LESCIEUX Xavier
ROBILLIARD Rémi
ROUET Yoann

Département informatique
1er année – Groupe A2

Année universitaire 2018 - 2019

Projet de Programmation

Cahier d'analyse et de conception

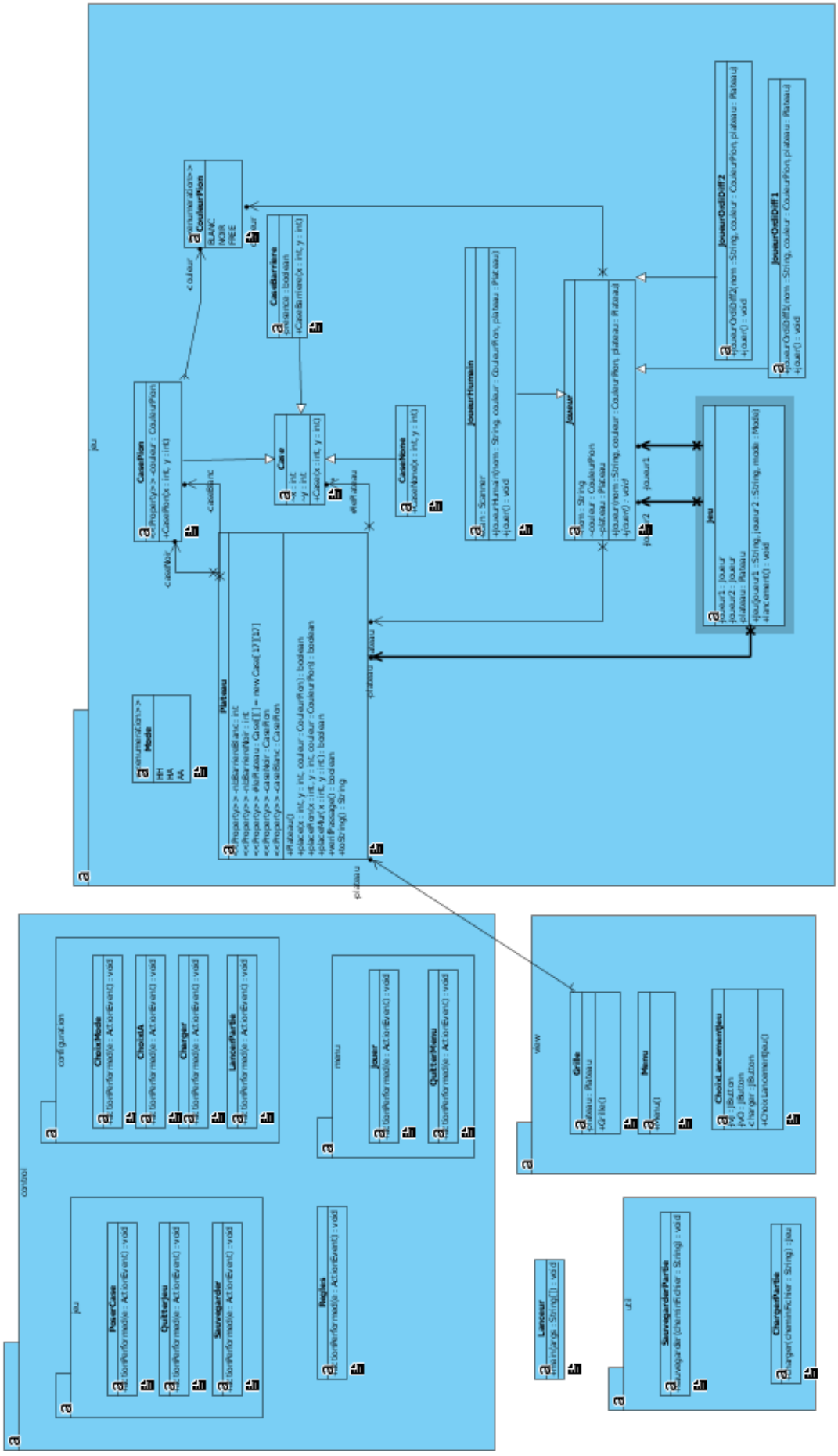
Destinataire

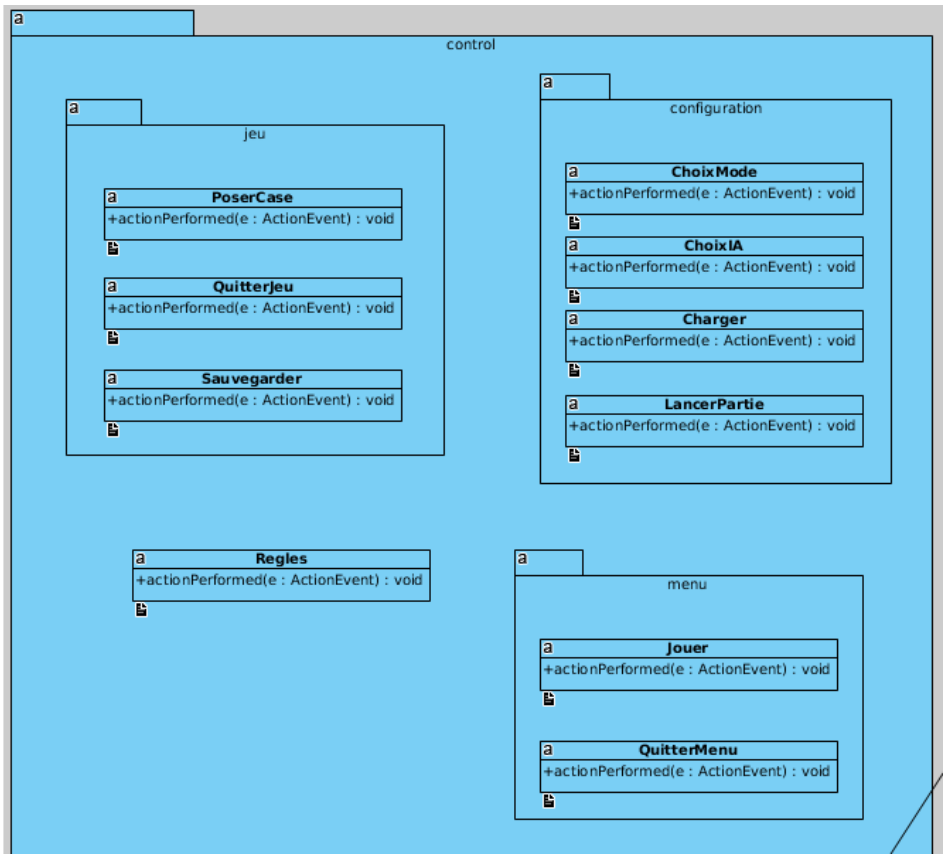
IUT de Vannes

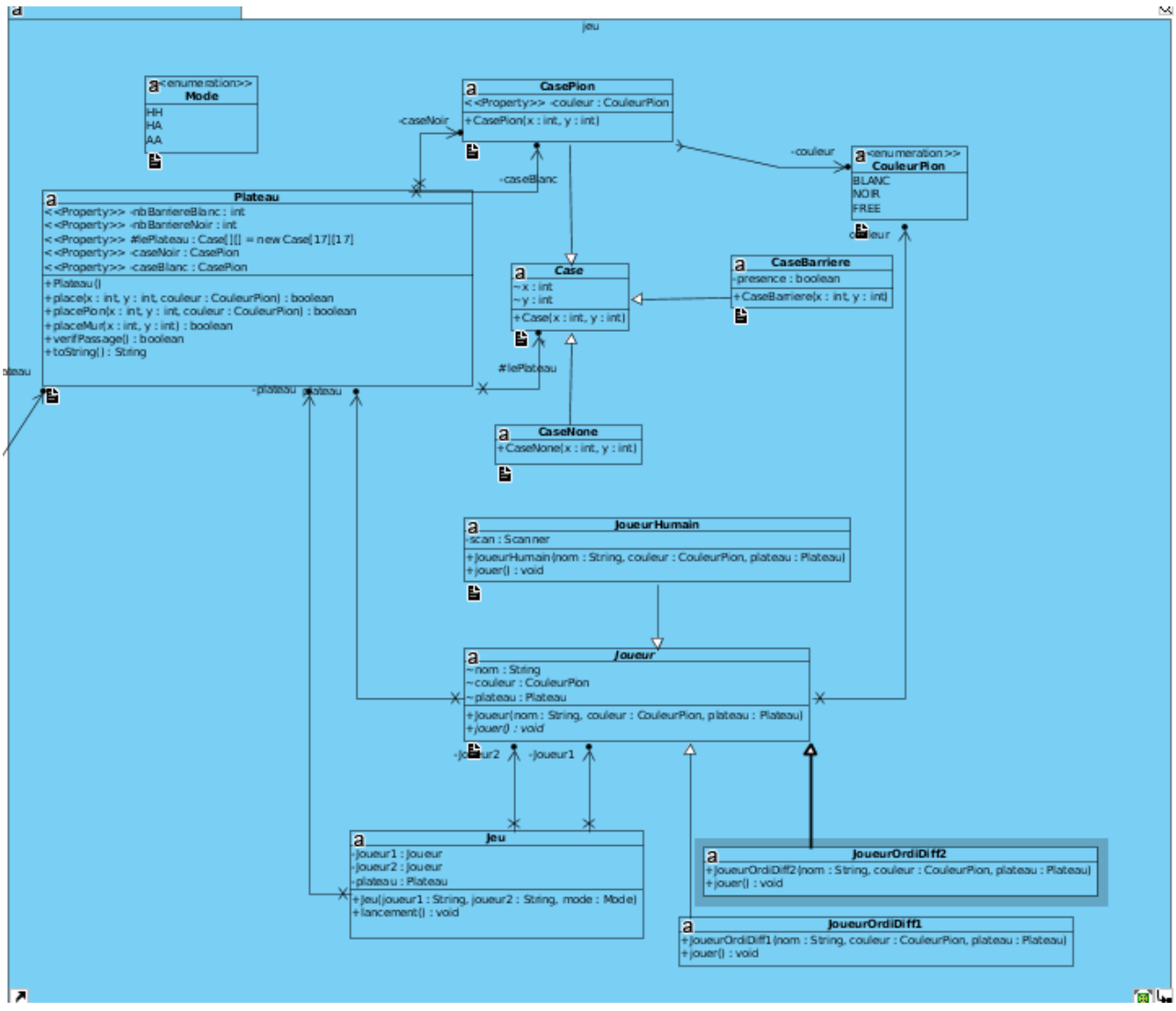
M. Sébastien LEFEVRE, professeur et client.

Table des matières

Diagramme de classes d’analyse :




control
jeu
PoserCase
+actionPerformed(e : ActionEvent) : void
QuitterJeu
+actionPerformed(e : ActionEvent) : void
Sauvegarder
+actionPerformed(e : ActionEvent) : void
configuration
ChoixMode
+actionPerformed(e : ActionEvent) : void
ChoixIA
+actionPerformed(e : ActionEvent) : void
Charger
+actionPerformed(e : ActionEvent) : void
LancerPartie
+actionPerformed(e : ActionEvent) : void
Regles
+actionPerformed(e : ActionEvent) : void
menu
Jouer
+actionPerformed(e : ActionEvent) : void
QuitterMenu
+actionPerformed(e : ActionEvent) : void


jeu
<<enumeration>>
Mode
HH
HA
AA
CasePion
<<Property>> -couleur : CouleurPion
+CasePion(x : int, y : int)
-caseNoir
-caseBlanc
-couleur
<<enumeration>>
CouleurPion
BLANC
NOIR
FREE
Plateau
<<Property>> -nbBarriereBlanc : int
<<Property>> -nbBarriereNoir : int
<<Property>> #lePlateau : Case[][] = new Case[17][17]
<<Property>> -caseNoir : CasePion
<<Property>> -caseBlanc : CasePion
+Plateau()
+place(x : int, y : int, couleur : CouleurPion) : boolean
+placePion(x : int, y : int, couleur : CouleurPion) : boolean
+placeMur(x : int, y : int) : boolean
+verifPassage() : boolean
+toString() : String
Case
~x : int
~y : int
+Case(x : int, y : int)
CaseBarriere
-presence : boolean
+CaseBarriere(x : int, y : int)
#lePlateau
-plateau
CaseNone
+CaseNone(x : int, y : int)
JoueurHumain
-scan : Scanner
+joueurHumain(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
Joueur
~nom : String
~couleur : CouleurPion
~plateau : Plateau
+Joueur(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
-Joueur2
-Joueur1
Jeu
-joueur1 : Joueur
-joueur2 : Joueur
-plateau : Plateau
+Jeu(joueur1 : String, joueur2 : String, mode : Mode)
+lancement() : void
JoueurOrdiDiff2
+JoueurOrdiDiff2(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
JoueurOrdiDiff1
+JoueurOrdiDiff1(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void

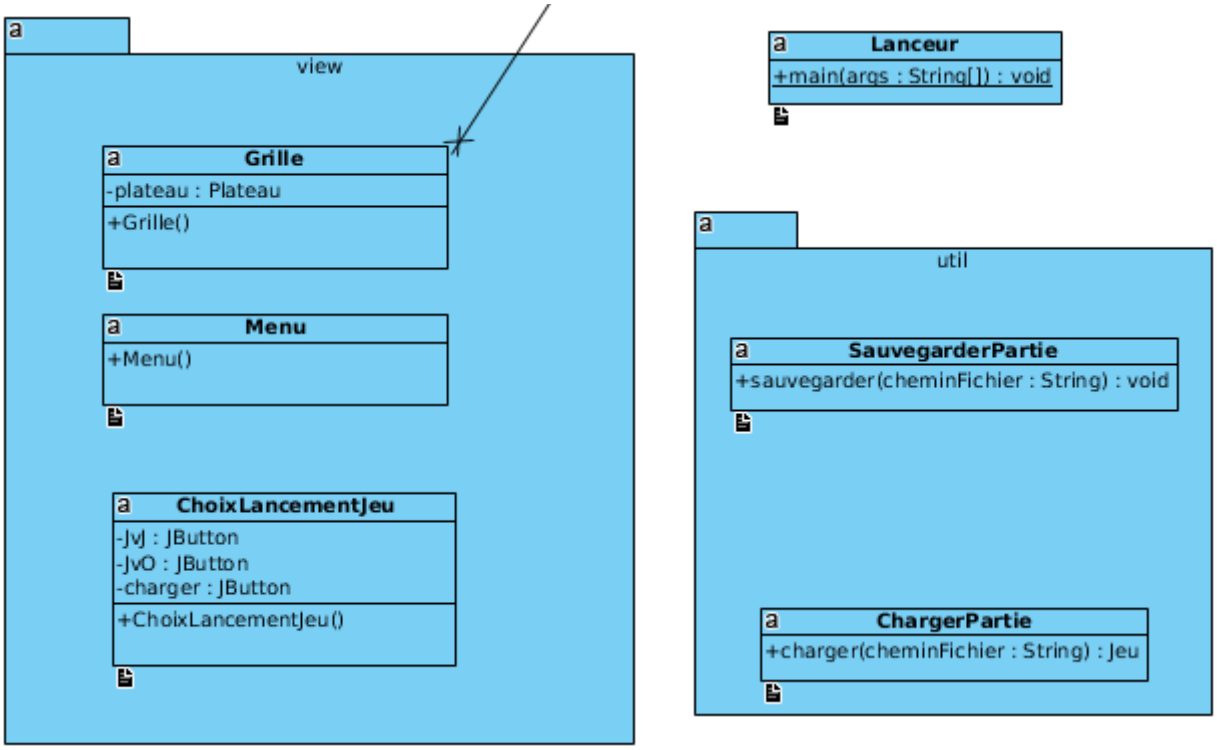
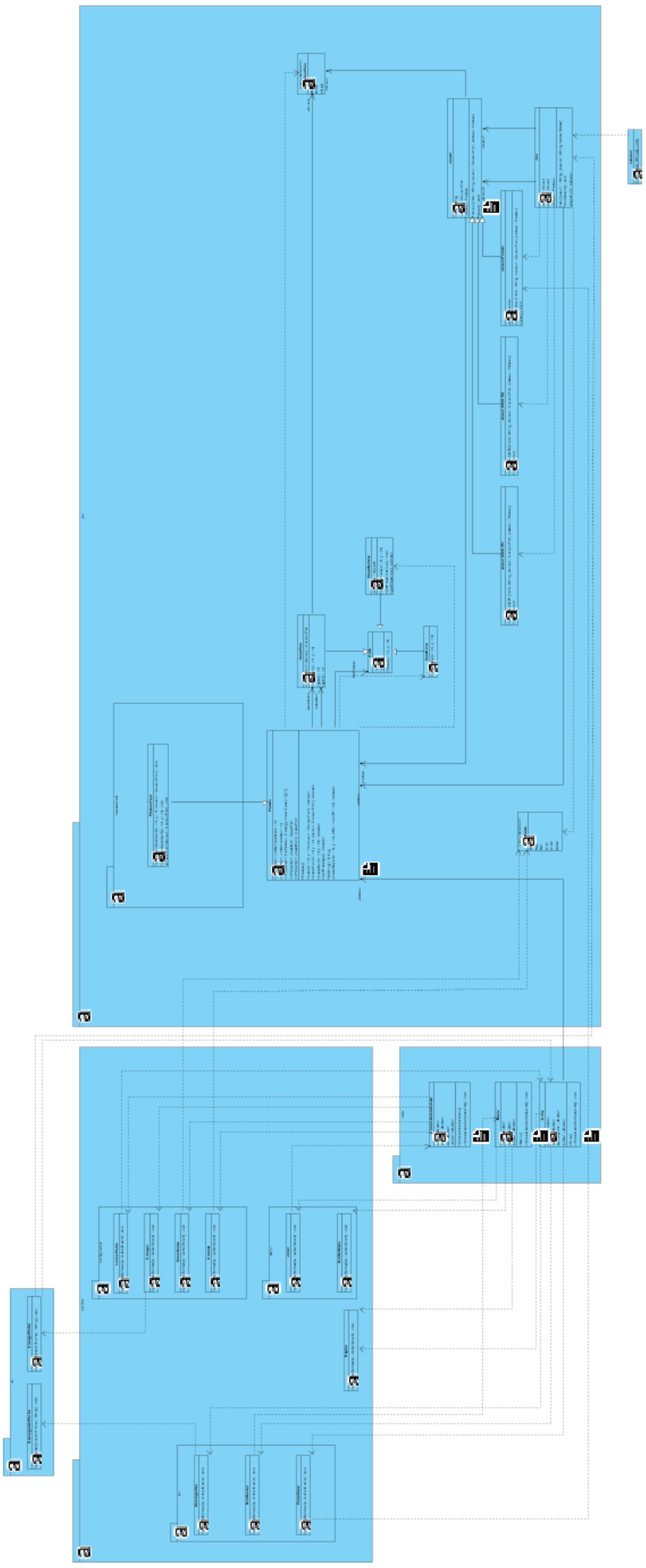
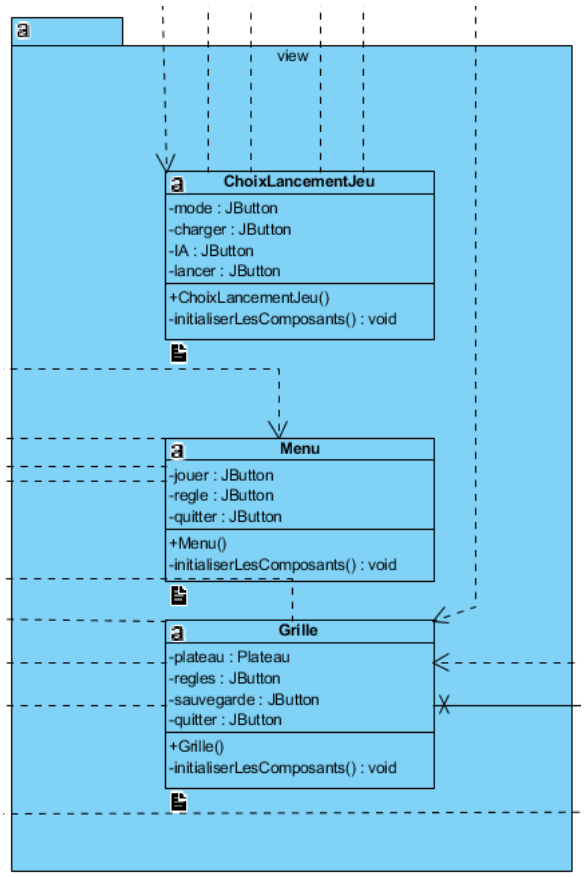
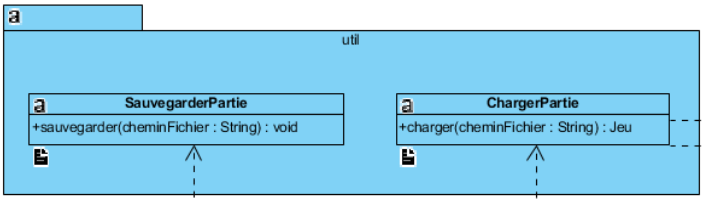
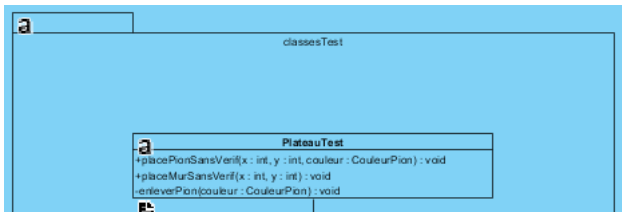

view
Grille
-plateau : Plateau
+Grille()
Menu
+Menu()
ChoixLancementJeu
-JvJ : JButton
-JvO : JButton
-charger : JButton
+ChoixLancementJeu()
Lanceur
+main(args : String[]) : void
util
SauvegarderPartie
+sauvegarder(cheminFichier : String) : void
ChargerPartie
+charger(cheminFichier : String) : Jeu

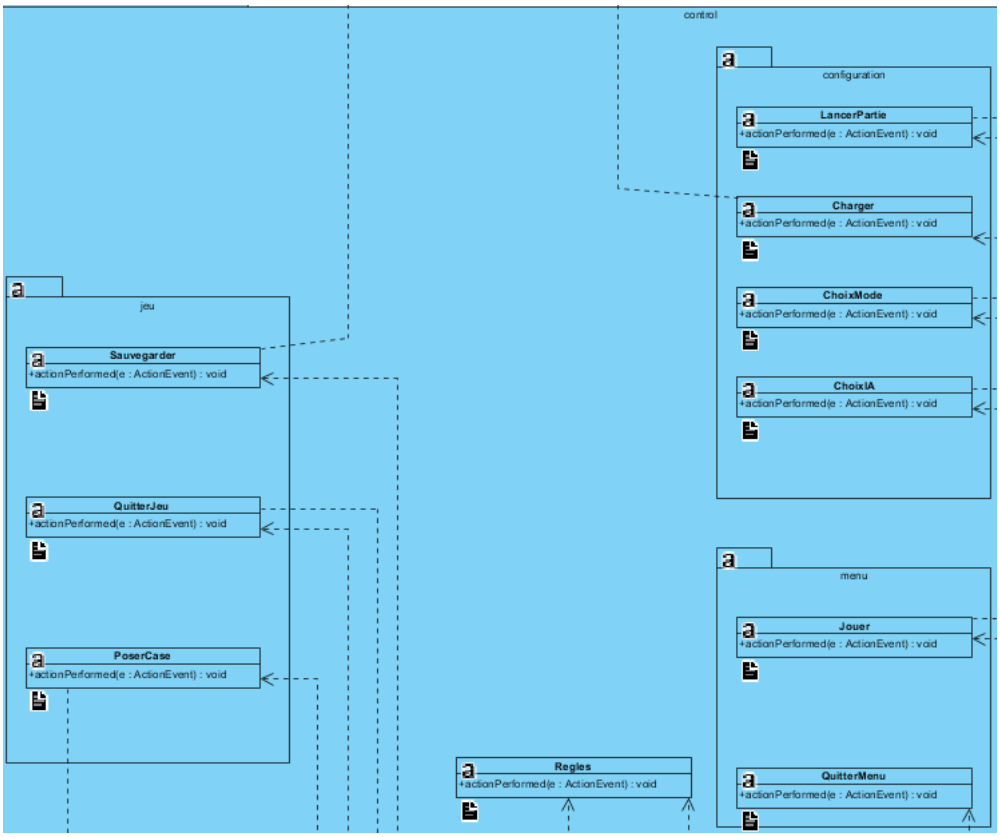
Diagramme de classes de conception :

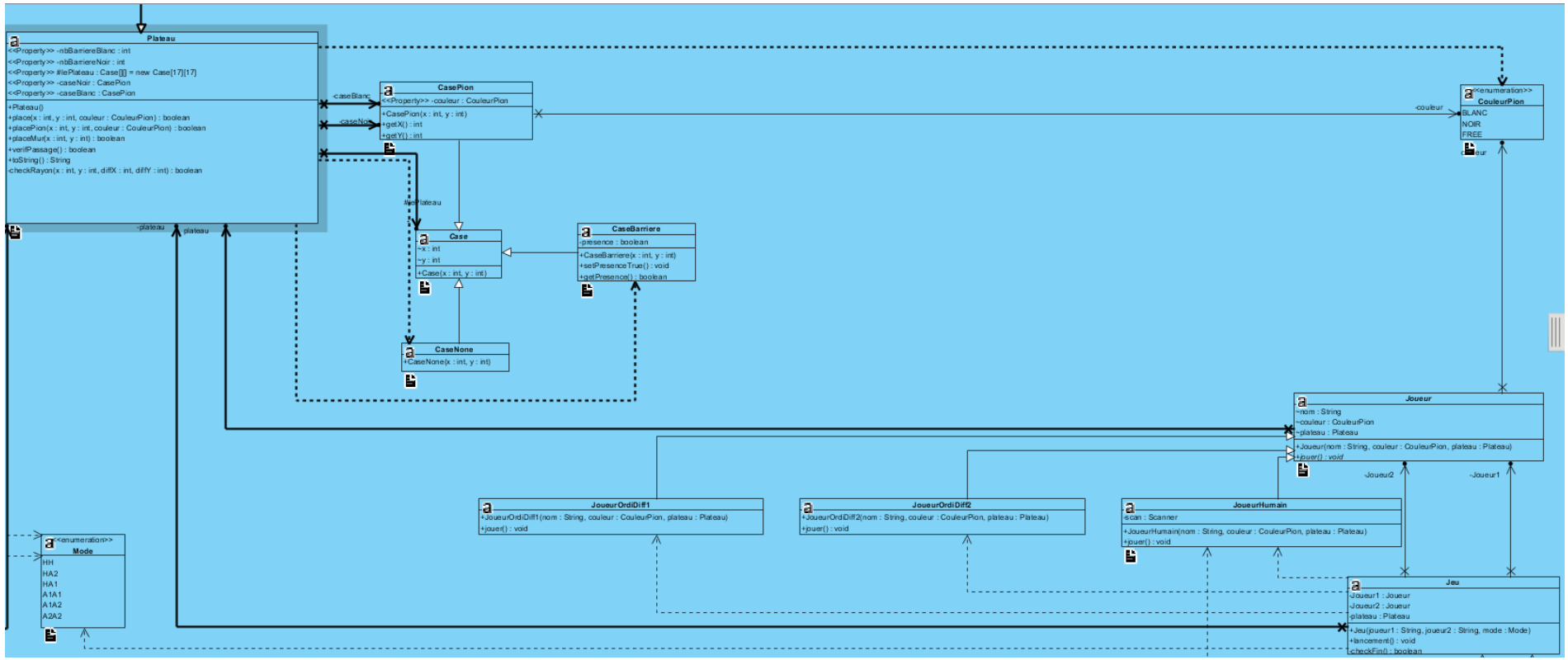



view
ChoixLancementJeu
-mode : JButton
-charger : JButton
-IA : JButton
-lancer : JButton
+ChoixLancementJeu()
-initialiserLesComposants() : void
Menu
-jouer : JButton
-regle : JButton
-quitter : JButton
+Menu()
-initialiserLesComposants() : void
Grille
-plateau : Plateau
-regles : JButton
-sauvegarde : JButton
-quitter : JButton
+Grille()
-initialiserLesComposants() : void

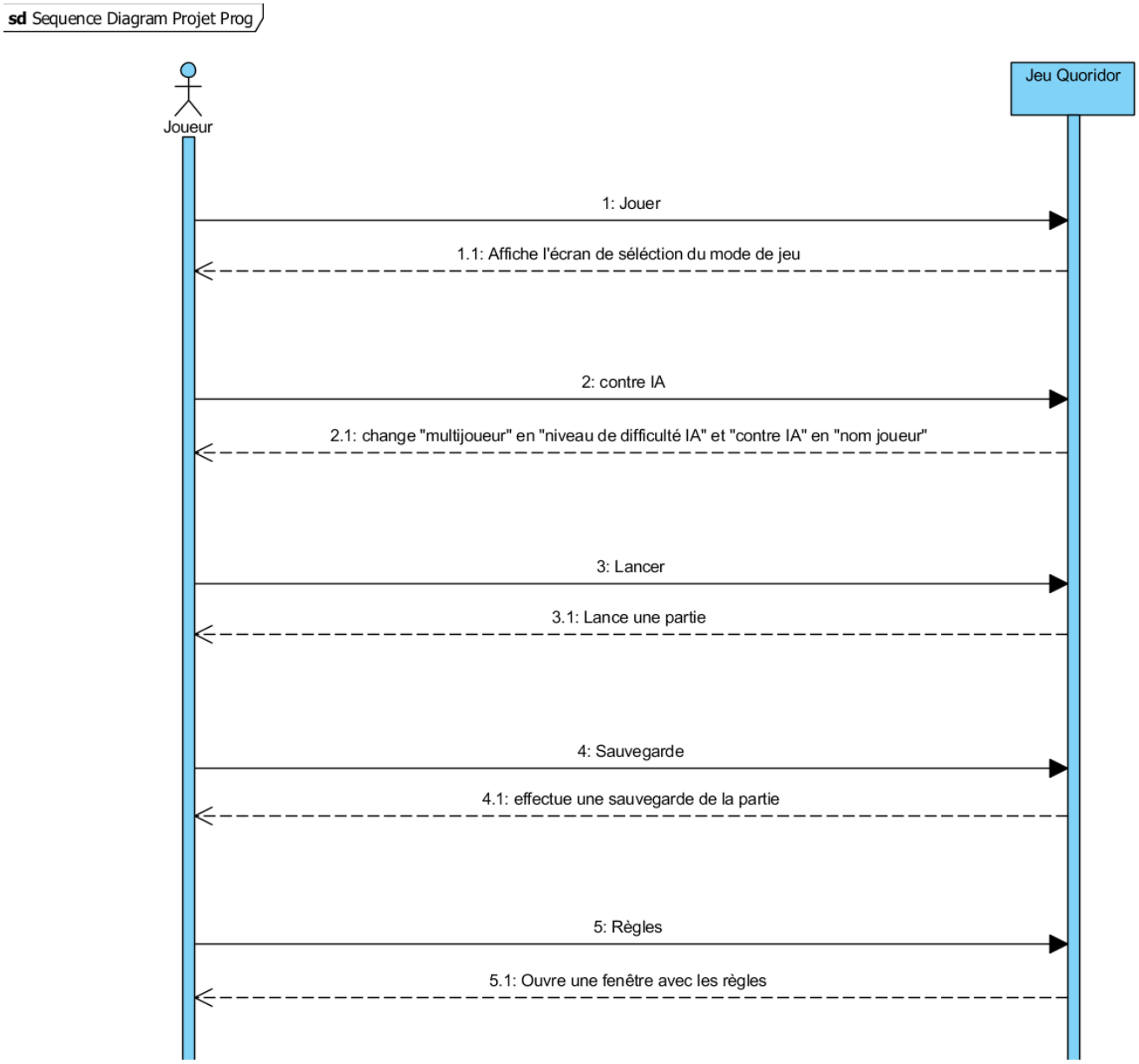

util
SauvegarderPartie
+sauvegarder(cheminFichier : String) : void
ChargerPartie
+charger(cheminFichier : String) : Jeu

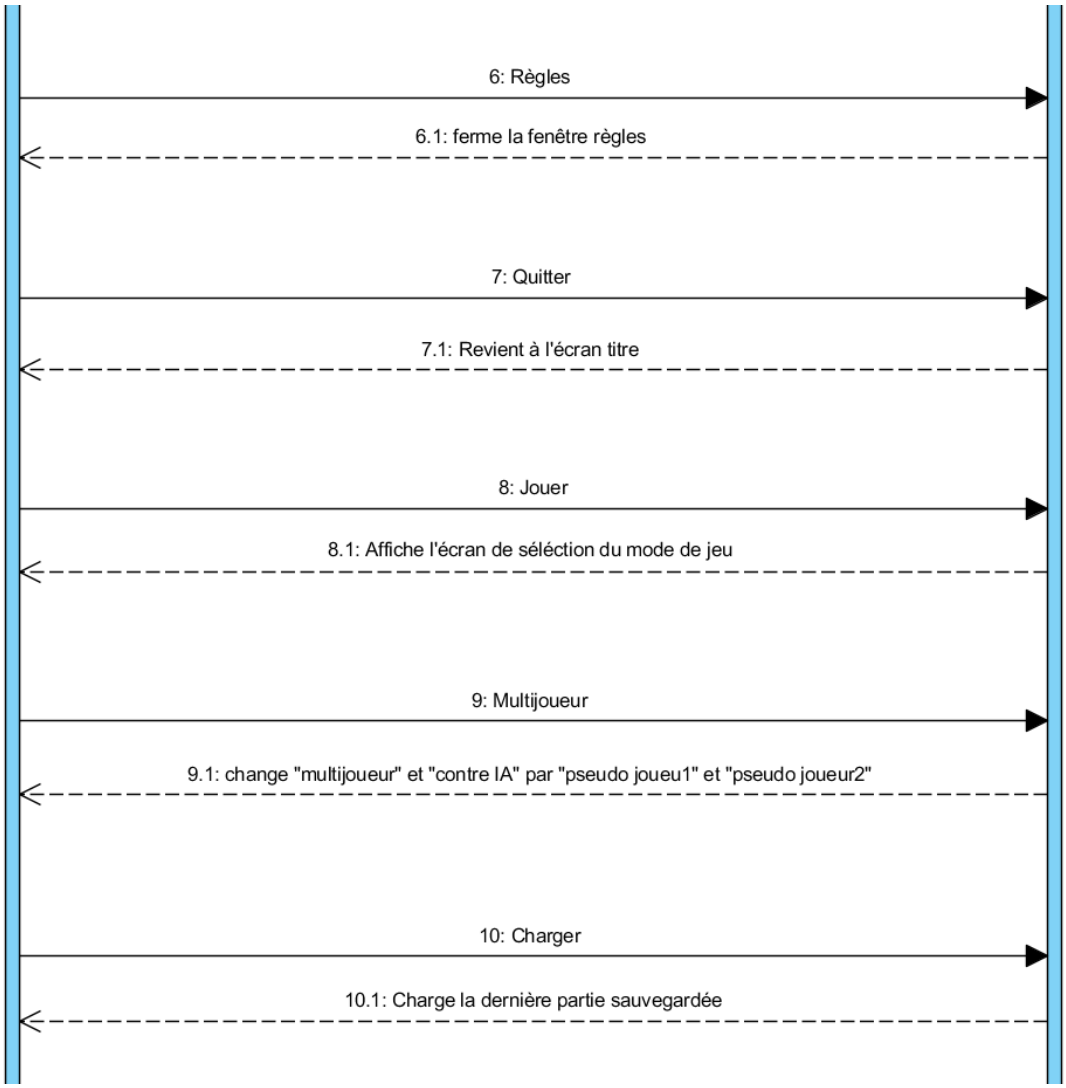

classesTest
PlateauTest
+placePionSansVerif(x : int, y : int, couleur : CouleurPion) : void
+placeMurSansVerif(x : int, y : int) : void
-enleverPion(couleur : CouleurPion) : void

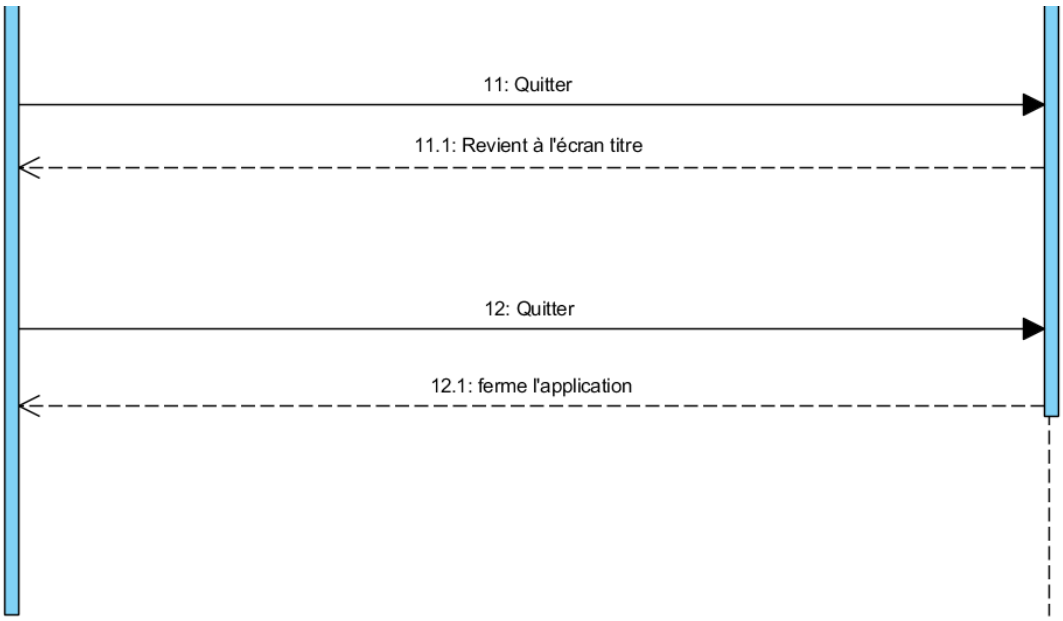

control
configuration
LancerPartie
+actionPerformed(e : ActionEvent) : void
Charger
+actionPerformed(e : ActionEvent) : void
ChoixMode
+actionPerformed(e : ActionEvent) : void
ChoixIA
+actionPerformed(e : ActionEvent) : void
jeu
Sauvegarder
+actionPerformed(e : ActionEvent) : void
QuitterJeu
+actionPerformed(e : ActionEvent) : void
PoserCase
+actionPerformed(e : ActionEvent) : void
menu
Jouer
+actionPerformed(e : ActionEvent) : void
Regles
+actionPerformed(e : ActionEvent) : void
QuitterMenu
+actionPerformed(e : ActionEvent) : void


Plateau
<<Property>> -nbBarriereBlanc : int
<<Property>> -nbBarriereNoir : int
<<Property>> #lePlateau : Case[][] = new Case[17][17]
<<Property>> -caseNoir : CasePion
<<Property>> -caseBlanc : CasePion
+Plateau()
+place(x : int, y : int, couleur : CouleurPion) : boolean
+placePion(x : int, y : int, couleur : CouleurPion) : boolean
+placeMur(x : int, y : int) : boolean
+verifPassage() : boolean
+toString() : String
-checkRayon(x : int, y : int, diffX : int, diffY : int) : boolean
-caseBlanc
-caseNoir
#lePlateau
-plateau
plateau
CasePion
<<Property>> -couleur : CouleurPion
+CasePion(x : int, y : int)
+getX() : int
+getY() : int
Case
#x : int
#y : int
+Case(x : int, y : int)
CaseBarriere
-presence : boolean
+CaseBarriere(x : int, y : int)
+setPresenceTrue() : void
+getPresence() : boolean
CaseNone
+CaseNone(x : int, y : int)
<<enumeration>>
CouleurPion
BLANC
NOIR
FREE
-couleur
Joueur
-nom : String
-couleur : CouleurPion
-plateau : Plateau
+Joueur(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
-Joueur2
-Joueur1
JoueurOrdiDiff1
+JoueurOrdiDiff1(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
JoueurOrdiDiff2
+JoueurOrdiDiff2(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
JoueurHumain
-scan : Scanner
+JoueurHumain(nom : String, couleur : CouleurPion, plateau : Plateau)
+jouer() : void
<<enumeration>>
Mode
HH
HA2
HA1
A1A1
A1A2
A2A2
Jeu
-Joueur1 : Joueur
-Joueur2 : Joueur
-plateau : Plateau
+Jeu(joueur1 : String, joueur2 : String, mode : Mode)
+lancement() : void
-checkFin() : boolean

Diagrammes de séquence boite noire :




6: Règles
6.1: ferme la fenêtre règles
7: Quitter
7.1: Revient à l'écran titre
8: Jouer
8.1: Affiche l'écran de séléction du mode de jeu
9: Multijoueur
9.1: change "multijoueur" et "contre IA" par "pseudo joueu1" et "pseudo joueur2"
10: Charger
10.1: Charge la dernière partie sauvegardée


11: Quitter
11.1: Revient à l'écran titre
12: Quitter
12.1: ferme l'application

Spécification des formats de fichier :

Le format des fichiers de sauvegardes est .qu.

Classes principales :

Squelette et documentation :

//CLASSE Jeu

```
package jeu;

public class Jeu {

	private Joueur Joueur1;
	private Joueur Joueur2;
	protected Plateau plateau;

	/**
	 * initialise le jeu
	 *
	 * @param joueur1 nom du joueur1
	 * @param joueur2 nom du joueur2
	 * @param mode    mode selectionne
	 */
	public Jeu(String joueur1 , String joueur2 , Mode mode) {

	}


	/**
	 * lance le jeu
	 */
	public void lancement() {
	}

	/**
	 * vérifie si le jeu est fini
	 *
	 * @return true si le jeu est fini, faux s'il ne l'est pas
	 */
	protected boolean checkFin() {
	}
}
```

//CLASSE Plateau

```
package jeu;

 /**
  * La classe plateau va permettre d'executer les principales actions de jeu.
On
```

```
 * passera par cette classe pour le déplacement des pions et le placement
des
 * murs.
 */
public class Plateau {

    protected Case[][] lePlateau = new Case[17][17];
    private int nbBarriereBlanc;
    private int nbBarriereNoir;
    protected CasePion caseNoir;
    protected CasePion caseBlanc;

    /**
     * initialise le plateau de jeu
     */
    public Plateau() {
    }

    /**
     * cette methode verifie si une case peut etre place. Si la case se
place
     * retourne vraie sinon faux.
     *
     * @param x       la coordonnee x de la case
     * @param y       la coordonnee x de la case
     * @param couleur la couleur du joueur qui veut placer la case
     * @return vraie si la case a ete place sinon false
     */
    public boolean place(int x, int y, CouleurPion couleur) {
    }

    /**
     * verifie si un pion peut etre place a une coordonnee donnee. Retourne
vrai si
     * il est place et faux sil ne peut pas etre place.
     *
     * @param x la position du pion sur l'axe des x souhaite
     * @param y la position du pion sur l'axe des y souhaite
     * @param couleur la couleur du pion
     * @return vrai si le pion est place et faux sil ne lest pas.
     */
    public boolean placePion(int x, int y, CouleurPion couleur) {

    }

    /**
     * Methode permettant de placer un mur sur une case appropriée, la
méthode
     * vérifiera donc la positio n du mur.
     *
     * @param x la position sur l'axe horizontal où l'on veut positionner le
mur
     * @param y la position sur l'axe vertical où l'on veut positionner le
mur
```

```
     * @return Si le mur ne peut pas etre pose retourne faux sinon pose le
mur et retourne vrai
     */
    public boolean placeMur(int x, int y) {

    }

    /**
     * Vérifie que les joueurs peuvent accéder à l'autre bord du plateau et
qu'ils
     * ne sont pas bloqués
     *
     * @return true s'il ne sont pas bloqué, et false s'ils le sont.
     */
    public boolean verifPassage() {
    }

    /**
     * @return la chaine de caractere qui decrit le tableau
     */
    public String toString() {

    }

    /**
     * @return la caseBlanc
     */
    public CasePion getCaseBlanc() {
    }

    /**
     * @return la caseNoir
     */
    public CasePion getCaseNoir() {
    }

    /**
     * @return le nbBarriereBlanc
     */
    public int getNbBarriereBlanc() {
    }

    /**
     * @return le nbBarriereNoir
     */
    public int getNbBarriereNoir() {
    }

    /**
     * @return the lePlateau
     */
    public Case[][] getLePlateau() {
    }
```

```
}
```

//CLASSE Joueur

```
package jeu;

/**
 * Joueur
 */
public abstract class Joueur {
	String nom;
	CouleurPion couleur;
	Plateau plateau;

	/**
	 * Créé un joueur
	 * @param nom son nom
	 * @param couleur la couleur de son pion
	 * @param plateau plateau de jeu
	 */
	public Joueur(String nom, CouleurPion couleur , Plateau plateau) {

	}

	/**
	 * cette méthode gère le tour de jeu de chacun
	 */
	public abstract void jouer();
}
```

//CLASSE JoueurHumain

```
package jeu;

import java.util.Scanner;
import java.lang.NumberFormatException;

/**
 * Human
 */
public class JoueurHumain extends Joueur {
    private Scanner scan;

    /**
     * créé un joueur humain
     *
     * @param nom     son nom
     * @param couleur sa couleur de pion
     * @param plateau plateau de jeu
     *
```

```
     */
    public JoueurHumain(String nom, CouleurPion couleur, Plateau plateau) {
    }

    /**
     * la méthode pour jouer
     */
    public void jouer() {

    }
}
```

//CLASSEjoueurOrdiDiff1

```
package jeu;

public class JoueurOrdiDiff1 extends Joueur {

    /**
     * créé un joueur ordinateur qui joue automatiquement
     *
     * @param nom     nom de l'ordinateur
     * @param couleur la couleur de l'ordinateur
     * @param plateau plateau de jeu
     */
    public JoueurOrdiDiff1(String nom, CouleurPion couleur, Plateau plateau) {
    }

    /**
     * joue un tour du joueur
     */
    public void jouer() {

    }
}
```

Tests unitaires :

Afin de tester certains cas particuliers, on utilise les classes se trouvant dans le package jeu.classesTest.

//CLASSES SE TROUVANT DANS LE PACKAGE test
//CLASSE DE TEST DE jeu

```
package test.jeu.jeu;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.Before;
import org.junit.Test;

import jeu.classesTest.jeuTest;

/**
 * TestFin
 */
public class TestFin {
	private static jeuTest jeu;

	@Before
	public void setUp() {
		jeu = new jeuTest();
	}

	/**
	 * teste si checkFin marche quand le jeu est fini
	 */
	@Test
	public void finir() {
		jeu.poseFin();
		assertTrue("Verifie si le jeu est bien fini", jeu.checkFinLancer());
	}

	/**
	 * teste si checkFin marche quand le jeu n'est pas fini
	 */
	@Test
	public void pasFinir() {
		jeu.jouerNormal();
		assertFalse("Verifie si le jeu peut continuer", jeu.checkFinLancer());
	}
}
```

//CLASSES DE TEST DE plateau

```
package test.jeu.plateau;

 import static org.junit.Assert.assertFalse;
 import static org.junit.Assert.assertTrue;

 import org.junit.Before;
 import org.junit.Test;

 import jeu.classesTest.PlateauTest;

 /**
  * TestPlaceMur
  */
 public class TestPlaceMur {
	private static PlateauTest plateau;

	@Before
	public void setUp() {
		plateau = new PlateauTest();
	}

	/**
	 * teste les lignes
	 */
	@Test
	public void testLigne() {
		// TestPlateau plateau = new TestPlateau();

		assertTrue("Mettre une barrière sur une ligne qui marche", plateau.placeMur(2, 9));
		assertFalse("Mettre une barrière alors que l'emplacement est déjà occupé",
plateau.placeMur(2, 9));
		assertFalse("Mettre une barrière sur une ligne alors que l'emplacement à droite est déja
occupé", plateau.placeMur(0, 9));
		assertFalse("Mettre une barrière sur une ligne alors que l'emplacement à droite est hors
du tableau", plateau.placeMur(16, 1));

		plateau.placeMurSansVerif(8, 5);
		assertFalse("Placer un mur à califourchon sur un autre mur", plateau.placeMur(9, 4));
		assertTrue("Placer un mur à côté d'un autre mur", plateau.placeMur(7, 4));
		assertTrue("Placer un mur à côté d'un autre mur", plateau.placeMur(11, 4));
	}

	/**
	 * teste les colonnes
	 */
	@Test
	public void testColonne() {
		assertTrue("Mettre une barrière sur une colonne qui marche", plateau.placeMur(3, 2));
		assertFalse("Mettre une barrière sur une colonne alors que l'emplacement est déja
occupé", plateau.placeMur(3, 2));
		assertFalse("Mettre une barrière sur une colonne alors que l'emplacement en dessous
est déja occupé", plateau.placeMur(3, 0));
```

```
            assertFalse("Mettre une barrière sur une colonne alors que l'emplacement en dessous
est hors plateau", plateau.placeMur(1, 16));

            plateau.placeMurSansVerif(7, 4);
            assertFalse("Placer un mur à califourchon sur un autre mur", plateau.placeMur(6, 5));
            assertTrue("Placer un mur à côté d'un autre mur", plateau.placeMur(6, 3));
            assertTrue("Placer un mur à côté d'un autre mur", plateau.placeMur(6, 7));
      }

      /**
       * teste les différentes cases
       */
      @Test
      public void testCases() {
            assertFalse("Mettre une barrière sur une case pion", plateau.placeMur(0, 2));
            assertFalse("Mettre une barrière sur une case none", plateau.placeMur(1, 5));
      }
 }
```

```
package test.jeu.plateau;

 import static org.junit.Assert.assertFalse;
 import static org.junit.Assert.assertTrue;

 import org.junit.Before;
 import org.junit.Test;

 import jeu.CouleurPion;
 import jeu.classesTest.PlateauTest;

 /**
  * TestPlacePion
  */
 public class TestPlacePion {
      private static PlateauTest plateau;

      @Before
      public void setUp() {
            plateau = new PlateauTest();
      }

      /**
       * teste le mouvement du pion dans les quatre directions, avec et sans murs
       */
      @Test
      public void testAvancerNormal() {
            CouleurPion couleur = CouleurPion.BLANC;
            plateau.placePionSansVerif(4, 4, couleur);
            assertTrue("Avancer un pion d'une case en arrière",
plateau.placePion(2, 4, couleur));

            plateau.placePionSansVerif(4, 4, couleur);
            assertTrue("Avancer un pion d'une case en avant", plateau.placePion(6,
4, couleur));
```

```
            plateau.placePionSansVerif(4, 4, couleur);
            assertTrue("Avancer un pion à droite", plateau.placePion(4, 2,
couleur));

            plateau.placePionSansVerif(4, 4, couleur);
            assertTrue("Avancer un pion à gauche", plateau.placePion(4, 6,
couleur));
        }

        /**
         * essaie de bouger le pion en dehors du plateau
         */
        @Test
        public void testAllerNull() {
            plateau.placePionSansVerif(0, 0, CouleurPion.BLANC);

            assertFalse("Sortir dans les x négatifs", plateau.placePion(-1, 0,
CouleurPion.BLANC));

            plateau.placePionSansVerif(0, 0, CouleurPion.BLANC);
            assertFalse("Sortir dans les y négatifs", plateau.placePion(0, -1,
CouleurPion.BLANC));

            plateau.placePionSansVerif(0, 16, CouleurPion.BLANC);
            assertFalse("Sortir dans les y positifs", plateau.placePion(16, 17,
CouleurPion.BLANC));

            plateau.placePionSansVerif(16, 16, CouleurPion.BLANC);
            assertFalse("Sortir dans les x positifs", plateau.placePion(17, 16,
CouleurPion.BLANC));
        }

        /**
         * essaie de sauter un pion adverse de manière normal
         */
        @Test
        public void testSauterPion() {
            plateau.placePionSansVerif(4, 8, CouleurPion.BLANC);
            plateau.placePionSansVerif(6, 8, CouleurPion.NOIR);
            assertTrue("Sauter un pion de manière normale", plateau.placePion(2, 8,
CouleurPion.NOIR));
        }

        /**
         * essaie de sauter un pion adverse par les diagonales
         */
        @Test
        public void testSauterPionDiagonal() {
            plateau.placePionSansVerif(4, 8, CouleurPion.BLANC);
            plateau.placePionSansVerif(6, 8, CouleurPion.NOIR);
            plateau.placeMurSansVerif(3, 8);
            assertFalse("Sauter un pion de manière normale avec un mur",
plateau.placePion(2, 8, CouleurPion.NOIR));
```

```
        assertTrue("Se place sur la diagonale à droite", plateau.placePion(4,
10, CouleurPion.NOIR));

        plateau.placePionSansVerif(6, 8, CouleurPion.NOIR);
        assertTrue("Se place sur la diagonale à gauche", plateau.placePion(4,
6, CouleurPion.NOIR));
    }

    /**
     * essaie de sauter un pion adverse par des diagonales inacessibles
     */
    @Test
    public void testSauterPionDiagonaleGaucheMauvais() {
        plateau.placePionSansVerif(4, 8, CouleurPion.BLANC);
        plateau.placePionSansVerif(6, 8, CouleurPion.NOIR);
        plateau.placeMurSansVerif(3, 8);
        plateau.placeMurSansVerif(4, 7);
        assertFalse("Essaie de se place sur la diagonale à gauche alors qu'il y
a un mur", plateau.placePion(4, 6, CouleurPion.NOIR));
    }

    @Test
    public void testSauterPionDiagonaleDroiteMauvais() {
        plateau.placePionSansVerif(4, 8, CouleurPion.BLANC);
        plateau.placePionSansVerif(6, 8, CouleurPion.NOIR);
        plateau.placeMurSansVerif(3, 8);
        plateau.placeMurSansVerif(4, 9);
        assertFalse("Essaie de se place sur la diagonale à gauche alors qu'il y
a un mur", plateau.placePion(4, 10, CouleurPion.NOIR));
    }
}
```

```
package test.jeu.plateau;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.BeforeClass;
import org.junit.Test;

import jeu.classesTest.PlateauTest;

/**
 * TestVerifPassage permet de tester la méthode verifPassage() de la class
Plateau
 */
public class TestVerifPassage {
    private static PlateauTest plateau;

    @BeforeClass
    public static void setUp() {
        plateau = new PlateauTest();
    }
```

```
	/**
	 * teste la methode avec des barrières sur toute la vertical
	 */
	@Test
	public void testeLigne() {
	    int moitiePlateau = (plateau.getLePlateau().length/2) + 1;
	    for (int i = 0; i < plateau.getLePlateau().length - 3; i+=2) {
	        plateau.placeMurSansVerif(moitiePlateau, i);
	    }

	    assertFalse("Essaie de poser un mur pour bloquer l'adversaire",
plateau.placeMur(moitiePlateau, plateau.getLePlateau().length - 3));
	    assertTrue("Essaie de poser un mur sans bloquer les joueurs",
plateau.placeMur(0, 1));
	}
 }
```

//CLASSES SE TROUVANT DANS LE PACKAGE jeu.classesTest
//CLASSE jeuTest

```
package jeu.classesTest;

import jeu.CouleurPion;
import jeu.Jeu;
import jeu.Mode;

/**
* jeu de test
*/
public class jeuTest extends Jeu {

	/**
	* créé un jeu pour les tests
	*/
	public jeuTest() {
	    super("joueur1", "joueur2", Mode.HH);
	    this.plateau = new PlateauTest() ;
	}

	/**
	* positionne le pion afin de finir le jeu
	*/
	public void poseFin() {
	    ((PlateauTest) this.plateau).placePionSansVerif(0, 0,
CouleurPion.NOIR);
	}

	/**
	* met un pion comme si les joueurs jouaient
	*/
	public void jouerNormal() {
	    ((PlateauTest) this.plateau).placePionSansVerif(2, 0,
CouleurPion.NOIR);
	}

	/**
```

```
	* lance checkFinLancer de Jeu
	* @return le résultat de checkFinLancer de Jeu
	*/
	public boolean checkFinLancer() {
		return this.checkFin();
	}
}
```

//CLASSE plateauTest

```
package jeu.classesTest;

import jeu.CasePion;
import jeu.CaseBarriere;
import jeu.CouleurPion;
import jeu.Plateau;

/**
* plateau de test
*/
public class PlateauTest extends Plateau {
	/**
	* permet de placer un pion sans vérification
	* @param x la coordonnée x
	* @param y la coordonnée y
	* @param couleur la couleur du pion
	*/
	public void placePionSansVerif(int x, int y, CouleurPion couleur) {
		if (x < 0 || y < 0 || x > this.lePlateau[1].length || y >
		this.lePlateau.length || couleur == null) {
			throw new IllegalArgumentException("placePionSansVerif a
des parametres non autorisés");
		}
		if (x % 2 == 0 && y % 2 == 0) {
			((CasePion) this.lePlateau[x][y]).setCouleur(couleur);
			if (couleur == CouleurPion.BLANC) {
				this.caseBlanc.setCouleur(CouleurPion.FREE);
				this.caseBlanc = ((CasePion)this.lePlateau[x][y]);
			} else {
				this.caseNoir.setCouleur(CouleurPion.FREE);
				this.caseNoir = ((CasePion)this.lePlateau[x][y]);
			}

		} else {
			System.err.println("pas une case pion !");
		}
	}

	/**
	* permet de placer un mur sans vérification
	* @param x x où on place le mur
	* @param y y où on place le mur
	*/
	public void placeMurSansVerif(int x, int y) {
		if (x < 0 || y < 0 || x > this.lePlateau[1].length || y >
		this.lePlateau.length) {
			throw new IllegalArgumentException("placeMurSansVerif a des
parametres non autorisés");
		}
		if ((x % 2 == 0 && y % 2 == 1) || (y % 2 == 0 && x % 2 == 1)) {
```

```
                ((CaseBarriere) this.lePlateau[x][y]).setPresenceTrue();
            } else {
                System.err.println("pas une case barriere !");
            }
        }
    }
```

Fichier build.xml :

```
<project name="Projet fin annee" default="." basedir=".">
      <description>
      Fait la javadoc, la compilation et les .jar du projet.
      </description>
      <property name="src" location="src" />
      <property name="javadoc" location="javadoc" />
      <property name="class" location="class" />
      <property name="jar" location="dossierJar" />

      <target name="initJavadoc">
            <delete dir="${javadoc}" />
            <mkdir dir="${javadoc}" />
      </target>

      <target name="initClass">
            <delete dir="${class}" />
            <mkdir dir="${class}" />
      </target>

      <target name="initJar">
            <delete dir="${jar}" />
            <mkdir dir="${jar}" />
      </target>

      <target name="doc" depends="initJavadoc">
            <javadoc sourcepath="${src}" destdir="${javadoc}">
                  <fileset dir="${src}" defaultexcludes="yes">
                        <include name="${src}/**java" />
                  </fileset>
            </javadoc>
      </target>

      <target name="compil" depends="initClass">
            <javac srcdir="${src}" destdir="${class}">
            </javac>
      </target>

      <target name="jar" depends="initJar,compil">
            <jar jarfile="./${jar}/QuoridorGraph.jar" basedir="${class}">
                  <fileset dir="${src}">
                        <include name="images/**/*.png" />
                  </fileset>
                  <manifest>
                        <attribute name="Main-Class" value="Menu"/>
                  </manifest>
            </jar>
            <jar jarfile="./${jar}/QuoridorTexte.jar" basedir="${class}">
                  <exclude name="${class}/view/**" />
                  <exclude name="${class}/control/**" />
                  <manifest>
                        <attribute name="Main-Class" value="Lanceur"/>
                  </manifest>
            </jar>
      </target>
</project>
```