Selvaratnam Akash groupe 111

Lucas Casenaz groupe 108

IUT de Paris – Rives de Seine Département Informatique ´ R2.01 – Développement orienté objets – Projet

6-qui-Prend !

Sommaire

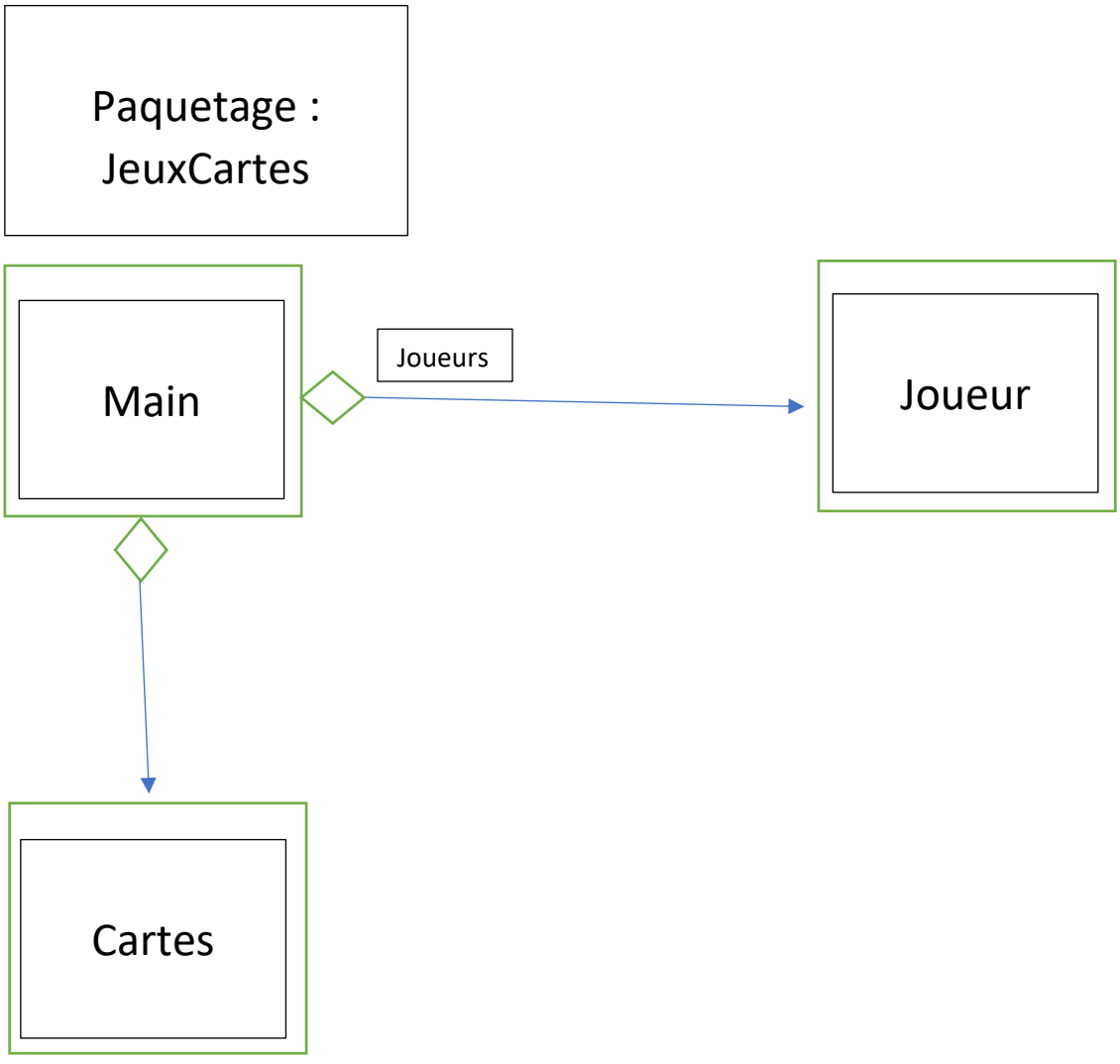
Introduction du projet

Le 6-qui-prend ! est un jeu de société créé par Wolfgang Kramer en 1994. Le 6-qui-prend ! est un jeu de cartes qui oppose 2 à 10 joueurs. Les 104 cartes du jeu contiennent des valeurs numériques de 1 à 104 permettant de repérer les cartes entre eux, chaque carte contient un nombre de tête de bœuf (entre 1 à 7), les têtes de bœufs représentent les pénalités du jeu.

Le paquet de carte doivent être mélangé puis distribué, 10 cartes par joueurs et les 4 dernière cartes restant du paquet sont déposer afin de définir les 4 séries du jeu. Les 4 séries du jeu est l'endroit où les joueurs déposeront leurs cartes à chaque tour, ces séries ne doivent pas comporter plus de 5 cartes. Au début de chaque tour, chaque joueur choisit une carte et la dépose face caché, puis a la fin du tour, chaque joueur retourne sa carte afin de voir la valeur numérique de chaque carte. Les joueurs déposent chacun leur tour leur carte dans une des séries, la carte peut être déposé dans la série seulement si la carte est supérieur à la dernière cartes de chaque série et elle doit être déposer ou la différence entre la carte jouée et la dernière carte de la série est la plus faible, si la carte jouée par le joueur est inférieure à toutes les dernière carte de la série alors le joueur doit récupérer toutes les cartes d'une série de son choix, il sera également pénalisé par le nombre de tête de bœufs associé au cartes récupéré, la carte jouée par le joueur définira la nouvelle série. Le deuxième cas d'être pénalisé est le fait que le joueur doit déposer sa carte dans une série qui contient déjà 5 cartes, alors le joueur doit récupérer toutes les cartes de cette série, et il sera également pénalisé par le nombre de tête de bœufs associé aux cartes récupéré par le joueur, la carte jouée par le joueur définira la nouvelle série. A chaque tour, ce scénario se répète.

Le jeu se termine au bout de 10 tours, il faudra à la fin du jeu avoir un aperçu du nombre total de tête de bœufs récupérer par chaque joueur. Le joueur ayant le moins de tête de bœuf a gagné la partie.

Diagramme UML



Test Unitaire

Notre programme débute avec la présentation des jouant à la partie du 6-qui-prend, l'entrée des joueurs jouant au jeu du 6-qui-prend se lit dans un fichier texte, puis le programme doit afficher le joueur qui doit jouer, puis un affichage « <pause> » s'affiche à l'écran, cet affichage permet de signaler au joueur d'appuyer pour continuer. Après cet affichage, les 4 séries du jeu sont affichés et on affiche également les cartes du joueur. Nous n'avons pas réussi à retirer les crochets des listes de nos séries, ni afficher le nombre de tête de bœufs sur les différentes cartes. On termine l'affichage par « Saissisez votre choix » et le joueur entre une cartes dont il possède.

```
Les 4 joueurs sont Samuel, Bob, Patrick et Camille. Merci de jouer à 6 qui prend !
A Samuel de jouer.
<pause>
- Série n° 1 : [59]
- Série n° 2 : [40]
- Série n° 3 : [90]
- Série n° 4 : [75]
- Vos cartes : [8, 10, 19, 23, 30, 46, 50, 63, 92, 97]
Saisissez votre choix :
4
Vous n'avez pas cette carte, saisissez votre choix :
8
<clearScreen>
A Bob de jouer.
<pause>
- Série n° 1 : [59]
- Série n° 2 : [40]
- Série n° 3 : [90]
- Série n° 4 : [75]
- Vos cartes : [5, 11, 22, 26, 28, 45, 61, 73, 95, 101]
Saisissez votre choix :
11
<clearScreen>
```

Dans le premier cas, si les cartes jouées par les joueurs sont supérieures à toutes les dernières cartes des séries, alors les cartes jouées par les joueurs seront déposées dans la série dans laquelle la différence entre la carte jouée et la dernière carte de la série est la plus faible, il faut également que la série dont elle doit être déposé doit être inférieur à la taille de 5 cartes. L'affichage consiste à afficher les cartes jouées par chaque joueur puis afficher les 4 série du jeu et déterminer le nombre de tête de bœufs récupéré dans ce cas zéros tête de bœufs à été récupéré. Nous n'avons pas réussi à afficher les cartes par ordre croissant, et nous n'avons également pas réussi à retirer les crochets et les têtes de bœufs à côté des cartes.

```
Les cartes 88 (Samuel), 55 (Bob), 64 (Patrick) et 27 (Camille) ont été posées
- Série n° 1 : [91]
- Série n° 2 : [51, 55, 64]
```

```
- Série n° 3 : [24, 27]
- Série n° 4 : [87, 88]
Aucun joueur ne ramasse de tête de boeufs
A Samuel de jouer.
```

Dans le deuxième cas, on peut voir la première façon d'être pénalisé, si les joueurs doivent déposer sa carte dans une série de taille 5 alors il doit récupérer toute la série. Comme on peut voir précédemment la série avait une taille de 4 cartes, avec comme dernière carte 79, puis la carte 84 est déposé dans la série 1 et pour finir la cartes 98 doit être également déposé dans la série 1 mais comme la taille de la série est égale a 5 alors il doit récupérer toute la série. Le programme doit afficher les cartes jouées par chaque joueur puis les 4 série et finir par le nombre de tête de bœuf que doit récupérer le joueur. Nous n'avons pas réussi à afficher les cartes par ordre croissant, et nous n'avons également pas réussi à retirer les crochets et les têtes de bœufs à côté des cartes.

```
A Camille de jouer.
<pause>
- Série n° 1 : [62, 73, 77, 79]
- Série n° 2 : [6]
- Série n° 3 : [14, 17]
- Série n° 4 : [32]
- Vos cartes : [8, 15, 39, 47, 74, 76, 78, 92, 103]
Saisissez votre choix :
39

Les cartes 84 (Samuel), 98 (Bob), 35 (Patrick) et 39 (Camille) ont été posées
- Série n° 1 : [98]
- Série n° 2 : [6]
- Série n° 3 : [14, 17]
- Série n° 4 : [32, 35, 39]
Bob a ramassé 9 têtes boeufs.
A Samuel de jouer.
```

Dans le troisième cas, on peut voir la deuxième façon d'être pénalisé, si les dépose une carte inférieure à toutes les dernières cartes des séries alors il doit récupérer une série de leur choix, c'est le cas de Patrick dans ce tour. Le programme affiche les cartes jouées par les joueurs puis nous signale que Patrick doit choisir la série qu'il veut ramasser pour pouvoir poser sa carte. Il nous affiche les 4 série et nous demande la série dont on a envie de récupéré, le joueur écrit le numéro de la série puis le programme nous rappelle les cartes jouées par les joueurs et le contenu des séries et le nombre de tête de bœufs récupéré par le joueur. Nous n'avons pas réussi à afficher les cartes par ordre croissant, et nous n'avons également pas réussi à retirer les crochets et les têtes de bœufs à côté des cartes.

```
Les cartes 20 (Samuel), 33 (Bob), 2 (Patrick) et 47 (Camille) ont été posées
Pour poser la carte 2, Patrick doit choisir la série qu'il va ramasser
- Série n° 1 : [98]
- Série n° 2 : [6]
- Série n° 3 : [14, 17, 20, 33]
- Série n° 4 : [32, 35, 39]
Saisissez votre choix :
3
Les cartes 20 (Samuel), 33 (Bob), 2 (Patrick) et 47 (Camille) ont été posées
```

```
- Série n° 1 : [98]
- Série n° 2 : [6]
- Série n° 3 : [2]
- Série n° 4 : [32, 35, 39, 47]
Patrick a ramassé 10 tête boeufs.
```

Le programme se termine après ce dernier affichage, il consiste à afficher le nombre de tête de bœufs totaux récupéré par chaque jouer. Nous n'avons pas réussi à afficher le total par ordre croissant.

```
** Score final
Samuel a ramassé 0 tête de boeufs
Bob a ramassé 3 tête de boeufs
Patrick a ramassé 0 tête de boeufs
Camille a ramassé 6 tête de boeufs
```

Bilan du projet

Nous avons effectué le projet du 6-qui-pred avec un réel plaisir car nous avons appris de nombreuses choses dans le langage Java grâce a ce projet, nous avons appris à réaliser des Arraylist (tableau dynamique), utilisation de ses nombreuses commandes telle que (add, remove, get …), ce qui va nous être bénéfique pour la suite, ce projet nous a permis de comprendre l'encapsulation des méthodes et des classes, de lire dans un fichier et de réaliser des getters et des setters.

Au cours de ce projet, nous avons rencontrer des complications dans l'affichage des listes des carte, par exemple : nous n'avons pas réussi à retirer les crochets des ArrayList ou d'ajoute le nombre de tête de bœufs près de chaque carte. Nous n'avons également pas réussi à placer les cartes jouées par chacun dans l'ordre croissant des cartes, donc les joueurs jouent toujours dans le même ordre et le nombre de tête de bœufs totaux de la fin n'est pas noter dans l'ordre croissant. Nous avons eu également rencontré quelque difficulté sur le traitement d'erreur de la classe Cartes.Java ou nous n'avons pas réussi à tester toutes les méthodes.

Au niveau des réussite du projet, nous avons réussi à réaliser toutes les listes demandé grâce à des ArrayList (Pioche (104 cartes), les cartes des joueurs, les séries), nous avons réussi à utiliser la bibliothèque collections pour mélanger ou trier les cartes par exemple, la lecture dans un fichier à était une réussite, nous avons réussi à faire jouer les joueurs dans n'importe quelle des cas. Premier cas réussi (si la carte jouée par les joueurs est correct alors on place ou la différence entre la carte jouée et la dernière de la série est la plus faible et la série doit avoir une taille inférieure à 5). Deuxième cas réussi (si la carte jouée par le joueur doit entrer dans une série dont la taille est de 5 alors il récupère toute la série). Troisième cas réussi (si la carte jouée par le joueur est inférieur à toutes les dernières cartes des séries alors il récupéré la série de son choix). Le calcul et l'affichage du nombre de tête bœufs récupérer à chaque tour a été une réussite et pour finir l'affichage du nombre de tête de bœufs totaux a la fin de la partie à été également une réussite.

On pourra améliorer notre programme tout d'abord au niveau des traitement d'erreur car nous avons choisi de faire le programme puis réaliser le traitement d'erreur, ce qui a été une erreur selon nous car cela aurait pu être plus facile de le traiter au moment ou on réaliser notre programme ce que nous rectifierons lors du prochain projet. Je pense que nos méthodes auraient pu, pour certain être plus courte à réaliser avec surement moins d'algorithme.

Joueur.Java

```java
package JeuxCartes;

public class Joueur {
	private String prenom;// le prenom du joueur
	/*
	 * @param[in] : prenom
	 * initialise le prenom du joueur
	 */
	public void initialise(String prenom) {
		this.prenom = prenom;
	}
	/*
	 * param[out] : this.prenom
	 * Retourne le prenom du joueur
	 */
	public String envoie() {
		return this.prenom;
	}
}
```

Cartes.Java

```
package JeuxCartes;

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



public class Cartes {
	private static ArrayList <Integer> tabCartes = new ArrayList<>();//
Tabcartes est la liste total des cartes du jeux
	private ArrayList <Integer> CarteJoueur = new ArrayList<>();//CartesJoueur
est la liste des cartes de chaque joueur
	private static ArrayList<Integer> Séries1 = new ArrayList<>();//Séries1 est
la liste des cartes de la série 1
	private static ArrayList<Integer> Séries2 = new ArrayList<>();//Séries2 est
la liste des cartes de la série 2
	private static ArrayList<Integer> Séries3 = new ArrayList<>();//Séries3 est
la liste des cartes de la série 3
	private static ArrayList<Integer> Séries4 = new ArrayList<>();//Séries4 est
la liste des cartes de la série 4
	private int CarteJoué;// La cartes joué par le joueur
	private int dist1;// La distance entre la carte joué par le joueur et la
dernière cartes de la série 1
	private int dist2;// La distance entre la carte joué par le joueur et la
dernière cartes de la série 2
	private int dist3;// La distance entre la carte joué par le joueur et la
dernière cartes de la série 3
	private int dist4;// La distance entre la carte joué par le joueur et la
dernière cartes de la série 4
	private int SérieChoisie;// La série choisie par le joueur si la carte joué
par le joueur est inférieur a toutes les derniere cartes des séries
	private int teteboeufsrecup;// Le nombre de tête de bouefs récupéré a chaque
tour
	private int teteboeufstot;// Le nombre de tête de boeufs total récupéré a la
fin de la partie
	private int Avertissement;// Si la carte joué doit entré dans une série
d'une taille 5, avertissement de définir dans quelle série elle doit rentré
	/*
	 * Verification que le nombre cartes attribué est de 104
	 */
	public boolean EstIlinitialisé() {
		if(Cartes.tabCartes.size() == 104) {
			return true;
		}
		return false;
	}
	/*
	 *Initialiser le nombre de cartes à 104
	 */
	public void initialiserCartes() {
		for(int i = 0; i < 104; ++i) {
			Cartes.tabCartes.add(i, (i+1));
		}
	}
	/*
	 * Mélanger les 104 cartes du jeux
	 */
```

```
	public void mélange(){
		Collections.shuffle(tabCartes);
	}
	/*
	 * Verification que les cartes de chaque joueurs ont bien été distribué
	 */
	public boolean estDistribuer() {
		if(this.CarteJoueur.size() == 10) {
			return true;
		}
		return false;
	}
	/*
	 * Distribuer 10 cartes par joueur des 104 cartes du jeux
	 */
	public void distribuerCartes() {
		for(int i = 0; i < 10; ++i) {
			this.CarteJoueur.add(i, tabCartes.get(i));
		}
	}
	/*
	 * Trier les 10 cartes du joueurs du plus pétit au plus grand
	 */
	public void trier(){
		Collections.sort(CarteJoueur);
	}
	/*
	 * Supprimer les cartes distribuer au joueur dans le jeu initiale des 104
cartes
	 */
	public void supprimer() {
		int j = 0;
		for(int i = 0; i < 104; ++i) {
			while(j < 10) {
				if(this.CarteJoueur.get(j) == Cartes.tabCartes.get(i)) {
					Cartes.tabCartes.remove(i);
				}
				j++;
			}
			if(j < 10 && i != 104) {
				j = 0;
			}
		}
	}
	/*
	 * Retourner le taille de la pioche total
	 */
	public int getTabCartessize() {
		return Cartes.tabCartes.size();
	}
	/*
	 * Defenir les série du jeux, en attibuant 4 cartes du jeux de cartes (64
cartes) pour chaque série
	 */
	public void defSerie() {
		for(int i = 0; i < 1; ++i) {
			Cartes.Séries1.add(Cartes.tabCartes.get(i));
			}
		for(int i = 1; i < 2; ++i) {
```

```
                Cartes.Séries2.add(Cartes.tabCartes.get(i));
            }
            for(int i = 2; i < 3; ++i) {
                Cartes.Séries3.add(Cartes.tabCartes.get(i));
            }
            for(int i = 3; i < 4; ++i) {
                Cartes.Séries4.add(Cartes.tabCartes.get(i));
            }
        }
        /*
         * Verification de la création des 4 série
         */
        public boolean SerieValide() {
            if(Cartes.Séries1.size() == 1 && Cartes.Séries2.size() == 1 &&
Cartes.Séries3.size() == 1 && Cartes.Séries4.size() == 1) {
                return true;
            }
            return false;
        }
        /*
         * Supprimer les cartes du jeux des 64 cartes identiques aux cartes des
séries
         */
        public void supprimeurs() {
            for(int i = 0; i < Cartes.tabCartes.size(); ++i) {
                if(Cartes.tabCartes.get(i) == Cartes.Séries1.get(0)) {
                    Cartes.tabCartes.remove(i);
                }
                if(Cartes.tabCartes.get(i) == Cartes.Séries2.get(0)) {
                    Cartes.tabCartes.remove(i);
                }
                if(Cartes.tabCartes.get(i) == Cartes.Séries3.get(0)) {
                    Cartes.tabCartes.remove(i);
                }
                if(Cartes.tabCartes.get(i) == Cartes.Séries4.get(0)) {
                    Cartes.tabCartes.remove(i);
                }
            }
        }
        /*
         * @param[out] : Cartes.Séries1
         * Retourner les cartes de la Série 1
         */
        public String envoie1() {
            return "- Série n° 1 : " + Cartes.Séries1;
        }
        /*
         * @param[out] : Cartes.Séries2
         * Retourner les cartes de la Série 2
         */
        public String envoie2() {
            return "- Série n° 2 : " + Cartes.Séries2;
        }
        /*
         * @param[out] : Cartes.Séries3
         * Retourner les cartes de la Série 3
         */
        public String envoie3() {
            return "- Série n° 3 : " + Cartes.Séries3;
```

```
}
/*
 * @param[out] : Cartes.Séries4
 * Retourner les cartes de la Série 4
 */
public String envoie4() {
	return "- Série n° 4 : " + Cartes.Séries4;
}
/*
 * Envoie de la Série 1
 */
public ArrayList<Integer> EnvoieReussi1() {
	return Cartes.Séries1;
}
public ArrayList<Integer> EnvoieReussi2() {
	return Cartes.Séries2;
}
public ArrayList<Integer> EnvoieReussi3() {
	return Cartes.Séries3;
}
public ArrayList<Integer> EnvoieReussi4() {
	return Cartes.Séries4;
}
/*
 * @param[out] : this.CarteJoueur
 * Retourner les cartes du joueur
 */
public String envoie5() {
	return "- Vos cartes : " + this.CarteJoueur;
}
/*
 * @param[in] : jeux
 * Le joueur joue une cartes de son deck
 * Verification si la carte joué par le joueur est presente dans son deck
 */
public boolean validejouer(int jeux) {
	this.CarteJoué = jeux;
	for(int i = 0; i < this.CarteJoueur.size(); ++i) {
		if(this.CarteJoueur.get(i) == this.CarteJoué) {
			return true;
		}
	}
	return false;
}

public String message() {
	return "Vous n'avez pas cette carte, saisissez votre choix : ";
}
/*
 * @param [out] : this.CarteJoué
 * Retourner la carte jouer par le joueur
 */
public int RetournerCarteJouée(){
	return this.CarteJoué;
}
/*
 * Supprimer la carte jouer par le joueur dans son deck
 */
public void SupprimerDeck() {
```

```
			for(int i = 0; i < this.CarteJoueur.size(); ++i) {
				if(this.CarteJoué == this.CarteJoueur.get(i)) {
					this.CarteJoueur.remove(i);
				}
			}
		}
		/*
		 * Verification si on peut poser la carte dans une des série
		 * Il faut que la cartes joué ne soient pas inférieur a la dernière carte de la série
		 */
		public boolean VerificationJeux() {
		for(int i = 0; i < Cartes.Séries1.size(); ++i) {
			if(i == Cartes.Séries1.size()-1 && this.CarteJoué > Cartes.Séries1.get(i) && Cartes.Séries1.size() < 6) {
				return true;
			}
		}
		for(int i = 0; i < Cartes.Séries2.size(); ++i) {
			if(i == Cartes.Séries2.size()-1 && this.CarteJoué > Cartes.Séries2.get(i) && Cartes.Séries2.size() < 6) {
				return true;
			}
		}
		for(int i = 0; i < Cartes.Séries3.size(); ++i) {
			if(i == Cartes.Séries3.size()-1 && this.CarteJoué > Cartes.Séries3.get(i) && Cartes.Séries3.size() < 6) {
				return true;
			}
		}
		for(int i = 0; i < Cartes.Séries4.size(); ++i) {
			if(i == Cartes.Séries4.size()-1 && this.CarteJoué > Cartes.Séries4.get(i) && Cartes.Séries4.size() < 6) {
				return true;
			}
		}
		return false;
}
		/*
		 * Calculer la distance entre la carte joué et la derniere d'une série s'il est plus grande que la carte de la série
		 */
		public void DistSerie(){
			this.dist1 = 110;
			this.dist2 = 110;
			this.dist3 = 110;
			this.dist4 = 110;

			for(int i = 0; i < Cartes.Séries1.size(); ++i) {
				if(i == Cartes.Séries1.size()-1 && this.CarteJoué > Cartes.Séries1.get(i)) {
					this.dist1 = this.CarteJoué - Cartes.Séries1.get(i);
				}
			}
			for(int i = 0; i < Cartes.Séries2.size(); ++i) {
				if(i == Cartes.Séries2.size()-1 && this.CarteJoué > Cartes.Séries2.get(i)) {
					this.dist2 = this.CarteJoué - Cartes.Séries2.get(i);
				}
```

```
			}
			for(int i = 0; i < Cartes.Séries3.size(); ++i) {
				if(i == Cartes.Séries3.size()-1 && this.CarteJoué >
Cartes.Séries3.get(i)) {
					this.dist3 = this.CarteJoué - Cartes.Séries3.get(i);
				}
			}
			for(int i = 0; i < Cartes.Séries4.size(); ++i) {
				if(i == Cartes.Séries4.size()-1 && this.CarteJoué >
Cartes.Séries4.get(i)) {
					this.dist4 = this.CarteJoué - Cartes.Séries4.get(i);
				}
			}
		}
		/*
		 * Verification de la bonne distance
		 */
		public boolean verifDistSeries() {
			for(int i = Cartes.Séries1.size()-1; i < Cartes.Séries1.size(); ++i)
{
				for(int a = Cartes.Séries2.size()-1; i < Cartes.Séries2.size();
++a) {
					for(int b = Cartes.Séries3.size()-1; i <
Cartes.Séries3.size(); ++b) {
						for(int c = Cartes.Séries4.size()-1; i <
Cartes.Séries4.size(); ++c) {


				if(dist1 == this.CarteJoué - Cartes.Séries1.get(i) && dist2 ==
this.CarteJoué - Cartes.Séries2.get(a) && dist3 == this.CarteJoué -
Cartes.Séries3.get(b) && dist4 == this.CarteJoué - Cartes.Séries1.get(c)) {
					return true;
				}
						}
					}
				}
			}
			return false;
		}

		/*
		 * Ajouter la carte joué par le joueur dans la série dans laquelle la
distance est la plus faible
		 */
		public void reussi(){
				if(this.dist1 < this.dist2 && this.dist1 < this.dist3 &&
this.dist1 < this.dist4) {
					Cartes.Séries1.add(this.CarteJoué);
				}
				else if(this.dist2 < this.dist1 && this.dist2 < this.dist3 &&
this.dist2 < this.dist4) {
					Cartes.Séries2.add(this.CarteJoué);
				}
				else if(this.dist3 < this.dist1 && this.dist3 < this.dist2 &&
this.dist3 < this.dist4) {
					Cartes.Séries3.add(this.CarteJoué);
				}
				else if(this.dist4 < this.dist1 && this.dist4 < this.dist2 &&
this.dist4 < this.dist3) {
```

```
                        Cartes.Séries4.add(this.CarteJoué);
                  }
}
      /*
       * Verififcation si la carte joué par le joueur est inférieur a la dernière
carte d'une série
       */
      public boolean inférieur() {
            int j = 0;
            for(int i = 0; i < Cartes.Séries1.size(); ++i) {
                  if(i == Cartes.Séries1.size()-1 && this.CarteJoué <
Cartes.Séries1.get(i)) {
                        j = 1;
                  }
            }
            for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                  if(i == Cartes.Séries2.size()-1 && this.CarteJoué <
Cartes.Séries2.get(i)) {
                        j = 1;
                  }
            }
            for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                  if(i == Cartes.Séries3.size()-1 && this.CarteJoué <
Cartes.Séries3.get(i)) {
                        j = 1;
                  }
            }
            for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                  if(i == Cartes.Séries4.size()-1 && this.CarteJoué <
Cartes.Séries4.get(i)) {
                        j = 1;
                  }
            }
            if (j == 1) {
                  return true;
            }
            return false;
      }
      /*
       * @param[in] : série
       * Le joueur choisir la série qu'il va récuperer, elle doit etre supérieur a
0 et inférieur a 5
       */
      public boolean inferieur(int série) {
            if(série > 0 && série < 5) {
                  this.SérieChoisie = série;
                  return true;
            }
            return false;
      }
      /*
       * Initialiser le nombre total de tete de boeufs a 0
       */
      public void initialisetotalteteboeufs() {
            this.teteboeufstot = 0;
      }
      /*
       * Calculer le nombre de tete de boeufs recuperer en fonction de la série
choisie
```

```
        * Comptez le nombre de cartes dans chaque cas
        * Multiplier le nombre de cartes par leur nombre de tête de boeufs (exemple
: divisible par 10 -> 3 tete boueufs
        *  Ajouter a nombre de tete boeufs total, le nombre de tête de boeufs
recuperer
        */
       public void nbteteboeufs() {
              int m = 0;
              int l = 0;
              int s = 0;
              int d = 0;
              int v = 0;
              if(this.SérieChoisie == 1) {
                    for(int i = 0; i < Cartes.Séries1.size(); ++i) {
                          if(Cartes.Séries1.get(i)%5==0 &&
Cartes.Séries1.get(i)%10!=0 && Cartes.Séries1.get(i) != 55) {
                                m++;
                          }
                          else if(Cartes.Séries1.get(i)%10==0) {
                                l++;
                          }
                          else if(Cartes.Séries1.get(i) == 11 ||
Cartes.Séries1.get(i) == 22 || Cartes.Séries1.get(i) == 33 ||
Cartes.Séries1.get(i) == 44 || Cartes.Séries1.get(i) == 66 ||
Cartes.Séries1.get(i) == 77 || Cartes.Séries1.get(i) == 88 ||
Cartes.Séries1.get(i) == 99) {
                                s++;
                          }
                          else if(Cartes.Séries1.get(i) == 55) {
                                d++;
                          }
                          else {
                                v++;
                          }
                    }
              }
              if(this.SérieChoisie == 2) {
                    for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                          if(Cartes.Séries2.get(i)%5==0 &&
Cartes.Séries2.get(i)%10!=0 && Cartes.Séries2.get(i) != 55) {
                                m++;
                          }
                          else if(Cartes.Séries2.get(i)%10==0) {
                                l++;
                          }
                          else if(Cartes.Séries2.get(i) == 11 ||
Cartes.Séries2.get(i) == 22 || Cartes.Séries2.get(i) == 33 ||
Cartes.Séries2.get(i) == 44 || Cartes.Séries2.get(i) == 66 ||
Cartes.Séries2.get(i) == 77 || Cartes.Séries2.get(i) == 88 ||
Cartes.Séries2.get(i) == 99) {
                                s++;
                          }
                          else if(Cartes.Séries2.get(i) == 55) {
                                d++;
                          }
                          else {
                                v++;
                          }
                    }
```

```
            }
            if(this.SérieChoisie == 3) {
                  for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                        if(Cartes.Séries3.get(i)%5==0 &&
Cartes.Séries3.get(i)%10!=0 && Cartes.Séries3.get(i) != 55) {
                              m++;
                        }
                        else if(Cartes.Séries3.get(i)%10==0) {
                              l++;
                        }
                        else if(Cartes.Séries3.get(i) == 11 ||
Cartes.Séries3.get(i) == 22 || Cartes.Séries3.get(i) == 33 ||
Cartes.Séries3.get(i) == 44 || Cartes.Séries3.get(i) == 66 ||
Cartes.Séries3.get(i) == 77 || Cartes.Séries3.get(i) == 88 ||
Cartes.Séries3.get(i) == 99) {
                              s++;
                        }
                        else if(Cartes.Séries3.get(i) == 55) {
                              d++;
                        }
                        else {
                              v++;
                        }
                  }
            }
            if(this.SérieChoisie == 4) {
                  for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                        if(Cartes.Séries4.get(i)%5==0 &&
Cartes.Séries4.get(i)%10!=0 && Cartes.Séries4.get(i) != 55) {
                              m++;
                        }
                        else if(Cartes.Séries4.get(i)%10==0) {
                              l++;
                        }
                        else if(Cartes.Séries4.get(i) == 11 ||
Cartes.Séries4.get(i) == 22 || Cartes.Séries4.get(i) == 33 ||
Cartes.Séries4.get(i) == 44 || Cartes.Séries4.get(i) == 66 ||
Cartes.Séries4.get(i) == 77 || Cartes.Séries4.get(i) == 88 ||
Cartes.Séries4.get(i) == 99) {
                              s++;
                        }
                        else if(Cartes.Séries4.get(i) == 55) {
                              d++;
                        }
                        else {
                              v++;
                        }
                  }
            }
            this.teteboeufsrecup = m*2 + l*3 + s*5 + d*7 + v*1;
            this.teteboeufstot += this.teteboeufsrecup;
      }
      /*
       * Le joueur doit récuperer toutes les cartes de la série choisis
       * Le joueur depose la carte dont il a joué
       */
      public void InférieurCartes() {
            if(this.SérieChoisie == 1) {
                  for(int i = 0; i < Cartes.Séries1.size(); ++i) {
```

```
                        this.CarteJoueur.add(Cartes.Séries1.get(i));
                }
                Cartes.Séries1.add(this.CarteJoué);
        }
        if(this.SérieChoisie == 2) {
                for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries2.get(i));
                }
                Cartes.Séries2.add(this.CarteJoué);
        }
        if(this.SérieChoisie == 3) {
                for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries3.get(i));
                }
                Cartes.Séries3.add(this.CarteJoué);
        }
        if(this.SérieChoisie == 4) {
                for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries4.get(i));
                }
                Cartes.Séries4.add(this.CarteJoué);
        }
    }
    /*
     * Verification de la récuperation total des cartes par le joueur
     */
    public boolean tailleCarteJoueur(){
        if(this.SérieChoisie == 1) {
                if(this.CarteJoueur.size() == this.CarteJoueur.size() +
Cartes.Séries1.size()) {
                        return true;
                }
        }
        if(this.SérieChoisie == 2) {
                if(this.CarteJoueur.size() == this.CarteJoueur.size() +
Cartes.Séries2.size()) {
                        return true;
                }
        }
        if(this.SérieChoisie == 3) {
                if(this.CarteJoueur.size() == this.CarteJoueur.size() +
Cartes.Séries3.size()) {
                        return true;
                }
        }
        if(this.SérieChoisie == 4) {
                if(this.CarteJoueur.size() == this.CarteJoueur.size() +
Cartes.Séries4.size()) {
                        return true;
                }
        }
        return false;
        }
    /*
     * Determiner si le nombre de cartes est supérieur a 1
     *
     */
    public boolean normaliser() {
        if(this.SérieChoisie==1) {
```

```
                if(Cartes.Séries1.size() > 1) {
                        return true;
                }
        }
        if(this.SérieChoisie==2) {
                if(Cartes.Séries2.size() > 1) {
                        return true;
                }
        }
        if(this.SérieChoisie==3) {
                if(Cartes.Séries3.size() > 1) {
                        return true;
                }
        }
        if(this.SérieChoisie==4) {
                if(Cartes.Séries4.size() > 1) {
                        return true;
                }
        }
        return false;
}
/*
 * Supprimer les cartes de la série choisi par le joueur
 * Supprimer toutes les cartes de la choisi mis a part la carte joué par le
joueur
 */
public void supprcartes() {
        if(this.SérieChoisie == 1) {
                for(int i = 0; i < Cartes.Séries1.size(); ++i) {
                        if(Cartes.Séries1.get(i) != this.CarteJoué) {
                                Cartes.Séries1.remove(i);
                        }
                }
        }
        if(this.SérieChoisie == 2) {
                for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                        if(Cartes.Séries2.get(i) != this.CarteJoué) {
                                Cartes.Séries2.remove(i);
                        }
                }
        }
        if(this.SérieChoisie == 3) {
                for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                        if(Cartes.Séries3.get(i) != this.CarteJoué) {
                                Cartes.Séries3.remove(i);
                        }
                }
        }
        if(this.SérieChoisie == 4) {
                for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                        if(Cartes.Séries4.get(i) != this.CarteJoué) {
                                Cartes.Séries4.remove(i);
                        }
                }
        }
}
/*
 * Determiner si le nombre de tete boeufs recupérer est supérieur a 0
 */
```

```
	public boolean recup() {
		if(this.teteboeufsrecup > 0) {
			return true;
		}
		return false;
	}
	/*
	 * @param[out] : this.teteboeufsrecup
	 * retourner le nombre de tete boeufs recuperer
	 */
	public int teteboeufs() {
		return this.teteboeufsrecup;
	}
	/*
	 * Mettre a 0 le nombre de tete de boeufs
	 */
	public void RemisaZero() {
		this.teteboeufsrecup = 0;
	}
	/*
	 * Si la carte joué par un joueur doit etre dans une série qui a supérieur
ou egal a 5 retourner faus sinon vrai
	 */
	public boolean depassement() {
		for(int i = 0; i < Cartes.Séries1.size(); ++i) {
			if(i >= 4 && this.dist1 < this.dist2 && this.dist1 < this.dist3
&& this.dist1 < this.dist4) {
				return false;
			}
		}
		for(int i = 0; i < Cartes.Séries2.size(); ++i) {
			if(i >= 4 && this.dist2 < this.dist1 && this.dist2 < this.dist3
&& this.dist2 < this.dist4) {
				return false;
			}

		}
		for(int i = 0; i < Cartes.Séries3.size(); ++i) {
			if(i >= 4 && this.dist3 < this.dist1 && this.dist3 < this.dist2
&& this.dist3 < this.dist4) {
				return false;
			}
		}
		for(int i = 0; i < Cartes.Séries4.size(); ++i) {
			if(i >= 4 && this.dist4 < this.dist1 && this.dist4 < this.dist2
&& this.dist4 < this.dist3) {
				return false;
			}
		}
		return true;
	}
	/*
	 * Defenir quelle serie doit être recuperer par le joueur selon la carte
joué et si la taille de la série est supérieur ou egal a 5
	 */
	public void RecupererSerie() {
		this.Avertissement = 0;
		for(int i = 0; i < Cartes.Séries1.size(); ++i) {
```

```
                  if(i >= 4 && this.dist1 < this.dist2 && this.dist1 < this.dist3
&& this.dist1 < this.dist4) {
                        this.Avertissement = 1;
                  }
            }
            for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                  if(i >= 4 && this.dist2 < this.dist1 && this.dist2 < this.dist3
&& this.dist2 < this.dist4) {
                        this.Avertissement = 2;
                  }

            }
            for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                  if(i >= 4 && this.dist3 < this.dist1 && this.dist3 < this.dist2
&& this.dist3 < this.dist4) {
                        this.Avertissement = 3;
                  }
            }
            for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                  if(i >= 4 && this.dist4 < this.dist1 && this.dist4 < this.dist2
&& this.dist4 < this.dist3) {
                        this.Avertissement = 4;
                  }
            }
      }

      /*
       * Calculer le nombre de tete de boeufs récuperer par le joueur
       * Comptez le nombre de cartes dans chaque cas
       * Multiplier le nombre de cartes par leur nombre de tête de boeufs (exemple
: divisible par 10 -> 3 tete boueufs
       *  Ajouter a nombre de tete boeufs total, le nombre de tête de boeufs
recuperer
       */
public void nbteteboeufDeppassement() {
      int m = 0;
      int l = 0;
      int s = 0;
      int d = 0;
      int v = 0;
      if(this.Avertissement == 1) {
            for(int i = 0; i < Cartes.Séries1.size(); ++i) {
                  if(Cartes.Séries1.get(i)%5==0 && Cartes.Séries1.get(i)%10!=0 &&
Cartes.Séries1.get(i) != 55) {
                        m++;
                  }
                  else if(Cartes.Séries1.get(i)%10==0) {
                        l++;
                  }
                  else if(Cartes.Séries1.get(i) == 11 || Cartes.Séries1.get(i) ==
22 || Cartes.Séries1.get(i) == 33 || Cartes.Séries1.get(i) == 44 ||
Cartes.Séries1.get(i) == 66 || Cartes.Séries1.get(i) == 77 ||
Cartes.Séries1.get(i) == 88 || Cartes.Séries1.get(i) == 99) {
                        s++;
                  }
                  else if(Cartes.Séries1.get(i) == 55) {
                        d++;
                  }
                  else {
```

```
                        v++;
                    }
                }
            }
            if(this.Avertissement == 2) {
                for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                    if(Cartes.Séries2.get(i)%5==0 && Cartes.Séries2.get(i)%10!=0 &&
Cartes.Séries2.get(i) != 55) {
                        m++;
                    }
                    else if(Cartes.Séries2.get(i)%10==0) {
                        l++;
                    }
                    else if(Cartes.Séries2.get(i) == 11 || Cartes.Séries2.get(i) ==
22 || Cartes.Séries2.get(i) == 33 || Cartes.Séries2.get(i) == 44 ||
Cartes.Séries2.get(i) == 66 || Cartes.Séries2.get(i) == 77 ||
Cartes.Séries2.get(i) == 88 || Cartes.Séries2.get(i) == 99) {
                        s++;
                    }
                    else if(Cartes.Séries2.get(i) == 55) {
                        d++;
                    }
                    else {
                        v++;
                    }
                }
            }
            if(this.Avertissement == 3) {
                for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                    if(Cartes.Séries3.get(i)%5==0 && Cartes.Séries3.get(i)%10!=0 &&
Cartes.Séries3.get(i) != 55) {
                        m++;
                    }
                    else if(Cartes.Séries3.get(i)%10==0) {
                        l++;
                    }
                    else if(Cartes.Séries3.get(i) == 11 || Cartes.Séries3.get(i) ==
22 || Cartes.Séries3.get(i) == 33 || Cartes.Séries3.get(i) == 44 ||
Cartes.Séries3.get(i) == 66 || Cartes.Séries3.get(i) == 77 ||
Cartes.Séries3.get(i) == 88 || Cartes.Séries3.get(i) == 99) {
                        s++;
                    }
                    else if(Cartes.Séries3.get(i) == 55) {
                        d++;
                    }
                    else {
                        v++;
                    }
                }
            }
            if(this.Avertissement == 4) {
                for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                    if(Cartes.Séries4.get(i)%5==0 && Cartes.Séries4.get(i)%10!=0 &&
Cartes.Séries4.get(i) != 55) {
                        m++;
                    }
                    else if(Cartes.Séries4.get(i)%10==0) {
                        l++;
                    }
                    }
```

```
                        else if(Cartes.Séries4.get(i) == 11 || Cartes.Séries4.get(i) ==
22 || Cartes.Séries4.get(i) == 33 || Cartes.Séries4.get(i) == 44 ||
Cartes.Séries4.get(i) == 66 || Cartes.Séries4.get(i) == 77 ||
Cartes.Séries4.get(i) == 88 || Cartes.Séries4.get(i) == 99) {
                              s++;
                        }
                        else if(Cartes.Séries4.get(i) == 55) {
                              d++;
                        }
                        else {
                              v++;
                        }
                  }
            }
            this.teteboeufsrecup = m*2 + l*3 + s*5 + d*7 + v*1;
            this.teteboeufstot += this.teteboeufsrecup;
      }
      /*
       * Le joueur récuperer toute la série
       * la serie récupere la carté joué par le joueur
       */
      public void DeppasementCartes() {
            if(this.Avertissement == 1) {
                  for(int i = 0; i < Cartes.Séries1.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries1.get(i));
                  }
                  Cartes.Séries1.add(this.CarteJoué);
            }
            if(this.Avertissement == 2) {
                  for(int i = 0; i < Cartes.Séries2.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries2.get(i));
                  }
                  Cartes.Séries2.add(this.CarteJoué);
            }
            if(this.Avertissement == 3) {
                  for(int i = 0; i < Cartes.Séries3.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries3.get(i));
                  }
                  Cartes.Séries3.add(this.CarteJoué);
            }
            if(this.Avertissement == 4) {
                  for(int i = 0; i < Cartes.Séries4.size(); ++i) {
                        this.CarteJoueur.add(Cartes.Séries4.get(i));
                  }
                  Cartes.Séries4.add(this.CarteJoué);
            }
      }
      /*
       * Si la taille de la série est superieur à 1 retourner vrai sinon faux
       */
      public boolean DepassementChoisi() {
            if(this.Avertissement == 1) {
                  if(Cartes.Séries1.size() > 1) {
                        return true;
                  }
            }
            if(this.Avertissement == 2) {
                  if(Cartes.Séries2.size() > 1) {
                        return true;
```

```
			}
		}
		if(this.Avertissement == 3) {
			if(Cartes.Séries3.size() > 1) {
				return true;
			}
		}
		if(this.Avertissement == 4) {
			if(Cartes.Séries4.size() > 1) {
				return true;
			}
		}
		return false;
}
/*
 * Supprimer les cartes de la série tant que la taille est égal a 1 et supprimer
toutes les cartes sauf la carte joué par le joueur
 */
public void supprcartesdepassement() {
		if(this.Avertissement == 1) {
			for(int i = 0; i < Cartes.Séries1.size(); ++i) {
				if(Cartes.Séries1.get(i) != this.CarteJoué) {
					Cartes.Séries1.remove(i);
				}
			}
		}
		if(this.Avertissement == 2) {
			for(int i = 0; i < Cartes.Séries2.size(); ++i) {
				if(Cartes.Séries2.get(i) != this.CarteJoué) {
					Cartes.Séries2.remove(i);
				}
			}
		}
		if(this.Avertissement == 3) {
			for(int i = 0; i < Cartes.Séries3.size(); ++i) {
				if(Cartes.Séries3.get(i) != this.CarteJoué) {
					Cartes.Séries3.remove(i);
				}
			}
		}
		if(this.Avertissement == 4) {
			for(int i = 0; i < Cartes.Séries4.size(); ++i) {
				if(Cartes.Séries4.get(i) != this.CarteJoué) {
					Cartes.Séries4.remove(i);
				}
			}
		}
}
/*
 * Retourner le nombre de tête de boeufs total
 */
public int nbteteboeufstot() {
		return this.teteboeufstot;
}
}
```

Main.java

```
package JeuxCartes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Scanner;

import static JeuxCartes.Console.clearScreen;
import static JeuxCartes.Console.pause;

public class Main {

	public static void main(String[] args) {
		try {
		Scanner sc = new Scanner(new
FileInputStream("C:\\ficSAE2.02/Config.txt"));//lecture des entré dans un fichier
		@SuppressWarnings("resource")
		Scanner sc1 = new Scanner(System.in);//Initialiser le Scanner
		Joueur j1 = new Joueur();//Création du joueur 1
		Joueur j2 = new Joueur();//Création du joueur 2
		Joueur j3 = new Joueur();//Création du joueur 3
		Joueur j4 = new Joueur();//Création du joueur 4
		int m = 0;// initialiser m a 0
		j1.initialise(sc.next());//Initialiser le prenom du joueur 1
		j2.initialise(sc.next());//Initialiser le prenom du joueur 2
		j3.initialise(sc.next());//Initialiser le prenom du joueur 3
		j4.initialise(sc.next());//Initialiser le prenom du joueur 4
		System.out.println("Les 4 joueurs sont " + j1.envoie() + ", " +
j2.envoie() + ", " + j3.envoie() + " et " + j4.envoie() + ". Merci de jouer à 6
qui prend !" );
		Cartes c1 = new Cartes();//Les cartes du joueur 1
		Cartes c2 = new Cartes();//Les cartes du joueur 2
		Cartes c3 = new Cartes();//Les cartes du joueur 3
		Cartes c4 = new Cartes();//Les cartes du joueur 4
		Cartes s = new Cartes();//les séries
		if(c1.EstIlinitialisé() == false) {//Si la taille des cartes n'est
pas égal a 10 alors
			c1.initialiserCartes();//initialiser les cartes a 104
			c1.mélange();//Mélanger les 104 cartes
		}
		if(c1.estDistribuer() == false) {//Si les cartes du joueur 1 n'ont
pas été distribué alors
			c1.distribuerCartes();// Distribué les 10 cartes du joueur1
			c1.supprimer();//Supprimer les cartes distibué au joueur 1 dans
le deck initiales (les 104 cartes)
		}
		if(c2.estDistribuer() == false) {//Si les cartes du joueur 2 n'ont
pas été distribué alors
			c2.distribuerCartes();// Distribué les 10 cartes du joueur 2
			c2.supprimer();//Supprimer les cartes distibué au joueur 2 dans
le deck initiales (les 104 cartes)
		}
		if(c3.estDistribuer() == false) {//Si les cartes du joueur 3 n'ont
pas été distribué alors
			c3.distribuerCartes();// Distribué les 10 cartes du joueur 3
			c3.supprimer();//Supprimer les cartes distibué au joueur 3 dans
le deck initiales (les 104 cartes)
```

```
            }
            if(c4.estDistribuer() == false) {//Si les cartes du joueur 4 n'ont
pas été distribué alors
                  c4.distribuerCartes();// Distribué les 10 cartes du joueur 4
                  c4.supprimer();//Supprimer les cartes distibué au joueur 4 dans
le deck initiales (les 104 cartes)
            }
            c1.trier();//Trier les cartes du joueur 1
            c2.trier();//Trier les cartes du joueur 2
            c3.trier();//Trier les cartes du joueur 3
            c4.trier();//Trier les cartes du joueur 4
            s.defSerie();//Definir les prémier cartes des 4 séries
            s.supprimeurs();//Supprimer les cartes donné aux 4 série dans le deck
initiales(des 104 cartes)
            c1.initialisetotalteteboeufs();//Initialiser le nombre de tete de
boeufs total a 0 pour le joueur 1
            c2.initialisetotalteteboeufs();//Initialiser le nombre de tete de
boeufs total a 0 pour le joueur 2
            c3.initialisetotalteteboeufs();//Initialiser le nombre de tete de
boeufs total a 0 pour le joueur 3
            c4.initialisetotalteteboeufs();//Initialiser le nombre de tete de
boeufs total a 0 pour le joueur 4
            while(m < 3) {// tant que m est inférieur a 10, les joueurs doivent
continuer a jouer
            System.out.println("A " + j1.envoie() + " de jouer.");
            pause();//appuyer une touche pour continuer
            System.out.println(s.envoie1());//affichage de la série 1
            System.out.println(s.envoie2());//affichage de la série 2
            System.out.println(s.envoie3());//affichage de la série 3
            System.out.println(s.envoie4());//affichage de la série 4
            System.out.println(c1.envoie5());//Affichage des cartes du jouer 1
            System.out.println("Saisissez votre choix :");
            while(c1.validejouer(sc1.nextInt()) == false) {//le jouer 1 joue une
cartes si il n'a pas cette cartes alors
                  System.out.println(c1.message());//il affiche ce message
d'erreur, puis  le joueur 1 recommence jusqu'a il pose une cartes qu'il a dans son
jeux de cartes
            }
            c1.SupprimerDeck();//Supprimer la carte joué par le joueur 1 sur son
deck
            clearScreen();
            System.out.println("A " + j2.envoie() + " de jouer.");
            pause();//appuyer une touche pour continuer
            System.out.println(s.envoie1());//affichage de la série 1
            System.out.println(s.envoie2());//affichage de la série 2
            System.out.println(s.envoie3());//affichage de la série 3
            System.out.println(s.envoie4());//affichage de la série 4
            System.out.println(c2.envoie5());//Affichage des cartes du jouer 2
            System.out.println("Saisissez votre choix :");
            while(c2.validejouer(sc1.nextInt()) == false) {//le jouer 2 joue une
cartes si il n'a pas cette cartes alors
                  System.out.println(c2.message());//il affiche ce message
d'erreur, puis  le joueur 2 recommence jusqu'a il pose une cartes qu'il a dans son
jeux de cartes
            }
            c2.SupprimerDeck();//Supprimer la carte joué par le joueur 2 sur son
deck
            clearScreen();
            System.out.println("A " + j3.envoie() + " de jouer.");
```

```
            pause();//appuyer une touche pour continuer
            System.out.println(s.envoie1());//affichage de la série 1
            System.out.println(s.envoie2());//affichage de la série 2
            System.out.println(s.envoie3());//affichage de la série 3
            System.out.println(s.envoie4());//affichage de la série 4
            System.out.println(c3.envoie5());//Affichage des cartes du jouer 3
            System.out.println("Saisissez votre choix :");
            while(c3.validejouer(sc1.nextInt()) == false) {//le jouer 3 joue une
cartes si il n'a pas cette cartes alors
                  System.out.println(c3.message());//il affiche ce message
d'erreur, puis  le joueur 3 recommence jusqu'a il pose une cartes qu'il a dans son
jeux de cartes
            }
            clearScreen();
            c3.SupprimerDeck();//Supprimer la carte joué par le joueur 3 sur son
deck
            System.out.println("A " + j4.envoie() + " de jouer.");
            pause();//appuyer une touche pour continuer
            System.out.println(s.envoie1());//affichage de la série 1
            System.out.println(s.envoie2());//affichage de la série 2
            System.out.println(s.envoie3());//affichage de la série 3
            System.out.println(s.envoie4());//affichage de la série 4
            System.out.println(c4.envoie5());//Affichage des cartes du jouer 4
            System.out.println("Saisissez votre choix :");
            while(c4.validejouer(sc1.nextInt()) == false) {//le jouer 4 joue une
cartes si il n'a pas cette cartes alors
                  System.out.println(c4.message());//il affiche ce message
d'erreur, puis  le joueur 4 recommence jusqu'a il pose une cartes qu'il a dans son
jeux de cartes
            }
            c4.SupprimerDeck();//Supprimer la carte joué par le joueur 4 sur son
deck
            clearScreen();
            c1.DistSerie();//Definir la distance entre la carte joué par le
joueur 1 et la dernière cartes des 4 série
            if(c1.VerificationJeux() == true && c1.depassement() == true) {//Si
la carte joué par le joueur 1 est supérieur a la cartes de la série dont il doit
être deposé et la taille de série dont il doit être deposé doit être infetrieur a
5
                  c1.reussi();//Poser la carte jouer par le joueur 1 dans la
série
            }
            else if(c1.inférieur() == true && c1.VerificationJeux() == false)
{//Si la cartes joué par le joueur est inférieur a toutes les dernieres cartes des
séries
                  System.out.println("Les cartes " + c1.RetournerCarteJouée() +
" (" + j1.envoie() + "), " + c2.RetournerCarteJouée() +  " (" + j2.envoie() + "), "
+ c3.RetournerCarteJouée() + " (" + j3.envoie() + ") et " +
c4.RetournerCarteJouée() + " (" + j4.envoie() + ") ont été posées" );
                  System.out.println("Pour poser la carte " +
c1.RetournerCarteJouée() + ", " + j1.envoie() + " doit choisir la série qu'il va
ramasser");
                  System.out.println(c1.envoie1());//Affichage série 1
                  System.out.println(c2.envoie2());//Affichage série 2
                  System.out.println(c3.envoie3());//Affichage série 3
                  System.out.println(c4.envoie4());//Affichage série 4
                  System.out.println("Saisissez votre choix : ");
                  while(c1.inferieur(sc1.nextInt()) == false){//le joueur 1 doit
choisir la série qu'il veut récuperer
```

```
                        System.out.println("Ce n'est pas une série valide,
saisissez votre choix : ");//Si la série choisi par le joueur 1 est incorrect
alors le joueur 1 doit choisir une série jusqu'a elle soit valide
                    }
                    c1.nbteteboeufs();//Calculer le nombre de tete de boeufs que
doit récupere le joueur 1
                    c1.InférieurCartes();//ajouter les cartes de la série demander
par le joueur 1 dans son deck puis ajouté la carte joué par le joueur 1 dans la
série choisie par le jouer
                    c1.trier();//Trier les cartes du joueur 1
                    while(c1.normaliser() == true) {// si la taille de série
choisie par le joueur 1 est supérieur a 1 alors
                        c1.supprcartes();//Supprimmer toutes les cartes de cette
série sauf la carte joué par le joueur
                    }
                }
                else if(c1.depassement() == false) {//Si la carte joué par le joueur
1 doit être deposé dans une série dont la taille est supérieur ou égal a 5 alors
                    c1.RecupererSerie();//Défenir quelle série le joueur 1 doit
récupérer
                    c1.nbteteboeufDeppassement();//Calculer le nombre de tête de
boeufs le joueur 1 doit récupérer
                    c1.DeppasementCartes();//Ajouter toutes les cartes de la série
dans le jeux de cartes du joueur 1 puis ajouter la carte joué par le joueur 1 dans
la série
                    c1.trier();//Trier les cartes du joueur 1
                    while(c1.DepassementChoisi() == true) {//Si la taille de la
série recuperer par le joueur est supérieur a 1 alors
                        c1.supprcartesdepassement();//Supprimer toutes les
cartes sauf la cartes joué par le joueur 1
                    }
                }
                c2.DistSerie();//Definir la distance entre la carte joué par le
joueur 2 et la dernière cartes des 4 série
                if(c2.VerificationJeux() == true && c2.depassement() == true) {//Si
la carte joué par le joueur 1 est supérieur a la cartes de la série dont il doit
être deposé et la taille de série dont il doit être deposé doit être infetrieur a
5
                    c2.reussi();//Poser la carte jouer par le joueur 2 dans la
série
                }
                else if(c2.inférieur() == true && c2.VerificationJeux() ==
false){//Si la cartes joué par le joueur 2 est inférieur a toutes les dernieres
cartes des 4 séries
                    System.out.println("Les cartes " + c1.RetournerCarteJouée() + "
(" + j1.envoie() + "), " + c2.RetournerCarteJouée() +  " (" + j2.envoie() + "), "
+ c3.RetournerCarteJouée() + " (" + j3.envoie() + ") et " +
c4.RetournerCarteJouée() + " (" + j4.envoie() + ") ont été posées" );
                    System.out.println("Pour poser la carte " +
c2.RetournerCarteJouée() + ", " + j2.envoie() + " doit choisir la série qu'il va
ramasser");
                    System.out.println(c1.envoie1());//affichage de la série 1
                    System.out.println(c2.envoie2());//affichage de la série 2
                    System.out.println(c3.envoie3());//affichage de la série 3
                    System.out.println(c4.envoie4());//affichage de la série 4
                    System.out.println("Saisissez votre choix : ");
                    while(c2.inferieur(sc1.nextInt()) == false){//le joueur 2 doit
choisir la série qu'il veut récuperer
```

```
                        System.out.println("Ce n'est pas une série valide,
saisissez votre choix : ");//Si la série choisi par le joueur 2 est incorrect
alors le joueur 1 doit choisir une série jusqu'a elle soit valide
                    }
                    c2.nbteteboeufs();//Calculer le nombre de tete de boeufs que
doit récupere le joueur 2
                    c2.InférieurCartes();//ajouter les cartes de la série demander
par le joueur 2 dans son deck puis ajouté la carte joué par le joueur 2 dans la
série choisie par le jouer
                    c2.trier();//Trier les cartes du joueur 2
                    while(c2.normaliser() == true) {// si la taille de série
choisie par le joueur 2 est supérieur a 1 alors
                        c2.supprcartes();//Supprimer toutes les cartes sauf la
cartes joué par le joueur 2
                    }
                }
                else if(c2.depassement() == false) {//Si la carte joué par le joueur
2 doit être deposé dans une série dont la taille est supérieur ou égal a 5 alors
                    c2.RecupererSerie();//Défenir quelle série le joueur 2 doit
récupérer
                    c2.nbteteboeufDeppassement();//Calculer le nombre de tête de
boeufs le joueur 2 doit récupérer
                    c2.DeppasementCartes();//Ajouter toutes les cartes de la série
dans le jeux de cartes du joueur 1 puis ajouter la carte joué par le joueur 1 dans
la série
                    c2.trier();//Trier les cartes du joueur 2
                    while(c2.DepassementChoisi() == true) {
                        c2.supprcartesdepassement();//Supprimer toutes les
cartes sauf la cartes joué par le joueur 2
                    }
                }
                c3.DistSerie();//Distance entre la carte joué par le joueur 3 et
derniere cartes de chaques séries
                if(c3.VerificationJeux() == true && c3.depassement() == true) {//Si
la cartes joué par le joueur 3 est inférieur a toutes les dernieres cartes des 4
séries
                    c3.reussi();//Poser la carte jouer par le joueur 3 dans la
série
                }
                else if(c3.inférieur() == true && c3.VerificationJeux() == false) {
                    System.out.println("Les cartes " + c1.RetournerCarteJouée() + "
(" + j1.envoie() + "), " + c2.RetournerCarteJouée() +  " (" + j2.envoie() + "), "
+ c3.RetournerCarteJouée() + " (" + j3.envoie() + ") et " +
c4.RetournerCarteJouée() + " (" + j4.envoie() + ") ont été posées" );
                    System.out.println("Pour poser la carte " +
c3.RetournerCarteJouée() + ", " + j3.envoie() + " doit choisir la série qu'il va
ramasser");
                    System.out.println(c1.envoie1());//affichage de la série 1
                    System.out.println(c2.envoie2());//affichage de la série 2
                    System.out.println(c3.envoie3());//affichage de la série 3
                    System.out.println(c4.envoie4());////affichage de la série 4
                    System.out.println("Saisissez votre choix : ");
                    while(c3.inferieur(sc1.nextInt()) == false){//le joueur 3 doit
choisir la série qu'il veut récuperer
                        System.out.println("Ce n'est pas une série valide,
saisissez votre choix : ");//Si la série choisi par le joueur 3 est incorrect
alors le joueur 3 doit choisir une série jusqu'a elle soit valide
                    }
```

```
                    c3.nbteteboeufs();//Calculer le nombre de tete de boeufs que
doit récupere le joueur 3
                    c3.InférieurCartes();//ajouter les cartes de la série demander
par le joueur 3 dans son deck puis ajouté la carte joué par le joueur 3 dans la
série choisie par le jouer
                    c3.trier();//Trier les cartes du joueur 3
                    while(c3.normaliser() == true) {// si la taille de série
choisie par le joueur 3 est supérieur a 1 alors
                          c3.supprcartes();//Supprimer toutes les cartes sauf la
cartes joué par le joueur 3
                    }
             }
             else if(c3.depassement() == false) {//Si la carte joué par le joueur
3 doit être deposé dans une série dont la taille est supérieur ou égal a 5 alors
                    c3.RecupererSerie();//Défenir quelle série le joueur 3 doit
récupérer
                    c3.nbteteboeufDeppassement();//Calculer le nombre de tête de
boeufs le joueur 3 doit récupérer
                    c3.DeppasementCartes();//Ajouter toutes les cartes de la série
dans le jeux de cartes du joueur 3 puis ajouter la carte joué par le joueur 3 dans
la série
                    c3.trier();//Trier les cartes du joueur 3
                    while(c3.DepassementChoisi() == true) {
                          c3.supprcartesdepassement();//Supprimer toutes les
cartes sauf la cartes joué par le joueur 3
                    }
             }
             c4.DistSerie();//Distance entre la carte joué par le joueur 4 et la
dernière cartes de chaque série
             if(c4.VerificationJeux() == true && c4.depassement() == true) {//Si
la cartes joué par le joueur 2 est inférieur a toutes les dernieres cartes des 4
séries
                    c4.reussi();//Poser la carte jouer par le joueur 4 dans la
série
             }
             else if(c4.inférieur() == true && c4.VerificationJeux() == false) {
                    System.out.println("Les cartes " + c1.RetournerCarteJouée() + "
(" + j1.envoie() + "), " + c2.RetournerCarteJouée() +  " (" + j2.envoie() + "), "
+ c3.RetournerCarteJouée() + " (" + j3.envoie() + ") et " +
c4.RetournerCarteJouée() + " (" + j4.envoie() + ") ont été posées" );
                    System.out.println("Pour poser la carte " +
c4.RetournerCarteJouée() + ", " + j4.envoie() + " doit choisir la série qu'il va
ramasser");
                    System.out.println(c1.envoie1());//Affichage Série 1
                    System.out.println(c2.envoie2());//Affichage Série 2
                    System.out.println(c3.envoie3());//Affichage Série 3
                    System.out.println(c4.envoie4());//Affichage Série 4
                    System.out.println("Saisissez votre choix : ");
                    while(c4.inferieur(sc1.nextInt()) == false){//le joueur 4 doit
choisir la série qu'il veut récuperer
                          System.out.println("Ce n'est pas une série valide,
saisissez votre choix : ");//Si la série choisi par le joueur 1 est incorrect
alors le joueur 1 doit choisir une série jusqu'a elle soit valide
                    }
                    c4.nbteteboeufs();//Calculer le nombre de tete de boeufs que
doit récupere le joueur 4
                    c4.InférieurCartes();//ajouter les cartes de la série demander
par le joueur 4 dans son deck puis ajouté la carte joué par le joueur 4 dans la
série choisie par le jouer
```

```
                    c4.trier();//Trier les cartes du joueur 4
                    while(c4.normaliser() == true) {// si la taille de série
choisie par le joueur 4 est supérieur a 1 alors
                          c4.supprcartes();//Supprimer toutes les cartes sauf la
cartes joué par le joueur 4
                    }
              }
              else if(c4.depassement() == false) {//Si la carte joué par le joueur
4 doit être deposé dans une série dont la taille est supérieur ou égal a 5 alors
                    c4.RecupererSerie();//Défenir quelle série le joueur 4 doit
récupérer
                    c4.nbteteboeufDeppassement();//Calculer le nombre de tête de
boeufs le joueur 4 doit récupérer
                    c4.DeppasementCartes();//Ajouter toutes les cartes de la série
dans le jeux de cartes du joueur 4 puis ajouter la carte joué par le joueur 4 dans
la série
                    c4.trier();//Trier les cartes du joueur 4
                    while(c4.DepassementChoisi() == true) {
                          c4.supprcartesdepassement();//Supprimer toutes les
cartes sauf la cartes joué par le joueur 4
                    }
              }
              System.out.println("Les cartes " + c1.RetournerCarteJouée() + " (" +
j1.envoie() + "), " + c2.RetournerCarteJouée() +  " (" + j2.envoie() + "), " +
c3.RetournerCarteJouée() + " (" + j3.envoie() + ") et " + c4.RetournerCarteJouée()
+ " (" + j4.envoie() + ") ont été posées" );
              System.out.println(c1.envoie1());//Affichage série 1
              System.out.println(c2.envoie2());//Affichage série 2
              System.out.println(c3.envoie3());//Affichage série 3
              System.out.println(c4.envoie4());//Affichage série 4
              if(c1.recup() == true) {//Si le joueur 1 à récupérer des tête de
bouefs alors
                    System.out.println(j1.envoie() + " a ramassé " +
c1.teteboeufs() + " tête boeufs.");//affiche le nombre de tête de boeufs récuperer
par le joueur 1
              }
              if(c2.recup() == true) {//Si le joueur 2 à récupérer des tête de
bouefs alors
                    System.out.println(j2.envoie() + " a ramassé " +
c2.teteboeufs() + " tête boeufs.");//affiche le nombre de tête de boeufs récuperer
par le joueur 2
              }
              if(c3.recup() == true) {//Si le joueur 3 à récupérer des tête de
bouefs alors
                    System.out.println(j3.envoie() + " a ramassé " +
c3.teteboeufs() + " tête boeufs.");//affiche le nombre de tête de boeufs récuperer
par le joueur 3
              }
              if(c4.recup() == true) {//Si le joueur 4 à récupérer des tête de
bouefs alors
                    System.out.println(j4.envoie() + " a ramassé " +
c4.teteboeufs() + " tête boeufs.");//affiche le nombre de tête de boeufs récuperer
par le joueur 4
              }
              if(c1.recup() == false && c2.recup() == false && c3.recup() == false
&& c4.recup() == false) {//Si aucun des jouer n'a récupérer de tête de boeufs
alors
                    System.out.println("Aucun joueur ne ramasse de tête de
boeufs");//afficherce message
```

```
            }
            c1.RemisaZero();//Remettre a zéro le nombre de tête de boeufs du
joueur 1
            c2.RemisaZero();//Remettre a zéro le nombre de tête de boeufs du
joueur 2
            c3.RemisaZero();//Remettre a zéro le nombre de tête de boeufs du
joueur 3
            c4.RemisaZero();//Remettre a zéro le nombre de tête de boeufs du
joueur 4
            if(m == 2) {//Si m est égal 10 alors on affiche le nombre total de
tête de boeufs récuperer par chaque joueur
                  System.out.println("** Score final");
                  System.out.println(j1.envoie() + " a ramassé " +
c1.nbteteboeufstot() + " tête de boeufs");
                  System.out.println(j2.envoie() + " a ramassé " +
c2.nbteteboeufstot() + " tête de boeufs");
                  System.out.println(j3.envoie() + " a ramassé " +
c3.nbteteboeufstot() + " tête de boeufs");
                  System.out.println(j4.envoie() + " a ramassé " +
c4.nbteteboeufstot() + " tête de boeufs");
                  }
            m++;
            }
            sc.close();
            }
            catch (FileNotFoundException e) {
                  System.out.println("Impossible d'ouvrir le fichier");
            }
      }
}
```

JoueurTest.java

```
package JeuxCartes;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

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

class JoueurTest {

	@Test
	void test() {
		Joueur j = new Joueur();
		try {
			Scanner sc = new Scanner(new
FileInputStream("C:\\ficSAE2.02/Config.txt"));
			j.initialise(sc.next());
			assertEquals("Samuel", j.envoie());//envoie du prenom
		} catch (FileNotFoundException e) {
			System.out.println("Impossible d'ouvrir le fichier");
		}
	}

}
```

CartesTest.Java

```
package JeuxCartes;

import static org.junit.Assert.assertEquals;
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;
import java.util.Scanner;

class CartesTest {

	@Test
	void test() {
		int m = 0;
		Cartes c = new Cartes();//Creation d'un joueur pour le test
		Scanner sc = new Scanner(System.in);
		assertTrue(c.EstIlinitialisé() == false);//Savoir si les 104 ont été
initialisé
		c.initialiserCartes();
		assertTrue(c.EstIlinitialisé() == true);//Les 104 cartes ont été
initialisé
		c.mélange();
		c.distribuerCartes();
		assertEquals(true, c.estDistribuer());//Distribution des 10 cartes

		c.supprimer();
		assertEquals(94, c.getTabCartessize());//Supprimer les 10 cartes
distribue au joueur
		c.trier();
		c.defSerie();
		assertTrue(c.SerieValide() == true);//Verification de la création des
4 série
		c.supprimeurs();
		assertEquals(90, c.getTabCartessize());//Verification de la
suppression des 4 cartes des séries
		c.initialisetotalteteboeufs();
		assertEquals(0, c.nbteteboeufstot());//Test initialisation totaux des
tete de boeufs
		//While(m < 3){
		assertEquals("- Série n° 1 : " +
c.EnvoieReussi1(),c.envoie1());//Verification d'un envoie de message correct
		System.out.println(c.envoie1());
		assertEquals("- Série n° 2 : " +
c.EnvoieReussi2(),c.envoie2());//Verification d'un envoie de message correct
		System.out.println(c.envoie2());
		assertEquals("- Série n° 3 : " +
c.EnvoieReussi3(),c.envoie3());//Verification d'un envoie de message correct
		System.out.println(c.envoie3());
		assertEquals("- Série n° 4 : " +
c.EnvoieReussi4(),c.envoie4());//Verification d'un envoie de message correct
		System.out.println(c.envoie4());
		System.out.println(c.envoie5());
		System.out.println("Saisissez votre choix : ");
		while(c.validejouer(sc.nextInt()) == false) {
			assertEquals("Vous n'avez pas cette carte, saisissez votre
choix : ", c.message());//Verifiaction de l'envoie du message
			System.out.println(c.message());
		}
```

```
            c.SupprimerDeck();
            c.DistSerie();
            assertEquals(false, c.verifDistSeries());//Verification de la bonne
distance
          if(c.inférieur() == true && c.depassement() == true) {
            c.reussi();
            assertTrue(true);
          }
          else if (c.inférieur() == true && c.VerificationJeux() == false) {
            System.out.println(c.envoie1());//Affichage série 1
                  System.out.println(c.envoie2());//Affichage série 2
                  System.out.println(c.envoie3());//Affichage série 3
                  System.out.println(c.envoie4());//Affichage série 4
            while(c.inferieur(sc.nextInt()) == false) {
                  System.out.println("Ce n'est pas une série valide, saisissez
votre choix : ");
            }
            c.nbteteboeufs();
            c.InférieurCartes();
            assertEquals(true, c.tailleCarteJoueur());
            c.trier();
            assertTrue(true);
          while(c.normaliser() == true) {
            c.supprcartes();
            assertTrue(true);
          }
          }
          else if(c.depassement() == true) {
            c.RecupererSerie();
            c.nbteteboeufDeppassement();
            c.DeppasementCartes();
            assertTrue(true);
            while(c.normaliser() == true) {
                  c.supprcartesdepassement();
                  assertTrue(true);
                }
          }


            }
}
```