INFO-F-203 : PROJET D'ALGORITHMIQUE 2
ÉVASION DU LABYRINTHE

Daniele Catanzaro Luciano Porretta Nikita Veshchikov

version du 13 Octobre 2014

1 Le problème

Monsieur *Pakkuman* se réveille au milieu d'un labyrinthe. Il ne sait pas comment il est arrivé là, mais il veut s'en sortir dès que possible, car pour lui tenir compagnie dans le labyrinthe il y a des petits monstres qui attaquent tout ce qui est dans leur voisinage. Comme ils sont très friands des bonbons qui se trouvent à l'intérieur du labyrinthe, une victime peut éviter d'être attaquée par un monstre en lui donnant à manger un bonbon recueilli auparavant.

Comment doit procéder M. Pakkuman pour arriver à sortir vivant du labyrinthe ?

```
Situation de départ:
+---+---+---+---+---+---+---+---+
|   |             o   4         |
+   +---+   +---+---+---+   +   +
|       |   |           |   |   |
+---+   +   +   +---+   +   +   +
| P |   |   |       |   |   |   |
+   +   +   +---+   +   +   +   +
|       | 1     | 3 |   |   |   |
+   +---+---+---+   +   +   +---+
|   |           |   |   |       |
+   +   +---+   +   +   +---+   +
| o     |       |   |       |   |
+---+---+   +---+   +---+   +   +
|       |   |       |   |     2 |
+   +---+   +   +---+   +---+   +
|     o         |               |
+---+---+---+---+---+---+---+   +
```

FIGURE 1 – Fichier contenant la situation initiale

2 Objectif du projet

Nous vous demandons de produire un algorithme efficace pour résoudre ce problème. L'algorithme en question reçoit en entrée la carte du labyrinthe et l'emplacement de M. Pakkuman, des petits monstres et des bonbons éparpillés le long du labyrinthe.
L'algorithme doit produire :

— un fichier qui représente la situation initiale (voir Figure 1) ;
— la séquence des déplacements de M. Pakkuman ;

— un fichier qui représente la situation finale et la liste des déplacements de M. Pakkuman (voir Figure 3) ;

En particulier, si un des petits monstres est présent sur le plus court chemin, vous devez identifier pour M. Pakkuman l'itinéraire alternatif le plus court lui permettant soit d'éviter tous les petits monstres, soit de récolter suffisamment de bonbons pour les nourrir. Dans le cas où aucun chemin ne permet à M. Pakkuman de sortir du labyrinthe, nous vous demandons d'expliciter la cause pour laquelle il est empêché de sortir.

3 Détails techniques

```
Labyrinthe: 8 fois 8
+---+---+---+---+---+---+---+---+
|   |                           |
+   +---+   +---+---+---+   +   +
|       |   |           |   |   |
+---+   +   +   +---+   +   +   +
|   |   |   |       |   |   |   |
+   +   +   +---+   +   +   +   +
|       |       |   |   |   |   |
+   +---+---+---+   +   +   +---+
|   |           |   |   |       |
+   +   +---+   +   +   +---+   +
|       |       |   |       |   |
+---+---+   +---+   +---+   +   +
|       |   |       |   |       |
+   +---+   +   +---+   +---+   +
|               |               |
+---+---+---+---+---+---+---+   +
Elements du Labyrinthe:
        Monstres: 4
        Bonbons: 3
Emplacements:
         Pakkuman: (2,0)
         Monstres: (3,2) (6,7) (3,4) (0,5)
         Bonbons: (5,0) (0,4) (7,1)
```

FIGURE 2 – Format du fichier d'entrée

Le programma devrait être implémenté en JAVA en suivant au maximum le paradigme Orienté Objet (OO). De plus, il est conseillé de bien concevoir votre code afin de minimiser le code redondant en pensant aux responsabilités de chaque classe et de chaque méthode et à l'utilité de l'héritage et de la composition. *Un conseil : ne faites jamais de copier-coller*.
Vous pouvez utiliser les structures de données des libraires standard fournies par JAVA.

Le données décrivant le labyrinthe et les positions de tous les éléments présents dans le labyrinthe (M. Pakkuman, monstres, bonbons) seront transmises à votre programme via la lecture d'un seul fichier (dont le nom sera passé en argument de votre programme). Le fichier contiendra la description du labyrinthe, composée :

— de la dimension du labyrinthe $n \times m$;
— d'une représentation graphique du labyrinthe ;
— d'une section contenant les nombres des éléments du labyrinthe (nombre de monstres et de bonbons) ;

— d'une section contenant les emplacements des éléments du labyrinthe (position du Pakkuman, des monstres et des bonbons) ;

Un exemple de fichier est montré dans la Figure 2. Un générateur de labyrinthe vous sera fourni pour tester votre code sur plusieurs scénarios.

Effectuez le *parsing* dans une ou plusieurs méthodes (pas dans le main) et placez-les dans les classes qui vous paraissent les plus à même d'effectuer ce travail (pensez aussi aux méthodes statiques). Vous pouvez supposer que le fichier d'entrée ne comporte pas d'erreur.

Votre programme doit recevoir le nom du fichier en argument et écrire sur l'output standard la solution de la manière montrée en Figure 4. En Figure 5 et Figure 6 vous avez un exemple de fichier et output, respectivement, à produire dans le cas où il n'y a pas de sortie au labyrinthe.

Utilisez les exceptions dans vos algorithmes en cas de problèmes (et évitez les retours de booléens). Votre code source sera évidemment accompagné d'un `makefile` permettant de compiler facilement votre projet.

Le projet peut être réalisé individuellement ou par groupe de deux (ce que nous recommandons).

Tout cela sera naturellement présenté dans un *rapport* à la présentation soignée. Pour rappel, un rapport est un *texte suivi en français* correct qui *décrit* votre travail. Cela va sans dire, le rapport fait partie *intégrale* de votre travail et une partie importante de la note finale portera sur le rapport (y compris sa présentation).

```
Situation finale:
+---+---+---+---+---+---+---+---+
|   |             o   4         |
+   +---+   +---+---+---+   +   +
|       |   | #   #   # |   |   |
+---+   +   +   +---+   +   +   +
| P |   |   | #   # | # |   |   |
+   +   +   +---+   +   +   +   +
| #     | 1     | 3 | # |   |   |
+   +---+---+---+   +   +   +---+
| # | #   #   # | # | # |       |
+   +   +---+   +   +   +---+   +
| o   # | #   # | # | #   # |   |
+---+---+   +---+   +---+   +   +
|       | # | #   # |   | #   2 |
+   +---+   +   +---+   +---+   +
|     o   #   # |             # |
+---+---+---+---+---+---+---+   +
Trouvé un plus court chemin de longueur 33.
M. Pakkuman a pris 2 Bonbons!
Déplacements de M. Pakkuman: (2,0) (3,0) (4,0) (5,0) (5,1) (4,1) (4,2) (4,3) (5,3)
(5,2) (6,2) (7,2) (7,1) (7,2) (7,3) (6,3) (6,4) (5,4) (4,4) (3,4) (2,4) (2,3)
(1,3) (1,4) (1,5) (2,5) (3,5) (4,5) (5,5) (5,6) (6,6) (6,7) (7,7)
```

FIGURE 3 – Fichier contenant la situation finale

```
~>java  pakkuman labyrinthe.txt

Le labyrinthe a un dimension 8 fois 8
Il contient 4 monstres e 3 bonbons.
M. Pakkuman se trouve en position: (2,0)
Le monstres se trouvent en position: (3,2) (6,7) (3,4) (0,5)
Le bonbons se trouvent en position: (5,0) (0,4) (7,1)

Déplacements de M. Pakkuman:
1. (2,0) Départ
2. (3,0) sud
3. (4,0) sud
4. (5,0) sud, bonbon récolté
5. (5,1) est
6. (4,1) nord
7. (4,2) est
8. (4,3) est
9. (5,3) sud
10. (5,2) ouest
11. (6,2) sud
12. (7,2) sud
13. (7,1) ouest, bonbon récolté
14. (7,2) est
15. (7,3) est
16. (6,3) nord
17. (6,4) est
18. (5,4) nord
19. (4,4) nord
20. (3,4) nord, bonbon donné au petit monstre
21. (2,4) nord
22. (2,3) ouest
23. (1,3) nord
24. (1,4) est
25. (1,5) est
26. (2,5) sud
27. (3,5) sud
28. (4,5) sud
29. (5,5) est
30. (5,6) est
31. (6,6) sud
32. (6,7) est, bonbon donné au petit monstre
33. (7,7) Sortie!

Trouvé un plus court chemin de longueur 33.
M. Pakkuman a récolté 2 bonbon et rencontré 2 monstres.
```

FIGURE 4 – Exemple d'execution ayant comme entrée le fichier en Figure 2

```
Situation finale:
+---+---+---+---+---+---+---+---+
|               | 4   o         |
+---+---+---+   +---+---+   +   +
|           |   |       |   | o |
+   +---+   +   +   +   +---+   +
|   |       | 1 | 2 | o         |
+---+   +---+   +   +---+---+---+
|       |       | #   #   #   # |
+   +   +   +---+   +---+---+   +
|   |   |   |   | #   #   # | # |
+   +---+   +   +---+---+   +   +
|           |   |     P   # | 3 |
+   +---+---+   +   +---+---+   +
|           |           |       |
+---+---+   +---+---+---+   +   +
|                           |   |
+---+---+---+---+---+---+---+   +
Il n'y a pas moyen de sortir.
M. Pakkuman à pris 0 Bonbons!
Déplacements de M. Pakkuman: (5,5) (5,6) (4,6) (4,5) (4,3) (3,3) (3,4) (3,5) (3,6)
(3,7) (4,7)
```

FIGURE 5 – Fichier contenant la situation finale quand il n'y a pas de plus court chemin valide

```
~>java  pakkuman labyrinthe.txt

Le labyrinthe a un dimension 8 fois 8
Il contient, 4 monstres e 3 bonbons.
M. Pakkuman se trouve en position: (5,5)
Le monstres se trouvent en position: (2,3) (2,4) (5,7) (0,4)
Le bonbons se trouvent en position: (1,7) (0,5) (2,5)

Il n'y a pas moyen de sortir vive du labyrinthe
car le monstre 3 nous empêche de sortir.
```

FIGURE 6 – Exemple d'exécution quand il n'y a pas de plus court chemin valide

Consignes pour la remise du projet

À respecter scrupuleusement !

1. Les modalités pour la remise du rapport sont :
   — Date : **le 18 Décembre 2014** (vous pouvez ne pas remettre votre projet dans le cas de fin du monde).
   — Lieu : **Université Virtuelle (**`uv.ulb.a.c.be`**)** et
   au Secrétariat « étudiants » du Département d'Informatique, local 2N8.104.
   — Heure : **Avant 13h**.
   Après 13h, les projets seront considérés comme **en retard**, et vous perdrez **la moitié des points** sur votre note finale (plus un point par jour de retard).
2. Les modalité pour la remise du code sont :
   — Tous les fichiers dans **une seule archive** au format **ZIP** (ni tar.gz, ni rar, ni autre chose !).
   — L'archive doit porter **le nom** `AG2.NomEtudiant1-NomEtudiant2.zip`, si deux étudiants ont réalisé le projet ; `AG2.NomEtudiant.zip` dans le cas contraire.
   Si vous ne respectez pas ces critères, nous considèrerons que vous n'avez pas rendu de code. Si votre projet ne compile pas avec la commande `make`, on ne corrige pas votre projet.

Bon travail !