
			boolean isInt = checkInt(s);
			int i = Integer.valueOf(s);
			boolean isCartInDeck = checkInDeckCart(i);// Check dans le deck si
la carte existe

			// Sinon reboucler sur la lecture d'entrée
			while(!isCartInDeck||!isInt){//
				System.out.print("Vous n'avez pas cette carte, saisissez
votre choix : ");
		sc = new Scanner(System.in);
		String re = sc.next();
				int reJouer = Integer.valueOf(re);
```

```
                                isCartInDeck = checkInDeckCart(reJouer);
                                isInt = checkInt(re);
                        }

                        // Si oui alors enlever du deck la carte
                        for(int j = 0; j < this.hand.size(); j++) {
                                if (this.hand.get(j).getCarteId() == i)
                                        this.hand.remove(j);
                        }

                }catch(java.lang.NumberFormatException e) {
                        System.out.println("Vous n'avez pas cette carte, saisissez votre
choix : ");

                }
        }

        /* Fonction boolean qui indique si une carte est dans la main du joueur
         * @param[in] int carteACheck
         */
        private boolean checkInDeckCart(int carteACheck) {
                for(Carte c: this.hand) {
                        if (carteACheck == c.getCarteId())
                                return true;
                }
                return false;
        }

        /* Fonction boolean qui indique si la valeur saisi est un entier
         * @param[in] String i
         */
        private boolean checkInt(String i) {
                for(char c : i.toCharArray()) {
                        if (Character.isDigit(c))
                                return true;
                }
                return false;
        }

        /* Permet d'avoir une chaine de caractère specifique pour le joueur
         * retourne le prenom
         */
        public String toString() {
                return prenom;
        }

}
```

```
package code;

import java.util.ArrayList;


public class Serie {
	public int serieID;                                          //l'indice de la serie
	public ArrayList<Carte> cartesDsSerie;                       //liste des cartes de la serie
	public int MAXCarteDsSerie = 6;       //Max de carte que peut contenir une serie


	//Constructeur de Serie
	public Serie(int serieID) {
		this.serieID = serieID;
		this.cartesDsSerie = new ArrayList<Carte>();
	}

	//get SerieId, permet de recuperer/retourner l'information "SerieId"
	public int getSerieId() {
		return serieID;
	}

	/* Permet d'avoir une chaine de caractère specifique pour la serie
	 * exemple :
	 * - Série n° 1 : 23
	 */
	public String toString() {
		String s = "";
		s+= "- Série n° " + getSerieId() + " : ";
		for (int i = 0; i< cartesDsSerie.size()-1; i++) {
			s+= cartesDsSerie.get(i).toString() + ",";
		}
		s+= cartesDsSerie.get(cartesDsSerie.size()-1).toString();
		return s;
	}
}
```

```
package code;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;
import java.io.BufferedReader;
import java.io.FileReader;
import util.Console;
```

```
public class Partie {
	private static int nbSerie = 4;
	//Nombre de serie sur dans la partie
	private static int maxCarteEnMain = 10;
	//Max de carte qu'un joueur peut avoir dans sa main
	private static int maxCarte = 104;
	//Max de carte dans la partie
	private ArrayList<Carte> deck = new ArrayList<Carte>();                    //deck
ou la pioche, liste de carte
	public ArrayList<Joueur> joueurArray = new ArrayList<Joueur>();   //liste des
joueurs dans la partie
	private Serie[] serieArray = new Serie[nbSerie];                           //liste de
series

	/* Methode : lecture d'un fichier texte
	 * recuperer les prenoms du fichier texte
	 */
	public void lire(String filename) throws FileNotFoundException, IOException {
		FileReader in = new FileReader(filename);
      BufferedReader bin = new BufferedReader(in);

		while (bin.ready()) {
	   String line = bin.readLine();
	   Joueur j = new Joueur(line);
	   joueurArray.add(j);
		}
      bin.close();
	   }


	//get nbSerie, permet de recuperer/retourner l'information nbSerie
	public static int getnbSerie() {
		return nbSerie;
	}

	/* Methode : chaine de caractere specifique
	 * annonce le nom des joueurs
	 */
	public String entreeJoueur() {
		String s = "";
		for (int i = 0; i < this.joueurArray.size() - 2 ; i++) {
			s+= this.joueurArray.get(i).toString()+", ";
		}
		s+= this.joueurArray.get(this.joueurArray.size()-2).toString() + " et " +
this.joueurArray.get(this.joueurArray.size()-1).toString();
		return s;
	}

	// Methode : Creer un paquet de cartes
	public void creerDeck() {
		for (int i = 1 ; i <= maxCarte ; i++) {
			Carte carte = new Carte(i);
			deck.add(carte);
```

```
        }
    }

    // Methode : melange le paquet de cartes
    public void melangerCarte() {
        Collections.shuffle(deck);
    }

    // Methode : distribuer des cartes aux joueurs
    public void distribuerCarte() {
        for (int i = 0; i < this.joueurArray.size(); i++) {
            for(int j = 0; j < maxCarteEnMain; j++) {
                Carte carte = deck.get(0);
                this.joueurArray.get(i).prendCarte(carte);
                deck.remove(0);
            }
        }
    }


    // Methode : initialise les series de la partie avec une carte
    public void initialiserSerie() {

        List<Carte> cartes = new ArrayList<Carte>();
        for (int i = 0; i < nbSerie; i++) {
            Carte carte = deck.get(0);
            deck.remove(0);
            cartes.add(carte);
        }

        // Mélange dans l'ordre croissant les cartes
        function_sort(cartes);

        // Les 4 séries ont été créer et ont reçu une carte
        for (int i = 0; i < nbSerie; i++) {
            this.serieArray[i] = new Serie(i+1);
            this.serieArray[i].cartesDsSerie.add(cartes.get(i));
        }
    }

    /* fonction de tri, tri les cartes
     * @param[in] list<Carte> cartes
     */
    private void function_sort(List<Carte> cartes) {

        for (int i = 0; i < cartes.size(); i++) {
            for (int j = 1; j < (cartes.size() - i); j++) {
                if (cartes.get(j - 1).getCarteId() > cartes.get(j).getCarteId()) {
                    //swap elements
                    Carte temp = cartes.get(j - 1);
                    cartes.set(j - 1, cartes.get(j));
                    cartes.set(j, temp);
                }
            }
```

```
		}
	}


	/* Methode : le joueur est appele
	 * Ses cartes lui sont presente
	 * Appelle de la methode "choisirCarte()" de la classe joueur
	 * le joueur
	 *  va pouvoir choisir une carte
	 */
	public void JoueurSelectionDesCartes(){
		for (int i = 0; i < this.joueurArray.size(); i++) {
			System.out.println("A " + this.joueurArray.get(i).toString() + " de
jouer");
			Console.pause();

			for (int j = 0; j < Partie.getnbSerie(); j++) {
				System.out.println(this.serieArray[j].toString());
			}
			System.out.println("Vos cartes sont :" + this.joueurArray.get(i).hand);
			this.joueurArray.get(i).choisirCarte();
			Console.clearScreen();
		}
	}
}
```

```
package code;

import java.io.FileNotFoundException;
import java.io.IOException;

public class Gestion {

	public static void play(String filename) throws FileNotFoundException, IOException
{

		Partie partie = new Partie();
		partie.lire(filename);
		System.out.println("Les " + partie.joueurArray.size() + " joueurs sont " +
partie.entreeJoueur() + ". Merci de jouer à 6 qui prend !");

		partie.creerDeck();
		partie.melangerCarte();
		partie.distribuerCarte();
		partie.initialiserSerie();
		partie.JoueurSelectionDesCartes();
	}
}
```

```
/* Projet DOO
 * Auteur : Vanessa Pham
 * Auteur : Xingtong Lin
 * Groupe : 103
 * "6 qui prend"
 * IUT Paris Rives de Seine, 13/03/2022
 */

package appli;
import java.io.FileNotFoundException;
import java.io.IOException;
import code.Gestion;


public class Application {

	public static void main(String[] args) throws FileNotFoundException, IOException {
		Gestion.play("./config.txt");

	}

}
```

+ le package util donné