TD 2

# Travaux Dirigés - Interfaces et Programmation Fonctionnelle

## Partie 1 : Interfaces

:hand: **Note** La première partie du TD est issu de ce site : [https://www.fil.univ-lille.fr/~routier/enseignement/licence/poo/tdtp/td-interfaces.pdf](https://https://www.fil.univ-lille.fr/~routier/enseignement/licence/poo/tdtp/td-interfaces.pdf)

### Exercice 1 :

On s’intéresse à un zoo qui peut accueillir des animaux. Certains de ces animaux peuvent être carnivores (carnivorous) d’autres non. On considère l’interface suivante qui permet de définir le type des animaux :

package zoo;  
   
public interface Animal {  
 /\*\* @return the number of legs of this animal \*/  
 public int getNbLegs();  
 /\*\* @return true if and only if this animal is carnivorous \*/  
 public boolean isCarnivorous();  
}

Le type Zoo dispose ainsi d’une méthode pour permettre l’ajout d’un animal au zoo :

public void addAnimal(Animal animal)

Un éléphant est un animal à 4 pattes (legs), non carnivore. Un éléphant est en plus caractérisé par un poids, en nombre entier de kilogrammes, qui est fourni à la construction. La classe Elephant permet de représenter de tels objets.

Un anaconda est un animal sans patte et carnivore. La classe Anaconda permet de représenter de tels objets.

### Q 1.

Sachant qu’il doit être possible de compiler et d’exécuter sans erreur les lignes de code suivantes :

Zoo aZoo = new Zoo( );  
aZoo.addAnimal(new Elephant (1500));  
aZoo.addAnimal(new Anaconda ( ) )

#### Q 1.1.

Donnez le code minimal pour une classe Elephant du paquetage zoo définie comme décrit ci-dessus. (Le code de la classe Anaconda n’est pas demandé.)

#### Q 1.2.

Est-il nécessaire de compléter la classe Zoo pour pouvoir compiler et exécuter sans erreur les lignes de code précédentes ? - Si oui, indiquez clairement pourquoi et donnez la signature des méthodes à ajouter (uniquement la signature). - Si non, indiquez clairement pourquoi ce n’est pas nécessaire.

#### Q 2.

On considère la déclaration suivante :

T ref = new Elephant(1000);

Quelles sont toutes les valeurs de type possibles pour T ?

### Exercice 2 :

On s’intéresse à des « transformations » de texte. Une opération de transformation consiste, à partir d’une chaîne de caractères initiale, à produire une nouvelle chaîne de caractères, résultat de la transformation de la chaîne initiale. Pour représenter ces transformations on fournit l’interface suivante :

package text;  
   
public interface Transformation {  
 /\*\*  
 \* Apply a transformation to the input String and return the resulting transformed String  
 \* @param input the String to transform  
 \* @return the transformed String  
 \*/  
 public String transform(String input);  
   
 /\*\*  
 \* Return a description of what the transformation does  
 \* @return a description of what the transformation does  
 \*/  
 public String getDescription();  
}

### Q 1.

On souhaite disposer d’une transformation qui remplace tous les caractères majuscules de la chaîne initiale par leur minuscule et laisse les autres caractères inchangés. Donnez le code java d’une classe ToLowerCaseTransformation qui permet de modéliser de telles transformations. NB : il existe une méthode toLowerCase() dans la classe String.

### Q 2.

Dans l’expression :

T ref = new ToLowerCaseTransformation();

Quelles sont toutes les valeurs possibles pour le type T telles que ce code puisse compiler ?

Pour la suite on suppose que d’autres classes permettant de modéliser d’autres transformations ont été définies de manière similaire. On a par exemple des transformations qui passent les caractères en majuscules ToUpperCaseTransformation ou qui appliquent un codage Rot-13 à la chaîne fournie Rot13Transformation, etc.

### Q 3.

On appelle MultiStringTransformer une classe qui dispose d’une méthode multiTransform qui prend en paramètres : - un tableau de chaînes de caractères - une transformation (au sens décrit ci-dessus) et dont le résultat est le tableau des chaînes de caractères transformées par la transformation fournie.

Donnez le code java de la méthode multiTransform.

### Q 3.1.

Donnez le code java de la classe MultiStringTransformer. Utilisez cette fois-ci la programmation fonctionnelle pour définir la méthode multiTransform. (Streams, lambda, etc.)

### Q 4.

On donne la variable suivante :

String[] data = new String[] { "Timoleon", "jaVa", "JRR Tolkien" };

Donnez une séquence de lignes de code qui : 1. crée un objet MultiStringTransformer, 2. définit une variable lowers avec pour valeur le tableau des chaînes de caractères issues de data transformées en minuscules, 3. définit une variable rot13s avec pour valeur le tableau des chaînes de caractères issues de data codées en Rot-13.

## Partie 2 : Programmation Fonctionnelle

:no\_entry\_sign: **Caution** Pour cette partie, vous ne devez utiliser que les classes de la bibliothèque stream de Java. Vous ne devez pas utiliser de boucles for ou while.

### Exercice 1:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant la longueur de chaque chaîne.

**Exemple d’entrée :** ["Java", "Python", "JavaScript", "C", "Ruby"] **Sortie attendue :** [4, 6, 10, 1, 4]

### Exercice 2:

Écrivez une méthode qui prend une liste d’entiers en entrée et renvoie une liste contenant le carré de chaque entier.

**Exemple d’entrée :** [1, 2, 3, 4, 5] **Sortie attendue :** [1, 4, 9, 16, 25]

### Exercice 3:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant uniquement les chaînes dont la longueur est supérieure à 5 caractères.

**Exemple d’entrée :** ["Apple", "Banana", "Strawberry", "Orange", "Kiwi"] **Sortie attendue :** ["Strawberry", "Banana", "Orange"]

### Exercice 4:

Écrivez une méthode qui prend une liste d’entiers en entrée et renvoie la somme de tous les entiers.

**Exemple d’entrée :** [1, 2, 3, 4, 5] **Sortie attendue :** 15

### Exercice 5:

Écrivez une méthode qui prend une liste d’entiers en entrée et renvoie la moyenne de tous les entiers.

**Exemple d’entrée :** [10, 20, 30, 40, 50] **Sortie attendue :** 30.0

### Exercice 6:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une seule chaîne contenant toutes les chaînes concaténées, séparées par des virgules.

**Exemple d’entrée :** ["Hello", "World", "Java", "Programming", "Language"] **Sortie attendue :** "Hello, World, Java, Programming, Language"

### Exercice 7:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant toutes les chaînes en majuscules.

**Exemple d’entrée :** ["apple", "banana", "orange", "kiwi", "strawberry"] **Sortie attendue :** ["APPLE", "BANANA", "ORANGE", "KIWI", "STRAWBERRY"]

### Exercice 8:

Écrivez une méthode qui prend une liste d’entiers en entrée et renvoie une liste contenant uniquement les nombres pairs, triés par ordre croissant.

**Exemple d’entrée :** [3, 8, 2, 5, 10, 7] **Sortie attendue :** [2, 8, 10]

## Partie 3 : Questions de cours

### Exercice :

Considérons une hiérarchie d’interfaces pour différents types de formes géométriques. Nous avons une interface de base Forme, et deux interfaces spécialisées Forme2D et Forme3D qui héritent de Forme. Déclarez ces interfaces et fournissez un exemple concret de classe qui les implémente.

#### Exemple de classe :

interface Forme {  
 double calculerSurface();  
}  
   
interface Forme2D extends Forme {  
 double calculerPerimetre();  
}  
   
interface Forme3D extends Forme {  
 double calculerVolume();  
}  
   
class Rectangle implements Forme2D {  
 private double longueur;  
 private double largeur;  
   
 public Rectangle(double longueur, double largeur) {  
 this.longueur = longueur;  
 this.largeur = largeur;  
 }  
   
 @Override  
 public double calculerSurface() {  
 return longueur \* largeur;  
 }  
   
 @Override  
 public double calculerPerimetre() {  
 return 2 \* (longueur + largeur);  
 }  
}

### Questions :

1. Quelle est la différence entre une classe abstraite et une interface en Java ?
2. Qu’est-ce qu’une interface ?
3. Quel est l’avantage d’utiliser une interface par rapport à une classe abstraite ?
4. Comment une interface peut-elle être utilisée pour définir un contrat ou un comportement commun pour plusieurs classes ?
5. Dans notre exemple, pourquoi avons-nous déclaré Forme2D et Forme3D comme des interfaces plutôt que des classes abstraites ?
6. Peut-on implémenter plusieurs interfaces dans une seule classe ? Si oui, comment cela s’appelle et quelles sont les restrictions associées ?
7. Dans notre exemple, que signifie implements Forme2D pour la classe Rectangle ?
8. Quelles méthodes la classe Rectangle doit-elle implémenter en raison de son implémentation de l’interface Forme2D ?

Ces questions permettront de vérifier la compréhension des étudiants sur les concepts d’interface en Java et leur application dans un exemple concret.