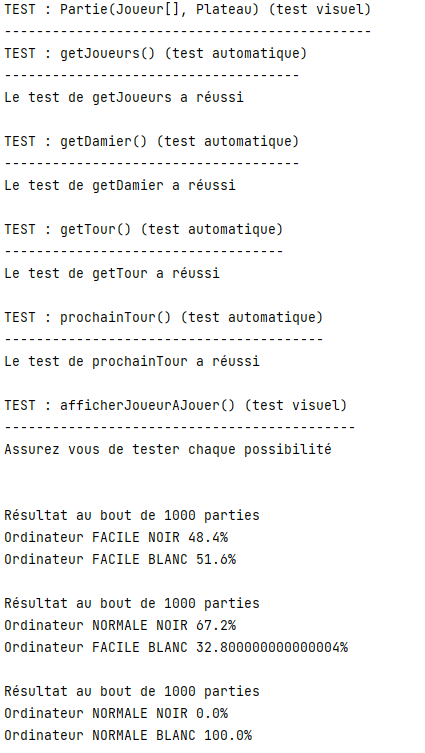
# Code Source

*/\*  
 \* TestPartie.java 27 mai 2021  
 \* IUT Rodez, pas de droits  
 \*/*package othello.tests;  
  
import othello.Partie;  
import othello.Plateau;  
import othello.caracteristiques.Couleur;  
import othello.joueur.Joueur;  
import othello.joueur.Ordinateur;  
  
*/\*\*  
 \* Tests unitaires de la classe Partie  
 \*  
 \* @author SIMON LAUNAY  
 \* YOHANN MAY  
 \* THEO MICHELLON  
 \* MAXIME MOSKALYK  
 \* DAVID SIMONIN  
 \* @version 1.0  
 \*/*public class TestPartie {  
  
 */\*\*  
 \* jeu d'essai avec un plateau initial  
 \*/* public static Plateau *damier* = new Plateau();  
 static Joueur[] *joueurs* = {  
 new Joueur("Ghalhyus", Couleur.*NOIR*),  
 new Joueur("CRAZYMat", Couleur.*BLANC*)  
 };  
 */\*\*  
 \* test du constructeur Partie Jeu de données pour les prochains test  
 \*/* public final static Partie *VALIDE* = new Partie(*joueurs*[0], *joueurs*[1],  
 *damier*);  
  
 */\*\*  
 \* Test du constructeur Partie avec des valeurs invalides  
 \*/* public static void testPartieJoueursPlateau() {  
 final Joueur[][] JOUEURS\_INVALIDES = {  
 null,  
 { new Joueur("Jean", Couleur.*NOIR*),  
 new Joueur("CRAZYMat", Couleur.*BLANC*) },  
 null,  
 { new Joueur("Ghalhyus", Couleur.*NOIR*),  
 new Joueur("Mausaille", Couleur.*NOIR*) },  
 { new Joueur("moska", Couleur.*BLANC*),  
 new Joueur("Guepaz", Couleur.*BLANC*) }  
 };  
  
 final Plateau[] DAMIERS\_INVALIDES = {  
 new Plateau(),  
 null,  
 null,  
 new Plateau(),  
 new Plateau()  
 };  
  
 final int TAILLE = JOUEURS\_INVALIDES.length;  
  
 int test = 0;  
  
 System.*out*.print("""  
 TEST : Partie(Joueur[], Plateau) (test visuel)  
 ----------------------------------------------  
 """);  
 */\* test avec des valeurs invalides \*/* for (int jdd = 0; jdd < TAILLE && jdd == test; jdd++) {  
 try {  
 new Partie(JOUEURS\_INVALIDES[jdd][0],  
 JOUEURS\_INVALIDES[jdd][1], DAMIERS\_INVALIDES[jdd]);  
 System.*out*.println("Erreur, l'un des tests a échoué\n");  
 } catch (RuntimeException echec) {  
 *// Test Ok* }  
 }  
 }  
  
 */\*\*  
 \* Test getJoueur  
 \*/* public static void testGetJoueurs() {  
 */\* jeu de donnée \*/* final Joueur[] COPIE\_JOUEUR = {  
 new Joueur("Ghalhyus", Couleur.*NOIR*),  
 new Joueur("CRAZYMat", Couleur.*BLANC*)  
 };  
  
 final int TAILLE = *VALIDE*.getJoueurs().length;  
  
 int jdd;  
  
 System.*out*.print("""  
 TEST : getJoueurs() (test automatique)  
 -------------------------------------  
 """);  
  
 for (jdd = 0; jdd < TAILLE && COPIE\_JOUEUR[jdd]  
 .equals(*VALIDE*.getJoueurs()[jdd]);  
 jdd++)  
 ; *// corps vide* if (jdd == TAILLE) {  
 System.*out*.println("Le test de getJoueurs a réussi\n");  
 } else {  
 System.*out*.println("Erreur, le test a échoué\n");  
 }  
  
 }  
  
 */\*\*  
 \* Test getDamier  
 \*/* public static void testGetDamier() {  
 */\* jeu de donnée \*/* final Plateau COPIE\_DAMIER = new Plateau();  
  
 System.*out*.print("""  
 TEST : getDamier() (test automatique)  
 -------------------------------------  
 """);  
  
 if (COPIE\_DAMIER.equals(*VALIDE*.getDamier())) {  
 System.*out*.println("Le test de getDamier a réussi\n");  
 } else {  
 System.*out*.println("Erreur, le test a échoué\n");  
 }  
 }  
  
 */\*\*  
 \* Test getTour  
 \*/* public static void testGetTour() {  
 System.*out*.print("""  
 TEST : getTour() (test automatique)  
 -----------------------------------  
 """);  
  
 if (0 == *VALIDE*.getTour()) {  
 System.*out*.println("Le test de getTour a réussi\n");  
 } else {  
 System.*out*.println("Erreur, le test a échoué\n");  
 }  
 }  
  
 */\*\*  
 \* Test prochainTour()  
 \*/* public static void testProchainTour() {  
 System.*out*.print("""  
 TEST : prochainTour() (test automatique)  
 ----------------------------------------  
 """);  
  
 *VALIDE*.prochainTour();  
 if (*VALIDE*.getTour() == 1) {  
 System.*out*.println("Le test de prochainTour a réussi\n");  
 } else {  
 System.*out*.println("Erreur, le test a échoué\n");  
 }  
 }  
  
 */\*\*  
 \* test de la méthode jouer()  
 \*/* public static void testJouer() {  
 System.*out*.println("""  
 TEST : afficherJoueurAJouer() (test visuel)  
 --------------------------------------------  
 Assurez vous de tester chaque possibilité  
 """);  
 }  
  
 */\*\*  
 \* Probabilités d'efficacite des ordinateurs  
 \*/* public static void probaOrdinateur() {  
 *lancerPartie*(Ordinateur.Difficulte.*FACILE*,  
 Ordinateur.Difficulte.*FACILE*);  
  
  
 *lancerPartie*(Ordinateur.Difficulte.*FACILE*,  
 Ordinateur.Difficulte.*NORMALE*);  
  
  
 *lancerPartie*(Ordinateur.Difficulte.*NORMALE*,  
 Ordinateur.Difficulte.*NORMALE*);  
 }  
  
 */\*\*  
 \* Lance une partie entre deux ordinateurs  
 \* @param diffOrdi1 difficulté du premier ordinateur  
 \* @param diffOrdi2 difficulté du second ordinateur  
 \*/* public static void lancerPartie(Ordinateur.Difficulte diffOrdi1,  
 Ordinateur.Difficulte diffOrdi2) {  
 int[] stats = {  
 0, 0  
 };  
 double nbPartie = 1000;  
 Ordinateur o1 = new Ordinateur(Couleur.*random*(), diffOrdi1);  
 Ordinateur o2 = new Ordinateur(o1.getCouleur().opposee(), diffOrdi2);  
  
 for (int i = 0; i < nbPartie; i++) {  
 Plateau damier = new Plateau();  
 Partie partie = new Partie(o1, o2, damier);  
 do {  
 partie.jouer(null);  
 partie.prochainTour();  
 } while (partie.isJouable());  
 if (partie.getVainqueur().getCouleur() == Couleur.*NOIR*) {  
 stats[0]++;  
 } else {  
 stats[1]++;  
 }  
 }  
  
 System.*out*.println("\nRésultat au bout de " + (int) nbPartie  
 + " parties");  
 if (o1.getCouleur() == Couleur.*NOIR*) {  
 System.*out*.println(o1.getPseudo() + " " + o1.getDifficulte() + " "  
 + o1.getCouleur() + " " + stats[0] / nbPartie \* 100 + "%");  
 System.*out*.println(o2.getPseudo() + " " + o2.getDifficulte() + " "  
 + o2.getCouleur() + " " + stats[1] / nbPartie \* 100 + "%");  
 } else {  
 System.*out*.println(o2.getPseudo() + " " + o2.getDifficulte() + " "  
 + o2.getCouleur() + " " + stats[0] / nbPartie \* 100 + "%");  
 System.*out*.println(o1.getPseudo() + " " + o1.getDifficulte() + " "  
 + o1.getCouleur() + " " + stats[1] / nbPartie \* 100 + "%");  
 }  
 }  
  
 */\*\*  
 \* Lance les tests unitaires  
 \*  
 \* @param args non utilisé  
 \*/* public static void main(String[] args) {  
 *testPartieJoueursPlateau*();  
 *testGetJoueurs*();  
 *testGetDamier*();  
 *testGetTour*();  
 *testProchainTour*();  
 *testJouer*();  
 *probaOrdinateur*();  
 }  
}

# Résultats



# Explications

Afin de réaliser des tests par la suite nous avons créer un jeu de donnée contenant un plateau et une partie valide.

1. Partie(Joueur, Joueur, Plateau)

Afin de réaliser des tests sur le constructeur nous avons créer deux tableaux contenants des joueurs invalides et un second avec des plateaux invalides.

Par la suite nous avons tenté à l’aide d’une boucle for de créer des parties invalides.

1. getJoueur

Afin de réaliser des tests nous avons créer un tableau contenant deux joueurs attendus. Pour la suite nous avons vérifié que le getJoueur renvoyé bien les joueurs attendus récupéré à l’aide de la méthode getJoueur sur le tableau contenant des joueurs valides.

1. getDamier

Afin de réaliser des tests nous avons créer un plateau initialisé par défaut puis nous avons vérifié que le plateau valide correspondait bien au damier attendu.

1. getTour

Afin de réaliser ce test nous avons vérifié que sur la partie valide initialisé au début, le tour actuel était bien à 0 comme aucun acteur n’a pu jouer.

1. prochainTour

Afin de réaliser ce test nous avons utilisé la même méthodologie que précédemment nous avons vérifié que le prochain tour de la partie valide était bien à 1 car aucun acteur n’a modifié la partie entre temps.

1. jouer

Afin de réaliser ce test nous avons simuler des parties. En effet ce test requiert à l’utilisateur de saisir manuellement des positions car la méthode jouer permet à l’utilisateur de jouer son tour.

1. lancerPartie

Cette méthode créée spécialement pour le test suivant permet de lancer une partie entre deux ordinateurs. Elle n’a ici pour but que de simplifier son utilisation pour la suite.

Nous avons donc créé une partie entre deux ordinateurs puis nous l’avons lancé 1000 fois tout en notant à chaque itération le vainqueur de cette partie (ordinateur1 ou ordinateur2), ensuite un affichage console est effectué pour observer les résultats.

Afin de permettre un peu d’aléatoire l’ordinateur 1 n’est pas forcément celui jouant les noirs et donc l’ordinateur 2 les blancs. Ceci a pu causer quelques soucis pour vérifier la différence entre deux ordinateurs de niveaux différents mais après quelques lancements on peut observer le résultat que les deux ordinateurs possèdent les pions noirs ou blancs.

*PS : Voir les probabilités des ordinateurs*

1. probaOrdinateur

Cette méthode n’est ici que pour vérifier les différents niveaux de l’ordinateur, elle ne permet donc pas de tester une méthode précise de Partie.

Nous avons donc utilisé cette méthode pour réaliser des tests sur les niveaux de l’ordinateur, on obtient ainsi des probabilités quand au taux de victoire de chaque ordinateur.

*PS : Voir les probabilités des ordinateurs*