DUT INFO 2015/2016
1ere ANNEE

PROJET JAVA
BOGGLE

FILIPOVICH Maxime
WANG Antoine
(Groupe 107)

Table des matières :

1) Présentation du projet

L'application développée pour ce projet a pour but de gérer une partie de Boggle dans son intégralité.

Lors de l'initialisation de la partie, plusieurs options sont paramétrables, comme le nombre de joueurs (1+), la taille du plateau de jeu (4+, taille en longueur et largeur), le choix de génération aléatoire ou de rédaction manuelle des cases du plateau, et le nom de chaque joueur.

Par la suite, chaque joueur est demandé de rentrer les mots qu'il voit dans le plateau, l'un après l'autre. Les mots doivent être écrits sur une même ligne, espacés.

Ensuite, les mots sont insérés dans une liste, et des algorithmes de suppression vont parcourir les listes des différents joueurs afin de retirer, dans l'ordre :

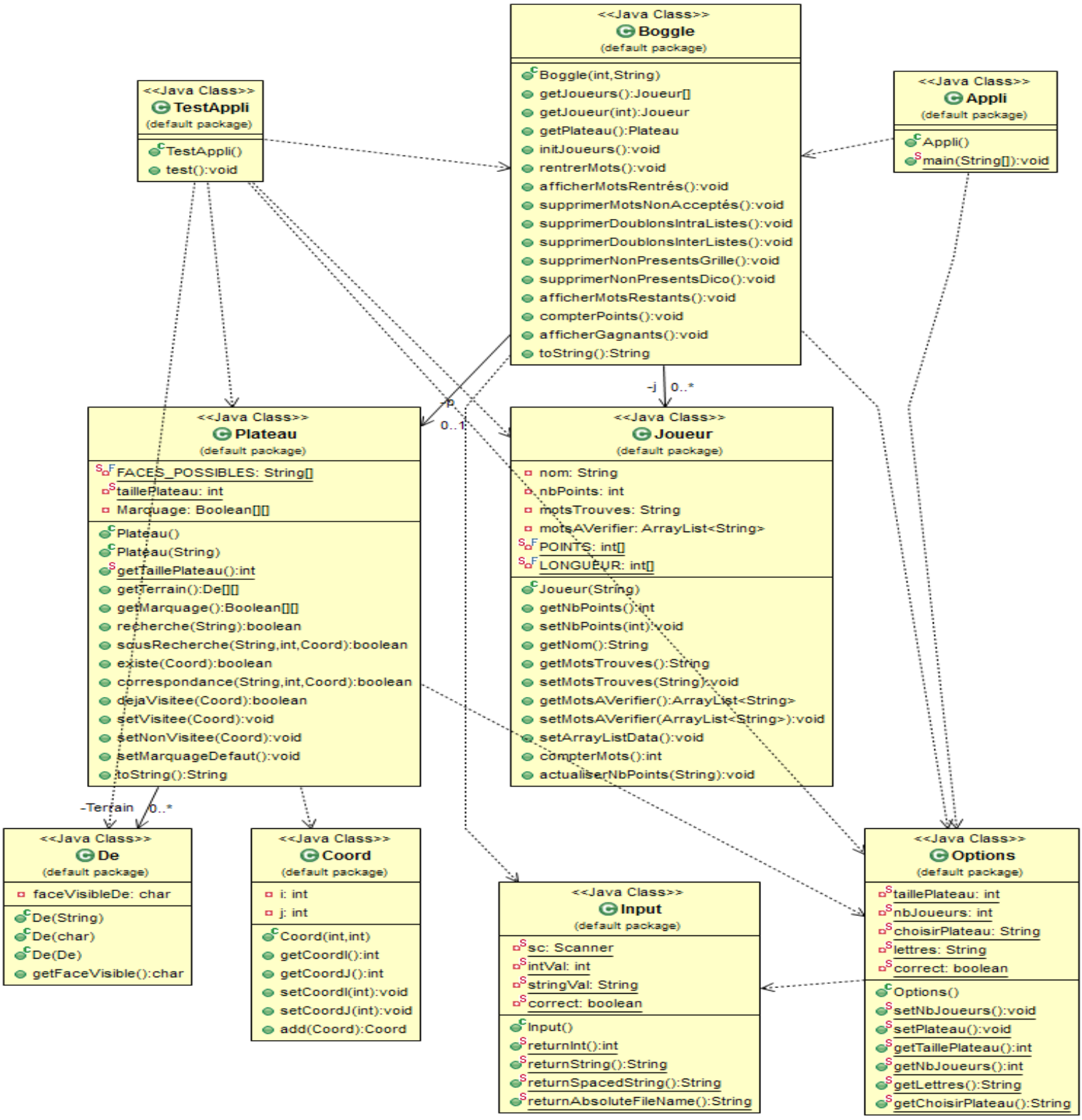
- Les mots écrits plusieurs fois au sein d'une liste (Un seul subsiste parmi ceux là)
- Les mots présents dans plusieurs listes (Ils sont tous supprimés)
- Les mots absents de la grille (Un mot ne doit pas passer deux fois par la même case, et chaque lettre doit être en contact direct ou en diagonale avec la suivante)
- Les mots non présents dans le dictionnaire fourni

Pour cette dernière étape on demande à l'utilisateur de rentrer l'adresse absolue d'un dictionnaire, dans notre cas un fichier .txt.

Enfin, les points sont comptés et le ou les vainqueurs sont affichés.

2) Diagramme UML des classes

Voici le diagramme UML des classes.



3) Listing des tests unitaires

Voici les tests unitaires tels que vus dans TestAppli.java.

On notera qu'étant donné que les appels de méthodes dans l'application se font sans entrée de paramètres, les tests unitaires effectués ne vérifient que la validité des données en sortie de ces méthodes.

Pour ce qui est des entrées, elles sont gérées par la classe input, qui ne permet pas de rentrer des données fautives, que ce soit pour le type ou le paterne, avec l'aide de boucles.

De plus si la donnée voulue en entrée est encore plus spécifique, comme par exemple un entier entre x et y, il est possible de boucler l'appel à la méthode input voulue.

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

import org.junit.Test;

public class TestAppli {

	@Test
	public void test() {
		Options.setNbJoueurs();
		assertTrue(Options.getNbJoueurs() == (int) Options.getNbJoueurs());
		assertTrue(Options.getNbJoueurs() >= 1);

		Options.setPlateau();
		assertTrue(Options.getTaillePlateau() == (int)
Options.getTaillePlateau());
		assertTrue(Options.getTaillePlateau() >= 4);
		assertTrue(Options.getChoisirPlateau().equals((String)
Options.getChoisirPlateau()));
		assertTrue(
				Options.getChoisirPlateau().equalsIgnoreCase("o") ||
Options.getChoisirPlateau().equalsIgnoreCase("n"));
		assertTrue(Options.getLettres().equals((String) Options.getLettres()));
		assertTrue(Options.getLettres().length() == 0
				|| Options.getLettres().length() ==
(Options.getTaillePlateau() * Options.getTaillePlateau()));

		Boggle b = new Boggle(Options.getNbJoueurs(), Options.getLettres());
```

```
            assertTrue(Options.getNbJoueurs() == b.getJoueurs().length);
            assertTrue(Options.getTaillePlateau() == Plateau.getTaillePlateau());
            assertTrue((Plateau.getTaillePlateau() * Plateau.getTaillePlateau()) ==
(b.getPlateau().getTerrain().length)
                        * (b.getPlateau().getTerrain().length));
            System.out.println(b);

            b.initJoueurs();
            for (int i = 0; i < b.getJoueurs().length; i++) {
                  assertTrue(b.getJoueur(i).getNom().equals((String)
b.getJoueur(i).getNom()));
                  assertTrue(b.getJoueur(i).getNom().length() >= 1);
            }

            b.rentrerMots();
            for (int i = 0; i < b.getJoueurs().length; i++) {
                  assertTrue(b.getJoueur(i).getMotsTrouves().equals((String)
b.getJoueur(i).getMotsTrouves()));


assertTrue(b.getJoueur(i).getMotsTrouves().equals(b.getJoueur(i).getMotsTrouves().trim(
)));
                  assertTrue(b.getJoueur(i).getMotsTrouves().matches("[a-zA-Z ]+$"));
                  assertTrue(b.getJoueur(i).getMotsTrouves().length() >= 1);
            }

            b.afficherMotsRentrés();
            for (int i = 0; i < b.getJoueurs().length; i++) {
                  assertTrue(b.getJoueur(i).getMotsAVerifier().size() >= 1);
                  for (int k = 0; k < b.getJoueur(i).getMotsAVerifier().size(); k++) {
                        assertTrue(b.getJoueur(i).getMotsAVerifier().get(k).length()
>= 1);
                  }
            }

            b.supprimerMotsNonAcceptés();
            b.compterPoints();
            for (int i = 0; i < b.getJoueurs().length; i++) {
                  assertTrue(b.getJoueur(i).getNbPoints() == ((int)
b.getJoueur(i).getNbPoints()));
                  assertTrue(b.getJoueur(i).getNbPoints() >= 0);
            }

            b.afficherGagnants();
      }

}
```

4) Listing des sources

Voici le listing des sources, rangées plus ou moins par ordre d'appel dans l'application.

1 – Appli.java

```
public class Appli {

	public static void main(String[] args) {
		System.out.println("Bienvenue dans le Boggle!");
		Options.setNbJoueurs();
		Options.setPlateau();
		Boggle b = new Boggle(Options.getNbJoueurs(), Options.getLettres());
		System.out.println(b);
		b.initJoueurs();
		b.rentrerMots();
		b.afficherMotsRentrés();
		b.supprimerMotsNonAcceptés();
		b.compterPoints();
		b.afficherGagnants();
	}

}
```

2 – Options.java

```
public class Options {

	private static int taillePlateau;
	private static int nbJoueurs;
	private static String choisirPlateau;
	private static String lettres;
	private static boolean correct;

	static {
		taillePlateau = 4;
		nbJoueurs = 1;
		choisirPlateau = "n";
		lettres = "";
		correct = false;
	}

	public static void setNbJoueurs() {
		System.out.println("Veuillez rentrer le nombre de joueurs");
		do {
			nbJoueurs = Input.returnInt();
			if (nbJoueurs < 1) {
				System.out.println("Veuillez rentrer un nombre supérieur ou
```

```
égal à 1");
                }
            } while (nbJoueurs < 1);
        }

        public static void setPlateau() {
            System.out.println("De quelle taille doit etre le plateau?");
            do {
                taillePlateau = Input.returnInt();
                if (taillePlateau < 4) {
                    System.out.println("Veuillez rentrer un nombre supérieur ou
égal à 4");
                }
            } while (taillePlateau < 4);
            System.out.println("Voulez vous rentrer votre propre plateau? (O/N)");
            do {
                choisirPlateau = Input.returnString();
                if (!choisirPlateau.equalsIgnoreCase("o") && !
choisirPlateau.equalsIgnoreCase("n")) {
                    System.out.println("Veuillez rentrer 'o' ou 'n'");
                }
            } while (!choisirPlateau.equalsIgnoreCase("o") && !
choisirPlateau.equalsIgnoreCase("n"));
            if (choisirPlateau.equalsIgnoreCase("o")) {
                System.out.println("Veuillez rentrer les lettres du plateau");
                System.out.println("Il doit y en avoir " + (taillePlateau *
taillePlateau));
                do {
                    lettres = Input.returnString();
                    if (lettres.length() == (taillePlateau * taillePlateau)) {
                        correct = true;
                    } else {
                        System.out.println("Il faut ni plus ni moins que " +
(taillePlateau * taillePlateau) + " lettres");
                        correct = false;
                    }
                } while (!correct);
            } else {
                System.out.println("Vous avez choisi de prendre un plateau
aleatoire");
            }
        }

        public static int getTaillePlateau() {
            return taillePlateau;
        }

        public static int getNbJoueurs() {
            return nbJoueurs;
        }

        public static String getLettres() {
            return lettres;
        }

        public static String getChoisirPlateau() {
            return choisirPlateau;
        }
```

```
}
```

3 – Input.java

```
import java.util.Scanner;

public class Input {

	private static Scanner sc;
	private static int intVal;
	private static String stringVal;
	private static boolean correct;

	static {
		sc = new Scanner(System.in);
		intVal = 0;
		stringVal = "";
		correct = false;
	}

	public static int returnInt() {
		correct = false;
		do {
			try {
				intVal = Integer.parseInt(sc.nextLine());
				correct = true;
			} catch (NumberFormatException e) {
				System.out.println("Veuillez rentrer un nombre entier");
			}
		} while (!correct);
		return intVal;
	}

	public static String returnString() {
		do {
			stringVal = sc.nextLine();
			if (!stringVal.matches("[a-zA-Z]+$")) {
				System.out.println("Veuillez rentrer des lettres
uniquement");
			}
		} while (!stringVal.matches("[a-zA-Z]+$"));
		return stringVal;
	}

	public static String returnSpacedString() {
		do {
			stringVal = sc.nextLine();
			if (!stringVal.matches("[a-zA-Z ]+$") || stringVal.trim().isEmpty())
{
				System.out.println("Veuillez rentrer des lettres
uniquement");
			}
		} while (!stringVal.matches("[a-zA-Z ]+$") || stringVal.trim().isEmpty());
		return stringVal.trim();
	}

	public static String returnAbsoluteFileName() {
```

```
		do {
			stringVal = sc.nextLine();
			if (!stringVal.matches(".*" + ".txt+$")) {
				System.out.println("Veuillez rentrer un chemin absolu de
fichier texte");
			}
		} while (!stringVal.matches(".*" + ".txt+$"));
		return stringVal;
	}

}
```

4 – Boggle.java

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

public class Boggle {

	private Joueur[] j;
	private Plateau p;

	public Boggle(int nbJoueurs, String faces) {
		this.j = new Joueur[nbJoueurs];
		if (faces.isEmpty()) {
			this.p = new Plateau();
		} else {
			this.p = new Plateau(faces);
		}
	}

	public Joueur[] getJoueurs() {
		return j;
	}

	public Joueur getJoueur(int i) {
		return j[i];
	}

	public Plateau getPlateau() {
		return p;
	}

	public void initJoueurs() {
		for (int i = 0; i < j.length; i++) {
			System.out.println("Rentrez le nom du joueur " + (i + 1));
			do {
				String nom = Input.returnString();
				nom = nom.toLowerCase();
				nom = nom.substring(0, 1).toUpperCase() + nom.substring(1);
				int verif = 0;
				for (int k = 0; k < i; k++) {
					if (!nom.equalsIgnoreCase(j[k].getNom())) {
						verif++;
					}
```

```
                        }
                        if (verif == i) {
                              j[i] = new Joueur(nom);
                        } else {
                              System.out.println("Veuillez rentrer un nom de joueur
different de celui des autres");
                        }
                  } while (j[i] == null);
            }
            for (int i = 0; i < Options.getNbJoueurs(); i++) {
                  System.out.println("Le joueur " + (i + 1) + " sera " +
j[i].getNom());
            }
      }

      public void rentrerMots() {
            for (int i = 0; i < j.length; i++) {
                  System.out
                              .println("C'est au joueur " + j[i].getNom() + " de
jouer, rentrez les mots sur une ligne, espacés");
                  String mots = Input.returnSpacedString();
                  mots = mots.toLowerCase();
                  mots = mots.replaceAll(" +", " ");
                  j[i].setMotsTrouves(mots);
            }
      }

      public void afficherMotsRentrés() {
            for (int i = 0; i < j.length; i++) {
                  j[i].setArrayListData();
                  System.out.println("Le joueur " + j[i].getNom() + " a rentré " +
j[i].compterMots() + " mots");
                  System.out.println("Ce sont les mots suivants: " +
j[i].getMotsAVerifier().toString());
            }
      }

      public void supprimerMotsNonAcceptés() {
            supprimerDoublonsIntraListes();
            afficherMotsRestants();
            supprimerDoublonsInterListes();
            afficherMotsRestants();
            supprimerNonPresentsGrille();
            afficherMotsRestants();
            supprimerNonPresentsDico();
            afficherMotsRestants();
      }

      public void supprimerDoublonsIntraListes() {
            System.out.println("Les doublons dans chaque liste seront supprimés");
            int index = 0;
            for (int i = 0; i < j.length; i++) {
                  ArrayList<Integer> indexDoublons = new ArrayList<Integer>();
                  for (int k = 0; k < j[i].getMotsAVerifier().size(); k++) {
                        for (int l = k + 1; l < j[i].getMotsAVerifier().size(); l++)
{
                              if
(j[i].getMotsAVerifier().get(k).equalsIgnoreCase(j[i].getMotsAVerifier().get(l))) {
                                    if (!indexDoublons.contains(l)) {
```

```
                                        indexDoublons.add(l);
                                    }
                                }
                            }
                        }
                        for (int l = 0; l < indexDoublons.size(); l++) {
                            index = indexDoublons.get(l);
                            j[i].getMotsAVerifier().remove(index);
                            for (int m = 0; m < indexDoublons.size(); m++) {
                                if (indexDoublons.get(m) >= index) {
                                    indexDoublons.set(m, indexDoublons.get(m) - 1);
                                }
                            }
                        }
                    }
                }

                public void supprimerDoublonsInterListes() {
                    System.out.println("Les mots présents dans plusieures listes seront
supprimés");
                    int index = 0;
                    ArrayList<ArrayList<Integer>> groupesIndexesDoublons = new
ArrayList<ArrayList<Integer>>(j.length);
                    for (int i = 0; i < j.length; i++) {
                        ArrayList<Integer> liste = new
ArrayList<Integer>(j[i].getMotsAVerifier().size());
                        groupesIndexesDoublons.add(liste);
                    }
                    for (int i = 0; i < j.length; i++) {
                        for (int k = i + 1; k < j.length; k++) {
                            for (int l = 0; l < j[i].getMotsAVerifier().size(); l++) {
                                for (int m = 0; m < j[k].getMotsAVerifier().size(); m+
+) {
                                    if
(j[i].getMotsAVerifier().get(l).equalsIgnoreCase(j[k].getMotsAVerifier().get(m))) {
                                        if (!
groupesIndexesDoublons.get(i).contains(l)) {

                groupesIndexesDoublons.get(i).add(l);
                                        }
                                        if (!
groupesIndexesDoublons.get(k).contains(m)) {

                groupesIndexesDoublons.get(k).add(m);
                                        }
                                    }
                                }
                            }
                        }
                    }
                    for (int i = 0; i < groupesIndexesDoublons.size(); i++) {
                        for (int k = 0; k < groupesIndexesDoublons.get(i).size(); k++) {
                            index = groupesIndexesDoublons.get(i).get(k);
                            j[i].getMotsAVerifier().remove(index);
                            for (int l = 0; l < groupesIndexesDoublons.get(i).size(); l+
+) {
                                if (groupesIndexesDoublons.get(i).get(l) >= index) {
                                    groupesIndexesDoublons.get(i).set(l,
groupesIndexesDoublons.get(i).get(l) - 1);
```

```
                                }
                        }
                }
        }
    }

    public void supprimerNonPresentsGrille() {
        System.out.println("Les mots non présents dans la grille seront
supprimés");
        int index = 0;
        ArrayList<ArrayList<Integer>> groupesIndexesNonPresents = new
ArrayList<ArrayList<Integer>>(j.length);
        for (int i = 0; i < j.length; i++) {
            ArrayList<Integer> liste = new
ArrayList<Integer>(j[i].getMotsAVerifier().size());
            groupesIndexesNonPresents.add(liste);
        }
        for (int i = 0; i < j.length; i++) {
            for (int k = 0; k < j[i].getMotsAVerifier().size(); k++) {
                if (!p.recherche(j[i].getMotsAVerifier().get(k))) {
                    groupesIndexesNonPresents.get(i).add(k);
                }
            }
        }
        for (int i = 0; i < groupesIndexesNonPresents.size(); i++) {
            for (int k = 0; k < groupesIndexesNonPresents.get(i).size(); k++) {
                index = groupesIndexesNonPresents.get(i).get(k);
                j[i].getMotsAVerifier().remove(index);
                for (int l = 0; l < groupesIndexesNonPresents.get(i).size();
l++) {
                    if (groupesIndexesNonPresents.get(i).get(l) >= index) {
                        groupesIndexesNonPresents.get(i).set(l,
groupesIndexesNonPresents.get(i).get(l) - 1);
                    }
                }
            }
        }
    }

    public void supprimerNonPresentsDico() {
        System.out.println("Les mots non présents dans le dictionnaire seront
supprimés");
        System.out.println("Rentrez l'adresse absolue du dictionnaire en .txt");
        ArrayList<ArrayList<String>> groupesMotsAGarder = new
ArrayList<ArrayList<String>>(j.length);
        for (int i = 0; i < j.length; i++) {
            ArrayList<String> liste = new
ArrayList<String>(j[i].getMotsAVerifier().size());
            groupesMotsAGarder.add(liste);
        }
        boolean correct = false;
        do {
            String filename = Input.returnAbsoluteFileName();
            try {
                for (int i = 0; i < j.length; i++) {
                    for (int k = 0; k < j[i].getMotsAVerifier().size(); k+
+) {
                        Scanner in = new Scanner(new
FileInputStream(filename));
```

```
                                        while (in.hasNextLine()) {
                                              if
(in.nextLine().equalsIgnoreCase(j[i].getMotsAVerifier().get(k))) {

      groupesMotsAGarder.get(i).add(j[i].getMotsAVerifier().get(k));
                                              }
                                        }
                                        in.close();
                                  }
                            }
                            correct = true;
                      } catch (FileNotFoundException e) {
                            System.out.println("Impossible d'ouvrir le fichier");
                      }
                } while (!correct);
                for (int i = 0; i < groupesMotsAGarder.size(); i++) {
                      j[i].setMotsAVerifier(groupesMotsAGarder.get(i));
                }
          }

          public void afficherMotsRestants() {
                for (int i = 0; i < j.length; i++) {
                      System.out.println("Voici les mots restants pour le joueur " +
j[i].getNom() + " : "
                                  + j[i].getMotsAVerifier().toString());
                }
          }

          public void compterPoints() {
                System.out.println("Les points du/des joueurs à présent calculés");
                for (int i = 0; i < j.length; i++) {
                      for (int k = 0; k < j[i].getMotsAVerifier().size(); k++) {
                            j[i].actualiserNbPoints(j[i].getMotsAVerifier().get(k));
                      }
                      System.out.println("Le joueur " + j[i].getNom() + " a " +
j[i].getNbPoints() + " points");
                }
          }

          public void afficherGagnants() {
                ArrayList<Integer> indexGagnants = new ArrayList<Integer>();
                int pointsMax = 0;
                for (int i = 0; i < j.length; i++) {
                      if (j[i].getNbPoints() > pointsMax) {
                            indexGagnants.clear();
                            pointsMax = j[i].getNbPoints();
                            indexGagnants.add(i);
                      } else if (j[i].getNbPoints() == pointsMax) {
                            indexGagnants.add(i);
                      }
                }
                if (indexGagnants.size() == 1) {
                      System.out.println("Le gagnant est " +
j[indexGagnants.get(0)].getNom() + " avec "
                                  + j[indexGagnants.get(0)].getNbPoints() + " points");
                } else {
                      System.out.println("Les gagnants sont :");
                      for (int i = 0; i < indexGagnants.size(); i++) {
                            System.out.println(j[indexGagnants.get(i)].getNom() + " avec
```

```
" + j[indexGagnants.get(i)].getNbPoints()
                                        + " points");
                    }
                }
                System.out.println("Fin de la partie, merci d'avoir joué!");
        }

        public String toString() {
                return p.toString();
        }

}
```

5 – Joueur.java

```
import java.util.ArrayList;

public class Joueur {

        private String nom;
        private int nbPoints;
        private String motsTrouves;
        private ArrayList<String> motsAVerifier;
        private static final int[] POINTS = { 1, 1, 2, 3, 5, 11 };
        private static final int[] LONGUEUR = { 3, 4, 5, 6, 7, 8 };

        public Joueur(String n) {
                this.nom = n;
                this.nbPoints = 0;
                this.motsTrouves = "";
                this.motsAVerifier = new ArrayList<String>();
        }

        public int getNbPoints() {
                return nbPoints;
        }

        public void setNbPoints(int nbPoints) {
                this.nbPoints = nbPoints;
        }

        public String getNom() {
                return nom;
        }

        public String getMotsTrouves() {
                return motsTrouves;
        }

        public void setMotsTrouves(String motsTrouves) {
                this.motsTrouves = motsTrouves;
        }

        public ArrayList<String> getMotsAVerifier() {
                return motsAVerifier;
        }
```

```
	public void setMotsAVerifier(ArrayList<String> arrayList) {
		this.motsAVerifier = arrayList;
	}

	public void setArrayListData() {
		String[] mots = motsTrouves.split(" ");
		for (int i = 0; i < mots.length; i++) {
			motsAVerifier.add(mots[i]);
		}
	}

	public int compterMots() {
		boolean motPresent = false;
		int nombreMots = 0;
		int finDeLigne = motsTrouves.length() - 1;
		for (int i = 0; i < motsTrouves.length(); i++) {
			if (Character.isLetter(motsTrouves.charAt(i)) && i != finDeLigne) {
				motPresent = true;
			} else if (!Character.isLetter(motsTrouves.charAt(i)) && motPresent)
{
				nombreMots++;
				motPresent = false;
			} else if (Character.isLetter(motsTrouves.charAt(i)) && i ==
finDeLigne) {
				nombreMots++;
			}
		}
		return nombreMots;
	}

	public void actualiserNbPoints(String mot) {
		for (int i = 0; i < LONGUEUR.length; i++) {
			if (mot.length() == LONGUEUR[i]) {
				this.nbPoints += POINTS[i];
			}
		}
		if (mot.length() > LONGUEUR[LONGUEUR.length - 1]) {
			this.nbPoints += POINTS[POINTS.length - 1];
		}
	}
}
```

6 – Plateau.java

```
import java.util.Random;

public class Plateau {

	private static final String[] FACES_POSSIBLES = { "TUPSEL", "MASROI", "GITNEV",
"YUNLEG", "DECPAM", "KEOTUN",
			"SERLAC", "LIREWU", "EAATOI", "DESTON", "SIHFEE", "NEHRIS",
"TIBRAL", "BOQMAJ", "ZENVAD", "FIXROA" };
	private static int taillePlateau;
	private De[][] Terrain;
	private Boolean[][] Marquage;

	static {
```

```
		taillePlateau = Options.getTaillePlateau();
	}

	public Plateau() {
		int k = 0;
		this.Terrain = new De[taillePlateau][taillePlateau];
		for (int i = 0; i < taillePlateau; i++) {
			for (int j = 0; j < taillePlateau; j++) {
				Terrain[i][j] = new De(FACES_POSSIBLES[k]);
				k++;
				if (k >= FACES_POSSIBLES.length) {
					k = 0;
				}
			}
		}
		Random r = new Random();
		int rand[] = { 0, 0 };
		for (int i = 0; i < (taillePlateau); i++) {
			for (int j = 0; j < (taillePlateau); j++) {
				rand[0] = r.nextInt(taillePlateau);
				rand[1] = r.nextInt(taillePlateau);
				De stockage = new De(Terrain[i][j]);
				Terrain[i][j] = Terrain[rand[0]][rand[1]];
				Terrain[rand[0]][rand[1]] = stockage;
			}
		}
		this.Marquage = new Boolean[taillePlateau][taillePlateau];
		for (int i = 0; i < taillePlateau; i++) {
			for (int j = 0; j < taillePlateau; j++) {
				Marquage[i][j] = false;
			}
		}
	}

	public Plateau(String faces) {
		int k = 0;
		this.Terrain = new De[taillePlateau][taillePlateau];
		for (int i = 0; i < taillePlateau; i++) {
			for (int j = 0; j < taillePlateau; j++) {
				Terrain[i][j] = new De(faces.toUpperCase().charAt(k));
				k++;
			}
		}
		this.Marquage = new Boolean[taillePlateau][taillePlateau];
		for (int i = 0; i < taillePlateau; i++) {
			for (int j = 0; j < taillePlateau; j++) {
				Marquage[i][j] = false;
			}
		}
	}

	public static int getTaillePlateau() {
		return taillePlateau;
	}

	public De[][] getTerrain() {
		return Terrain;
	}
```

```
	public Boolean[][] getMarquage() {
		return Marquage;
	}

	public boolean recherche(String mot) {
		setMarquageDefaut();
		for (int i = 0; i < taillePlateau; i++) {
			for (int j = 0; j < taillePlateau; j++) {
				Coord coord = new Coord(i, j);
				if (sousRecherche(mot, 0, coord)) {
					return true;
				}
			}
		}
		return false;
	}

	public boolean sousRecherche(String mot, int pos, Coord coord) {
		if (pos >= mot.length()) {
			return true;
		}
		if (!existe(coord)) {
			return false;
		}
		if (!correspondance(mot, pos, coord)) {
			return false;
		}
		if (dejaVisitee(coord)) {
			return false;
		}
		setVisitee(coord);
		Coord c = new Coord(0, 0);
		for (int x = -1; x <= 1; x++) {
			for (int y = -1; y <= 1; y++) {
				c.setCoordI(x);
				c.setCoordJ(y);
				if (x == 0 && y == 0) {
					continue;
				}
				if (sousRecherche(mot, pos + 1, coord.add(c))) {
					return true;
				}
			}
		}
		setNonVisitee(coord);
		return false;
	}

	public boolean existe(Coord coord) {
		if (coord.getCoordI() >= 0 && coord.getCoordI() < taillePlateau &&
coord.getCoordJ() >= 0
				&& coord.getCoordJ() < taillePlateau) {
			return true;
		} else {
			return false;
		}
	}

	public boolean correspondance(String mot, int pos, Coord coord) {
```

```
			String face = Character.toString(Terrain[coord.getCoordI()]
[coord.getCoordJ()].getFaceVisible());
			String lettre = Character.toString(mot.charAt(pos));
			return lettre.equalsIgnoreCase(face);
		}

		public boolean dejaVisitee(Coord coord) {
			return Marquage[coord.getCoordI()][coord.getCoordJ()] == true;
		}

		public void setVisitee(Coord coord) {
			Marquage[coord.getCoordI()][coord.getCoordJ()] = true;
		}

		public void setNonVisitee(Coord coord) {
			Marquage[coord.getCoordI()][coord.getCoordJ()] = false;
		}

		public void setMarquageDefaut() {
			for (int i = 0; i < taillePlateau; i++) {
				for (int j = 0; j < taillePlateau; j++) {
					Marquage[i][j] = false;
				}
			}
		}

		public String toString() {
			String s = new String();
			s = "";
			for (int i = 0; i < taillePlateau; i++) {
				for (int j = 0; j < taillePlateau; j++) {
					s += Terrain[i][j].getFaceVisible();
					if (j < taillePlateau - 1) {
						s = s + " ";
					}
				}
				if (i < taillePlateau - 1) {
					s += System.lineSeparator();
				}
			}
			return s;
		}

}
```

7 – De.java

```
import java.util.Random;

public class De {

		private char faceVisibleDe;

		public De(String str) {
			Random r = new Random();
			this.faceVisibleDe = str.charAt(r.nextInt(str.length()));
		}
```

```
	public De(char c) {
		this.faceVisibleDe = c;
	}

	public De(De de) {
		this.faceVisibleDe = de.faceVisibleDe;
	}

	public char getFaceVisible() {
		return faceVisibleDe;
	}

}
```

8 – Coord.java

```
public class Coord {

	private int i;
	private int j;

	public Coord(int i, int j) {
		this.i = i;
		this.j = j;
	}

	public int getCoordI() {
		return i;
	}

	public int getCoordJ() {
		return j;
	}

	public void setCoordI(int i) {
		this.i = i;
	}

	public void setCoordJ(int j) {
		this.j = j;
	}

	public Coord add(Coord c) {
		return new Coord(this.i + c.i, this.j + c.j);
	}

}
```

5) Bilan du projet

En fin de compte, nous considérons que nous avons réussi ce premier projet de Java, car nous possédons un Boggle fonctionnel et ne laissant pas de place aux erreurs pouvant être rentrées par un utilisateur malveillant.

De manière générale, nous sommes satisfaits de notre application, mais elle peut potentiellement être améliorée d'un point de vue algorithmique, comme par exemple pour les différents algorithmes de suppression.

Les difficultés principales que nous avons rencontrées se situent de manière générale au niveau du choix de l'architecture globale de l'application (Imbrication des classes, classes utilitaires, etc...). Nous avons aussi eu quelques difficultés lors de l'implantation des algorithmes, encore une fois, de suppression, qui sont complexes dans leur fonctionnement.
Une dernière chose qui à été fastidieuse à été de faire en sorte que l'utilisateur ne puisse pas rentrer les données qu'il veut, mais uniquement celles demandées par le programme.