Programmation Java
A. Bourmaud, J. Maitre, P. Rodriguez
Ch. Sempé, J-F. Viaud

Introduction à la programmation et à la modélisation en Java

TD°3 : Objets

On cherche à écrire un programme qui simule le déroulement d'un jeu.

1. Règles du jeu à simuler

Le jeu doit permettre au joueur de se déplacer du monde 0 au monde 4.

Dans cette version, l'unique joueur lance 2 dés. Le score obtenu grâce à ces dés détermine le déplacement du joueur d'un monde à un autre :

- Si le score est inférieur à 6, c'est la stagnation : le joueur reste dans le même monde
- Si le score appartient à [6 ; 8], le joueur progresse dans les mondes en empruntant le trajet long
- Si le score est supérieur à 8, le joueur progresse dans les mondes en empruntant le trajet court

Les connexions entre les mondes sont les suivantes :

- Le monde 0 est ouvert sur le monde 2 (trajet court) et sur le monde 1 (trajet long)
- Le monde 1 est ouvert sur le monde 3 (trajet court) et sur le monde 2 (trajet long)
- Le monde 2 est ouvert sur le monde 3 (trajet court) et sur le monde 1 (trajet long)
- Le monde 3 est ouvert sur le monde 4 (trajet court) et sur le monde 1 (trajet long)

La partie est terminée lorsque le joueur atteint le monde 4.

2. Modélisation

Dessinez un graphe représentant les différents mondes du jeu et les différents trajets entre ces mondes.

3. Test du modèle

Quel est l'état final si la succession des lancers de dés génère les scores 9, 4, 6, 5, 10, 7, 8, 10, 5, 7, 8 ? Cette série est-elle valide pour une partie gagnante ?

La série 5, 7, 8, 8, 12, 5, 10 est elle valide pour une partie gagnante ?

4. Programmation

A.Les classes

Le projet est organisé en 3 classes :

- La classe Joueur possédant un attribut privé entier, mondeCourant, un constructeur sans paramètre qui initialise le monde courant au monde initial

(0), un getter et un setter pour le monde courant, ainsi que deux autre méthodes : gagne et mondeSuivant détaillées ci-dessous.
- La classe Partie possédant un attribut privé Joueur, nommé jean, un constructeur qui initialise le joueur, une méthode privée unLancer ainsi que deux autres méthodes jouer et simuler. Les trois méthodes unLancer, jouer et simuler sont détaillées ci-dessous.
- Enfin, la classe jeu qui contiendra la méthode « main ».

B.Fonction de transition

Ecrivez la classe Joueur excepté : gagne et mondeSuivant.

Ecriver la méthode gagne qui retourne vrai si et seulement si le joueur est dans le monde final (4).

Le prochain monde du joueur dépend du monde dans lequel il se trouve et du score que les dés lui donnent.

Ecrivez la fonction monde*Suivant*, qui reçoit le monde actuel du joueur et le score au dés, et qui renvoie le prochain monde du joueur.
Vous utiliserez une structure *switch*.

```
switch(variable à évaluer)
{
        case valeur1 : instructions ;
break
        case valeur2 : instructions ;
break
        case valeur3 : instructions ;
break
         …
        default : instructions ;
}
```

La structure *switch*

C.Jeu complet

Vous disposez de la fonction *unLancer* dans la classe Partie, qui renvoie le score obtenu par le lancer d'un dé.

Ecrivez la classe Partie, exceptées les méthodes unLancer, jouer et simuler.

Ecrivez la procédure *jouer*, qui simule le déplacement d'un joueur entre les mondes.

L'affichage généré par la procédure *jouer* pourra prendre la forme ci-contre.

Dans la méthode main de la classe Jeu, appelez cette méthode

```
Score 9  : je passe dans le monde 1
Score 8  : je passe dans le monde 2
Score 4  : je passe dans le monde 2
          …
Je suis dans le monde 4, la partie est terminée  !
```

Transformez cette procédure en une fonction qui renvoie le nombre de coups joués avant d'atteindre le monde final.

D.Statistiques

Ecrivez la fonction *simuler de la classe Partie* qui joue *n* parties et renvoie le nombre moyen de coups permettant de gagner une partie.

Faites appel à cette méthode dans le main.