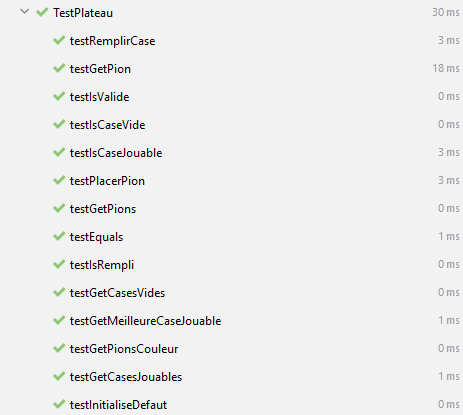
# Code Source

*/\*  
 \* TestPlateau.java 27 Avril 2021  
 \* IUT Rodez, pas de droits  
 \*/*import junit.framework.TestCase;  
import othello.Pion;  
import othello.Plateau;  
import othello.caracteristiques.Couleur;  
import othello.caracteristiques.Position;  
import othello.interfacejeu.console.InterfaceConsole;  
  
import java.util.ArrayList;  
import java.util.LinkedList;  
import java.util.*List*;  
  
import static othello.Plateau.*isValide*;  
  
public class TestPlateau extends TestCase {  
 */\*\* Plateau valide \*/* final static Plateau *damier* = new Plateau();  
  
 */\*\* Tableau contenant des positions invalides \*/* static final Position[] *POSITIONS\_INVALIDES* = {  
 new Position(-1, -1),  
 new Position(-1, 5),  
 new Position(4, -5),  
 new Position(9, 9),  
 new Position(10, 5),  
 new Position(8, 12),  
 new Position(74, -87),  
 new Position(80, 32) };  
  
 */\*\* Tableau contenant des positions valides \*/* static final Position[] *POSITIONS\_VALIDES* = {  
 new Position(7, 7),  
 new Position(0, 0),  
 new Position(3, 4),  
 new Position(4, 3),  
 new Position(3, 3),  
 new Position(4, 4),  
 new Position(4, 5),  
 new Position(4, 7),  
 new Position(2, 1) };  
  
 public void testIsValide() {  
 for (Position invalide : *POSITIONS\_INVALIDES*) {  
 *assertFalse*(*isValide*(invalide));  
 }  
  
 for (Position valide : *POSITIONS\_VALIDES*) {  
 *assertTrue*(*isValide*(valide));  
 }  
 }  
  
 public void testInitialiseDefaut() {  
 Plateau plateau = new Plateau();  
 plateau.initialiseDefaut();  
 *assertEquals*(*damier*, plateau);  
 }  
  
 public void testRemplirCase() {  
 Plateau p = new Plateau();  
 while (p.getPions().size() < 30) {  
 p.remplirCase(new Pion(  
 new Position((int) (Math.*random*() \* Plateau.*TAILLE\_RANGEE*),  
 (int) (Math.*random*() \* Plateau.*TAILLE\_RANGEE*)),  
 Couleur.*values*()[(int) (Math.*random*() \*  
 Couleur.*values*().length)]));  
 }  
 InterfaceConsole.*afficherPlateau*(p, new ArrayList<>());  
 }  
  
 public void testGetPion() {  
 Plateau p = new Plateau();  
  
 */\* Insertion d'un pion en 8H pour réaliser un test ci dessous \*/* p.remplirCase(new Pion(new Position(7, 7), Couleur.*NOIR*));  
 */\* Test sur le pion ajouté en 8H \*/  
 assertEquals*(new Pion(new Position(7, 7), Couleur.*NOIR*),  
 p.getPion(new Position(7, 7)));  
  
 */\* Verification que le pion en 4D soit bien de couleur blanche \*/  
 assertEquals*(new Pion(new Position(3, 3), Couleur.*BLANC*),  
 p.getPion(new Position(3, 3)));  
  
 */\* Verification que le pion en 3D soit bien inexistant (null) \*/  
 assertNull*(p.getPion(new Position(3, 2)));  
 }  
  
 public void testIsCaseVide() {  
 */\* Vérification sur des cases vides \*/  
 assertTrue*(*damier*.isCaseVide(new Position(3, 2)));  
 *assertTrue*(*damier*.isCaseVide(new Position(4, 5)));  
 *assertTrue*(*damier*.isCaseVide(new Position(0, 7)));  
 *assertTrue*(*damier*.isCaseVide(new Position(1, 5)));  
  
 */\* Vérification sur des cases non vides \*/  
 assertFalse*(*damier*.isCaseVide(new Position(3, 3)));  
 *assertFalse*(*damier*.isCaseVide(new Position(3, 4)));  
 *assertFalse*(*damier*.isCaseVide(new Position(4, 3)));  
 *assertFalse*(*damier*.isCaseVide(new Position(4, 4)));  
 }  
  
 public void testIsRempli() {  
 Plateau p = new Plateau();  
 */\* Test sur le plateau de base \*/  
 assertFalse*(p.isRempli());  
  
 */\* Ajout d'un pion mais damier reste non rempli \*/* p.remplirCase(new Pion(new Position(7, 7), Couleur.*NOIR*));  
 *assertFalse*(p.isRempli());  
  
 */\* Test sur un plateau plein \*/  
  
 /\* On rempli en entier le tableau de pions noirs \*/* for (Position c : p.getCasesVides()) {  
 p.remplirCase(new Pion(c, Couleur.*NOIR*));  
  
 }  
 */\* Verification que le tableau est bien plein \*/  
 assertTrue*(p.isRempli());  
 }  
  
 public void testGetPionsCouleur() {  
 Plateau p = new Plateau();  
 *List*<Pion> PIONS\_ATTENDU\_NOIR = new LinkedList<>();  
 *List*<Pion> PIONS\_ATTENDU\_BLANC = new LinkedList<>();  
 PIONS\_ATTENDU\_NOIR.add(new Pion(new Position(3, 4), Couleur.*NOIR*));  
 PIONS\_ATTENDU\_NOIR.add(new Pion(new Position(4, 3), Couleur.*NOIR*));  
 PIONS\_ATTENDU\_BLANC.add(new Pion(new Position(3, 3), Couleur.*BLANC*));  
 PIONS\_ATTENDU\_BLANC.add(new Pion(new Position(4, 4), Couleur.*BLANC*));  
  
 *assertEquals*(PIONS\_ATTENDU\_NOIR, p.getPions(Couleur.*NOIR*));  
 *assertEquals*(PIONS\_ATTENDU\_BLANC, p.getPions(Couleur.*BLANC*));  
  
 }  
  
 public void testGetPions() {  
 Plateau p = new Plateau();  
 *List*<Pion> PIONS\_ATTENDUS = new LinkedList<>();  
 PIONS\_ATTENDUS.add(new Pion(new Position(3, 3), Couleur.*BLANC*));  
 PIONS\_ATTENDUS.add(new Pion(new Position(4, 4), Couleur.*BLANC*));  
 PIONS\_ATTENDUS.add(new Pion(new Position(3, 4), Couleur.*NOIR*));  
 PIONS\_ATTENDUS.add(new Pion(new Position(4, 3), Couleur.*NOIR*));  
  
 *assertEquals*(PIONS\_ATTENDUS, p.getPions());  
  
 }  
  
 public void testGetCasesVides() {  
 Plateau p = new Plateau();  
  
 */\* Test sur un plateau contenant 4 pions donc 60 cases vides \*/  
 assertEquals*(60, p.getCasesVides().size());  
  
 */\* Test sur un plateau contenant 6 pions donc 58 cases vides \*/* p.remplirCase(new Pion(new Position(7, 7), Couleur.*NOIR*));  
 p.remplirCase(new Pion(new Position(1, 1), Couleur.*NOIR*));  
 *assertEquals*(58, p.getCasesVides().size());  
  
 */\* Test sur un plateau contenant 7 pions donc 57 cases vides \*/* p.remplirCase(new Pion(new Position(0, 0), Couleur.*NOIR*));  
 *assertEquals*(57, p.getCasesVides().size());  
 }  
  
 public void testGetCasesJouables() {  
 Plateau p = new Plateau();  
 *List*<Position> positionJouablesNoir = new LinkedList<>();  
 *List*<Position> positionJouablesBlanc = new LinkedList<>();  
 positionJouablesNoir.add(new Position(2, 3));  
 positionJouablesNoir.add(new Position(3, 2));  
 positionJouablesNoir.add(new Position(4, 5));  
 positionJouablesNoir.add(new Position(5, 4));  
  
 positionJouablesBlanc.add(new Position(2, 4));  
 positionJouablesBlanc.add(new Position(3, 5));  
 positionJouablesBlanc.add(new Position(4, 2));  
 positionJouablesBlanc.add(new Position(5, 3));  
  
  
 *assertEquals*(4, p.getCasesJouables(Couleur.*NOIR*).size());  
 *assertEquals*(positionJouablesNoir, p.getCasesJouables(Couleur.*NOIR*));  
 *assertEquals*(4, p.getCasesJouables(Couleur.*BLANC*).size());  
 *assertEquals*(positionJouablesBlanc, p.getCasesJouables(Couleur.*BLANC*));  
 }  
  
 public void testGetMeilleureCaseJouable() {  
 */\* Initialisation d'un plateau normal \*/* Plateau p = new Plateau();  
  
 */\* Ajout d'un pion blanc en 3D creant une possibilite de retourner 2  
 pions en un seul coup pour les noirs \*/* p.remplirCase(new Pion(new Position(2, 3), Couleur.*BLANC*));  
  
 */\* Initialisation de la meilleure position jouable par le joueur noir \*/* Position meilleurePosition = new Position(1, 3);  
  
 *assertEquals*(meilleurePosition,  
 p.getMeilleureCaseJouable(Couleur.*NOIR*));  
 }  
  
 public void testIsCaseJouable() {  
 Plateau p = new Plateau();  
 *List*<Position> positionsPossibles = p.getCasesJouables(Couleur.*NOIR*);  
 *assertTrue*(p.isCaseJouable(new Position(2, 3), positionsPossibles));  
 *assertTrue*(p.isCaseJouable(new Position(3, 2), positionsPossibles));  
 *assertTrue*(p.isCaseJouable(new Position(4, 5), positionsPossibles));  
 *assertTrue*(p.isCaseJouable(new Position(5, 4), positionsPossibles));  
 }  
  
 public void testPlacerPion() {  
 Plateau p = new Plateau();  
 */\* Pions possibles a plcaer : \*/* Pion aTester = new Pion(new Position(2, 3), Couleur.*NOIR*);  
 *assertTrue*(p.placerPion(aTester));  
 aTester = new Pion(new Position(2, 4), Couleur.*BLANC*);  
 *assertTrue*(p.placerPion(aTester));  
 aTester = new Pion(new Position(4, 5), Couleur.*NOIR*);  
 *assertTrue*(p.placerPion(aTester));  
 aTester = new Pion(new Position(4, 2), Couleur.*BLANC*);  
 *assertTrue*(p.placerPion(aTester));  
  
 */\* Pions impossibles a placer : \*/* aTester = new Pion(new Position(1, 2), Couleur.*BLANC*);  
 *assertFalse*(p.placerPion(aTester));  
 aTester = new Pion(new Position(7, 2), Couleur.*BLANC*);  
 *assertFalse*(p.placerPion(aTester));  
 }  
  
 public void testEquals() {  
 Plateau premier = new Plateau();  
 Plateau second = new Plateau();  
 */\* Test sur deux plateau initialisée par défaut : \*/  
 assertTrue*(premier.equals(second));  
 *assertTrue*(second.equals(premier));  
  
 */\* Test avec deux plateaux différents \*/* Pion aPlacer = new Pion(new Position(2, 3), Couleur.*NOIR*);  
 premier.placerPion(aPlacer);  
 *assertFalse*(premier.equals(second));  
 *assertFalse*(second.equals(premier));  
  
 */\* Test sur deux plateaux avec un pion supplémentaire chacun : \*/* second.placerPion(aPlacer);  
 *assertTrue*(premier.equals(second));  
 *assertTrue*(second.equals(premier));  
  
 }  
}

# Résultats



# Explications

Afin de réaliser des tests par la suite nous avons créé un jeu de donnée contenant un plateau initialisé par défaut, un tableau de positions valides et un autre de positions invalides.

1. initialiseDefaut

Pour vérifier que le constructeur initialise initialise bien par défaut nous avons créé un tableau puis nous l’avons comparé avec celui initialisé plus tôt : damier.

1. isValide

Afin de tester la validité de la méthode nous avons utilisé un for each sur les positions invalides et valides. Pour chaque for each nous avons vérifié que chaque position était soit invalide soit valide.

1. remplirCase

Afin de tester la méthode remplirCase nous avons créer un nouveau plateau puis nous avons essayé de remplir n’importe quelle case par un pion tant qu’elle est vide.

1. getPion

Afin de tester getPion nous avons initialisée un plateau par défaut puis nous y avons ajouté un pion en 8H afin de vérifier qu’il y est bien présent. Nous avons ensuite utilisé la méthode getPion afin de vérifier qu’un pion était bien présent en 8H, nous avons fait de même pour un pion en 4D et en 3D qui sont de base dans un plateau, blanc pour l’un et null pour l’autre (inexistant).

1. isCaseVide

Afin de tester la méthode iscasevide nous avons utiliser le plateau damier et avons réalisé des tests sur des cases vides puis des cases non vides.

1. isRempli

Afin de tester la méthode isRempli nous avons initialisé un nouveau plateau par défaut puis vérifié s’il était bien vide. Ensuite nous avons ajouté un pion et réessayé de vérifier s’il était vide. Enfin nous avons rempli tout le tableau à l’aide d’un for each puis nous avons réessayé de vérifier si le tableau était vide, surprise il était plein.

1. getPions(Couleur)

Afin de tester la méthode getPions(Couleur) nous avons créer une liste de pions attendus des deux couleurs, noir et blanc. Par la suite nous avons vérifié que la méthode getPions(noir) renvoyé bien la liste attendue et de même pour getPions(Blanc).

1. getPions

Afin de tester la méthode getPions nous avons utilisé la même méthode que précédemment, nous avons créé une liste de pion contenant la totalité des pions attendus puis nous avons vérifié que la méthode renvoyée bien la bonne liste.

1. getCasesVides

Afin de tester la méthode getCasesVides nous avons créé un plateau par défaut puis nous avons vérifié que la taille de la liste renvoyée par la méthode getCasesVides correspondait bien à 60. Nous avons utilisé cette méthode pour vérifier la quantité de cases vides, 60 car il existe 64 cases dans un plateau de 8 par 8 dont 4 cases utilisées par 4 pions en début de partie. Ce principe nous a permis de calculer facilement le nombre de cases vides et vérifier sa validité.

Par la suite nous avons ajoutés des pions sur le plateau et nous avons vérifié que le nombre de cases vides diminuées en conséquence.

1. getMeilleureCaseJouable

Afin de tester la méthode nous avons créé un plateau initialisé par défaut puis nous avons rempli une case par un pion en 3D afin de créer la possibilité de retourner 2 pions en un coup pour les pions de couleurs noirs. Par la suite nous avons vérifié que la meilleure position retournée par la fonction correspondait bien à la case attendue.

1. iCasesJouable

Afin de tester la méthode CasesJouable nous avons créé deux listes de positions contenant les différentes meilleures positions. Nous avons ensuite vérifié la validité de la méthode à l’aide des listes créés précédemment.

1. placerPion

Afin de tester la méthode placerPion nous avons tenté de placer des pions qui pouvaient l’être puis nous avons essayer de placer des pions invalides.

1. Equals

Afin de tester la méthode equals nous avons créé deux plateaux par défaut, puis nous les avons comparés, le premier au second puis le second au premier afin d’éviter les erreurs. Par la suite nous avons ajouter des pions différents aux deux plateaux puis vérifié la non-égalités des deux par le même procédé. Enfin nous avons ajoutés des pions afin de faire correspondre les deux plateaux et nous avons réalisé le même procédé.