Diplôme : Licence SPI 3ème année	2020-2021

UE : Ateliers de programmation

Ateliers de programmation, Programmation orientée objet

Type de document : Feuille support exercices

- Classe
- Constructeur
- Objet

Enseignants : Paul-Antoine BISGAMBIGLIA, Marie-Laure NIVET, Evelyne VITTORI

Exercice 1 : Création d'une classe permettant de représenter des Dés

On vous demande de créer une classe permettant de manipuler des Dés, c'est-à-dire de les créer et de jouer avec en les lançant. Chaque question donne lieu à un codage spécifique. Pour vous y retrouver commentez vos codes en fonction des questions.

Question 1. Squelette d'un Dé

Un Dé est caractérisé par un nombre de faces `nbFaces` qui est un entier compris entre 3 (nombre minimum de faces) et 120 (nombre maximum de faces sur un dé physique ?).

- On doit pouvoir connaitre son nombre de faces (getNbFaces) et éventuellement le changer (setNbFaces) si ce nombre est correct c'est-à-dire compris entre les bornes autorisées (3 et 120). Si le nombre n'est pas correct on affichera une erreur sur la sortie err (variable statique) de la classe System.

Question 2. Classe de Test

Créez une classe de test de vos codes, nommée TestDe.java dans laquelle vous allez créer une nouvelle instance d'un Dé à 6 faces. Ce code de test doit être écrit dans la méthode main :

```
public static void main(String[] args){}
```

Question 3. Constructeurs de Dé

Une instance de Dé peut être créée sans spécifier aucun paramètre. Dans ce cas, par défaut, le dé aura 6 Faces.
Une instance de Dé peut être créée en spécifiant un nombre de faces entier passé en paramètre, sous réserve qu'il soit compris dans les bornes (3 et 120).
Testez ces nouveaux constructeurs dans votre classe de test.

Question 4. Ajout d'un champs nom au dé

Ce nom est une chaîne de caractères (String). Il doit être modifiable, mais ne doit jamais être une chaîne vide, ni nulle. On doit pouvoir accéder à sa valeur mais sans pouvoir la modifier.

Question 5. Lancé d'un dé

On doit pouvoir lancer un Dé via la méthode lancer(), ce qui a pour effet de retourner la valeur du lancer c'est-à-dire un entier. Pour simuler ce lancé nous allons utiliser un générateur de nombre aléatoire. Ci-dessous le code à ajouter à votre classe.

```
//En tête de fichier
import java.util.*;
…
//Dans les attributs de classe
private static Random r = new Random();

//Pour générer un nombre aléatoire entre 0 et nb (exclu)
int nbAleatoire= r.nextInt(nb) ;
```

Testez le lancer de dés dans votre classe de test.

Question 6. Contraintes sur le nom, nombres de dés créés

Si aucun nom n'est spécifié à la construction du Dé alors le Dé s'appellera automatiquement Dé « n°i » ou i est le n° de création du Dé. Si le nom est donné en paramètre du constructeur (en plus du nombre de faces), qu'il est non vide et non null alors le nom du dé sera celui donné.
Il nous faut donc savoir à tout moment combien de Dés différents ont été créés jusqu'à présent. Créez la variable permettant à la classe de connaitre ce nombre et à l'utilisateur de le lui demander. Personne ne devra pouvoir modifier ce nombre directement.

Question 7. Nb Lancés successifs, meilleur lancer gardé

On doit pouvoir lancer le dé un nombre nb donné de fois via la méthode lancer et ne garder que le meilleur des lancés qui sera alors retourné.
Écrivez cette nouvelle version de la méthode lancer (surchage) et testez la ensuite dans votre classe de test.

Question 8. Les méthodes à redéfinir héritées de la classe Objetc.

Test de toString et de equals.

Question 9. Héritage Dés pipés

On souhaite maintenant modéliser le comportement d'un dé pipé. C'est un dé qui retourne toujours une valeur supérieure à une valeur donnée. Cette valeur borne minimale doit être spécifiée à la construction du dé (en plus de son nombre de faces et de son nom) et ne dois plus pouvoir être modifiée par la suite. On veillera bien sûr à ce que cette borne ait une valeur correcte.
Proposez une modélisation pour y arriver et testez votre code dans votre classe de test.

Question 10. Héritage Dés à effet mémoire

Modélisation des Dés à effet mémoire, c'est un dé avec lequel on est toujours sûr de ne jamais retomber deux fois de suite sur la même valeur. Proposez une modélisation pour y arriver et testez votre code dans votre classe de test.

Question 11. Héritage Dés à faces autres

On souhaite maintenant pouvoir gérer des dés dont les faces ne sont pas nécessairement des chiffres entiers qui se suivent de façon continue de 1 à nbFaces, mais pourquoi pas des chaines de caractères, par exemple "Gagné", "Perdu", "Relancez", "Passez votre tour" ; ou bien des suites d'entiers pairs, ou impairs ou suivant une autre logique.
Comment pourrait-on procéder ?

Exercice 2 : Création d'une classe permettant de représenter des entiers

Question 1 : On vous demande de créer une classe permettant de manipuler des entiers.

- Un Entier est défini par un champ valeur de type int, compris entre deux bornes spécifiées à la construction de l'Entier, non modifiables mais librement consultables.
- La valeur de l'Entier est soit spécifiée à la construction de l'objet, soit donnée après la construction via une méthode de type set.
    - Si rien n'est spécifié à la construction, la valeur sera temporairement mise à 0. Si la valeur donnée via la méthode set n'est pas être comprise dans les bornes spécifiées à la construction, la valeur interne ne sera pas modifiée.
- On souhaite pouvoir demander à l'Entier d'incrémenter sa valeur interne. On vous demande pour cela de définir une méthode incrémente, qui incrémente la valeur interne de l'Entier de 1 tout en vous assurant toujours qu'elle reste dans les bornes. Si ce n'est pas le cas, la valeur interne ne sera pas modifiée.
- On souhaite également pouvoir demander l'incrémentation par pas de n, n étant un entier passé en paramètre. Surchargez la méthode incrémente précédemment définie.

Question 2 : On hérite d'Object

Redéfinissez les méthodes nécessaires héritées de la classe Object, à savoir, toString et equals.

Question 3 : Héritage

On vous demande maintenant de définir le comportement d'EntierFou. Ils se comportent exactement de la même façon que les Entiers sauf qu'ils possèdent en plus du champs valeur, un champ `niveauDeFolie` de type entier également. Et que lorsqu'on leur demande de s'incrémenter ils augmentent leur valeur d'un nombre aléatoire[1] compris entre 0 et leur niveau de folie.

- [1] Pour manipuler et générer des nombres aléatoires allez voir du côté de la classe `java.lang.Math` et de sa méthode `random`