Thomas LELIEVRE & Florian LELOUP 14 novembre 2013

RAPPORT TP3 V&V

Question 1

18	Private field 'xSize' could be made final; it is only initialized in the declaration or constructor. **Ok : private int xSize => private final int xSize**
23	Private field 'ySize' could be made final; it is only initialized in the declaration or constructor. **Ok : private int ySize => private final int ySize**
28	Private field 'pawns' could be made final; it is only initialized in the declaration or constructor. **Ok : private ArrayList<Pawn> pawns => private final ArrayList<Pawn> pawns**
33	Private field 'xBonusSquare' could be made final; it is only initialized in the declaration or constructor. **Ok : private int xBonusSquare => private final int xBonusSquare**
38	Private field 'yBonusSquare' could be made final; it is only initialized in the declaration or constructor. **Ok : private int yBonusSquare => private final int yBonusSquare**
64	Parameter 'numberOfPawns' is not assigned and could be declared final **Ok : public Board(int numberOfPawns => public Board(final int numberOfPawns**
64	Parameter 'sizeX' is not assigned and could be declared final **Ok : int sizeX, => final int sizeX,**
65	Parameter 'sizeY' is not assigned and could be declared final **Ok : int sizeY, => final int sizeY,**
66	Local variable 'random' could be declared final **Ok : Random random = new Random() => final Random random = new Random()**
73	Avoid instantiating new objects inside loops **Faux positif, l'object pawn doit être instancié à chaque passage dans la boucle**
73	Local variable 'pawn' could be declared final **Ok : Pawn pawn = new Pawn => final Pawn pawn = new Pawn**
87	Parameter 'x' is not assigned and could be declared final **Ok : (int x => (final int x**
87	Parameter 'y' is not assigned and could be declared final **Ok : , int y) { => , final int y) {**

88	Local variable 'p' could be declared final **Ok : for (Pawn p : pawns) => for (final Pawn p : pawns)**
100	Parameter 'pawn' is not assigned and could be declared final **OK : removePawn(Pawn pawn) { => removePawn(final Pawn pawn) {**
108	Parameter 'pawn' is not assigned and could be declared final **OK : addPawn(Pawn pawn) { => addPawn(final Pawn pawn) {**
120	Parameter 'x' is not assigned and could be declared final **OK : isBonusSquare(int x => isBonusSquare(final int x**
120	Parameter 'y' is not assigned and could be declared final **OK : , int y) { => , final int y) {**
139	Local variable 'p' could be declared final **OK : for (Pawn p : pawns) { => for (final Pawn p : pawns) {**
155	Local variable 'result' could be declared final **OK : Pawn result = currentPawn; => final Pawn result = currentPawn;**
168	Parameter 'x' is not assigned and could be declared final **OK : (int x => (final int x**
168	Parameter 'y' is not assigned and could be declared final **OK : , int y) { => , final int y) {**
170	Local variable 'content' could be declared final **OK : Pawn content = => final Pawn content =**
177	Use equals() to compare object references. **OK : content == currentPawn => content.equals(currentPawn)**
193	Prefer StringBuffer over += for concatenating strings **C'est un faux positif car à la compilation, le compilateur java transforme automatiquement l'opérateur += en StringBuffer.**
195	Prefer StringBuffer over += for concatenating strings **C'est un faux positif car à la compilation, le compilateur java transforme automatiquement l'opérateur += en StringBuffer.**

Question 2

Lorsque l'on applique PMD sur ces propres sources on observe que PMD viole de nombreuses règles qu'ils définissent eux-mêmes.

On peut donc en déduire que de nombreuses règles provoque des faux positifs.