UFR ST - Besançon- L2 Info - Année 2014/15
Programmation Orientée Objets
TP 5 - Jeu de Dés

L'objectif de ce TP est en premier de fournir un objet « sécurisé » représentant un dé, à des programmeurs qui voudraient programmer un jeu de Dés. Vous aurez ensuite à utiliser cet objet pour programmer des jeux de dés.
Les exercices sont liés et devront être réalisés dans l'ordre.
N'oubliez pas de rédiger un programme principal qui teste vos classes au fur et à mesure que vous les écrivez.

Exercice 1. Une classe Hasard pour générer des entiers au hasard

Réalisez une classe nommée Hasard pour générer des nombres entiers au hasard, *sans qu'il soit nécessaire de l'instancier*. Par exemple,
- Hasard.entierRel() générera un entier relatif,
- Hasard.entier() générera un entier positif.

Vous devrez aussi surcharger ces méthodes pour qu'elles acceptent en paramètre une borne supérieure : par exemple, Hasard.entier(50) générera un entier entre 0 et 49 compris.

Vous devrez vous appuyer sur la classe Java Random, dont la documentation peut être consultée à l'URL : `http://java.sun.com/javase/6/docs/api/java/util/Random.html`

Exercice 2. Une classe modélisant un dé

Vous devez écrire un classe `De` représentant un dé. Un dé possède un certain nombre constant de faces, (en général 6, mais il faut prévoir des dés moins communs), et possède une valeur (celle qu'il affiche). Le dé peut être « roulé », auquel cas il change aléatoirement de valeur.

Un objet de classe `De` doit pouvoir être construit de deux manières : par défaut, avec un nombre de faces égal à 6, ou bien avec un nombre de faces fourni en paramètre.

On ne doit pouvoir modifier l'état du dé qu'en le roulant, pour éviter les tricheries. Un *getter* permettra de récupérer la valeur du dé. La méthode toString() représentera le dé en entourant sa valeur d'une paire de crochets. Par exemple, la chaîne "[6]" sera calculée pour un dé de valeur 6.

Exercice 3. Une classe modélisant un jeu de dés

Vous devez écrire une classe `JeuDe` qui doit permette de réaliser des opérations communes à tous les jeux de dés : lancer un certain nombre de dés, mettre de côté certains dés et relancer les autres, comparer deux jets de dés... Pour simplifier la réalisation, on fait l'hypothèse que tous les dés d'un jeu ont le même nombre de faces.

La classe doit aussi faciliter l'affichage des dés du jeu.

Propositions d'implantation

L'ensemble des dés participant au jeu est vu comme un tableau de dés nommé jet, c'est à dire comme un tableau d'objets de type De. Chaque dé du jeu possède donc un indice particulier, correspondant à sa place dans le tableau.

Un tableau de booléens nommé remiseEnJeu permet, pour chaque dé du jeu, de savoir s'il doit être relancé ou pas au prochain jet. Par exemple, la valeur `true` à l'indice 2 du tableau remiseEnJeu indique que le dé d'indice 2 dans le tableau jet sera relancé au prochain tour. La valeur `false` à l'indice 1 dans le tableau remiseEnJeu indique que le dé d'indice 1 dans le tableau jet ne sera pas relancé au prochain tour.

Le nombre de dés participant au jeu (3 pour le 421, 5 pour le Yam's, etc.) est enregistré dans un attribut constant nommé `NB_DES`.

Constructeurs

Par défaut, un jeu de dé comporte 3 dés à 6 faces.

On doit pouvoir préciser à la création d'un objet JeuDe qu'on souhaite un nombre différent de dés.

Il doit également être possible de créer un jeu de dés avec un nombre différent de dés, et avec des dés qui n'ont pas forcément 6 faces (mais qui ont cependant tous le même nombre de faces).

Enfin, quelque soit la constructeur, le tableau de booléens indiquant si les dés doivent être relancés ou pas est initialisé à `true` pour chaque dé du jeu.

Lancer et relancer les dés

Lors du premier jet de dés, tous les dés sont lancés car aucun n'est bloqué.

N.B. On lance un dé en le faisant *rouler*, par la méthode permettant de rouler un dé.

Quand on relance les dés, il ne faut relancer que ceux qui n'ont pas été bloqués. Une méthode nommée `bloquerDe()` permet de bloquer (dans le tableau `remiseEnJeu`) un dé dont l'indice est donné en paramètre.

Une méthode nommée `conserverDes()` permet de bloquer en un seul appel plusieurs dés. Elle bloque les dés désignés en paramètre, le paramètre étant un tableau d'entiers tel que chaque case de ce tableau donne l'indice d'un dé à bloquer. Ainsi, un appel à `conserverDes(new int[] {1,3})` bloque les dés d'indice 1 et 3.

Une méthode nommée `lancer()` lance (en les roulant) tous les dés, sauf ceux qui ont été bloqués.

toString()

La méthode toString() doit représenter les dés sous une forme qui ressemble à (pour un jeu de trois dés, aux valeurs 4, 2, et 1) :

```
"1: [4]     *2: [2]*     3: [1]"
```

Avant un deux-points est indiqué l'indice du dé dans le jeu, et après (entre crochets) sa valeur. Les dés bloqués sont entourés d'astérisques $*$. Dans l'exemple, le dé d'indice 2 est bloqué.

Comparer deux jets de dés

Doter la classe d'une méthode equals(...) qui compare le jeu à un jeu dont la référence est passée en paramètre.

ATTENTION. Les jeux où les valeurs sont identiques à une permutation près doivent être considérés comme égaux. Par exemple, $< 4, 2, 1 >$ et $< 1, 2, 4 >$ sont deux jeux égaux.

Surcharger ensuite la méthode equals(...) pour qu'elle accepte en paramètre de manière alternative un tableau d'entiers représentant des valeurs de dés.

Soit par exemple un jeu de 3 dés $jd1$ valant $< 1, 2, 4 >$. Alors un appel à

```
jd1.equals(new int[] {4, 2, 1});
```

doit renvoyer la valeur true.