Rendu du 13 juin 2019 :

· Création d'un espace GitHub.

· Gestion des Exceptions dans les classes *Barriere*, *Coordonnee*, *IA*, *Joueur*, *Partie*, *Pion* & *Plateau*.

· Ajout de Setters dans la classe *Coordonnee.*

· Ajout des méthodes *toString()* et *drawCase()* dans la classe *Plateau* (Ces méthodes ne sont pas terminées).

Classes modifiées :

Classe Barriere

```
package quoridor;

/**
  * Cette classe gère les barrières utilisées par les joueurs
  * @author AlexMO2 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public class Barriere {

    private String COULEUR;
    private Coordonnee coordonnee;

    /**
      * Créé un nouvel objet Barriere
      * @param couleur la couleur du joueur
      * @param coordonnee les coordonnées de la barrière sur le plateau (null si non posée)
      */
    public Barriere(String couleur, Coordonnee coordonnee) {
      try {
        if (couleur == null) {
          throw new Exception("Barriere constructeur - La couleur de la barrière doit
exister.");
        }
        else if (coordonnee == null) {
          throw new Exception("Barriere constructeur - La barrière doit posséder des
coordonnées valides.");
        }
        else {
          this.COULEUR = couleur;
          this.coordonnee = coordonnee;
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }
```

```
    /**
      * Retourne la couleur de la barrière
      * @return la couleur de la barrière
      */
    public String getCouleur() {
        return COULEUR;
    }


    /**
      * Retourne les coordonnées de la barrière
      * @return les coordonnées de la barrière sous la forme d'un objet Coordonnee
      */
    public Coordonnee getCoordonnee() {
        return coordonnee;
    }

    /**
      * Modifie les coordonnées de la barrière
      * @param coordonnee les nouvelles coordonnées de la barrière
      */
    public void setCoordonnee(Coordonnee coordonnee) {
      try {
        if (coordonnee == null) {
          throw new Exception("Barriere setCoordonnee() - Les coordonnees a changer doivent
existe r.");
        }
        else {
          this.coordonnee = coordonnee;
        }
      }
      catch(Exception e) {
        System.err.println();
      }
    }
}
```

Classe Coordonnee

```
package quoridor;

/**
  * Cette classe gère les coordonnées utilisées par les barrières et les pions
  * @author AlexM02 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public class Coordonnee {

    private int x1;
    private int y1;
    private int x2;
    private int y2;
```

```java
    /**
      * Créé un nouvel objet Coordonnee. Si les coordonnées sont utilisées par un pion
alors les valeurs de x2 et y2 seront initialisées à -1 et ne seront pas utilisées.
      * @param x1 La première coordonnée x de l'objet
      * Pour une barrière il s'agit de la coordonnée x de son point de départ
      * @param y1 La première coordonnée y de l'objet
      * Pour une barrière il s'agit de la coordonnée y de son point de départ
      * @param x2 La seconde coordonnée x de l'objet
      * Pour une barrière il s'agit de la coordonnée x de son point d'arrivée
      * @param y2 La seconde coordonnée y de l'objet.
      * Pour une barrière il s'agit de la coordonnée y de son point d'arrivée
      */
    public Coordonnee(int x1, int y1, int x2, int y2) {
      try {
        if ((x1 < 0) || (x1 > 9) || (x2 < -1) || (x2 > 9) || (y1 < 0) || (y1 > 9) || (y2 <
-1) || (y2 > 9)) {
          throw new Exception("Coordonnee constructeur - Les coordonées doivent comprises
entre 0 (ou -1 s'il s'agit d'un pion) et 9.");
        }
        else {
          this.x1 = x1;
          this.y1 = y1;
          this.x2 = x2;
          this.y2 = y2;
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Renvoie la coordonnée X1 de l'objet
      * @return la coordonnée X1
      */
    public int getX1() {
        return x1;
    }

    /**
      * Renvoie la coordonnée Y1 de l'objet
      * @return la coordonnée Y1
      */
    public int getY1() {
        return y1;
    }

    /**
      * Renvoie la coordonnée X2 de l'objet (ou -1 si celui-ci n'en possède pas)
      * @return la coordonnée X2 ou -1
      */
    public int getX2() {
```

```
        return x2;
    }

    /**
      * Renvoie la coordonnée Y2 de l'objet (ou -1 si celui-ci n'en possède pas)
      * @return la coordonnée Y2 ou -1
      */
    public int getY2() {
        return y2;
    }

    /**
      * Re-définie la coordonnée x1 de l'objet
      * @param x1 la coordonée x1 que l'on souhaite attribué à l'objet
      */
    public void setX1(int x1) {
      try {
        if ((x1 < 0) && (x1 > 9)) {
          throw new Exception("Coordonnee setX1() - Les coordonées doivent comprises entre
0 et 9.");
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Re-définie la coordonnée x2 de l'objet
      * @param x2 la coordonée x2 que l'on souhaite attribué à l'objet
      */
    public void setX2(int x2) {
      try {
        if ((x2 < -1) && (x2 > 9)) {
          throw new Exception("Coordonnee setX2() - Les coordonées doivent comprises entre
0 (ou -1 s'il s'agit d'un pion) et 9.");
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Re-définie la coordonnée y1 de l'objet
      * @param y1 la coordonée y1 que l'on souhaite attribué à l'objet
      */
    public void setY1(int y1) {
      try {
        if ((y1 < 0) && (y1 > 9)) {
          throw new Exception("Coordonnee setY1() - Les coordonées doivent comprises entre
0 et 9.");
        }
```

```
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Re-définie la coordonnée y2 de l'objet
      * @param y2 la coordonée y2 que l'on souhaite attribué à l'objet
      */
    public void setY2(int y2) {
      try {
        if ((y2 < -1) && (y2 > 9)) {
          throw new Exception("Coordonnee setY2() - Les coordonées doivent comprises entre
0 (ou -1 s'il s'agit d'un pion) et 9.");
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }
}
```

Classe IA

```
package quoridor;

import java.util.ArrayList;

/**
  * Cette classe gère les joueurs IA
  * @author AlexM02 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public class IA extends Joueur {

    private Difficulte DIFFICULTE;
    private int[][] plusCourtChemin;

    /**
      * Créé un nouvel objet IA
      * @param nom le nom du joueur
      * @param numero le numéro du joueur defini selon l'ordre de création (ex joueur 1 ,
joueur 2 ...)
      * @param couleur la couleur du joueur (indique la forme du pion en mode texte)
      * @param barrieres liste contenant les barrières restantes du joueur
      * @param pion le pion utilisé par le joueur
      * @param plateau le plateau de jeu
      * @param difficulte le niveau de difficulté de cette IA
      */
    public IA(String nom, int numero, String couleur, ArrayList<Barriere> barrieres, Pion
pion, Plateau plateau, Difficulte difficulte) {
```

```
      super(nom, numero, couleur, barrieres, pion, plateau);
      try {
        if (difficulte == null) {
          throw new Exception("Erreur IA(), parametre null");
        }
        else {
            this.DIFFICULTE = difficulte;
        }
      }
      catch (Exception e) {
        System.err.println(e.getMessage());
      }

    }

    /**
      * Retourne la difficulté de l'IA
      * @return la difficulté de l'IA
      */
    public Difficulte getDifficulte() {
        return this.DIFFICULTE;
    }

    /**
      * Retourne le plus court chemin en déplacement de pion pour gagner que l'IA a prévu
      * @return un tableau a deux dimensions contenant le plus court chemin identifié par
l'IA
      */
    public int[][] getPlusCourtChemin() {
        return plusCourtChemin;
    }

    /**
      * Modifie le plus court chemin en déplacement de pion que l'IA prévoie
      * @param plusCourtChemin un tableau a deux dimensions contenant le plus court chemin
que l'IA doit identifier
      */
    public void setPlusCourtChemin(int[][] plusCourtChemin) {
      try {
        if (plusCourtChemin == null) {
          throw new Exception("Erreur setPlusCourtChemin(), parametre null");
        }
        else {
          this.plusCourtChemin = plusCourtChemin;
        }
      }
      catch (Exception e) {
        System.out.println(e.getMessage());
      }
    }

    /**
```

```
      * Identifie le plus court chemin pour chacun des joueurs et planifie les actions de
l'IA en conséquences
      */
    public void plannification() {

    }
}
```

Classe Joueur

```
package quoridor;

import java.util.ArrayList;

/**
  * Classe abstraite gérant les joueurs
  * @author AlexM02 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public abstract class Joueur {

    protected String nom;
    protected int NUMERO;
    protected String COULEUR;
    protected ArrayList<Barriere> barrieres;
    protected Pion pion;
    protected Plateau plateau;

    /**
      * Créé un nouvel objet Humain
      * @param nom le nom du joueur
      * @param numero le numéro du joueur defini selon l'ordre de création (ex joueur 1 ,
joueur 2 ...)
      * @param couleur la couleur du joueur (indique la forme du pion en mode texte)
      * @param barrieres liste contenant les barrières restantes du joueur
      * @param pion le pion utilisé par le joueur
      * @param plateau le plateau de jeu
      */
    public Joueur(String nom, int numero, String couleur, ArrayList<Barriere> barrieres,
Pion pion, Plateau plateau) {
      try {
        if (nom == null) {
          throw new Exception("Joueur constructeur - Le joueur doit avoir un nom.");
        }
        else if ((numero < 0) || (numero > 4)) {
          throw new Exception("Joueur constructeur - Le numéro d'un joueur est compris
entre 0 et 4.");
        }
        else if (couleur == null) {
          throw new Exception("Joueur constructeur - Le joueur doit avoir une couleur.");
        }
        else if (barrieres == null) {
```

```
          throw new Exception("Joueur constructeur - Le joueur doit posséder une liste de
barrières.");
        }
        else if (pion == null) {
          throw new Exception("Joueur constructeur - Le joueur doit posséder un pion qui
existe.");
        }
        else if (plateau == null) {
          throw new Exception("Joueur constructeur - Le joueur doit se trouver sur un
plateau existant");
        }
        else {
          this.nom = nom;
          this.NUMERO = numero;
          this.COULEUR = couleur;
          this.barrieres = barrieres;
          this.pion = pion;
          this.plateau = plateau;
        }
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Retourne le nom du joueur
      * @return le nom du joueur
      */
    public String getNom() {
        return nom;
    }

    /**
      * Retourne le numéro du joueur
      * @return le numero du joueur
      */
    public int getNumero() {
        return this.NUMERO;
    }

    /**
      * Retourne la couleur du joueur
      * @return la couleur du joueur
      */
    public String getCouleur() {
        return this.COULEUR;
    }

    /**
      * Retourne le pion utilisé par le joueur
      * @return le pion utilisé par le joueur
      */
```

```
    public Pion getPion() {
        return this.pion;
    }

    /**
      * Retourne la liste des barrières restantes du joueur
      * @return la liste des barrières restantes du joueur
      */
    public ArrayList<Barriere> getBarrieres() {
        return this.barrieres;
    }

    /**
      * Déplace le pion vers de nouvelles coordonnées
      * si celles-ci sont atteignables
      * @param coordonnee les coordonnées à atteindre
      */
    public void deplacerPion(Coordonnee coordonnee) {
      this.pion.setCoordonnee(coordonnee);
    }

    /**
      * Place une barrière aux coordonnées sélectionnées s'il en reste une au joueur
      * @param coordonnee les coordonnées où placer la barrière
      */
    public void placerBarriere(Coordonnee coordonnee) {
      if (!(this.barrieres.isEmpty())) {
        this.barrieres.get(0).setCoordonnee(coordonnee);
        this.barrieres.remove(0);
      }
    }

    /**
      * Permet au joueur de jouer
      */
    public void jeu() {

    }
}
```

Classe Partie

```
package quoridor;

import java.util.ArrayList;
import java.io.FileNotFoundException;

/**
  * Cette classe gère les différents aspect de la partie
  * @author AlexMO2 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
```

```java
public class Partie {

    private int tour;
    private Mode mode;
    private Plateau plateau;
    private ArrayList<Joueur> joueur;

    /**
      * Créé un nouvel objet Partie
      * @param fileName le nom du fichier de configuration
      */
    public Partie(String fileName) {
      try {
        if (fileName == null) {
          throw new Exception("Partie constructeur - Le nom du fichier doit être valide
pour pouvoir être utilisé.");
        }
        else {

        }
      }
      catch(FileNotFoundException e) {
        System.err.println("Partie construceur - Fichier non trouvé (" + fileName + ")");
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }

    }

  /**
    * Retourne le numéro du tour actuel
    * @return le numéro du tour
    */
  public int getTour() {
        return tour;
    }

    /**
      * Retourne le mode de jeu utilisé
      * @return le mode de jeu utilisé
      */
    public Mode getMode() {
        return mode;
    }

    /**
      * Sauvegarde la partie
      */
    public void sauvegarder() {

    }
```

```java
    /**
      * Charge les données de sauvegarde contenues dans le fichier sélectionné
      * @param filename le fichier contenant les données à charger
      */
    public void charger(String filename) {
      try {
        if (filename == null) {
          throw new Exception("Partie constructeur - Le nom du fichier doit être valide
pour pouvoir être utilisé.");
        }
        else {

        }
      }
      catch(FileNotFoundException e) {
        System.err.println("Partie construceur - Fichier non trouvé (" + filename + ")");
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }

    }

    /**
      * Initialise les différents éléments constants de la partie
      */
    private void initialisation() {

    }

    /**
      * Configure les éléments non constants de la partie à l'aide du fichier de
configuration
      * @param filename le nom du fichier de configuration
      */
    private void configuration(String filename) {
      try {
        if (filename == null) {
          throw new Exception("Partie constructeur - Le nom du fichier doit être valide
pour pouvoir être utilisé.");
        }
        else {

        }
      }
      catch(FileNotFoundException e) {
        System.err.println("Partie construceur - Fichier non trouvé (" + filename + ")");
      }
      catch(Exception e) {
        System.err.println(e.getMessage());
      }
    }
```

```
    /**
      * Lance la partie
      */
    public void start() {

    }

    /**
      * Termine la partie
      */
    public void fin() {

    }
}
```

Classe Pion

```
package quoridor;

/**
  * Cette classe gère les pions utilisés par les joueurs
  * @author AlexM02 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public class Pion {

    private String COULEUR;
    private int[][] deplacementPossibles;
    private Coordonnee coordonnee;


    /**
      * Créé un nouvel objet Pion
      * @param couleur la couleur du joueur (désigne une forme en mode texte)
      * @param coordonnee les coordonnées de départ du pion
      */
    public Pion(String couleur, Coordonnee coordonnee) {
      try {
        if (couleur == null || coordonnee == null) {
          throw new Exception("Erreur Pion(), parametre null");
        }
        else {
          this.COULEUR = couleur;
          this.coordonnee = coordonnee;
        }
      }
      catch (Exception e) {
        System.err.println(e.getMessage());
      }

    }
```

```java
    /**
      * Retourne la couleur du pion
      * @return la couleur du pion
      */
    public String getCouleur() {
        return this.COULEUR;
    }

    /**
      * Retourne les coordonnées du pion
      * @return les coordonnées du pion sous la forme d'un objet Coordonnee
      */
    public Coordonnee getCoordonnee() {
        return this.coordonnee;
    }

    /**
      * Modifie les coordonnées du pion
      * @param coordonnees les nouvelles coordonnées du pion
      */
    public void setCoordonnee(Coordonnee coordonnees) {
      try {
        if (coordonnees == null) {
          throw new Exception("Pion setCoordonnee() - Les coordonnees a changer doivent
exister.");
        }
        else {
          this.coordonnee = coordonnees;
        }
      }
      catch (Exception e) {
        System.err.println(e.getMessage());
      }
    }

    /**
      * Retourne les différents déplacements possibles du pion
      * @return les différents déplacements possibles du pion sous la forme d'un tableau à
deux dimensions
      */
    public int[][] getDeplacementPossibles() {
        return this.deplacementPossibles;
    }

    /**
      * Identifie les nouveaux déplacements possibles du pion avant ou après un déplacement
      */
    public void nextCoup() {

    }
}
```

Classe Plateau

```
package quoridor;
import java.util.ArrayList;

/**
  * Cette classe gère le plateau de jeu
  * @author AlexM02 , Drmarsupial35 , Eclixal , griffin568
  * @version 0.1.0
  */
public class Plateau {

    private int TAILLE;
    private int[][] DAMIER;

    /**
      * Créé un nouvel objet Plateau
      * @param taille la taille du plateau (longueur et largeur car le plateau est
forcément un carré)
      */
    public Plateau(int taille) {
      try {
        if (taille < 2) {
          throw new Exception("Erreur Plateau(), taille trop petite");
        }
        else {
          this.TAILLE = taille;
          this.DAMIER = new int[taille][taille];
        }
      }
      catch (Exception e) {
        System.err.println(e.getMessage());
      }

    }

    /**
      * Retourne la taille du plateau
      * @return la taille du plateau
      */
    public int getTaille() {
        return TAILLE;
    }

  /**
    * Le plateau actuel en ASCII art
    * @return une String avec ces informations
    */
    public String toString(ArrayList<Pion> listePion) {
      ArrayList<ArrayList<String>> cases = new ArrayList<ArrayList<String>>();
      String ret = "\n\n\n";
      ret += "\t\t   1    2    3    4    5    6    7    8    9 ";
      ret += "\t\t                                              ";
```

```
      String[] letters = {"A","B","C","D","E","F","G","H","I"};
      for (int i = 0 ; i < this.TAILLE ; i++) {
        for (int j = 0 ; j < this.TAILLE ; j++) {


        }
      }
      return ret;
    }

  /**
    * Créé les cases dans pour le plateau en mode texte
    * @param pion le pion se trouvant dans la case, null s'il n'y en a aucun
    * @return une case sous la forme d'une String
    */
    private String drawCase() {
      String Acase = "   _ \n";
      Acase += "  | | \n";
      Acase += "   - \n";
      return Acase;
    }

}
```