Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2016-2017**

TP - Séance n°3

Démineur

Si vous n'avez pas fait la partie facultative du dernier TD, commencez par lire *entièrement* ce sujet de TP. Avant de vous mettre à programmer, réfléchissez aux classes qu'il faut définir, et choisissez pour chaque méthode la classe dans laquelle vous allez l'écrire. Vous trouverez quelques suggestions à la fin du sujet. Évidemment, il y a plusieurs manières de faire, mais certaines solutions sont plus judicieuses que d'autres. Si vous avez un doute, parlez-en à votre enseignant.

1 Jeu du démineur

Le démineur est un jeu où le joueur doit trouver les mines d'un terrain miné sans les déclencher.

Au départ, le terrain est complêtement invisible. Á chaque tour, le joueur peut décider de découvrir une case. Si cette case est dépourvue de mine, la case est révélée, s'il y avait une mine, le jeu s'arrête, et on a perdu.

Une case révélée indique le nombre de mines dans les 8 cases adjacentes à la case révélée. S'il n'y a aucune mine dans les 8 cases adjacentes, alors on révèle aussi les 8 cases autour de la première case révélée, car il n'y a aucun risque à le faire. Si dans ces 8 cases il y a encore une case sans mines autour d'elle, alors on révèle aussi ses voisins etc etc ...

Le joueur a aussi la possibilité de poser un drapeau sur certaines cases pour se souvenir qu'il soupçonne qu'il y ait une mine sur cette case. Ce drapeau sert comme une sécurité : si le joueur demande à révéler une case avec un drapeau dessus, le jeu refuse de révéler la case. Le joueur doit d'abord enlever le drapeau.

Le jeu s'achève par une victoire lorsque toutes les cases dépourvues de mines sont révélées.

2 Implémentation

Nous vous proposons deux voies : réfléchir vous-même à ce que vous voulez faire et comment vous voulez le faire, ou suivre notre guide pas à pas qui vous donne des classes à impplémenter au fur et à mesure.

Vous pouvez aussi rester autonome tout en vous inspirant de ce que nous vous proposons.

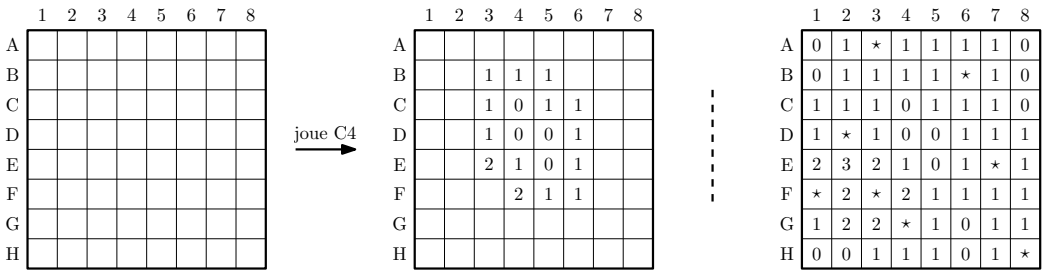

FIGURE 1 – À gauche, le joueur joue son premier coup en C4 et découvre toute une zone. À droite, le terrain complètement découvert

2.1 Les grandes lignes

On va implémenter quatres classes : `Lanceur`, `Jeu`, `Joueur` et `Plateau`.

`Lanceur` permettra de lancer le jeu en demandant à l'utilisateur quelques paramètres (comme la taille du plateau, le nombre de mines). Sa fonction `main` demandera en boucle à l'utilisateur s'il veut jouer, si oui quels paramètres il veut, et lancera alors le jeu. Il reposera les mêmes questions quand le jeu s'arrête. Quand le joueur dira non, le programme s'arrêtera.

`Joueur` contiendra une fonction qui demandera en ligne de commande quelle action le joueur veut effectuer, ainsi qu'éventuellement quelques éléments personnalisant le joueur (un nom, le nombre de parties gagnées et perdues ...).

`Plateau` contiendra des tableaux représentant l'état du jeu, ainsi que des fonctions pour agir sur le plateau, pour savoir si la condition de victoire est remplie, ou si on a perdu.

`Jeu` contiendra un `Joueur` et un `Plateau`, et fera interagir les deux entre eux.

2.2 Le plateau

On va maintenant implémenter le plateau. Nous vous proposons de partir du squelette suivant :

```
public class Plateau {
    public final int hauteur, largeur, nbMines;

    private final boolean[][] mines;
        /*indique où sont les mines sur le plateau*/
    private final int[][] etats;
        /*indique dans quel état est chaque case
          (cachée, révélée, avec/sans drapeau)*/
    private final int[][] adja;
        /*indique le nombre de mines adjacentes
          à chaque case*/
```

```
    public Plateau(int hauteur, int largeur, int nbMines){
        //A remplir
    }

    public boolean agir(int[] action){
        //A remplir
    }

    public boolean jeuFini(){
        //A remplir
    }

    public boolean jeuGagne(){
        //A remplir
    }

    public String affichage(){
        //A remplir
    }
}
```

Ces quelques fonctions publiques devrait suffir pour interagir avec un `Plateau`, mais libre à vous de rajouter des champs et des fonctions privés !

Pour l'affichage, nous vous proposons de faire un affichage textuel de la sorte, où `'?'` indique un drapeau et `'.'` une case cachée.

```
****************
*Mines/Drapeaux*
*  8  /  1     *
****************
  12345678
A ........
B ..111?..
C ..1011..
D ..1001..
E ..2101..
F ...211..
G ........
H ........
```

Utile : pour sauter des lignes dans une chaîne de caractère, on utilise le caractère spécial `'\n'`. (Peut être différent sous Windows, Mac, ou dans un éditeur particulier)

2.3 Le joueur

Nous vous proposons de créer la classe Joueur suivante, qui permet de recevoir des réponses de l'utilisateur au clavier. Les questions qu'on pourra poser seront de type 'oui ou non' (pour demander si on veut continuer à jouer, par exemple), 'quel nombre ?' (pour recevoir un nombre, par exemple le nombre de mines), et 'quelle action ?' (pour savoir quelle case révéler, ou munir/démunir d'un drapeau). On pourra également demander au joueur de s'identifier (donner son nom).

```
public class Joueur {
    private String nom;

    public Joueur(){
        //A remplir
    }

    public void setNom(){
        //A remplir
    }

    public String getNom(){
        //A remplir
    }

    public boolean ouiNon(){
        //A remplir
    }

    public int nombreChoisi(){
        //A remplir
    }

    public int[] actionChoisie(){
        //A remplir
    }
}
```

2.4 Le jeu

La classe `Jeu` sera assez simple : elle comprendra la fonction `jouer` qui fera interagir le joueur et le plateau jusqu'à la fin du jeu, et renverra un booléen pour indiquer si le jeu s'est terminée par victoire ou défaite. La fonction `jouer` s'assurera notamment que le joueur demande des coups valides, et redemandera au joueur de choisir une action s'il a demandé n'importe quoi.

```
public class Jeu {
```

```
    private Joueur leJoueur;
    private Plateau lePlateau;

    public Jeu(Joueur leJoueur, Plateau lePlateau){
        //A remplir
    }

    public boolean jouer(){
        //A remplir
    }
}
```

2.5 Le lanceur

Le lanceur contiendra un `main`, qui créera d'abord un joueur, puis lui demandera :

— Son nom
— S'il veut jouer
— Si oui, les paramètres du jeu

À la fin d'une partie, on demandera au joueur s'il veut rejouer, si oui, s'il veut changer de paramètres ou non, et alors on relancera une partie.

Quand le joueur ne veut plus jouer, on quitte le programme.

```
public class Lanceur {
    public static main (String[] args){
        //A remplir
    }
}
```

3 Des conseils d'implémentation

Si vous choisissez de représenter des actions ou les états des cases par des entiers, il peut être utile de définir des constantes entières de la façon suivante : `final int POSER_DRAPEAU = 2`. On pourra ainsi habilement utiliser un `switch case` comme ceci :

```
switch(typeAction) {
    case POSERDRAPEAU :
        //du code
    break;
    case REVELER :
        //du code
    break;
    //encore d'autres cas
    default :
```

```
        break;
    }
```

Note : On pourra alternativement utiliser des `enum`, si cela vous est familier.

Dans cet énoncé, nous vous proposons de représenter une action du joueur sur le plateau par un triplet d'entiers `{x,y,action}`. Il est également possible de définir une classe `Action` qui contiendrait trois champs (les 2 coordonnées, et l'action effectuée).

C'est souvent une bonne idée de séparer en plusieurs sous-fonctions une fonction complexe. Par exemple, la génération du plateau pourra se faire avec une sous-fonction posant aléatoirement des mines et une autre calculant le nombre de mines adjacentes à chaque case. La fonction agissant sur le plateau pourra appeler une sous-fonction ayant pour rôle de révéler une case du plateau, qui s'appellera elle même récursivement si nécessaire.

4 Pour aller plus loin ...

4.1 Affichage textuel enrichi

Pour améliorer votre affichage textuel, vous pouvez utiliser les caractères spéciaux ANSI (si vous exécutez votre programme depuis une console sous Linux). Ces caractères spéciaux peuvent changer la couleur du texte, effacer la console, et bien d'autres choses !

Pour effectuer une action spéciale, on commencera par afficher le caractère spécial `'\033'`, suivie d'un `'['` puis de paramètres et enfin de l'action qu'on veut effectuer. Par exemple :

- `System.out.print("\033[2J")` effacera la console
- `System.out.print("\033[;H")` positionnera le curseur en haut à gauche de la console.
- `System.out.print("\033[31m")` fera qu'on écrira en rouge ensuite
- `System.out.print("\033[m")` rétablira la couleur d'écriture par défaut

Cherchez 'ANSI escape code' sur internet pour découvrir tous les codes permettant d'améliorer votre affichage !

4.2 Affichage textuel adaptatif

Imaginez que vous voulez dessiner un démineur qui remplirait tout pile votre console ... comment pourrait-on faire ?

Une solution à cela est de d'abord appeler `'export LINES COLUMNS'` dans la console, puis d'utiliser `System.getenv("LINES")` et `System.getenv("COLUMNS")` dans votre votre programme, qui vous retourneront chacun le nombre de lignes et de colonnes de votre console sous forme d'une chaîne de caractères. Le résultat de ces appels change lorsque vous changez la taille de votre console.

4.3 Coopération

Si vous vous êtes contenté d'implémenter les méthodes publiques que nous vous avons conseillées, vous devriez pouvoir mélanger votre code avec celui de vos camarades. Á la fin du TP, lorsque votre implémentation et celle d'un de vos camarades fonctionnent, essayez de prendre vos classes `Lanceur`, `Joueur`, `Jeu`, mais d'utiliser la classe `Plateau` de votre camarade. Ou encore mieux : prenez chaque classe chez quelqu'un de différent !

L'un des buts de la programmation orientée objet est de permettre ce genre d'échange, car on peut ainsi se répartir le travail sur un gros projet. Si votre code et celui d'un de vos camarades ne sont pas compatibles, essayez de voir ce qui bloque et de corriger le problème.

4.4 Optimisation de la révélation des cases

Pour révéler les cases, on pourra procéder différemment qu'utiliser une fonction récursive. Une manière est par exemple d'utiliser une pile ou une file qui contient les cases sur lesquelles on doit passer, puis de la remplir et de la vider petit à petit. On pourra ainsi implémenter un parcours en largeur ou en profondeur (BFS et DFS en anglais). Pour de grands plateaux, cela peut beaucoup améliorer la vitesse et l'utilisation de mémoire du programme.