Nom : .....

Prénom : .....

Groupe : .....

Identifiant : .....

/ 50

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

2015 – 2016

Développement – 1ère

Examen

Le jeu des nains et de Blanche-Neige

La méchante Reine s'est transformée en vieille sorcière. Elle part à la recherche de Blanche-Neige pour lui donner une pomme empoisonnée. Avertis par les animaux de la forêt, les 7 nains quittent la mine et courent prévenir leur amie.

Il s'agit d'un jeu collaboratif qui se joue de 2 à 7 joueurs, les nains. Ils doivent parcourir le chemin en suivant les cases jusqu'à la fin (case 90). Si un nain arrive à voir Blanche-Neige (toujours sur la case 90) avant la sorcière, les nains gagnent. Si c'est la sorcière qui y arrive en premier, c'est la sorcière qui gagne.

Chaque joueur choisit un nain et le pose sur la mine, toujours en case 18. La sorcière est posée sur les souterrains du château, toujours en case 0.

Le jeu se joue avec 3 dés. Un premier dé à 2 couleurs, noir et rouge, désignant si

Version du document : 24 janvier 2016

c'est le nain (si c'est la couleur rouge) qui avancera ou la sorcière (si c'est la couleur noire). Suivant que ce soit le nain ou la sorcière, le second dé lancé sera différent ; il y a un dé pour les nains (un nain peut avancer de 1, 2 ou 3 cases) et un dé pour la sorcière (la sorcière a une chance sur 3 d'avancer de 1 case et 2 chances sur 3 d'avancer de 4 cases).

Chaque joueur joue à tour de rôle dans le sens des aiguilles d'une montre. Quand vient son tour, on lance un premier dé à 2 couleurs.

Si le résultat est rouge, on lance le dé des nains et on avance son nain du nombre de cases indiqué en suivant bien le chemin. Plusieurs nains peuvent se trouver sur une même case. Mais la cohabitation est impossible entre les nains et la sorcière. Si un nain doit finir son déplacement sur une case occupée par la sorcière, il ne bouge pas de sa position.

Si le résultat est noir, on lance le dé de la sorcière et on avance la sorcière du nombre de cases indiqué en suivant bien le chemin. Si la sorcière doit finir son déplacement sur une case occupée par un nain, elle ne bouge pas de sa position.

FIGURE 1 – Jeu original. On y voit le plateau de jeu, la sorcière, les nains et Blanche-Neige en case 90. Nous n'utiliserons pas ici les cartes Joker ni les échelles ni les pièges. Contrairement à ce qui est représenté ici (1 dé jaune et 1 dé blanc), nous utiliserons 3 dés tels qu'ils sont décrits dans l'énoncé.

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.

1 Afficher les positions des nains et de la sorcière (4 points)

Écrivez un algorithme **afficherPersonnages** qui reçoit un tableau de **n** entiers des positions de la sorcière et des nains sur le plateau de jeu et qui affiche la position de chaque personnage. La position de la sorcière se trouve toujours dans la première case du tableau. Les autres cases contiennent la position de chacun des nains.

Par exemple : si le tableau reçu contient {21, 26, 46, 26, 52}, votre algorithme affichera :
La sorcière est en position 21
Le nain 1 est en position 26
Le nain 2 est en position 46
Le nain 3 est en position 26
Le nain 4 est en position 52

```
module afficherPersonnages(positions : tableau de n entiers)
    afficher "La sorcière est en position ", positions[0]
    pour i de 1 à n-1 faire
        afficher "Le nain ", i, " est en position ", positions[i]
    fin pour
fin module
```

2 Initialiser les positions des nains et de la sorcière (4 points)

Écrivez un algorithme **initialiserPersonnages** qui reçoit **nbNains**, le nombre de nains qui jouent en paramètre, qui crée un tableau de **nbNains+1** entiers (nbNains + 1 pour la sorcière en plus des nbNains), qui, sans utiliser la notation raccourcie, initialise la première case du tableau à 0 et les autres cases à 18 ; et qui retourne ce tableau.

```
module initialiserPersonnages(nbNains : entier) → tableau de nbNains+1 entiers
    personnages : tableau de nbNains+1 entiers
    personnages[0] ← 0
    pour i de 1 à nbNains faire
        personnages[i] ← 18
    fin pour
    retourner personnages
fin module
```

3 Un nain est-il déjà sur la case ? (7 points)

Écrivez un algorithme **estCaseSansNain** qui reçoit le tableau de **n** entiers des positions de la sorcière et des nains sur le plateau de jeu et le numéro de la case dont il faut vérifier la disponibilité et qui retourne vrai si la case n'est pas déjà occupée par un nain et faux sinon. Cet algorithme ne regarde pas la position de la sorcière.

```
module estCaseSansNain(positions : tableau de n entiers, case : entier) → booléen
    trouvé : booléen
    i : entier
    trouvé ← faux
    i ← 1
    tant que NON trouvé ET i<n faire
        trouvé ← positions[i] = case
        i ← i+1
    fin tant que
    retourner NON trouvé
fin module
```

4 Jouer 1 coup (5 points)

Écrivez un algorithme **jouerCoup** qui reçoit le tableau de **n** entiers des positions de la sorcière et des nains sur le plateau de jeu et le numéro du nain qui joue le coup (le numéro commence à 1).

Pour jouer un coup,

- Tirez au sort pour savoir si c'est la sorcière ou un nain qui avance (lancez un dé à 2 valeurs/couleurs). La probabilité est la même que ce soit la sorcière ou le nain qui avance.
  - Si c'est la sorcière, tirez au sort pour savoir si elle avance de 1 ou 4 cases. Elle a une chance sur 3 d'avancer d'une case et donc, 2 chances sur 3 d'avancer de 4 cases.

    Si il n'y a aucun nain sur la case d'arrivée, avancez la sorcière.
  - Si c'est le nain, tirez au sort pour savoir si il avance de 1, 2 ou 3 cases. La probabilité est la même pour chacun des choix.

    Si la sorcière n'est pas sur la case d'arrivée, avancez le nain.

Vous pouvez faire appel à l'algorithme **hasard(n : entier)**.

```
module jouerCoup(positions↓↑ : tableau de n entiers, numNain↓ : entier)
    sorcièreOuNain, nbCases : entiers

    sorcièreOuNain ← hasard(2)
    // si c'est la sorcière qui avance :
    si sorcièreOuNain = 1 alors
        afficher "C'est la sorcière qui avance"
        nbCases ← hasard(3)
        si nbCases ≠ 1 alors
            nbCases ← 4
        fin si
        si estCaseSansNain(positions, positions[0]+nbCases) alors
            positions[0] ← positions[0]+nbCases
        fin si
    sinon
        // si c'est le nain qui avance :
        afficher "C'est le nain qui avance"
        nbCases ← hasard(3)
        si positions[0] ≠ positions[numNains]+nbCases alors
            positions[numNains] ← positions[numNains]+nbCases
        fin si
    fin si
fin module
```

5 Jouer une partie (5 points)

Écrivez un algorithme **jouerPartie** qui

- reçoit en paramètre le nombre de nains qui jouent **nbNains** ;
- crée et initialise un tableau de **nbNains+1** entiers des positions de la sorcière et des nains sur le plateau de jeu, en faisant appel à l'algorithme **initialiserPersonnages** écrit ci-dessus ;
- affiche les positions de la sorcière et des nains sur le plateau de jeu ;
- fait appel à l'algorithme

  algorithme isOver(positions : **tableau** de n entiers) → booléen

  Vous ne devez pas écrire le code de l'algorithme **isOver** qui retourne vrai si la partie est finie et faux sinon. La partie est finie quand un nain ou la sorcière est arrivé à voir Blanche-Neige, en case 90.
- tant que la partie n'est pas finie, joue un coup, affiche les positions et passe au joueur suivant.

```
module jouerPartie(nbNains : entier)
    numNain : entier
    positions : tableau de nbNains+1 entiers
    numNain ← 1
    positions ← initialiserPersonnages(nbNains)
    afficherPersonnages(positions)

    tant que NON isOver(positions) faire
        jouerCoup(positions, numNain)
        afficherPersonnages(positions)
        numNain ← (numNain MOD nbNain) + 1
    fin tant que
fin module
```

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.*

6 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.

7 Travailler dans un package (3 points)

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

Votre programme s'appellera *BlancheNeige.java* et sera situé dans
`/home/g12345/evaluations/janvier/sources/` [1]
et *BlancheNeige.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 ;

— la commande (complète et précise) que vous allez utiliser pour **compiler** votre classe ;
javac -d ~/evaluations/janvier/classes BlancheNeige.java

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

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

8 BlancheNeige (6 points)

Dans votre classe *BlancheNeige*, écrivez les méthodes :

— public static void afficherPersonnages(int[] positions) ;
— public static int[] initialiserPersonnages(int nbNains) qui lancera une IllegalArgumentException si le **nbNain** n'est pas compris entre 2 et 7 ;
— public static boolean estCaseSansNain(int[] positions, int laCase) ;
— public static void jouerCoup(int[] positions, int numNain) ;
— public static void jouerPartie(int nbNains) ;

9 UtilBlancheNeige (6 points)

Nous avons écrit pour vous une classe *UtilBlancheNeige* qui contient une méthode

public static boolean isOver(int[] positions)

permettant de savoir si le jeu est fini. Nous savons qu'elle 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 UtilBlancheNeige.class ?

— Que devrez-vous ajouter au contenu de la variable d'environnement *CLASSPATH* pour pouvoir exécuter la classe UtilBlancheNeige ?

10 Javadoc (4 points)

Écrivez la javadoc pour la méthode public static int[] initialiserPersonnages(int nbNains).

11 JUnit (6 points)

Écrivez une série de tests **JUnit** pour la méthode public static int[] initialiserPersonnages(int nbNains).

Écrivez ici :

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

— 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 java.util.Scanner;
4  import util.blanche.UtilBlancheNeige;
5
6  public class BlancheNeige{
7
8      public static void main(String[] args){
9          jouerPartie(2);
10     }
11     /** affiche les positions de la sorcière et des nains
12      * @param positions les positions à afficher
13      */
14     public static void afficherPersonnages(int[] positions){
15         System.out.println("La sorcière est en position " + positions[0]);
16         for(int i=1; i< positions.length; i++){
17             System.out.println("Le nain " + (i) +
18                     " est en position " + positions[i]);
19         }
20     }
21
22     /** initialise les positions de la sorcière et des nains
23      * @param nbNains le nombre de nains
24      * @return un tableau des positions de départ de la sorcière (en 0)
25      * et des nains (en 18)
26      * @throws IllegalArgumentException si le nbNains
27      */
28     public static int[] initialiserPersonnages (int nbNains){
29         if (nbNains < 2 || nbNains > 7)
30             throw new IllegalArgumentException ("paramètre invalide");
31
32         int[] personnages = new int[nbNains + 1];
33         for(int i=1; i<=nbNains; i++){
34             personnages[i] = 18;
35         }
36         return personnages;
37     }
38
39     /** crée un tableau de nbNains + 1 personnages
40      * intitalis é à 0 pour la sorcière et à 18 pour les nbNains.
41      * @param nbNains le nombre de nains àcréer dans le tableau.
42      */
43     public static boolean estCaseSansNain(int[] positions, int laCase){
44         boolean trouvé=false;
45         int i = 1;
46         while(!trouvé && i<positions.length){
47             trouvé = positions[i]==laCase;
```

```
48              i++;
49          }
50          return !trouvé;
51      }
52
53      private static int hasard(int nb){
54          return (int)(Math.random() * nb + 1);
55      }
56
57      /** joue un coup
58       * @param positions le tableau des positions de la sorcière et des nains
59       qui sera modifié avec la position après le coup
60       * @param numNain le numéro du nain àbouger
61       entre 1 et positions.length - 1
62       */
63      public static void jouerCoup(int[] positions, int numNain){
64          int sorcièreOuNain;
65          int nbCases;
66          sorcièreOuNain = hasard(2);
67
68          if(sorcièreOuNain == 1){
69              System.out.println("C'est la sorcière qui avance ");
70              nbCases = hasard(3);
71              if(nbCases != 1)
72                  nbCases = 4;
73              if(estCasesansNain(positions, positions[0] + nbCases))
74                  positions[0] = positions[0] + nbCases;
75          } else {
76              System.out.println("C'est le nain qui avance ");
77              nbCases = hasard(3);
78              if(positions[0] != positions[numNain] + nbCases)
79                  positions[numNain] = positions[numNain] + nbCases;
80          }
81      }
82
83
84      public static void jouerPartie(int nbNains){
85          int[] positions = initialiserPersonnages(nbNains);
86          int numNain = 1;
87          afficherPersonnages(positions);
88
89          while(! UtilBlancheNeige.isOver(positions)){
90              jouerCoup(positions, numNain);
91              afficherPersonnages(positions);
92              numNain = ((numNain + 1)%nbNains) + 1;
93          }
94      }
95  }
```

12 Javadoc ()

Générez la javadoc dans le répertoire `doc`.

```
javadoc -charset utf-8 -d doc BlancheNeige.java
```