Programmation Orientée Objet

1 Introduction

1.1 Objectifs

- Mise en oeuvre de sous-programmes.
- Distinction entre boucles `for` et `while`.
- Mise en oeuvre d'un algorithme "online".
- Création de classes.
- Fonctionnement d'un automate, vu comme machine à état.

1.2 Organisation

Dans ce TP, vous allez créer trois nouveaux projets sous Netbeans :

- Calcul : trois sous-programmes calculant des moyennes de nombres.
- Interrupteurs : création d'une classe implémentant un automate.
- MaisonHantee : création d'une classe implémentant un automate.

2 Projet : Calcul "online" de la moyenne

Remarques :

- Un algorithme "online" est un algorithme qui traite les données au fur et à mesure qu'elles sont saisies.
- Ces calculs portent sur des notes, vous devrez donc vérifier que les valeurs saisies sont comprises en 0 et 20.
- Notez l'usage de la surcharge (overload en anglais) de la fonction moyenne dans ce projet.

Questions :

1. Créez un projet Netbeans que vous nommerez "Calculs".
2. Créez une classe `public class App` qui sera la classe principale de votre projet.
3. Dans cette classe, vous ajouterez la méthode `main` dans laquelle vous testerez les méthodes de classe de calcul.
4. Créez une seconde classe `public class Calculateur`, avec un constructeur ne contenant rien.
5. Ecrivez une procédure `public void moyenne()` qui demande demande à l'utilisateur 10 valeurs et qui affiche au fur et à mesure la moyenne des valeurs saisies précédemment. Si l'utilisateur donne des valeurs extravagantes, elles ne sont pas prises en compte dans le calcul, mais par contre elles sont comptabilisées dans les 10 valeurs.
6. Testez votre procédure. En particulier, vous testerez le rejet des notes supérieures à 20, le rejet des notes négatives, et bien sûr vous vous assurerez que les calculs sont justes.
7. Ré-investissez le code précédent pour écrire une fonction `public double moyenne(int n)` qui demande à l'utilisateur n valeurs et qui en calcule la moyenne. Cette fois, aucun affichage ne sera fait, et la moyenne finale sera retournée. De plus, il faudra que l'utilisateur saisisse effectivement n valeurs correctes.

8. Testez votre fonction sur le même modèle que précédemment.

9. Enfin, écrivez une dernière fonction `public double moyenneV3()` qui, à nouveau, calcule la moyenne des valeurs saisies par l'utilisateur. A nouveau, aucun affichage ne sera fait, seule la valeur finale sera retournée. Par contre, la valeur -1 (et uniquement cette valeur) servira à arrêter le calcul. Autrement dit, l'utilisateur saisit autant de notes qu'il le souhaite et lorsqu'il veut mettre fin au calcul, il saisit la valeur -1.

10. Testez également cette fonction.

3 Projet : Interrupteurs

3.1 Description

Le joueur dispose d'un panneau constitué d'une rangée de 5 lampes au-dessus d'une rangée de 5 interrupteurs. On conviendra qu'une lampe allumée est notée X et une lampe éteinte O.

X	X	O	O	O
1	2	3	4	5

La règle du jeu est la suivante :

- L'interrupteur 1 change l'état de la lumière 2
- L'interrupteur 2 change l'état des lumières 2 et 3
- L'interrupteur 3 change l'état des lumières 3 et 4
- L'interrupteur 4 change l'état des lumières 4 et 5
- L'interrupteur 5 change l'état des lumières 1 et 5

Le panneau démarre dans une configuration aléatoire et le joueur doit actionner les interrupteurs de manière à allumer toutes les lumières.

3.2 Questions

11. Créez un projet Netbeans que vous nommerez "Interrupteurs".

12. Créez une classe `public class App` qui sera la classe principale de votre projet.

13. Dans cette classe, vous ajouterez la méthode `main` dans laquelle vous testerez les méthodes de classe `Panneau`. Puis dans laquelle se déroulera la boucle principale du jeu.

14. Créez une seconde classe `public class Panneau`.

15. Dans cette classe `public class Panneau`, ajoutez 5 champs booléens `lumiere1`, ..., `lumiere5`.

16. Ecrivez un constructeur `public Panneau()` qui initialise l'état des lumieres de manière aléatoire (utlisez la méthode `Math.random()` dont vous chercherez le fonctionnement sur Internet).

17. Ecrivez la méthode `public String toString()` qui retourne une représentation du panneau sous forme d'une chaine de caractères. Lorsque les lumières 2 et 5 sont allumées et les autres éteintes, la chaine sera $[OXOOX]$.

18. Dans la méthode `main()` de votre classe `App`, testez votre constructeur et la méthode `toString()`.

19. Ecrivez 5 setters `public void interrupteur1()`, ..., `public void interrupteur5()` qui mettent en oeuvre les règles du jeu. Par exemple, le setter `public void interrupteur3()` change l'état des lumières 3 et 4.

20. Vérifiez le bon fonctionnement de vos 5 setteurs dans la méthode `main()`.

21. Ecrivez la méthode `public boolean allumees()` qui retourne `true` si et seulement si toutes les lumières sont allumées.

22. Dans la méthode `main()`, conservez vos tests mais transformez les en commentaires afin qu'il n'interfèrent pas dans la suite de l'exécution du programme.

23. Dans la méthode `main()`, mettez en oeuvre la boucle principale du jeu. Après avoir mis en route le jeu, le programme demande à l'utilisateur un numéro d'interrupteur à actionner, met à jour l'état du panneau et l'affiche. Vous pouvez utiliser la structure de contrôle `switch (...) {case ...}`. Le programme recommence jusqu'à ce que toutes les lampes soient allumées.

Voici un exemple de déroulement d'une partie :

```
Etat du panneau :  [OXXOO]
Interrupteur à actionner :  2
Etat du panneau :  [OOOOO]
Interrupteur à actionner :  5
Etat du panneau :  [XOOOX]
Interrupteur à actionner :  3
Etat du panneau :  [XOXXX]
Interrupteur à actionner :  1
Gagné !
```

4 Maison hantée

L'histoire de la maison hantée vous est racontée dans cette lettre que vous avez reçue d'un vieil ami :

> Je viens de faire l'acquisition d'une maison superbe, dans un cadre merveilleux. Malheureusement, elle est hantée par deux bruits d'outre-tombe, un rire sardonique et un chant grivois qui la rendent inhabitable. Lorsque l'un ou l'autre se fait entendre seul, c'est déjà difficile à supporter, mais vous imaginez l'état de mes nerfs lorsque les deux retentissent en même temps ! S'il n'y avait parfois le silence complet, je crois que je n'aurais pas pu tenir.
>
> Cependant, j'ai constaté que ces deux phénomènes obéissaient à des lois obscures mais immuables, liées à mes passions, que vous connaissez bien pour l'orgue et l'encens.
>
> Minuit est l'heure où ces bruits changent (éventuellement) d'état. Celui qui se ferait alors entendre durerait 24 longues heures. Celui qui se tairait nous laisserait en paix pour la même durée.
>
> Le chant conservera le même état (présent ou absent) sauf si, la veille, j'ai joué de l'orgue sans que le rire se fasse entendre, auquel cas, le chant prendrait l'état opposé.
>
> Quant au rire, si l'encens brûlait, il aura aujourd'hui le même état qu'avait hier le chant. Si l'encens ne brûlait pas, le rire fera alors le contraire de ce que faisait le chant hier.
>
> Vous m'avez convaincu, naguère, de l'infaillibilité de votre intelligence en me montrant comment traverser la rivière avec mon loup, ma chèvre et ma salade. Je suis sûr que, là encore,

vous saurez me faire parvenir la solution et me montrer comment obtenir le silence dont j'ai besoin, et comment le conserver.

Avec mon amitié et ma gratitude,

PS : Voilà près d'une semaine que j'entends à la fois le rire et le chant. Par pitié, faites vite.

24. En vous inspirant du projet "Interrupteurs" précédent, répondez à cette lettre.