Reversi 1

TP du module 8 - Les énumérations

Proposition de solution

2 - Création d'une énumération `Pion` :

```
package fr.eni.ecole.reversi;

/**
 * Énumération permettant de définir les états possibles pour une case
 * du plateau de jeu : soit elle est libre, soit elle est occupée
 * par un pion blanc, soit par un pion noir
 *
 * @date 24 sept. 2018
 * @version POO - V1.0
 * @author hboisgontier
 *
 */
public enum Pion {
	LIBRE, BLANC, NOIR;

	// nombre de pions de cette couleur
	private int nombre = 2;

	/**
	 * Getter pour nombre.
	 * @return le nombre de pions de cette couleur
	 */
	public int getNombre() {
		return nombre;
	}

	/**
	 * Donne le symbole utilisé pour afficher ce pion
	 * @return le symbole utilisé pour afficher ce pion
	 */
	public char getSymbole() {
		char ret;
		switch (this) {
		case BLANC:
			ret = 'o';
			break;
		case NOIR:
			ret = '●';
			break;
		default:
			ret = '.';
			break;
		}
		return ret;
	}
```

```
	/**
	 * Méthode pour connaître le pion opposé :
	 * Blanc pour Noir et Noir pour Blanc
	 * @return le pion opposé
	 */
	public Pion autrePion() {
		Pion autre;
		switch (this) {
		case BLANC:
			autre = Pion.NOIR;
			break;
		case NOIR:
			autre = Pion.BLANC;
			break;
		default:
			autre = Pion.LIBRE;
			break;
		}
		return autre;
	}

	/**
	 * Modifie l'attribut nombre pour ajouter le nombre de pions acquis
	 * grâce au coup du joueur
	 * L'adversaire perd ce même nombre de pions
	 * L'attribut nombre de ce pion est augmenté de 1 en raison du pion
	 * posé par le joueur
	 * @param nombre
	 *            le nombre de pions qui changent de couleur
	 *            suite à un coup effectué par un joueur
	 */
	public void gagne(int nombre) {
		this.nombre += nombre + 1;
		this.autrePion().nombre -= nombre;
	}
}
```

Cette énumération ne se contente pas de définir un type avec un ensemble de valeurs possibles, mais utilise le fait qu'en Java une énumération est une classe dans laquelle il est possible de définir des attributs, des constructeurs, des méthodes…

Pour la méthode **gagne**, il ne faut pas oublier que tous les pions qui sont gagnés par un joueur sont perdus pour l'autre. Il faut donc mettre à jour l'attribut **nombre** pour les deux joueurs. Un autre élément à ne pas oublier dans le compte est que le joueur pose un pion sur le plateau de jeu pour en capturer d'autres. Il ne faut donc pas oublier de faire **+ 1** !

3 - Création de la classe `PlateauDeReversi` :

```
package fr.eni.ecole.util;

import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * Classe utilitaire proposant des méthodes simplifiant la saisie de valeurs
 * de l'utilisateur sur la console
 * @date 4 juil. 2018
```

```
 * @version Outils - V1.0
 * @author hboisgontier
 */
public class Outils {
	private static Scanner s = new Scanner(System.in);

	/**
	 * Fait saisir à l'utilisateur une valeur comprise entre les valeurs
	 * {@code min} et {@code max} passées en paramètre. En cas d'erreur,
	 * l'utilisateur devra ressaisir jusqu'à ce que la valeur soit correcte.
	 * @param message
	 *            message à afficher à l'utilisateur pour l'inviter à saisir
	 * @param min
	 *            valeur minimale acceptable (incluse)
	 * @param max
	 *            valeur maximale acceptable (incluse)
	 * @return la valeur saisie par l'utilisateur
	 */
	public static int saisie(String message, int min, int max) {
		System.out.printf("%s ", message);
		int val = 0;
		boolean ok;
		do {
			try {
				val = Outils.s.nextInt();
				ok = val >= min && val <= max;
			} catch (InputMismatchException e) {
				ok = false;
			} finally {
				Outils.s.nextLine();
			}
			if (!ok)
				System.err.printf(
    "La valeur doit être un entier compris entre %d et %d%nRessaisissez... ", min, max);
		} while (!ok);
		return val;
	}
}
```

```
package fr.eni.ecole.reversi;

import fr.eni.ecole.util.Outils;

/**
 * Modélise un plateau de jeu permettant de jouer au jeux du Reversi.
 *
 * @date 24 sept. 2018
 * @version POO - V1.0
 * @author hboisgontier
 *
 */
public class PlateauDeReversi {
	public static final int TAILLE = 8;
	private Pion[][] plateau;

	/**
	 * Constructeur : Initialise le plateau de jeu
```

```
	 * dans la configuration de début de partie <br>
	 * · · · · · · · · <br>
	 * · · · · · · · · <br>
	 * · · · · · · · · <br>
	 * · · · o ● · · · <br>
	 * · · · ● o · · · <br>
	 * · · · · · · · · <br>
	 * · · · · · · · · <br>
	 * · · · · · · · ·
	 */
	private PlateauDeReversi() {
		this.plateau = new Pion[TAILLE][TAILLE];
		for (int j = 0; j < TAILLE; j++) {
			for (int i = 0; i < TAILLE; i++) {
				plateau[j][i] = Pion.LIBRE;
			}
		}
		this.plateau[TAILLE / 2][TAILLE / 2] = Pion.BLANC;
		this.plateau[TAILLE / 2 - 1][TAILLE / 2] = Pion.NOIR;
		this.plateau[TAILLE / 2][TAILLE / 2 - 1] = Pion.NOIR;
		this.plateau[TAILLE / 2 - 1][TAILLE / 2 - 1] = Pion.BLANC;
	}

	/**
	 * Méthode permettant d'effectuer une partie de Reversi
	 */
	private void jouer() {
		Pion courant = Pion.NOIR;
		int nbPasseTour = 0;
		while (nbPasseTour < 2
		     && Pion.BLANC.getNombre() + Pion.NOIR.getNombre() < TAILLE * TAILLE) {
			System.out.printf("Au tour de %s...%n", courant.getSymbole());
			int nbRetournes = 0;
			boolean ok = false;
			do {
				this.afficher();
				if (this.peutJouer(courant)) {
					int ligne = Outils.saisie("ligne", 1, TAILLE)-1;
					int colonne = Outils.saisie("colonne", 1, TAILLE)-1;
					nbRetournes = this.tester(courant, ligne, colonne);
					if (nbRetournes > 0) {
						this.poser(courant, ligne, colonne);
						courant.gagne(nbRetournes);
						nbPasseTour = 0;
						ok = true;
					} else {
						System.err.println("Position illégale");
					}
				} else {
					System.out.printf(
		"%s n'a aucune position où poser un de ses pions. Il passe son tour.%n",
							courant.getSymbole());
					nbPasseTour++;
					ok = true;
				}
			} while (!ok);
			// changement de joueur
			courant = courant.autrePion();
```

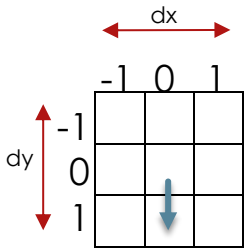
```
		}
		if (Pion.BLANC.getNombre() > Pion.NOIR.getNombre()) {
			System.out.printf("%s gagne !%n", Pion.BLANC.getSymbole());
		} else if (Pion.BLANC.getNombre() < Pion.NOIR.getNombre()) {
			System.out.printf("%s gagne !%n", Pion.NOIR.getSymbole());
		} else {
			System.out.println("Égalité !");
		}
		this.afficher();
	}

	/**
	 * Affiche le plateau de jeu et le score
	 */
	private void afficher() {
		// Affichage des scores
		System.out.printf("%2d %s%n%2d %s%n", Pion.NOIR.getNombre(),
		  Pion.NOIR.getSymbole(), Pion.BLANC.getNombre(), Pion.BLANC.getSymbole());
		// Affichage du plateau de jeu
		System.out.print("  ");
		for (int i = 1; i <= TAILLE; i++)
			System.out.printf("%d ", i);
		System.out.println();
		for (int j = 0; j < TAILLE; j++) {
			System.out.printf("%d ", j + 1);
			for (int i = 0; i < TAILLE; i++) {
				System.out.printf("%s ", plateau[j][i].getSymbole());
			}
			System.out.println();
		}
	}

	/**
	 * Indique le nombre de pions changeant de couleur si le joueur
	 * ayant les pions de la couleur passée en paramètre pose un pion
	 * sur la case dont les coordonnées sont passées en paramètre
	 *
	 * @param couleur
	 *            couleur du pion qui serait posé
	 * @param y
	 *            ligne pour le pion
	 * @param x
	 *            colonne pour le pion
	 * @return le nombre de pions qui changeraient de couleur
	 */
	public int tester(Pion couleur, int y, int x) {
		int nbPions = 0;
		if (plateau[y][x] == Pion.LIBRE)
			for (int dy = -1; dy <= 1; dy++)
				for (int dx = -1; dx <= 1; dx++)
					if (dx != 0 || dy != 0)
						nbPions += testerDirection(couleur,y,x,dy,dx);
		return nbPions;
	}

	/**
	 * Indique le nombre de pions changeant de couleur si le joueur
	 * ayant les pions de la couleur passée en paramètre pose un pion
```

```
     * sur la case dont les coordonnées sont passées en paramètre
     * pour une direction donnée avec les paramètres dx et dy
     *
     * @param couleur
     * @param y
     *            ligne pour le pion
     * @param x
     *            colonne pour le pion
     * @param dy
     *            -1 pour indiquer un décalage vers la gauche, 1 vers la droite,
     *            et 0 pas de décalage
     * @param dx
     *            -1 pour indiquer un décalage vers le haut, 1 vers le bas,
     *            et 0 pas de décalage
     * @return le nombre de pions qui changeraient de couleur
     */
    private int testerDirection(Pion couleur, int y, int x, int dy, int dx) {
            Pion couleurOpp = couleur.autrePion();
            int nbAutres = 0;
            int i = x + dx;
            int j = y + dy;
            while (0 <= i && i < TAILLE && 0 <= j && j < TAILLE &&
                                                    this.plateau[j][i] == couleurOpp) {
                    nbAutres++;
                    i += dx;
                    j += dy;
            }
            if (i < 0 || i >= TAILLE || j < 0 || j >= TAILLE ||
                                                            this.plateau[j][i] != couleur)
                    nbAutres = 0;
            return nbAutres;
    }

    /**
     * Cherche s'il y a au moins une case sur laquelle le joueur
     * passé en paramètre peut jouer
     *
     * @param joueur
     *            le joueur dont c'est le tour de jeu
     * @return s'il y a au moins une case sur laquelle le joueur
     *         passé en paramètre peut jouer
     */
    private boolean peutJouer(Pion joueur) {
            boolean positionJouable = false;
            int j = 0;
            while (j < TAILLE && !positionJouable) {
                    int i = 0;
                    while (i < TAILLE && !positionJouable) {
                            positionJouable = tester(joueur, j, i) > 0;
                            i++;
                    }
                    j++;
            }
            return positionJouable;
    }

    /**
     * Pose un pion sur le plateau et change la couleur des pions
```

```
         * nouvellement encadrés entre ce pion et un autre de la même couleur
         * dans toutes les directions
         *
         * @param couleur
         *            couleur du pion à poser
         * @param y
         *            ligne où poser le pion
         * @param x
         *            colonne où poser le pion
         */
        private void poser(Pion couleur, int y, int x) {
                this.plateau[y][x] = couleur;
                int nbPions;
                for (int dy = -1; dy <= 1; dy++)
                        for (int dx = -1; dx <= 1; dx++) {
                                nbPions = 0;
                                if (dx != 0 || dy != 0) {
                                        nbPions += testerDirection(couleur, y, x, dy, dx);
                                        for (int k = 1; k <= nbPions; k++)
                                                this.plateau[y + dy * k][x + dx * k] = couleur;
                                }
                        }
        }

        /**
         * Méthode principale : lance une partie de Reversi
         *
         * @param args Rien du tout
         */
        public static void main(String[] args) {
                PlateauDeReversi p = new PlateauDeReversi();
                p.jouer();
        }
}
```

La principale difficulté consiste à tester une position et à savoir combien de pions sont capturés. Pour cela, deux méthodes sont utilisées : **`tester()`** et **`testerDirection()`**.

La méthode **`testerDirection()`** se contente de tester l'effet de la pose du pion pour une seule direction. Les paramètres **`dx`** et **`dy`** sont utilisés pour indiquer la direction à tester. Ils indiquent le décalage suivant l'axe des x (horizontal) et l'axe des y (vertical). **`dx=0`** et **`dy=1`** par exemple correspond à la direction vers le bas.



Le tableau est parcouru à partir des coordonnées de départ (x et y) dans cette direction jusqu'à ce que les pions ne soient plus des pions de l'adversaire. Si le pion suivant est un pion de ce joueur, tous les pions parcourus seront capturés ; sinon, aucun pion ne sera capturé dans cette direction.

La méthode **`tester()`** fait appel à la méthode **`testerDirection()`** pour l'ensemble des huit directions possibles.

La méthode **`poser()`** est basée sur le même principe, sauf que les pions capturés changent de couleur.

La méthode `peutJouer()` fait appel à la méthode `tester()` sur chacune des cases du tableau jusqu'à trouver une case où il est possible de poser un pion et, dans ce cas, retourne vrai. Si l'ensemble du plateau de jeu est parcouru sans trouver une telle case, alors la méthode retourne faux.

4 - Les différents types de joueurs :

```
package fr.eni.ecole.reversi;

/**
 * Interface représentant un joueur pour le Reversi
 *
 * @date 26 sept. 2018
 * @version POO - V1.0
 * @author hboisgontier
 *
 */
public interface Joueur {
	static final int LIGNE = 0;
	static final int COLONNE = 1;

	/**
	 * Méthode retournant les coordonnées où le joueur souhaite
	 * positionner son pion
	 *
	 * @param p
	 *            le plateau de jeu (la seule méthode d'instance publique
	 *            qu'il est possible d'appeler est la méthode tester() indiquant
	 *            la possibilité de poser son pion et le gain associé).
	 * @param couleur
	 *            la couleur du pion du joueur
	 * @return un couple de coordonnées
	 *         la case de coordonnée {@code LIGNE} contient la ligne choisie
	 *         et la case de coordonnée {@code COLONNE} contient la colonne choisie.
	 *         Ces coordonnées sont basées à partir de 0.
	 */
	int[] jouer(PlateauDeReversi p, Pion couleur);

	/**
	 * Donne le nom du joueur
	 * @return le nom du joueur
	 */
	String getNom();
}
```

```
package fr.eni.ecole.reversi;

import fr.eni.ecole.util.Outils;

/**
```

```
 * Classe modélisant un joueur humain pour le jeu du Reversi.
 * C'est le joueur qui renseigne les coordonnées pour positionner
 * son pion à chaque tour de jeu.
 *
 * @date 26 sept. 2018
 * @version POO - V1.0
 * @author hboisgontier
 *
 */
public class JoueurHumain implements Joueur {

	private String nom;

	/**
	 * Constructeur
	 */
	public JoueurHumain() {
		this.nom = Outils.saisie("Nom du joueur ?");
	}

	/**
	 * {@inheritDoc}
	 *
	 * @see fr.eni.ecole.reversi.Joueur#jouer(fr.eni.ecole.reversi.PlateauDeReversi)
	 */
	@Override
	public int[] jouer(PlateauDeReversi p, Pion couleur) {
		int[] coordonnees = new int[2];
		coordonnees[LIGNE] = Outils.saisie("ligne", 1, PlateauDeReversi.TAILLE)-1;
		coordonnees[COLONNE]=Outils.saisie("colonne",1,PlateauDeReversi.TAILLE)-1;
		return coordonnees;
	}

	/**
	 * {@inheritDoc}
	 *
	 * @see fr.eni.ecole.reversi.Joueur#getNom()
	 */
	@Override
	public String getNom() {
		return this.nom;
	}
}
```

```
package fr.eni.ecole.reversi;

import java.util.Random;

/**
 * Classe modélisant un joueur ordinateur pour le jeu du Reversi.
 * Cet IA n'a d'IA que le nom puisqu'elle choisit aléatoirement
 * la position de son pions jusqu'à trouver une position légale.
 *
 * @date 26 sept. 2018
 * @version POO - V1.0
 * @author hboisgontier
 *
```

```
 */
public class IAalea implements Joueur {

	private static Random r = new Random();

	/**
	 * {@inheritDoc}
	 * @see fr.eni.ecole.reversi.Joueur#jouer(fr.eni.ecole.reversi.PlateauDeReversi)
	 */
	@Override
	public int[] jouer(PlateauDeReversi p, Pion couleur) {
		int[] coord = new int[2];
		do {
			coord[LIGNE] = r.nextInt(PlateauDeReversi.TAILLE);
			coord[COLONNE] = r.nextInt(PlateauDeReversi.TAILLE);
		} while(p.tester(couleur, coord[LIGNE], coord[COLONNE])==0);
		return coord;
	}

	/**
	 * {@inheritDoc}
	 * @see fr.eni.ecole.reversi.Joueur#getNom()
	 */
	@Override
	public String getNom() {
		return "IA Aléatoire";
	}

}
```

```
public enum Pion {
		…
	// joueur associé à cette couleur
	private Joueur joueur;

	/**
	 * Getter pour joueur.
	 *
	 * @return le joueur associé à ce pion
	 */
	public Joueur getJoueur() {
		return joueur;
	}

	public void choixJoueur() {
		String m = String.format("Quel joueur pour les pions %s ?%n " +
					"1 - Humain%n 2 - Ordinateur", this.getSymbole());
		int c = Outils.saisie(m, 1, 2);
		if (c == 1)
			this.joueur = new JoueurHumain();
		else
			this.joueur = new IAalea();
	}

}
```

```
public class PlateauDeReversi {
		…
	/**
	 * Méthode permettant d'effectuer une partie de Reversi
	 */
	private void jouer() {
		Pion.NOIR.choixJoueur();
		Pion.BLANC.choixJoueur();
		Pion courant = Pion.NOIR;
		int nbPasseTour = 0;
		while (nbPasseTour < 2 &&
			Pion.BLANC.getNombre() + Pion.NOIR.getNombre() < TAILLE * TAILLE) {
			System.out.printf("Au tour de %s (%s)...%n",
					courant.getJoueur().getNom(), courant.getSymbole());
			int nbRetournes = 0;
			boolean ok = false;
			do {
				this.afficher();
				if (this.peutJouer(courant)) {
					int[] coord = courant.getJoueur().jouer(this,courant);
					nbRetournes = this.tester(courant,
							coord[Joueur.LIGNE], coord[Joueur.COLONNE]);
					if (nbRetournes > 0) {
						this.poser(courant, coord[Joueur.LIGNE],
								coord[Joueur.COLONNE]);
						courant.gagne(nbRetournes);
						nbPasseTour = 0;
						ok = true;
					} else {
						System.err.println("Position illégale");
					}
				} else {
					System.out.printf("%s n'a aucune position où poser " +
				"un de ses pions. Il passe son tour.%n", courant.getSymbole());
					nbPasseTour++;
					ok = true;
				}
			} while (!ok);
			// changement de joueur
			courant = courant.autrePion();
		}
		if (Pion.BLANC.getNombre() > Pion.NOIR.getNombre()) {
			System.out.printf("%s (%s) gagne !%n",
				Pion.BLANC.getJoueur().getNom(), Pion.BLANC.getSymbole());
		} else if (Pion.BLANC.getNombre() < Pion.NOIR.getNombre()) {
			System.out.printf("%s (%s) gagne !%n",
				Pion.NOIR.getJoueur().getNom(), Pion.NOIR.getSymbole());
		} else {
			System.out.println("Égalité !");
		}
		this.afficher();
	}
		…
}
```

```
/**
 * Classe utilitaire proposant des méthodes simplifiant la saisie de valeurs
```

```
 * de l'utilisateur sur la console et la génération de valeurs aléatoires
 * @date 4 juil. 2018
 * @version Outils - V1.0
 * @author hboisgontier
 *
 */
public class Outils {
	private static Scanner s = new Scanner(System.in);

		…

	public static String saisie(String message) {
		System.out.println(message);
		return Outils.s.nextLine();
	}
}
```

Dans l'interface `Joueur`, deux constantes ont été déclarées pour éviter l'ambigüité entre les lignes et les colonnes dans le tableau retourné par la méthode `jouer()`.

Les classes `JoueurHumain` et `IAalea` implémentent l'interface `Joueur`. Elles implémentent donc les méthodes `jouer()` et `getNom()`. La méthode `jouer()` de la classe `JoueurHumain` demande à l'utilisateur de saisir les coordonnées de la case où déposer son pion alors que celle de `IAalea` tire des coordonnées aléatoirement jusqu'à trouver une case valide. Cette manière de procéder est vraiment naïve ; il est tout à fait possible de réaliser des actions plus évoluées en cherchant à privilégier des coups maximisant les captures de pions ou permettant de positionner son pion sur une case stratégique…

L'énumération a été quelque peu modifiée pour associer un joueur à une couleur de pion.