Nom : .....

Prénom : .....

Groupe : .....

Identifiant : .....

/ 50

Haute École de Bruxelles
École Supérieure d'Informatique
Bachelor en Informatique

2015 – 2016

Développement – 1ère

Examen

Star Wars

Consignes

Pour l'ensemble de l'examen,

- Vous avez 4 heures de temps ;
- Vous gérerez ce temps selon votre meilleure convenance pour réaliser les solutions en algorithmique et en Java.

Le jeu à implémenter consiste en la mission de deux *jedis* (Luke et Yoda) sur un plateau de jeu mêlé d'ennemis. Il s'agit d'un jeu de plateau à deux joueurs, le premier joueur jouant Luke représenté à gauche sur la figure 1 et le second joueur jouant Yoda représenté à droite sur la figure 1 . Comme on le voit sur la figure 1 le plateau de jeu est divisé en 10 zones [1] représentées par des positions allant de 1 à 10. Chaque personnage du jeu (*jedis* ou ennemis) possède une position. Deux ennemis

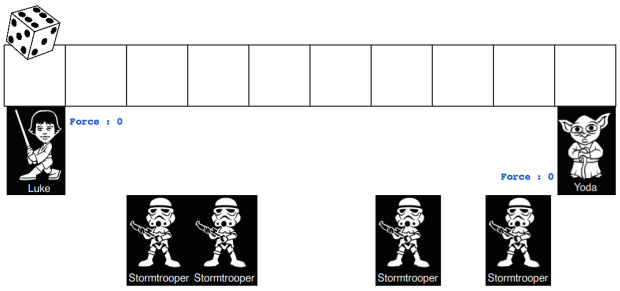

FIGURE 1 – En début de jeu les deux *jedis* sont placés au bord du plateau de jeu. Les ennemis (Stormtroopers) sont placés au hasard sur le plateau de jeu (sauf en première et dernière case). Le dé permettra de mettre en mouvement les *jedis*.

ne peuvent occuper la même case. En début de partie les deux *jedis* sont placés aux bords opposés du terrain (première et dernière case du plateau). Ces deux cases sont choisies comme case départ car aucun ennemi ne peut s'y trouver.

À chaque tour un joueur (Luke ou Yoda) lance le dé et a le droit de se déplacer du

1. Par la suite la taille du plateau de jeu sera variable.

Version du document : 19 janvier 2016

nombre de case(s) indiqué par le dé, soit vers la droite soit vers la gauche, comme il le désire. Luke débute la partie. Sur la figure 2 on voit que Luke obtient un 3 avec le dé et décide d'aller vers la droite.

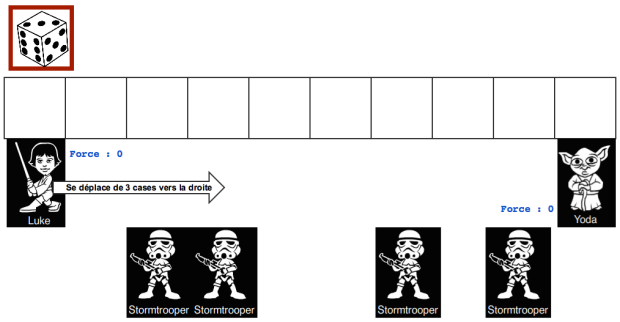

FIGURE 2 – Luke obtient un 3 avec le dé, il décide d'aller vers la droite.

Après avoir effectué son déplacement le *jedi* sécurise la zone (c'est-à-dire la case sur laquelle il vient d'arriver). L'ennemi disparait de celle-ci. Via cette action la force du *jedi* est modifiée, si un ennemi se trouve sur la zone il gagne un point de force.

Sur la figure 3, Luke après être arrivé sur la case numéro 4 sécurise cette zone. Un ennemi se trouve sur cette case, il gagne un point de force et l'ennemi disparaît.

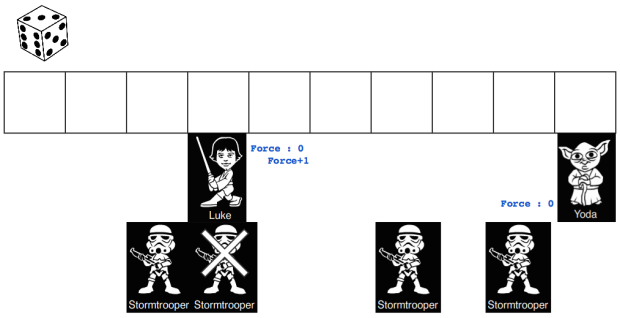

FIGURE 3 – Luke obtient 1 point de force supplémentaire en sécurisant sa zone occupée par un ennemi.

Aucune action particulière n'est à prévoir si l'autre *jedi* occupait cette case.

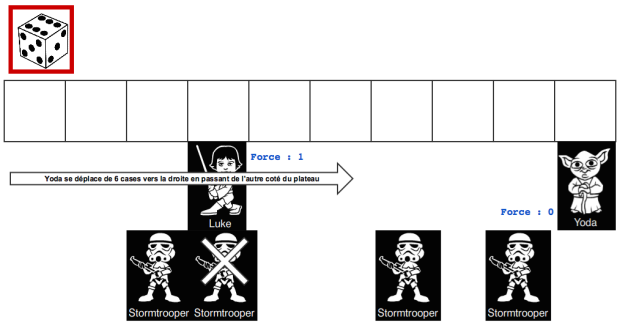

FIGURE 4 – Yoda obtient un 6 avec le dé et décide de se déplacer vers la droite, il fait le tour du plateau de jeu.

Si son déplacement le fait traverser un des bords du plateau, le *jedi* apparait à l'autre bout du plateau. Le plateau est refermé sur lui même. Sur la figure 4 on observe Yoda qui obtient un 6 après avoir lancé le dé.

Il décide de partir vers la droite et vu que le plateau est fermé sur lui-même, il se retrouve en case 6. Sur la figure 5 Yoda sécurise la case numéro 6, sa force reste

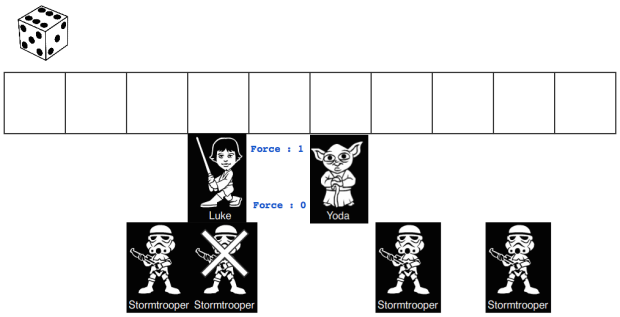

FIGURE 5 – Yoda sécurise la zone, sa force n'est pas modifiée vu qu'aucun ennemi n'est présent.

inchangée vu qu'aucun ennemi ne s'y trouve.

Le jeu se termine lorsqu'il n'y a plus d'ennemis. Le vainqueur est le *jedi* dont la force est la plus grande.

Durant la conception de ce jeu on utilisera des tableaux contenant la position des personnages du jeu :

— un tableau pour la position des deux *jedis*. Dans notre exemple en début de partie ce tableau vaut $\{1, 10\}$.

— un tableau pour la position des ennemis. Dans notre exemple en début de partie ce tableau vaut $\{3, 4, 7, 9\}$.

Le nombre d'ennemis sera compris entre 1 et la moité de la taille du plateau de jeu et sera choisi aléatoirement en début de partie. Notez que tous ces tableaux seront indicés en commençant à 0.

Le contenu d'une case d'un de ces tableaux est un nombre compris entre 1 et 10 qui représente la position du personnage. Si le personnage disparaît la case prend la valeur 0. Dans notre exemple au moment de la figure 5 le tableau de la position des ennemis vaut $\{3, 0, 7, 9\}$

Un tableau supplémentaire de deux cases contiendra la valeur de la force de chaque *jedi*. En début de partie la force des deux *jedis* vaut 0.

I
Algorithmique

Consignes

Pour la partie algorithmique,

- Vous ne pouvez pas utiliser de notes.
- Vos réponses se feront au bic bleu ou noir sur la feuille de réponses.
- Sauf spécification du contraire, les données lues ou reçues ne comportent pas d'erreurs.
- Les noms en gras (variables et types) doivent être respectés.
- Veillez à travailler de manière modulaire.

Algorithmes disponibles

Dans le cadre de cet examen vous pouvez utiliser si nécessaire les algorithmes ci-dessous sans les écrire :

- `algorithme` **`hasard`**`(n↓ : entier)` $\rightarrow$ `entier` qui renvoie un entier au hasard entre 1 et la valeur n ;
- `algorithme` **`estLibre`**`(ennemis↓ : tableau de nbEnnemis entiers, position↓ : entier)` $\rightarrow$ `booléen` qui reçoit un tableau de **nbEnnemis** cases et une position en paramètre et qui renvoie un booléen indiquant si la position est occupée par un ennemi du tableau passé en paramètre ;
- `algorithme` **`securiseZone`**`(ennemis↓↑ : tableau de nbEnnemis entiers, position↓ : entier)` $\rightarrow$ `booléen` qui retourne 0 si aucun ennemi ne se trouvait à la position donnée en paramètre et 1 sinon. Cet algorithme met également à jour le tableau **ennemis** en mettant à 0 la position de l'ennemi occupant la position donnée en paramètre, cette action fera disparaitre l'éventuel ennemi qui s'y trouve. Si on donne le tableau d'ennemis $\{3, 4, 7, 9\}$ et la position 4 en paramètre de cet algorithme, il renverra la valeur 1 et le tableau ennemis vaudra $\{3, 0, 7, 9\}$;
- `algorithme` **`affichePlateau`**`( jedi↓ : tableau de 2 entiers, force↓ : tableau de 2 entiers, ennemis↓ : tableau de nbEnnemis entiers, taille↓ : entier, joueur↓ : entier)` qui affiche le plateau de jeu sur base des différentes données de la partie (position des deux *jedis*, force des deux *jedis*, position des ennemis, taille du plateau de jeu et joueur courant).

1 Compte des ennemis (5 points)

Écrivez un algorithme **compteEnnemis** qui reçoit un tableau **ennemis** de **nbEnnemis** entiers représentant les positions des ennemis et qui compte le nombre d'ennemis non éliminés. Au début du jeu tous les ennemis ont une position (un nombre de 2 à **taille du plateau-1**, les positions 1 et **taille du plateau** étant occupée par les *jedis*). Après une élimination la position de l'ennemi passe à 0.

```
module compteEnnemis(ennemis↓ : tableau de nbEnnemis entiers) → entier
    nb : entier
    nb ← 0
    pour i de 0 à nbEnnemis-1 faire
        si ennemis[i] ≠ 0 alors
            nb ← nb + 1
        fin si
    fin pour
    retourner nb
fin module
```

2 Déplacement des *jedis* (5 points)

Écrivez un algorithme **calculePosition** qui :

- reçoit **jedi** un entier supérieur ou égal à 1 correspondant à la position du *jedi* à déplacer ;
- reçoit **taille** un entier correspondant à la taille du plateau de jeu ;
- lance un dé à 6 faces ;
- demande au joueur s'il souhaite se déplacer vers la droite, dans le cas contraire il se déplace vers la gauche ;
- calcule et retourne la nouvelle position du *jedi* en prenant garde aux conditions aux bords du plateau de jeu (rappelez-vous le déplacement de Yoda à la figure 4).

```
module calculePosition(jedi↓ :entier,taille↓ :entier,) → entier
    nbCase,nouvellePosition : entier
    aDroite : booléen
    nbCase ← hasard(6)
    afficher "Résultat du lancer de dé : ", nbCase
    afficher "Voulez-vous déplacer votre Jedi de " , nbCase , " case(s) vers la droite ?
(vrai ou faux)"
    demander aDroite
    si aDroite alors
        nouvellePosition ← jedi + nbCase
        si nouvellePosition > taille alors
            nouvellePosition ← nouvellePosition MOD taille
        fin si
    sinon
        nouvellePosition ← jedi - nbCase
        si nouvellePosition < 1 alors
            nouvellePosition ← nouvellePosition + taille
        fin si
    fin si
    retourner nouvellePosition
fin module
```

3 Initialiser le tableau des ennemis (5 points)

Écrivez un algorithme **placerEnnemis** qui :

- reçoit **nbEnnemis** le nombre d'ennemis à générer ;
- reçoit **taille** la taille du plateau du jeu ;
- retourne un tableau de **nbEnnemis** d'entiers contenant les positions des différents ennemis.

Ces positions sont choisies **aléatoirement** et ne peuvent se répéter. Si un ennemi occupe la position numéro 4, les ennemis suivants devront occuper une position différente. La première et la dernière position du plateau de jeu sont interdites vu qu'elles sont réservées à Luke et Yoda.

Pour écrire cet algorithme n'hésitez pas à utiliser l'algorithme **estLibre**(ennemis : tableau de nbEnnemis entiers, position : entier) → booléen. Pour ce faire n'oubliez pas d'initialiser votre tableau de **nbEnnemis** d'entiers à 0 en début d'algorithme.

Si on demande à l'algorithme de générer un tableau de 4 ennemis pour un plateau de taille 10, il retournera par exemple le tableau $\{3, 4, 7, 9\}$, qui ne contient ni la position 1 ni la position 10.

```
module placerEnnemis(nbEnnemis↓ :entier,taille↓ :entier) → tableau de nbEnnemis
entiers
    position : entier
    ennemis : tableau de nbEnnemis entiers
    ennemis ← {0, ..., 0}
    position ← 0
    pour i de 0 à nbEnnemis-1 faire
        tant que non estLibre(ennemis, position) faire
            position ← hasard(taille-2)+1
        fin tant que
        ennemis[i] ← position
    fin pour
    retourner ennemis
fin module
```

4 Jouer une partie (5 points)

Écrivez un algorithme **jouePartie** qui

- reçoit la taille du plateau de jeu [2] ;
- crée et initialise le tableau des positions des deux *jedis* Luke et Yoda ;
- crée et initialise le tableau des forces des deux *jedis* ;
- crée et initialise le tableau des positions des ennemis via l'algorithme **placerEnnemis**, le nombre d'ennemis est un nombre aléatoire compris entre 1 et la moité de la taille du plateau de jeu [3] ;
- tant que sur le plateau de jeu se trouve des ennemis faire jouer les deux *jedis* chacun à leur tour :
  - afficher le plateau de jeu ;
  - déplacer le *jedi* via l'algorithme **calculePosition** ;
  - mettre à jour la force du *jedi* après l'appel à l'algorithme **securiseZone**, un *jedi* gagnant un point de force si la zone était occupée par un ennemi ;
  - changer le joueur courant.
- affiche le résultat de la partie, c'est-à-dire le nom du vainqueur (Luke ou Yoda) ou la mention "Match nul".

```
module jouePartie(taille↓ : entier)
    joueur, nbEnnemis : entier
    jedi,force : tableau de 2 entiers
    nbEnnemis ← hasard(taille DIV 2)
    ennemis : tableau de nbEnnemis entiers

    jedi ← {1, taille}
    force ← {0, 0}
    ennemis ← placerEnnemis(nbEnnemis, taille)
    joueur ← 0

    tant que comptePersonne(ennemi) ≠ 0 faire
        affichePlateau(jedi, force, ennemis, taille, joueur)
        jedi[joueur] ← calculePosition(jedi[joueur], taille)
        force[joueur] ← force[joueur] + securiseZone(ennemis, jedi[joueur])
        joueur ← (joueur + 1) MOD 2
    fin tant que

    si force[0] > force[1] alors
        afficher "Vainqueur Luke " , force[0]
    sinon
        si force[0] = force[1] alors
            afficher "Match Null " , force[0] , " " , force[1]
        sinon
            afficher "Vainqueur Yoda " , force[1]
        fin si
    fin si
fin module
```

2. Fixée à 10 dans notre exemple elle peut prendre d'autres valeurs entières. La valeur minimale étant 3.
3. Si le plateau comporte un nombre impair de cases, le résultat est tronqué.

II
Java et laboratoire

Consignes

Pour la partie java,

- *Vous réaliserez votre travail sur* `linux1` *et le déposerez dans le casier linux de votre professeur par la commande* `casier`.
- *Vous disposez de toutes vos notes ainsi que de l'aide en ligne.*
- *Il ne suffit pas que votre code compile. Testez-le pour identifier d'éventuelles erreurs à l'exécution.*
- *La cotation tiendra compte aussi du style de programmation que vous avez acquis.*
- *Respectez bien les noms de package, classe, méthodes demandés dans l'énoncé.*
- *Vous remplacerez bien sûr g12345 par votre numéro d'étudiant.*

5 Sine qua non (0 point)

Créez un répertoire `evaluations/janvier`. Changez les droits sur votre répertoire `janvier` pour donner les permissions de lecture et d'exécution aux professeurs mais aucun droit aux autres étudiants. Appelez votre professeur pour lui montrer que vos permissions ont bien été changées.

Vous ne continuerez pas l'examen tant que cette question n'a pas été validée par votre professeur.

6 Travailler dans un package (2 points)

Dans la suite de l'examen, votre classe fera partie du package `evaluations.janvier`.

Votre programme s'appellera *StarWars.java* et sera situé dans
`/home/g12345/evaluations/janvier/sources/` [4]
et *StarWars.class* dans
`/home/g12345/evaluations/janvier/classes/evaluations/janvier`.

Écrivez ici :

— l'instruction que doit contenir votre classe pour faire partie du package demandé ;
package evaluations.janvier ;

4. Vous remplacerez évidemment *g12345* par **votre** identifiant

— la commande (complète et précise) que vous allez utiliser pour **compiler** votre classe ;
javac -d ~/evaluations/janvier/classes StarWars.java
— la commande (complète et précise) que vous allez utiliser pour **exécuter** votre classe ;
java evaluations.janvier.StarWars
— le contenu minimal de votre variable d'environnement *CLASSPATH*.
~/evaluations/janvier/classes

7 StarWarsUtils (2 points)

Nous avons écrit pour vous une classe *StarWarsUtils* qui contient les méthodes :

— public static boolean estLibre(int[] ennemis, int position) permettant de savoir si la position position est occupée par un personnage du tableau ennemis ;
— public static int securiseZone(int[] ennemis, int position) permettant de mettre à jour le tableau ennemis si un ennemi se trouve à la position position. Cette méthode retourne 0 si aucun ennemi ne se trouvait à la position position et 1 sinon ;
— public static void affichePlateau(int[] jedi, int[] force, int[] ennemis, int taille, int joueur) permettant d'afficher le plateau de jeu.

Nous savons que la classe *StarWarsUtils* se trouve sur la machine *linux1* dans un des sous-répertoires de `/eCours`.

La javadoc de la classe est à votre disposition dans un répertoire `doc` situé à côté de la classe.

— Quelle commande allez-vous utiliser pour retrouver le fichier *StarWarsUtils.class* ?
find /eCours -name StarWarsUtils.class
— Que devrez-vous ajouter au contenu de la variable d'environnement *CLASSPATH* pour pouvoir exécuter la classe *StarWarsUtils* ?
/eCours/java/outils
— Que s'affiche-t-il à l'écran lors de l'exécution de la classe *StarWarsUtils* ?
'Do. Or do not. There is no try.' - Yoda

8 StarWars (14 points)

Lors du td10, vous avez écrit pour vous une classe *Util* qui contient des méthodes utilitaires. N'hésitez pas à les utiliser dans le cadre de l'examen sans oublier de copier le fichier *Util.java* et ses dépendances dans le dossier `/home/g12345/evaluations/janvier/sources/` .

Dans votre classe *StarWars*, écrivez les méthodes :

— public static int compteEnnemis(int[] ennemis) ;

— public static int calculePosition(int jedi, int taille), comme pour le td10 les lectures d'entrées au clavier seront robustes ;

— public static int[] placerEnnemis(int nbEnnemis, int taille) qui lancera une IllegalArgumentException si **nbEnnemis** n'est pas compris entre 1 et **taille/2** ;

— public static int[] jouePartie(int taille) .

9 Javadoc (3 points)

Écrivez la javadoc pour la méthode
public static int[] placerEnnemis(int nbEnnemis, int taille).

Générez la javadoc dans le répertoire `myDoc`. Ecrivez la commande complète que vous avez utilisez pour générer cette documentation.

javadoc -charset utf-8 -d myDoc StarWars.java

10 JUnit (4 points)

Écrivez une série de 4 tests **JUnit** afin de valider la méthode
public static int compteEnnemis(int[] ennemis).

Écrivez ici :

— la commande permettant d'afficher le résultat des tests JUnit ;
java org.jUnit.runner.JUnitCore evaluations.janvier.StarWarsTest

— le contenu de votre variable d'environnement *CLASSPATH* pour pouvoir exécuter ces tests.
/usr/share/java/junit4.jar

```
1  package evaluations.janvier;
2
3  import g12345.dev1.janvier.StarWarsUtils;
4  import java.util.Scanner;
5
6  /**
7   * Jeu consistant en la mission de deux jedis (Luke et Yoda) sur
8   * un plateau de jeu m?l\u00e9 d'ennemis.
9   */
10 public class StarWars {
11
12     /**
13      * Compte le nombre de personnages non \u00e9limin\u00e9s.Au d\u00e9but du
14      * jeu tous les
15      * personnages ont une position (nombre de 1 \u00e0 10).Apr\u00e8s une
16      * \u00e9limination la
17      * position du personnage passe \u00e0 0.
18      * <p>
19      * @param ennemis tableau de n entiers repr\u00e9sentant les positions d'un
20      *                groupe de
21      *                personnages
22      * <p>
```

```
23      * @return le nombre de personnages non \u00e9limin\u00e9s
24      */
25     public static int compteEnnemis(int[] ennemis) {
26         int nb = 0;
27         for (int i = 0; i < ennemis.length; i++) {
28             if (ennemis[i] != 0) {
29                 nb++;
30             }
31         }
32         return nb;
33     }
34
35     /**
36      * Calcule et retourne la nouvelle position du jedi en prenant garde aux
37      * conditions aux bords du plateau de jeu.
38      * <p>
39      * @param jedi   un entier correspondant \u00e0 la position du jedi \u00e0
40      *               d\u00e9placer
41      * @param taille un entier correspondant \u00e0 la taille du plateau de jeu
42      * <p>
43      * @return la nouvelle position du jedi
44      */
45     public static int calculePosition(int jedi, int taille) {
46         int nouvellePosition = 0;
47         int nbCase = hasard(6);
48         System.out.println("R\u00e9sultat du lancer de d\u00e9 : " + nbCase);
49         String msg = "Voulez-vous d\u00e9placer votre Jedi de " +
50                 nbCase + " case(s) vers la droite \u2794 ? (y or n)";
51         boolean aDroite = Read.isYes(msg, new Scanner(System.in));
52         if (aDroite) {
53             nouvellePosition = jedi + nbCase;
54             if (nouvellePosition > taille) {
55                 nouvellePosition = nouvellePosition % taille;
56             }
57         } else {
58             nouvellePosition = jedi - nbCase;
59             if (nouvellePosition < 1) {
60                 nouvellePosition = nouvellePosition + taille;
61             }
62         }
63         return nouvellePosition;
64
65      // Une autre solution possible
66      //  if (!aDroite) {
67      // nbCase = taille - nbCase;
68      // }
69      // return (((jedi - 1) + nbCase) % (taille)) + 1;
70     }
71
72     /**
73      * Calcul les positions d'un groupe de personnages.
74      * Ces positions sont choisies al\u00e9atoirement et ne peuvent se
75      * r\u00e9p\u00e9ter.
76      * Si un ennemi occupe la position num\u00e9ro 4 les ennemis suivants
77      * devront
78      * occuper une position diff\u00e9rente.
79      * <p>
80      * @param nbEnnemis le nombre de personnages \u00e0 g\u00e9n\u00e9rer
```

```
81      * @param taille        la  taille  du plateau du jeu
82      * <p>
83      * @return un tableau de nbEnnemis d'entiers contenant les positions des
84  diff \u00e9rents personnages
85      */
86     public static int[] placerEnnemis(int nbEnnemis, int taille) {
87         if (nbEnnemis < 1 || nbEnnemis > taille / 2) {
88             throw new IllegalArgumentException("Le nombre de personnages " +
89                                                "doit \u00eatre compris " +
90                                                "entre 1 et " +
91                                                (taille / 2) + " " + taille);
92         }
93         int[] tab = new int[nbEnnemis];
94         int position;
95         for (int i = 0; i < tab.length; i++) {
96             do {
97                 position = hasard(taille - 2) + 1;
98             } while (StarWarsUtils.estLibre(tab, position));
99             tab[i] = position;
100        }
101        return tab;
102    }
103
104    /**
105     * Affiche le r\u00e9sultat de la partie (vainqueur ou match null).
106     * <p>
107     * @param force tableau de deux entiers repr\u00e9sentant la Force des deux
108     *              jedis
109     *              en fin de partie
110     */
111    public static void afficheVainqueur(int[] force) {
112        if (force[0] > force[1]) {
113            System.out.println("Vainqueur Luke " + force[0]);
114        } else {
115            if (force[0] == force[1]) {
116                System.out.println("Match Null " + force[0] + " " + force[1]);
117            } else {
118                System.out.println("Vainqueur Yoda " + force[1]);
119            }
120        }
121    }
122
123    /**
124     * Lance un tour le joueur courant.
125     * <p>
126     * @param jedis     position des deux jedis
127     * @param force     force des deux jedis
128     * @param ennemis position des ennemis
129     * @param taille    taille du plateau de jeu
130     * @param joueur joueur courant
131     */
132    public static void joueTour(int[] jedis, int[] force,
133                                int[] ennemis,
134                                int taille, int joueur) {
135        StarWarsUtils.affichePlateau(jedis, force, ennemis, taille, joueur);
136        jedis[joueur] = calculePosition(jedis[joueur], taille);
137        force[joueur] = force[joueur] +
138                        StarWarsUtils.securiseZone(ennemis, jedis[joueur]);
```

```
139     }
140
141     /**
142      * Lance une partie du jeu.
143      * <p>
144      * @param taille  taille  du plateau de jeu
145      */
146     public static void jouePartie(int taille ) {
147         int [] jedis = {1, taille };
148         int [] force = {0, 0};
149         int [] ennemis = placerEnnemis(hasard(taille / 2), taille );
150         int joueur = 0;
151         while (compteEnnemis(ennemis) != 0) {
152             joueTour(jedis, force , ennemis, taille , joueur);
153             joueur = (joueur + 1) % 2;
154         }
155         afficheVainqueur(force );
156     }
157
158     /**
159      * G\u00e9n\u00e8re un nombre entier al\u00e9atoire entre 1 et
160      * max.
161      * <p>
162      * @param max borne maximale
163      * <p>
164      * @return nombre al\u00e9atoire entre 1 et max
165      */
166     public static int hasard(int max) {
167         return (int) (Math.random() * max) + 1;
168     }
169
170     /**
171      * Point d'entr\u00e9e du programme.
172      * <p>
173      * @param args no arguments
174      */
175     public static void main(String[] args) {
176         int sizeField = 10;
177         jouePartie( sizeField );
178     }
179 }
```

```
1  package evaluations.janvier;
2
3  import org.junit.Test;
4
5  import static org.junit.Assert.assertEquals;
6
7  public class StarWarsTest {
8
9      @Test
10     public void testCount01() {
11         System.out.println("count01");
12         int [] tab = {0,0,0,0,0,0,0,0,0,0};
13         int expResult = 0;
14         int result = StarWars.compteEnnemis(tab);
15         assertEquals(expResult, result );
```

```
16      }
17
18      @Test
19      public void testCount02() {
20          System.out.println("count02");
21          int [] tab =  {1,2,3,4,5,6,7,8,9,10};
22          int expResult = 10;
23          int result  = StarWars.compteEnnemis(tab);
24          assertEquals(expResult, result );
25      }
26
27      @Test
28      public void testCount03() {
29          System.out.println("count03");
30          int [] tab =  {0,1,2,3,4,5,6,7,8,0};
31          int expResult = 8;
32          int result  = StarWars.compteEnnemis(tab);
33          assertEquals(expResult, result );
34      }
35
36      @Test(expected = NullPointerException.class)
37      public void testCount04() {
38          System.out.println("count04");
39          int [] tab = null;
40          StarWars.compteEnnemis(tab);
41      }
42
43
44  }
```