Mathématiques pour l'informatique **Licence 3 d'informatique**

Complément au TP 2 — Éléments d'entrées-sorties en Java (1)

Ce court document explique comment accéder à un fichier et à l'entrée standard dans le même programme, en utilisant des instances différenciées de la classe *Scanner*. Toute suggestion ou toute amélioration sont les bienvenues.

La classe *TestFile01* contient deux fonctions statiques (méthodes de classe), la fonction *usage* qui affiche un petit mode d'emploi du programme, et la fonction *main*, le programme principal. Le programme implémenté par la classe peut être exécuté avec un argument sur la ligne de commande, désignant un fichier, ou bien sans argument sur la ligne de commande. Voici les différentes étapes du fonctionnement du programme :

— le programme commence par lire la spécification d'un graphe, donné par sa matrice d'adjacence ; cette lecture a lieu, soit dans le fichier spécifié sur la ligne de commande, soit sur l'entrée standard s'il n'y a aucun argument sur la ligne de commande ;

— ceci effectué, le programme interroge répétitivement l'utilisateur en lui demandant un numéro de sommet, et il affiche alors la liste d'adjacence de ce sommet ; ce dialogue se fait en utilisant l'entrée standard du programme (qui sera généralement le clavier, eu égard à ce que l'on recherche).

On voit que dans le programme on utilise deux instances de la classe *Scanner*, l'une qui désigne le canal sur lequel on lit la description du graphe (qui peut être, soit le fichier désigné sur le ligne de commande, soit l'entrée standard du programme), et une seconde instance qui désigne l'entrée standard.

```
1:  package test ;
2:  import java.util.Scanner ;
3:  import java.io.File ;
4:  import java.io.FileNotFoundException ;
5:  import graph.GraphSimple ;
6:  import graph.GraphSimpleIO ;
7:  import graph.MiscellaneousInput ;
8:  import graph.MiscellaneousOutput ;
9:  public class TestFile01 {
10:   private static void Usage () {
11:     System.out.println ("Usage: ") ;
12:     System.out.println (" ** " + TestFile01.class.getName()) ;
13:     System.out.println ("    Everything read from stdin") ;
14:     System.out.println (" ** " + TestFile01.class.getName()
15:                         + " <filename>") ;
16:     System.out.println ("    Graph read from <filename>, "
17:                         + "questions asked on stdin") ;
18:   }
19:   public static void main (String[] arg) {
20:     Scanner stdin, file ;
21:     if (arg.length <= 1) {
22:       stdin = new Scanner (System.in) ;
23:       if (arg.length == 0) {
24:         file = stdin ;
25:       }
26:       else {
27:         try {
```

```
28:           file = new Scanner (new File (arg[0])) ;
29:         }
30:         catch (FileNotFoundException e) {
31:           System.err.println ("File \"" + arg[0] + "\" not found") ;
32:           return ;
33:         }
34:       }
35:     }
36:     else { // arg.length > 1
37:       Usage () ; return ;
38:     }
39:     int n = MiscellaneousInput.getInteger (file) ;
40:     GraphSimple graph = GraphSimpleIO.getGraphAmatrix (file, n) ;
41:     System.out.print("Give a vertex: ") ;
42:     int x = MiscellaneousInput.getInteger() ;
43:     while (x != 0) {
44:       MiscellaneousOutput.printlnIntegers (graph.getAdjacencyList (x)) ;
45:       System.out.print("Give a vertex: ") ;
46:       x = MiscellaneousInput.getInteger() ;
47:     }
48:   }
49: }
```

Voici une petite description des différentes instructions de la partie essentielle de la fonction *main* :

— on commence par lire la taille du graphe (sur le canal où doit être lu le graphe) ; la classe *MiscellaneousInput* fournit quelques routines de lecture (lecture d'entiers, lecture de suite d'entiers) :

```
39: int n = MiscellaneousInput.getInteger (file) ;
```

— puis on construit une instance de la classe *Graph* pour un graphe d'ordre n, que l'on construit à partir de la donnée de sa matrice d'adjacence :

```
40: GraphSimple graph = GraphSimpleIO.getGraphAmatrix (file, n) ;
```

— ensuite, on interroge répétitivement l'utilisateur pour un numéro de sommet (jusqu'à ce que l'utilisateur donne 0 pour numéro de sommet), et pour chaque sommet fourni, on affiche sa liste d'adjacence (la classe *MiscellaneousOutput* fournit quelques routines d'affichage liées à nos graphes) :

```
41: System.out.print("Give a vertex: ") ;
42: int x = MiscellaneousInput.getInteger() ;
43: while (x != 0) {
44:   MiscellaneousOutput.printlnIntegers (graph.getAdjacencyList (x)) ;
45:   System.out.print("Give a vertex: ") ;
46:   x = MiscellaneousInput.getInteger() ;
47: }
```

— on note, dans cette boucle, l'utilisation de la méthode *getAdjacencyList* qui s'applique à une instance de la classe *GraphSimple* :

```
44: MiscellaneousOutput.printlnIntegers (graph.getAdjacencyList (x)) ;
```