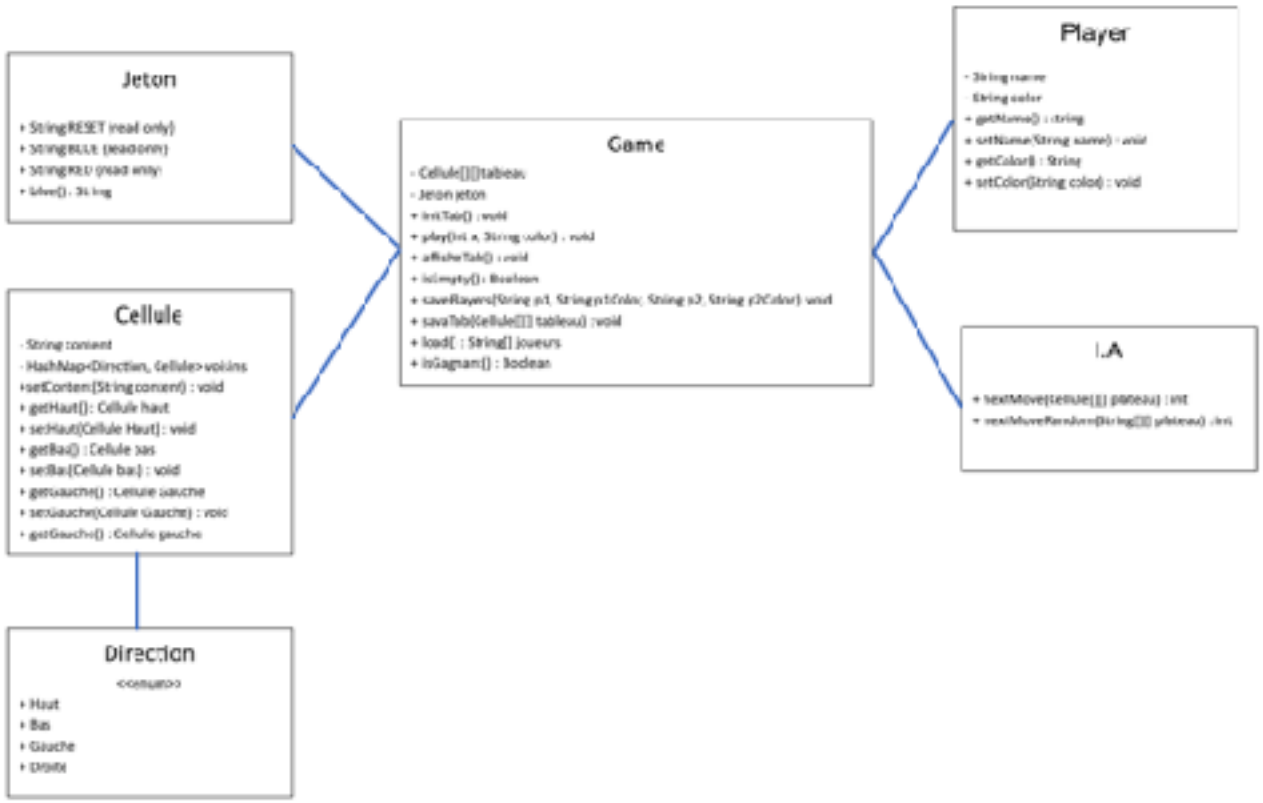
JEU DE PUISSANCE 4

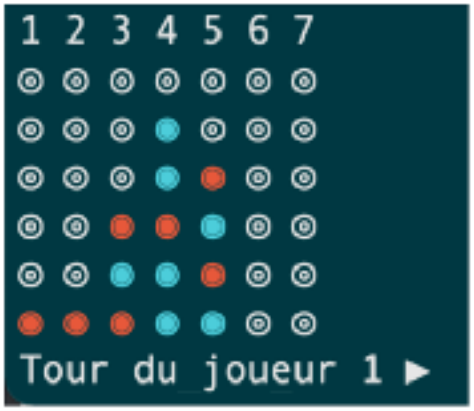
Ce programme en Java vous permet de jouer au célèbre jeu de stratégie où deux joueurs s'affrontent pour aligner quatre jetons de leur couleur horizontalement, verticalement ou en diagonale. Dans ce projet, vous pourrez jouer contre un autre joueur ou contre une intelligence artificielle qui utilisera des algorithmes pour essayer de vous battre. Nous avons utilisé une matrice de cellules pour représenter le plateau de jeu et avons implémenté différentes fonctions pour gérer la logique du jeu. Nous avons également ajouté une fonctionnalité de sauvegarde et de chargement pour que vous puissiez continuer vos parties ultérieurement. Que vous soyez un joueur expérimenté ou un débutant, notre programme vous offrira une expérience de jeu agréable et stimulante. Alors, êtes-vous prêt à jouer ?

DIAGRAMME UML :



J'ai choisi d'utiliser une matrice de cellules qui fera office de plateau de jeu, chaque cellule de la grille sera initialisée avec le symbole ◎ pour dire qu'elle est vide, et grâce à la HashMap, chaque cellule contiendra une référence vers les cellules qui l'entourent.

Chaque joueur a un nom et une couleur. Grâce au code ANSI, mes cellules seront soit bleues soit rouges, affichées par le symbole ◉.



Afin de savoir si le plateau est encore vide, j'ai ajouté une fonction qui parcourt la première ligne de notre plateau à la recherche d'une cellule vide (cellule contenant le symbole ◎).

Pour savoir qui a gagné, j'ai choisi d'implémenter une fonction isGagnant qui parcourt notre plateau à la recherche d'une suite de quatre jetons de la même couleur.

Pour sauvegarder, j'ai choisi de mettre le tableau dans un fichier texte et les informations sur les joueurs dans un autre.

Pour recharger, je recharge ligne par ligne le fichier texte dans notre plateau et je mets les informations joueurs dans un tableau afin de les réattribuer aux joueurs par la suite.

Pour l'IA, j'ai voulu faire un algorithme MinMax, mais faute de temps, je me suis plus orienté vers une approche plus simple. En premier lieu, j'ai fait un algorithme qui joue aléatoirement, puis j'ai fait un algorithme qui essaye de monter le plus possible.

IA.java

```java
public class IA {
    /*
     ************************************************
        nextMove() : permet a l'IA de joueur un tour
            Prend comme parametres le plateau
            Retourne le choix de l'IA
     ************************************************
     */
    public int nextMove(Cellule[][] plateau) {
        float min = 800;
        int col = 0;
        int lignes = 6;
        int colonnes = 7;
        float midx = (float)colonnes / 2;
        float midy = (float)lignes / 2;
        for (int i = 0; i < lignes; i++) {
            for (int j = 0; j < colonnes; j++) {
                if (plateau[i][j].getContent().equals
                    float valx = Math.abs(i-midx);
                    float valy = Math.abs(j-midy);
                    float finalVal= valx+ valy;
                    if (finalVal < min) {
                        min = finalVal;
                        col = j;
                    }
                }
            }
        }
        return col;
    }
}

```

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

import static java.lang.Thread.sleep;
public class Game {
    Cellule[][] tableau = new Cellule[6][7];
    Jeton jeton = new Jeton();

    /*
     ************************************************
        InitTab() : Initialisation du tableau de jeu
            Ne prend rien en parametre
            Ne retourne rien
     ************************************************
     */
    public void initTab(){
        //plateau.initPlateau();
        for (int i = 0; i <6;i++){
            for (int j = 0; j <7;j++) {
                tableau[i][j] = new Cellule();
            }
        }
        for (int i = 0; i<6;i++){
            for (int j = 0; j<7;j++){
                tableau[i][j].setContent("◎");
                try {
                    tableau[i][j].setHaut(tableau[i -
                }catch(Exception ignored){}
                try{
                    assert tableau != null;
                    assert tableau[i] != null;
                    assert tableau[i][j] != null;
                    tableau[i][j].setBas(tableau[i +
                }catch(Exception ignored){}
                try {
                    tableau[i][j].setGauche(tableau[i
                }catch(Exception ignored){}
                try{
                    assert tableau != null;
```

```
42                      assert tableau[i] != null;
43                      assert tableau[i][j] != null;
44                      tableau[i][j].setDroite(tableau[i
45                  }catch(Exception ignored){}
46              }
47          }
48      }
49
50      /*
51       ************************************************
52          Play() : Deroulement d'un tour
53              Prend comme parametres le choix du joueur
54              Ne retourne rien
55       ************************************************
56       */
57      public void play(int x,String color) throws FileN
58
59          if(x == -1){  //save
60              saveTab(tableau);
61          }
62          else{
63              for(int i = 5; i>=0;i--){
64                  if(tableau[i][x].getContent().equals(
65                      if(color.equals("Rouge")){
66                          tableau[i][x].setContent(jeto
67                          break;
68                      }
69                      if(color.equals("Bleu")){
70                          tableau[i][x].setContent(jeto
71                          break;
72                      }
73                  }
74              }
75          }
76      }
77
78      /*
79       ************************************************
80          afficheTab() : Affichage du plateau apres cha
81              Ne prend rien en parametre
82              Ne retourne rien
```

```
83       ************************************************
84       */
85      public void affichTab(){
86          System.out.println("1 2 3 4 5 6 7");
87          for(int i=0; i<6; i++){
88              for(int j=0; j<7; j++){System.out.print(
89              System.out.println();
90          }
91      }
92
93      /*
94       ************************************************
95          IsEmpty() : Verifie s'il y a encore de la pl
96              Ne prend rien en parametre
97              Retourne un Boolean
98       ************************************************
99       */
100     public boolean isEmpty() {
101         for (int i = 0; i < 7; i++) {
102             if (tableau[0][i].getContent().equals("©
103                 return true;
104             }
105         }
106         return false;
107     }
108
109     /*
110      ************************************************
111         savePlayer() : Permet de sauvegarder les joe
112             Prend comme parametres les noms des joue
113             Ne retourne rien
114      ************************************************
115      */
116     public void savePlayers(String p1, String p1Colo
117         File joueurs = new File("Save_J.txt");
118         PrintWriter pr = new PrintWriter(joueurs);
119         pr.println(p1);
120         pr.println(p1Color);
121         pr.println(p2);
122         pr.println(p2Color);
123         pr.close();
```

```
124     }
125
126     /*
127          ******************************************
128             saveTabr() : Permet de sauvegarder le Pl
129                 Prend comme parametres le plateau
130                 Ne retourne rien
131          ******************************************
132          */
133     public void saveTab(Cellule[][] tableau) throws
134         File tab = new File("Save_T.txt");
135         PrintWriter pr = new PrintWriter(tab);
136         for (int i = 0; i <6;i++) {
137             for (int j = 0; j < 7; j++) {
138                 pr.println(tableau[i][j].getContent(
139             }
140         }
141         pr.close();
142     }
143
144     /*
145      ************************************************
146         load() : Permet de recharger une partie
147             Ne prend rien comme parametre
148             Retourne les joueurs
149      ************************************************
150      */
151     public String[] load() throws FileNotFoundExcept
152         String[] joueurs = new String[4];
153         Scanner loaderTab = new Scanner(new File("Sa
154         Scanner loaderP = new Scanner(new File("Save
155         for (int i = 0; i <6;i++){
156             for (int j = 0; j <7;j++) {
157                 if(loaderTab.hasNext()){
158                     tableau[i][j].setContent(loaderT
159                 }
160                 else{
161                     System.out.println("Il n y a pas
162                 }
163                 if(loaderP.hasNext()){
164                     for(int k = 0; k <4; k++){
```

```
165                         joueurs[k]= loaderP.next();
166                     }
167                 }
168             }
169         }
170         return joueurs;
171     }
172 
173     /*
174      ************************************************
175         isGagnant() : Permet de savoir si un joueur
176             Ne prend pas de parametre
177             Retourne un boolean
178      ************************************************
179      */
180     public boolean isGagnant() {
181         for(int i = 0; i <6;i++){
182             for(int j = 0; j <7;j++){
183                 if(! tableau[i][j].getContent().equa
184                     if(tableau[i][j].getContent().eq
185                             && tableau[i][j+1].getCo
186                             && tableau[i][j+2].getCo
187                     ){
188                         return true;
189                     }
190                 }
191             }
192         }
193         for(int i = 0; i <3;i++){
194             for(int j = 0; j <7;j++){
195                 if(! tableau[i][j].getContent().equa
196                     if(tableau[i][j].getContent().eq
197                             && tableau[i+1][j].getCo
198                             && tableau[i+2][j].getCo
199                     ){
200                         return true;
201                     }
202                 }
203             }
204         }
205         for(int i = 0; i <3;i++){
```

```
206             for(int j = 0; j <7;j++){
207                 if(! tableau[i][j].getContent().equa
208                     if(tableau[i][j].equals(tableau[
209                             && tableau[i+1][j+1].get
210                             && tableau[i+2][j+2].get
211                     ){
212                         return true;
213                     }
214                 }
215 
216             }
217         }
218         for(int i = 0; i <3;i++){
219             for(int j = 3; j <7;j++){
220                 if(! tableau[i][j].getContent().equa
221                     if(tableau[i][j].equals(tableau[
222                             && tableau[i+1][j-1].get
223                             && tableau[i+2][j-2].get
224                     ){
225                         return true;
226                     }
227                 }
228             }
229         }
230         return(false);
231     }
232 
233     public void easterEgg() throws InterruptedExcept
234 
235         String[][] easter = new String[6][7];
236         for (int i = 0; i <6;i++){
237             for (int j = 0; j <7;j++) {
238                 easter[i][j]=jeton.rouge();
239             }
240         }
241         easter[1][1] = jeton.bleu();
242         easter[2][1] = jeton.bleu();
243         easter[1][2] = jeton.bleu();
244         easter[2][2] = jeton.bleu();
245         easter[1][4] = jeton.bleu();
246         easter[2][4] = jeton.bleu();
```

```
247             easter[1][5] = jeton.bleu();
248             easter[2][5] = jeton.bleu();
249             easter[4][1] = jeton.bleu();
250             easter[4][2] = jeton.bleu();
251             easter[4][3] = jeton.bleu();
252             easter[4][4] = jeton.bleu();
253             easter[4][5] = jeton.bleu();
254             System.out.println("1 2 3 4 5 6 7");
255             for(int i=0; i<6; i++){
256                 for(int j=0; j<7; j++){System.out.print(
257                 System.out.println();
258             }
259             sleep(2000);
260         }
261 }
262
```

Main.java

```java
import java.io.FileNotFoundException;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) throws Fil
        String load,adversaire; // toujours le p1 qui
        String[] T;
        int choix;
        Scanner input = new Scanner(System.in);
        Player p1 = new Player();
        Player p2 = new Player();
        IA Machine = new IA();
        Game jeu = new Game();
        System.out.println();
        System.out.println("PUISSANCE 4-2000");
        System.out.println("****************");
        jeu.initTab();
        do {
            System.out.print("Voulez vous recharger l
            load = input.next();
        } while (!(load.equals("Oui") || load.equals(

        if (load.equals("Oui")) { // PAS DE GOTO LES
            T = jeu.load();
            p1.setName(T[0]);
            p1.setColor(T[1]);
            p2.setName(T[2]);
            if (p2.getName().equals("Machine")){
                adversaire = "Machine";
            }
            else{
                adversaire = "Humain";
            }

            p2.setColor(T[3]);
        } else {
            System.out.println();
            System.out.print("Player 1 ▶ ");
            p1.setName(input.next());

            do {
                System.out.print("Color (Rouge/Bleu)
```

```
42                  p1.setColor(input.next());
43              } while (!(p1.getColor().equals("Rouge")
44
45              do {
46                  System.out.print("Voulez vous joueur
47                  adversaire = input.next();
48              } while (!(adversaire.equals("Humain") ||
49
50              if (adversaire.equals("Humain")) {
51                  System.out.print("Player 2 ▶ ");
52                  p2.setName(input.next());
53              } else {
54                  p2.setName("Machine");
55              }
56              if (p1.getColor().equals("Bleu")) {
57                  p2.setColor("Rouge");
58              }
59              if (p1.getColor().equals("Rouge")) {
60                  p2.setColor("Bleu");
61              }
62          }
63          System.out.println(p1.getName() + " joue " +
64          System.out.println(p2.getName() + " joue " +
65          System.out.println("Vous pouvez sauvegarder a
66          jeu.affichTab();
67          do {
68              do {
69                  System.out.print("Tour du joueur 1 ▶
70                  choix = input.nextInt();
71                  if (choix == 42) {   //EasterEgg
72                      jeu.easterEgg();
73                      jeu.affichTab();
74                  }
75                  if (choix == 0) {
76                      jeu.savePlayers(p1.getName(), p1.
77                  }
78              } while (!(choix >= 1 && choix <= 7 || ch
79              jeu.play(choix - 1, p1.getColor());
80              jeu.affichTab();
81              if(jeu.isGagnant()){
82                  System.out.println(p1.getName() + " g
```

```
                    break;
                }
                if (adversaire.equals("Humain")) {
                    do {
                        System.out.print("Tour du joueur
                        choix = input.nextInt();
                        if (choix == 42) {   //EasterEgg
                            jeu.easterEgg();
                            jeu.affichTab();
                        }
                        if (choix == 0) {
                            jeu.savePlayers(p1.getName()
                        }
                    } while (!(choix >= 1 && choix <= 7
                    jeu.play(choix - 1, p2.getColor());
                    jeu.affichTab();
                    if(jeu.isGagnant()){
                        System.out.println(p2.getName()
                        break;
                    }
                } else {
                    jeu.play(Machine.nextMove(jeu.tablea
                    jeu.affichTab();
                    if(jeu.isGagnant()){
                        System.out.println("l'Ordinateur
                        break;
                    }
                }
            }while (jeu.isEmpty()) ;
            System.out.println("Fin Partie");
        }
    }
```

Jeton.java

```java
public class Jeton {

    public static final String RESET = "\u001B[0m";
    public static final String BLUE = "\u001B[36m";
    public static final String RED = "\033[0;31m";

    public String bleu(){
        return(BLUE+"◉"+RESET);
    }
    public String rouge(){
        return(RED+"◉"+RESET);
    }
}

```

Player.java

```java
public class Player {
    String name;
    String color;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }
}

```

Cellule.java

```java
import java.util.HashMap;

public class Cellule {
    String content;
    HashMap<Direction, Cellule> voisins = new HashMap
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public void setHaut(Cellule haut) {
        voisins.put(Direction.Haut, haut);
    }
    public void setBas(Cellule bas) {
        voisins.put(Direction.Bas, bas);
    }
    public void setGauche(Cellule gauche) {
        voisins.put(Direction.Gauche, gauche);
    }
    public void setDroite(Cellule droite) {
        voisins.put(Direction.Droite, droite);
    }
}

enum Direction {
    Haut,
    Bas,
    Gauche,
    Droite
}
```