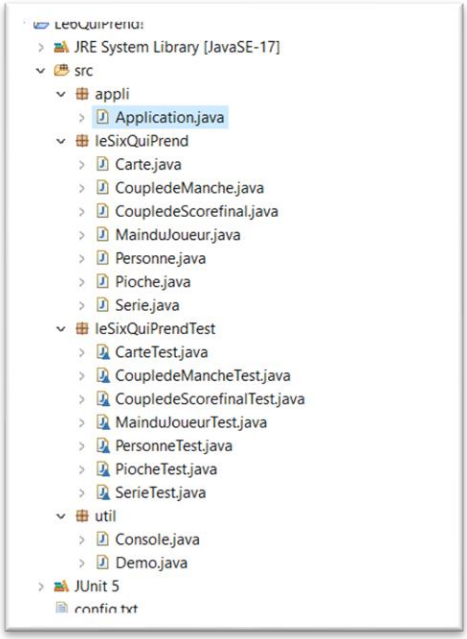
Année scolaire : 2021-2022 (période C)

Université de Paris

RABIARIVELO Ilo Andrianaly

Groupe 35

Table des matières :



Introduction au projet :

Le six qui prend (résumé du jeu) :

Le jeu peut se jouer de 2 à 10 joueurs. Il y a 104 cartes numérotés et possédant chacune un nombre de têtes de bœufs qui seront une sorte de malus pour le score final. Les têtes de bœufs sont définies par le numéro de la carte (3 si elle se termine par un 0, 2 si elle se termine par un 5, 7 si elle correspond au numéro 55 ,5 si elle possède le même chiffre deux fois par exemple 22 ou 11 et le reste 1 tête de bœuf).

Le jeu se déroule de cette façon :

Tous les joueurs reçoivent 10 cartes par personne, et 4 cartes sont posées sur la table pour former des séries. Durant chaque manche, chaque joueur par tours va choisir une carte et la place en face cachée. Lorsque tous les joueurs ont posé leurs cartes cachées, toutes les cartes sont retournées et elles vont être placé dans l'ordre croissant de leurs valeurs sur les séries.

Lorsqu'on pose les cartes dans la série correspondante, il y a quatre règles :

- La carte doit être jouée sur une rangée dont la valeur de la dernière carte est inférieure à celle de la carte jouée

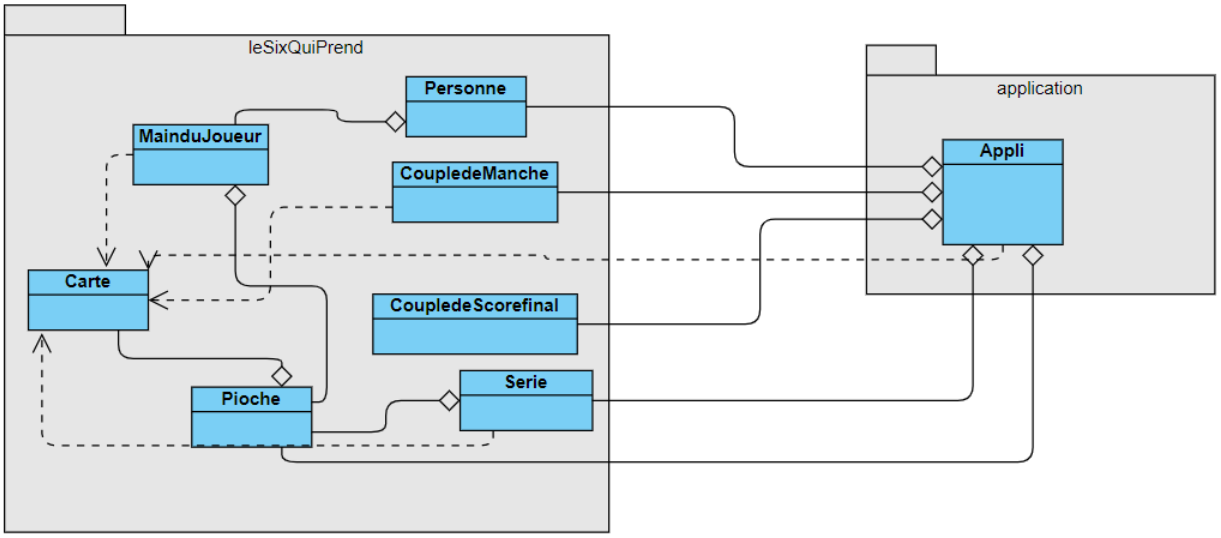
- Entre les rangées dont la dernière carte est inférieure à la carte jouée, il faut choisir celle où l'écart avec la carte jouée est le plus faible

- Si la carte à jouer est de valeur inférieure à la dernière carte de toutes les rangées, le joueur choisit une rangée qu'il ramasse et pose sa carte comme première carte de la nouvelle rangée et récupère toutes les cartes de la série pour son score finales

- Si la carte à jouer et la sixième, il récupère aussi toutes les têtes de bœufs de la série.

Le jeu se termine lorsque plus aucun joueur n'a de cartes soit au bout de la dixième manche. Chaque joueur comptabilise le nombre de têtes de bœufs récupéré et le classement final se fait dans l'ordre croissant (celui qui on a le moins finit premier alors que celui qui on a le plus finit dernier).

Diagramme UML :



Code Source :

Test unitaires :

```
package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */

import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;

class CarteTest {

	@Test
	/**
	 * Test du constructeur de la carte
	 */
	void testCarte() {
		Carte c4 = new Carte();
		assertEquals("Erreur sur la carte",c4.toString()," 4");
	}

	@Test
	/**
	 * Test pour récuperer la valeur des tetes de boeufs
	 */
	void testGetTetedeboeuf() {
		Carte c8=new Carte();
		Carte c9=new Carte();
		Carte c10=new Carte();
		assertEquals("Erreur sur la carte",c10.getTetedeboeuf(),3);
```

```
	assertEquals("Erreur sur la carte",c9.getTetedeboeuf(),1);
	assertEquals("Erreur sur la carte",c8.getTetedeboeuf(),1);
}

@Test
/**
 * Test pour récuperer le numéro de la carte
 */
void testGetNumdecarte() {
	Carte c1=new Carte();
	Carte c2=new Carte();
	Carte c3=new Carte();
	assertEquals("Erreur sur la carte",c3.getNumdecarte(),3);
	assertEquals("Erreur sur la carte",c2.getNumdecarte(),2);
	assertEquals("Erreur sur la carte",c1.getNumdecarte(),1);
}

@Test
/**
 * Test pour l'affichage de la carte
 */
void testToString() {
	Carte c5=new Carte();
	Carte c6=new Carte();
	Carte c7=new Carte();
	assertEquals("Erreur sur la carte",c5.toString()," 5 (2)");
	assertEquals("Erreur sur la carte",c6.toString()," 6");
	assertEquals("Erreur sur la carte",c7.toString()," 7");
}

@Test
/**
 * Test pour la comparaison des cartes
 */
```

```
	void testCompareCarte() {
		Carte c11=new Carte();
		Carte c12=new Carte();
		assertEquals("Erreur sur la carte",Carte.compareCarte(c11,c12),-1);
	}

}
package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */
import static org.junit.Assert.assertEquals;
import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;

class CoupledeMancheTest {

	@Test
	/**
	 * Test pour le constructeur du couple
	 */
	void testCoupledeManche() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le
couple",cdm1.toString(),c1.getNumdecarte()+" (Bernard)");
	}

	@Test
	/**
	 * Test pour l'accès du nom du joueur
```

```
	 */
	void testGetPseudodujoueur() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le
couple",cdm1.getPseudodujoueur(),"Bernard");
	}

	@Test
	/**
	 * Test si le couple a acces au joueur
	 */
	void testGetJoueur() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le
couple",cdm1.getJoueur().toString(),"Bernard");
	}

	@Test
	/**
	 * Test si le couple a accès au numéro de la carte
	 */
	void testGetNumerodecarte() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le
couple",cdm1.getNumerodecarte(),c1.getNumdecarte());
```

```
	}

	@Test
	/**
	 * Test la comparaison de deux noms de joueurs
	 */
	void testComparePseudo() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		Personne j2 = new Personne("Bob",p);
		Carte c2 = p.piocher();
		CoupledeManche cdm2 = new CoupledeManche(c2,j2);
		//System.out.print(CoupledeManche.comparePseudo(cdm1, cdm2));
		assertEquals("Erreur sur le
couple",CoupledeManche.comparePseudo(cdm1, cdm2),-10);
	}

	@Test
	/**
	 * Test pour la comparaison des cartes
	 */
	void testCompareCarte() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		Personne j2 = new Personne("Bob",p);
		Carte c2 = p.piocher();
		CoupledeManche cdm2 = new CoupledeManche(c2,j2);
		assertEquals("Erreur sur le couple",CoupledeManche.compareCarte(cdm1,
cdm2),c1.getNumdecarte()-c2.getNumdecarte());
	}
```

```
	@Test
	/**
	 * test l'affichage des couples
	 */
	void testToString() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le
couple",cdm1.toString(),c1.getNumdecarte()+" (Bernard)");

	}

	@Test
	/**
	 * test le nombre de tête de boeuf par manche dans le couple
	 */
	void testGetTeteDeBoeuf() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1 = p.piocher();
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		assertEquals("Erreur sur le couple",cdm1.getTeteDeBoeuf(),0);
	}

	@Test
	/**
	 * test le nombre de tête de boeuf dans la manche
	 */
	void testScoredeTetedeBoeuf() {
		Pioche p = new Pioche();
		Personne j1 = new Personne("Bernard",p);
		Carte c1=new Carte();
```

```
		Serie s1=new Serie(p);
		CoupledeManche cdm1 = new CoupledeManche(c1,j1);
		cdm1.scoredeTetedeBoeuf(s1);
		assertEquals("Erreur sur le
couple",cdm1.getTeteDeBoeuf(),s1.getSerie().get(0).getTetedeboeuf());



	}


}
package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */

import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;

class CoupledeScorefinalTest {

	@Test
	/**
	 * Test pour le constructeur du score final
	 */
	void testCoupledeScorefinal() {
		CoupledeScorefinal c1 = new CoupledeScorefinal(2,"Bernard");
		assertEquals("Erreur sur le couple final",c1.toString(),"Bernard a
ramassé 2 têtes de boeufs");
	}

	@Test
	/**
```

```
	 * Test pour la comparaison de pseudo du joueur
	 */
	void testComparePseudo() {
		CoupledeScorefinal c1 = new CoupledeScorefinal(2,"Bernard");
		CoupledeScorefinal c2 = new CoupledeScorefinal(8,"Jack");
		assertEquals("Erreur sur le couple final",CoupledeScorefinal.comparePseudo(c1,c2),-8);
	}

	@Test
	/**
	 * Test pour la comparaison du score
	 */
	void testCompareScore() {
		CoupledeScorefinal c1 = new CoupledeScorefinal(2,"Bernard");
		CoupledeScorefinal c2 = new CoupledeScorefinal(8,"Jack");
		assertEquals("Erreur sur le couple final",CoupledeScorefinal.compareScore(c1,c2),-6);
	}

	@Test
	/**
	 * Test pour l'affichage des scores finaux
	 */
	void testToString() {
		CoupledeScorefinal c1 = new CoupledeScorefinal(2,"Bernard");
		assertEquals("Erreur sur le couple final",c1.toString(),"Bernard a ramassé 2 têtes de boeufs");
	}

}
```

```
package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */
```

```
import static org.junit.Assert.assertEquals;
import leSixQuiPrend.*;

import org.junit.jupiter.api.Test;

class MainduJoueurTest {

	@Test
	/**
	 * Test pour le constructeur de la main du joueur
	 */
	void testMainduJoueur() {
		Pioche p = new Pioche();
		MainduJoueur m1= new MainduJoueur(p);
		assertEquals("Erreur sur la main",m1.getcartes().size(),10);
	}

	@Test
	/**
	 * Test pour la récupération des têtes de boeufs dans le score final
	 */
	void testScoredetetedeboeuf() {
		Pioche p = new Pioche();
		MainduJoueur m1= new MainduJoueur(p);
		Serie s1= new Serie(p);
		m1.scoredetetedeboeuf(s1);
		assert(m1.getScoretotaldeboeuf()>0);
		assert(s1.getSerie().size()==0);
	}

	@Test
	/**
	 * Test pour voir si la carte est dans la main
```

```
 */
void testEstdanslamain() {
	Pioche p = new Pioche();
	MainduJoueur m1= new MainduJoueur(p);
	//test sur une carte inexistante
	assertEquals("Erreur sur la main",m1.estdanslamain(205),(false));

}

@Test
/**
 * Test pour voir si la main peut jouer sur une série
 */
void testJouerunecarte() {
	Pioche p = new Pioche();
	MainduJoueur m1= new MainduJoueur(p);
	Serie s1 = new Serie(p);
	m1.jouerunecarte(m1.getcartes().get(3).getNumdecarte(), s1);
	//la taille de la série à augmenter
	assertEquals("Erreur sur la main",s1.getSerie().size(),2);
}


@Test
/**
 * Test si la carte a été prisede la main
 */
void testPrendrelacarte() {
	Carte M1;
	int lavaleurdeM1;
	Pioche p = new Pioche();
	MainduJoueur m1 = new MainduJoueur(p);
	lavaleurdeM1=m1.getcartes().get(3).getNumdecarte();
	M1=m1.prendrelacarte(m1.getcartes().get(3).getNumdecarte());
```

```
		assertEquals("Erreur sur la main",M1.toString()," "+lavaleurdeM1);
		assertEquals("Erreur sur la main",m1.getcartes().size(),10);
	}

	@Test
	/**
	 * Test si la main de carte est possible en vérifiant sa taille
	 */
	void testGetcartes() {
		Pioche p = new Pioche();
		MainduJoueur m1 = new MainduJoueur(p);
		assertEquals("Erreur sur la main",m1.getcartes().size(),10);
	}

	@Test
	/**
	 * Test le score total de tete de boeuf initialisé à 0
	 */
	void testGetScoretotaldeboeuf() {
		Pioche p = new Pioche();
		MainduJoueur m1 = new MainduJoueur(p);
		Serie s1 = new Serie(p);
		assertEquals("Erreur sur la main",m1.getScoretotaldeboeuf(),0);
		m1.scoredetetedeboeuf(s1);
		assert(m1.getScoretotaldeboeuf()>0);
		assert(s1.getSerie().size()==0);
	}

}

package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */
```

```
import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;

class PersonneTest {

	@Test
	/**
	 * Test sur le constructeur de personne
	 * Test sur la carte
	 * Test sur l'affichage du pseudo du joueur
	 */
	void testPersonne() {
		Pioche p = new Pioche();
		Personne p1 = new Personne("Ilo",p);
		assertEquals("Erreur sur la personne",p1.toString(),"Ilo");
		assertEquals("Erreur sur la
personne",p1.getmain().getcartes().size(),10);
		assertEquals("Erreur sur la personne",p1.getPseudo(),"Ilo");
	}

}
package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */
import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;
```

```
class PiocheTest {

	@Test
	/**
	 * Test du constructeur de la pioche
	 * Test pour piocher
	 */
	void testPioche() {
		Pioche p = new Pioche();
		assertEquals("Erreur sur la pioche",p.pioche.size(),104);
		@SuppressWarnings("unused")
		Carte c1 = p.piocher();
		assertEquals("Erreur sur la pioche",p.pioche.size(),103);
	}
}

package leSixQuiPrendTest;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */
import static org.junit.Assert.assertEquals;

import org.junit.jupiter.api.Test;

import leSixQuiPrend.*;

class SerieTest {

	@Test

	/**Test du constructeur de serie
	 * Test pose la série
	 * Test vide la série
	 * Test pour la dernière carte de la série
```

```
	 * Test pour le numéro de la série
	 */
	void testSerie() {
		Pioche p = new Pioche();
		Serie s1 = new Serie(p);
		Carte c1 = new Carte();
		assertEquals("Erreur sur la série",s1.getSerie().size(),1);
		s1.poserdanslaserie(c1);
		assertEquals("Erreur sur la série",s1.getSerie().size(),2);
		assertEquals("Erreur sur la
série",s1.getDerniereCarte(),c1.getNumdecarte());
		s1.videserie();
		assertEquals("Erreur sur la série",s1.getSerie().size(),0);
		assertEquals("Erreur sur la série",s1.getNum_serie(),1);
	}

}
```

leSixQuiPrend :

```
package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 *
 */

public class Carte {
	public static final int nombre_de_carte=104;
	public static int comptecarte=1;
	private int numdecarte;
	private int tetedeboeuf;

	/**@brief crée une carte du sixquiprend
	 */
	public Carte() {
		this.numdecarte=comptecarte++;
		if(this.getNumdecarte()%10!=5 &&
this.getNumdecarte()%10!=this.getNumdecarte()/10 && this.getNumdecarte()%10!=0) {
			this.tetedeboeuf=1;
		}
		if (this.getNumdecarte()%10==5) {
			if (this.getNumdecarte()==55){
				this.tetedeboeuf=7;
				}
			else{
```

```
                        this.tetedeboeuf=2;
                }
            }
            if (this.getNumdecarte()%10==this.getNumdecarte()/10 &&
this.getNumdecarte()!=55) {
                this.tetedeboeuf=5;
            }
            if (this.getNumdecarte()%10==0) {
                this.tetedeboeuf=3;
            }
            }

      /**
       * @brief renvoie le nombre de tête de boeuf d'une carte
       * @return entier correspondant au tête de boeuf de la carte
       */
      public int getTetedeboeuf() {
            return tetedeboeuf;
      }

      /**
       * @brief permet d'accéder au numéro de la carte
       * @return un entier qui correspond au chiffre sur la carte
       */
      public int getNumdecarte() {
            return numdecarte;
      }

      /**
       * @brief affiche la carte avec le chiffre et entre parenthèse les têtes de
boeufs
       * @return un String de la carte.
       */
      public String toString() {
            if (this.tetedeboeuf>1) {
            return " " + this.getNumdecarte()+" "+"("+this.tetedeboeuf+")";
      }
            else {
                  return " "+this.getNumdecarte();
                  }
      }

      /**
       * @brief fais la comparaison entre deux cartes
       * @param c1 la première carte
       * @param c2 la deuxième carte
       * @return un entier permettant de savoir l'ordre
       */
      public static int compareCarte(Carte c1, Carte c2) {
            return c1.numdecarte-c2.numdecarte;
      }


      }

package leSixQuiPrend;

/**
 * @author RABIARIVELO Ilo Andrianaly
```

```
 *
 */

public class CoupledeManche {
		private Carte carte;
		private Personne joueur;
		private int tetedeboeuf;

		/**
		 * @brief crée le tour du joueur pendant la manche
		 * @param c La carte du joueur joué pendant la manche
		 * @param p La personne qui joue la carte
		 */

		public CoupledeManche(Carte c, Personne p) {
			this.carte= c;
			this.joueur = p;
			this.tetedeboeuf= 0;
		}

		/**
		 * @brief permet d'accéder au nom du joueur
		 * @return String du nom du joueur
		 */
		public String getPseudodujoueur() {
			return joueur.getPseudo();
		}

		/**
		 * @brief permet d'accéder au joueur
		 * @return le joueur
		 */
		public Personne getJoueur() {
			return joueur;
		}

		/**
		 * @brief permet d'accéder au numéro de la carte
		 * @return int correspondant au numéro de carte
		 */
		public int getNumerodecarte() {
			return carte.getNumdecarte();
		}


		/**
		 * @brief compare le pseudo de deux attributs de type coupledemanche
		 * @param c1 le premier coupledemanche
		 * @param c2 le deuxième coupledemanche
		 * @return un entier permettant de savoir l'ordre
		 */
		public static int comparePseudo(CoupledeManche c1, CoupledeManche c2)
{
		  return c1.getPseudodujoueur().compareTo(c2.getPseudodujoueur());
	    }

		/**
		 * @brief compare le pseudo et la carte de deux attributs de type
coupledemanche
```

```
             * @param c1 le premier coupledemanche
             * @param c2 le deuxième coupledemanche
             * @return un entier permettant de savoir l'ordre
             */
            public static int compareCarte(CoupledeManche c1, CoupledeManche c2)
{
             if (c1.getNumerodecarte() == c2.getNumerodecarte())
                 return comparePseudo(c1, c2);
             return c1.getNumerodecarte() - c2.getNumerodecarte();
          }

            /**
             * @brief affiche un CoupledeManche sous la forme du numéro de la
carte joué pendant la manche et le prénom du joueur entre parenthèse
             * @return un string du tour jouer par le joueur
             */
            public String toString() {
                  StringBuilder affcouple=new StringBuilder();
                  affcouple.append(this.getNumerodecarte()+"
"+"("+this.getPseudodujoueur()+")");
                  return affcouple.toString();
            }

            /**
             * @brief renvoie les têtes de boeufs que le joueur a eu pendant la
manche
             * @return un entier qui est le nombre de têtes de boeufs
             */
            public int getTeteDeBoeuf() {
                  return this.tetedeboeuf;
            }

            /**
             * @brief prend toutes les têtes de boeufs correspondantes a la série
             * @param serie la serie correspondante
             */
            public void scoredeTetedeBoeuf(Serie serie) {

                  for(int i=0;i<serie.getSerie().size();i++) {

      this.tetedeboeuf+=serie.getSerie().get(i).getTetedeboeuf();
                  }
            }
      }

package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 *
 */

public class CoupledeScorefinal{
      private int score;
      private String perso;

      /**
       * @brief crée le couple final pour l'affichage des scores
       * @param s score de tête de boeuf du joueur
       * @param p le nom du joueur
```

```
	 */
	public CoupledeScorefinal(int s, String p) {
		this.score=s;
		this.perso=p;
	}

	/**
	 * @brief compare deux prénoms de couple pour connaitre l'ordre
	 * @param p1 Le premier couple
	 * @param p2 Le deuxième couple
	 * @return un entier correspondant à la comparaison
	 */
	public static int comparePseudo(CoupledeScorefinal p1, CoupledeScorefinal
p2) {
	  return p1.perso.compareTo(p2.perso);
    }

	/**
	 * @brief compare deux couple par rapport au nom mais aussi au score
	 * @param p1 Le premier couple
	 * @param p2 Le deuxième couple
	 * @return un entier correspondant à la comparaison
	 */
	public static int compareScore(CoupledeScorefinal p1, CoupledeScorefinal p2)
{
	  if (p1.score == p2.score)
	      return comparePseudo(p1, p2);
	  return p1.score - p2.score;
    }

	/**
	 * @brief affiche le score final
	 * @return un string correspondant au joueur et son score
	 */
	public String toString() {
		StringBuilder s=new StringBuilder();
		s.append(this.perso+" a ramassé "+this.score+" têtes de boeufs");
		return s.toString();
	}
}

package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 *
 */

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

public class MainduJoueur {
	private int nombredecarte = 10;
	private ArrayList<Carte> cartes;
	private int scoretotaldeboeuf;

	/**
	 * @brief crée une main pour le joueur en piochant le nombre de carte
correspondant dans la pioche
	 * @param pioche la pioche de la partie
```

```
	 */
	public MainduJoueur(Pioche pioche) {
		this.scoretotaldeboeuf=0;
		this.cartes = new ArrayList<>();
		for(int i=0;i<nombredecarte;i++) {
			cartes.add(pioche.piocher());
		}
		Collections.sort(cartes,Carte::compareCarte);
	}

	/**
	 * @brief ajoute dans le score de tête de boeuf de la main et vide la série
correspondante
	 * @param serie la série dans laquel on récupère les têtes de boeufs et qui
va être vidé
	 */
	public void scoredetetedeboeuf(Serie serie) {
		for(int i=0;i<serie.getSerie().size();i++) {

	this.scoretotaldeboeuf+=serie.getSerie().get(i).getTetedeboeuf();
		}
		serie.videserie();
	}

	/**
	 * @brief vérifie si une carte est bien dans la main du joueur
	 * @param num le numéro de la carte
	 * @return boolean vrai ou faux si elle est bien présente
	 */
	public boolean estdanslamain(int num) {
		for(int i=0;i<this.nombredecarte;i++) {
			if(num==this.cartes.get(i).getNumdecarte()) {
				return true;
			}
		}
		return false;
	}

	/**
	 * @brief joue une carte dans une série
	 * @param num le numéro de la carte
	 * @param serie la série ou la carte va être posé
	 */
	public void jouerunecarte(int num, Serie serie) {
			for(int i=0;i<this.nombredecarte;i++) {
				if(num==this.cartes.get(i).getNumdecarte()) {
				serie.poserdanslaserie(this.cartes.get(i));
				this.cartes.remove(i);
				this.nombredecarte-=1;
				}
			}
	}

	/**
	 * @brief prend une carte la main et l'enlève mais qui renvoie une carte
	 * @param n le numéro de la carte correspondante
	 * @return la carte enlevé de la main
	 */
	public Carte prendrelacarte(int n) {
```

```
		Carte cartearetourner=new Carte();
		for(int i=0;i<this.nombredecarte;i++) {
			if(n==this.cartes.get(i).getNumdecarte()) {
				cartearetourner=this.cartes.get(i);
			}
		}
		return cartearetourner;
	}

	/**
	 * @brief permet d'accéder aux cartes de la main
	 * @return la liste de cartes qui correspond à la main
	 */
	public ArrayList<Carte> getcartes() {
		return this.cartes;
	}

	/**
	 * @brief permet l'accès aux scores totales de têtes de boeuf
	 * @return l'entier correspondant au score total
	 */
	public int getScoretotaldeboeuf() {
		return scoretotaldeboeuf;
	}

	/**
	 * @brief affiche le contenu de la main
	 * @return String toutes les cartes de la main
	 */
	public String toString() {
		StringBuilder lamain = new StringBuilder();
		lamain.append("- Vos cartes :");
		for(int i=0; i<cartes.size();i++) {
			lamain.append(this.cartes.get(i));
			if (this.cartes.size()-1==i) {
				lamain.append("");
			}
			else {
				lamain.append(",");
			}
		}
		return lamain.toString();
	}



}

package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */

public class Personne {
	private String pseudo;
	private MainduJoueur main;

	/**
	 * @brief crée le joueur donc sa main et son nom
```

```
	 * @param nom le nom du joueur enregistré
	 * @param pioche la pioche de la partie
	 */
	public Personne(String nom, Pioche pioche) {
		this.pseudo=nom;
		this.main=new MainduJoueur(pioche);
	}

	/**
	 * @brief permet d'accéder à la main du joueur
	 * @return la main du joueur
	 */
	public MainduJoueur getmain() {
		return main;
	}

	/**
	 * @brief permet d'accéder au pseudo du joueur
	 * @return String le nom du joueur
	 */
	public String getPseudo() {
		return pseudo;
	}


	/**
	 * @brief affiche le nom du joueur
	 * @return String le pseudo du joueur
	 */
	public String toString() {
		return this.pseudo;
	}
}
```

```
package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 *
 */

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

public class Pioche {
	public ArrayList<Carte> pioche;

	/**
	 * @brief crée la pioche de la partie avec des cartes et la mélange
	 */
	public Pioche() {
		this.pioche = new ArrayList<>();
		for(int i=1;i<Carte.nombre_de_carte+1;i++) {
			pioche.add(new Carte());
		}
		Collections.shuffle(pioche);
	}

	/**
	 * @brief permet de piocher une carte du dessus dans la pioche
```

```
	 * @return la carte pioché
	 */
	public Carte piocher() {
		Carte Cartedudessus = pioche.get(0);
		pioche.remove(0);
		return Cartedudessus;
	}

}

package leSixQuiPrend;
/**
 * @author RABIARIVELO Ilo Andrianaly
 *
 */

import java.util.ArrayList;

public class Serie {
	private ArrayList<Carte> serie;
	private int num_serie;
	public static int compteserie=1;

	/**
	 * @brief crée une série à partir d'une pioche
	 * @param pioche la pioche de la partie
	 */
	public Serie(Pioche pioche) {
		this.serie= new ArrayList<>();
		this.getSerie().add(pioche.piocher());
		this.num_serie=compteserie++;
	}

	/**
	 * @brief permet de poser une carte dans la série
	 * @param carte la carte à poser
	 */
	public void poserdanslaserie(Carte carte) {
		this.getSerie().add(carte);
	}

	/**
	 * @brief vide la série de tous ces élements
	 */
	public void videserie() {
		serie.clear();
	}

	/**
	 * @brief renvoie la position de la carte dans la série
	 * @param c le numéro de la carte
	 * @return un entier qui correspond à la position de la carte dans la série
	 */
	public int positiondanslaserie(int c) {
		int pos=1;
		for(int i=0;i<serie.size();i++) {
			if(c==serie.get(i).getNumdecarte()) {
				pos=i;
			}
```

```
			}
			return pos+1;
		}

		/**
		 * @brief permet d'accéder à la série
		 * @return la série
		 */
		public ArrayList<Carte> getSerie() {
			return serie;
		}

		/**
		 * @brief affiche la série avec son numéro et ce qu'elle contient
		 * @return String la série
		 */
		public String toString() {
			StringBuilder laserie=new StringBuilder();
			laserie.append("- série n° "+ this.num_serie+" :");
			for(int i=0; i<this.getSerie().size();i++) {
				laserie.append(this.getSerie().get(i));
				if (this.getSerie().size()-1==i) {
					laserie.append("");
				}
				else {
					laserie.append(",");
				}
			}
			return laserie.toString();
		}

		/**
		 * @brief affiche le numéro de cartes de la dernière carte de la série
		 * @return la dernière carte de la série
		 */
		public int getDerniereCarte() {
			return (serie.get(serie.size()-1).getNumdecarte());
		}

		/**
		 * @brief permet d'accéder au numéro de la série
		 * @return le numéro de la série
		 */
		public int getNum_serie() {
			return num_serie;
		}
}
```

Application:

```
package appli;
/**
 * @author RABIARIVELO Ilo Andrianaly
 */

import static util.Console.clearScreen;
import static util.Console.pause;

import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

import leSixQuiPrend.*;

public class Application {
	public static final int nombredemanche = 10;
	public static final int nombredeserie = 4;

	public static void main(String[] args) throws FileNotFoundException {

		//Création de la table de jeu
		Scanner texte=new Scanner(new FileInputStream("config.txt"));
		Scanner num=new Scanner(System.in);
		Scanner numserie=new Scanner(System.in);
		ArrayList<Personne> joueurs= new ArrayList<>();
		ArrayList<CoupledeScorefinal> classementf=new ArrayList<>();
		Pioche pioche = new Pioche();
		ArrayList<Serie> series = new ArrayList<>();


		//Affichage de la première ligne d'introduction au jeu
		for(int i=0;i<4;i++) {
			series.add(new Serie(pioche));
		}
		while(texte.hasNext()) {
			joueurs.add(new Personne(texte.nextLine(),pioche));
		}
		System.out.print("Les "+joueurs.size()+" joueurs sont ");
		for (int i=0;i<joueurs.size();i++) {
			System.out.print(joueurs.get(i));
			if(i<joueurs.size()-2) {
				System.out.print(", ");
			}
			if(i==joueurs.size()-2) {
				System.out.print(" et ");
			}
			if(i==joueurs.size()-1) {
				System.out.println(". Merci de jouer à 6 qui prend !");
			}
		}

	//Début de manche
	for(int i=0;i<nombredemanche;i++) {
		ArrayList<CoupledeManche> manche=new ArrayList<>();
		for(int a = 0;a<joueurs.size();a++) {
			int numcarte=0;

			System.out.println("A "+joueurs.get(a)+" de jouer.");
			pause();
			for(int j=0; j<nombredeserie;j++) {
				System.out.println(series.get(j));
			}
			System.out.println(joueurs.get(a).getmain());
			System.out.print("Saisissez votre choix : ");
			//Vérifie si la carte est bien un int et est dans la main
			do {
```

```
                    try {
                        numcarte = Integer.parseInt(num.nextLine());
                        if(numcarte < 0)
                            numcarte = -1;
                    }
                    catch(NumberFormatException e) {
                        numcarte = -1;
                    }
                    if(joueurs.get(a).getmain().estdanslamain(numcarte)==false) {
                     numcarte = -1;
                    }
                    if(numcarte==-1) {
                     System.out.print("Vous n'avez pas cette carte, saisissez votre
choix : ");
                    }
                    if(joueurs.get(a).getmain().estdanslamain(numcarte)==true) {
                                manche.add(new
CoupledeManche(joueurs.get(a).getmain().prendrelacarte(numcarte),joueurs.get(a)));
                            }

                     }while(numcarte==-1);

                clearScreen();
                }
                Collections.sort(manche,CoupledeManche::compareCarte);
                //Pose les cartes automatiquement dans les séries (règle numéro 2)
                for(int m=0;m<manche.size();m++) {
                      int entiermax=104;
                      int compteserie=-1;
                      boolean sixiemeposition=false;
                      for(int s=0;s<nombredeserie;s++) {
                            if(manche.get(m).getNumerodecarte()-
series.get(s).getDerniereCarte()>0 && manche.get(m).getNumerodecarte()-
series.get(s).getDerniereCarte()<entiermax) {
                                entiermax=manche.get(m).getNumerodecarte()-
series.get(s).getDerniereCarte();
                                compteserie=series.get(s).getNum_serie()-1;

          if(series.get(s).positiondanslaserie(series.get(s).getDerniereCarte())==5) {
                                      sixiemeposition=true; //Vérifivation de la
règle numéro 3
                                }
                            }
                     }
                     //Affichage du contenu de la manche si la règle numéro 4 est
activé
                     if(compteserie<0) {
                            System.out.print("Les cartes");
                            int taillemanche=0;
                            for(int m1=0;m1<manche.size();m1++) {
                                taillemanche++;
                                if(taillemanche==manche.size())
                                      System.out.println(" et "+manche.get(m1)+"
vont être posées.");
                                if(taillemanche==manche.size()-1)
                                      System.out.print(" "+manche.get(m1));
                                if(taillemanche!=manche.size()-1 &&
taillemanche!=manche.size())
                                      System.out.print(" "+manche.get(m1)+",");
```

```
                        }
                        //Règle numéro 4
                        System.out.println("Pour poser la carte
"+manche.get(m).getNumerodecarte()+", "+manche.get(m).getPseudodujoueur()+" doit
choisir la série qu'il va ramasser.");
                        for(int j=0; j<nombredeserie;j++) {
                              System.out.println(series.get(j));
                        }
                        System.out.print("Saisissez votre choix : ");
                        //Vérifie la saisie de la série pour la règle 4
                        do {
                       try {
                        compteserie = Integer.parseInt(numserie.nextLine());
                           if(compteserie < 0)
                              compteserie = -1;
                       }
                       catch(NumberFormatException e) {
                           compteserie = -1;
                       }
                       if(compteserie>=1 && compteserie<=4) {
                                    compteserie-=1;

      manche.get(m).scoredeTetedeBoeuf(series.get(compteserie));

      manche.get(m).getJoueur().getmain().scoredetetedeboeuf(series.get(compteseri
e));

      manche.get(m).getJoueur().getmain().jouerunecarte(manche.get(m).getNumerodec
arte(), series.get(compteserie));
                              }
                       else {
                        compteserie = -1;
                       }
                       if(compteserie==-1) {
                        System.out.print("Ce n'est pas une série valide,
saisissez votre choix : ");
                        }
                     }while(compteserie==-1);
                  }
                  else {
                        //Règle numéro 3
                        if(sixiemeposition==true) {

      manche.get(m).scoredeTetedeBoeuf(series.get(compteserie));

      manche.get(m).getJoueur().getmain().scoredetetedeboeuf(series.get(compteseri
e));

      manche.get(m).getJoueur().getmain().jouerunecarte(manche.get(m).getNumerodec
arte(), series.get(compteserie));
                        }
                        //Règle numéro 1 et 2 si les autres règles ne sont pas
activés
                        else{

      manche.get(m).getJoueur().getmain().jouerunecarte(manche.get(m).getNumerodec
arte(), series.get(compteserie));
                        }
                  }
```

```
            }
            //Affichage pour la fin d'une manche
            int taillemanche=0;
            System.out.print("Les cartes");
            for(int m=0;m<manche.size();m++) {
                  taillemanche++;
                  if(taillemanche==manche.size())
                        System.out.print(" et "+manche.get(m)+" ont été
posées.");
                  if(taillemanche==manche.size()-1)
                        System.out.print(" "+manche.get(m));
                  if(taillemanche!=manche.size()-1 &&
taillemanche!=manche.size())
                        System.out.print(" "+manche.get(m)+",");
            }
            System.out.println();
            for(int j=0; j<nombredeserie;j++) {
                  System.out.println(series.get(j));
            }
            //Score des joueurs en fin de manche
            int ceuxquiramasse=0;
            for(int m=0;m<manche.size();m++) {
                  if(manche.get(m).getTeteDeBoeuf()>0) {
                        ceuxquiramasse+=1;
                        System.out.println(manche.get(m).getPseudodujoueur()+" a
ramassé "+manche.get(m).getTeteDeBoeuf()+" têtes de boeufs");
                  }
            }
            if(ceuxquiramasse==0) {
                  System.out.println("Aucun joueur ne ramasse de tête de
boeufs.");
            }
      }
      num.close();
      numserie.close();
      //Afficahge du score final
      System.out.println("** Score final");
      for(int f=0;f<joueurs.size();f++) {
            classementf.add(new
CoupledeScorefinal(joueurs.get(f).getmain().getScoretotaldeboeuf(),joueurs.get(f).
getPseudo()));
      }
      Collections.sort(classementf,CoupledeScorefinal::compareScore);
      for(int f=0;f<classementf.size();f++) {
            System.out.println(classementf.get(f));
      }
      }

}
```

Util :

```
package util;

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() {
        }
}
package util;

import java.util.Arrays;

import java.util.Scanner;


import static util.Console.clearScreen;

import static util.Console.pause;


public class Demo {

        // Démonstration

        public static void main(String[] args) {

                pause();

                clearScreen(); // ou util.Cleaner.clearScreen(); voir le 'import
static' ci-dessus

                compteur();

                ventilateur();

                pause();

                baffe();

                System.out.println("c'est fini");

        }


        private static void ventilateur() {

                for (int i = 1; i <= 20; ++i)

                        for (char c : Arrays.asList('-', '\\', '|', '/')) {

                                System.out.println(c);

                                patience(50);

                                clearScreen();

                        }

        }


        private static void compteur() {

                for (int i = 1; i <= 100; ++i) {
```

```
                System.out.println(String.format("%03d", i));
                patience(50);
                clearScreen();
        }
    }

    private static void baffe() {
        @SuppressWarnings("resource")
        Scanner sc = new Scanner(System.in);
        System.out.print("Voulez vous une baffe (O/N) : ");
        String rep = sc.next();
        clearScreen();
        if ("O".equalsIgnoreCase(rep))
            System.out.println("vous vous la prenez");
        else
            System.out.println("tout va bien");
        patience(1000);
        clearScreen();
    }

    // Utilitaire
    private static void patience(long ms) {
        try {
            Thread.sleep(ms);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
```

Bilan du projet :

Les difficultés rencontrées :

Durant ce projet, la difficulté principale fut de le faire seul. Forcément, puisque je suis seul j'ai dû réfléchir seul et créer mon code seul. Bien sûr les cours aidaient beaucoup mais le fait de chercher des solutions et de vérifier la syntaxe du code demande plus d'énergie que si j'avais un binôme. Comme autres difficultés, j'ai pu constater la création des tests unitaires, on n'en a pas beaucoup parlé en cours mais j'ai fait de mon mieux pour en faire. J'ai aussi eu du mal concernant la manipulation des objets dans des listes mais grâce au cours, j'ai su m'en sortir.

Ce qui est réussit :

Le code du projet respecte bien toutes les règles, affiche l'interface de la bonne façon et peut bien être jouer par 2 à 10 joueurs. Toutes les méthodes créées ont été utilisés de la bonne façon et fonctionnent toutes.

Ce qui peut être amélioré :

La qualité du code peut être amélioré en effet peut être en créant une classe « Table» je pourrais stocké les manches et les couples pour rendre plus léger le main. Ensuite peut être la manière penser peut-être améliorer, certes le projet fonctionne mais il pourrait être améliorer en pensant d'une différente manière.