TD 2

Exercice 1 : Interfaces fonctionnelles

Question 1 : écrire une interface fonctionnelle `Somme` et utiliser cette dernière pour calculer la somme de deux entiers, de deux doubles, de deux longs et de deux chaînes de caractères (vous définirez 4 lambdas qui implantent `Somme`). Comparer `Somme` et `Sommable` vu au TD1. Discuter.

Question 2 : écrire une interface fonctionnelle `ToString` générique sur un type `T` et permettant de convertir un `T` en `String`. Créer deux lambdas implémentations de `ToString`, une pour les listes de `String` (`l2s`) et une pour les map `String → Integer` (`m2s`) dont l'effet est respectivement de créér une chaîne de la forme `"e1, e2, ⋯"` et de la forme `"k1: v1, k2: v2, ⋯"`.

Question 3 : pour chaque modèle dans {`Function`, `Predicate`, `Consumer`, `Supplier`}, donner une description de l'utilisation, dire si le modèle a des arguments et s'il retourne une valeur.

Exercice 2 : Prédicats

Question 1 : la direction d'une fête foraine souhaite vérifier automatiquement l'accès à ses manèges. Elle dispose de capteurs pour mesurer la taille des clients (`taille`, entier, en cm) et leur poids (`poids`, double, en kg). Définir les prédicats suivants :

- taille trop petite (`taille < 100`)
- taille trop grande (`taille > 200`)
- taille incorrecte (par composition)
- taille correcte (par composition)
- poids trop lourd (`poids > 150.0`)
- poids correct (par composition)
- accès autorisé (par composition)

Pour représenter les clients vous pourrez utiliser une `Paire`.

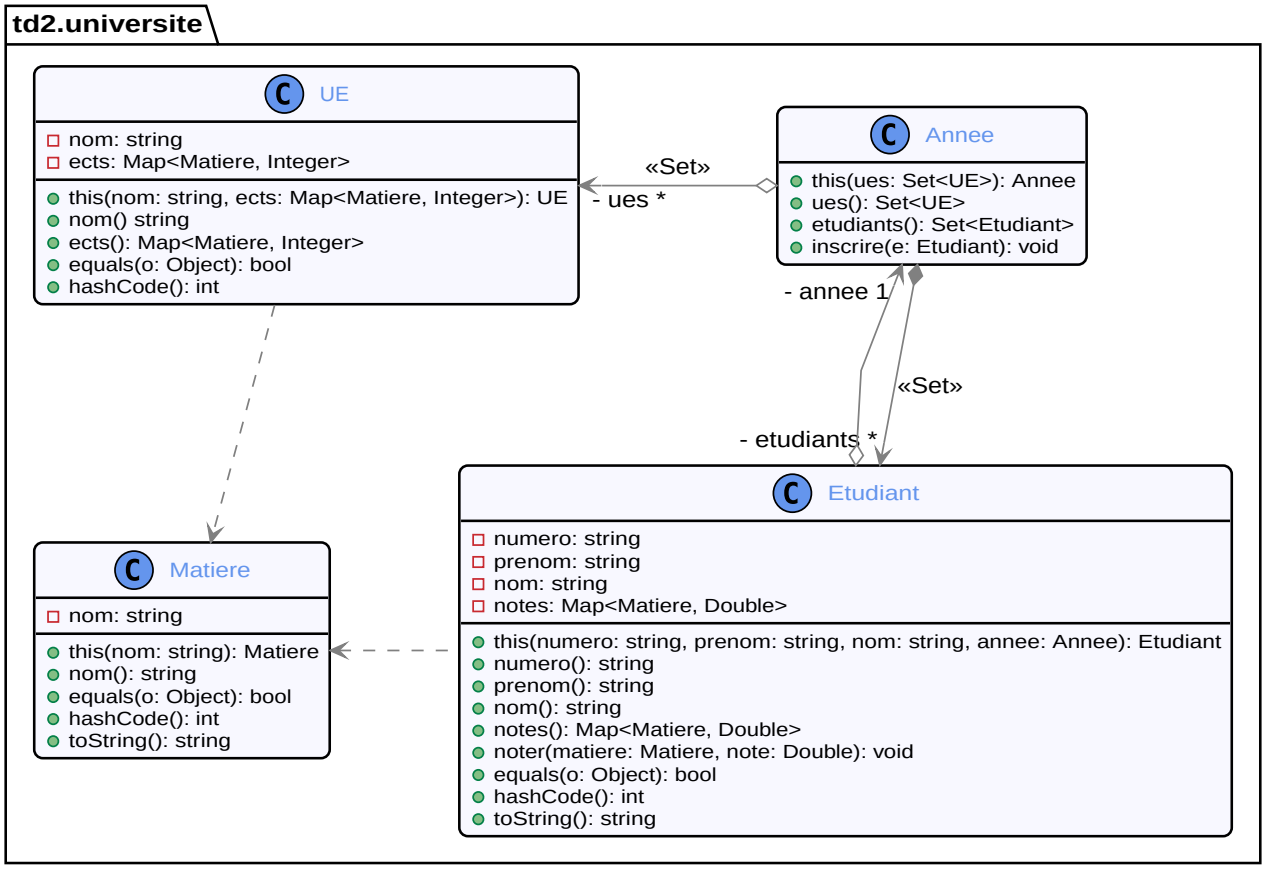
```
public class Paire<T,U> {
    public T fst;
    public U snd;
    public Paire(T fst, U snd) {
        this.fst = fst;
        this.snd = snd;
    }
    @Override public String toString() {
        return String.format("(%s,%s)",fst.toString(),snd.toString());
    }
}
```

Testez l'accès avec le bon nombre de tests.

Question 2 : on désire généraliser cela au travers d'une méthode `filtragePredicatif` qui prend une liste de prédicats sur un type `T`, une liste d'éléments de type `T`, et qui renvoie la liste des éléments qui vérifient la conjonction des prédicats. Utiliser cette méthode sur le cas de la fête foraine.

Exercice 3 : Etude de cas

Dans cet exercice, utiliser les classes du paquetage `td2.universite` qui vous sera fourni.



Les données à utiliser sont :

```
Matiere m1 = new Matiere("MAT1");
Matiere m2 = new Matiere("MAT2");
UE ue1 = new UE("UE1", Map.of(m1, 2, m2, 2));
Matiere m3 = new Matiere("MAT3");
UE ue2 = new UE("UE2", Map.of(m3, 1));
Annee a1 = new Annee(Set.of(ue1, ue2));
Etudiant e1 = new Etudiant("39001", "Alice", "Merveille", a1);
e1.noter(m1, 12.0);
e1.noter(m2, 14.0);
e1.noter(m3, 10.0);
System.out.println(e1);
Etudiant e2 = new Etudiant("39002", "Bob", "Eponge", a1);
e2.noter(m1, 14.0);
e2.noter(m3, 14.0);
Etudiant e3 = new Etudiant("39003", "Charles", "Chaplin", a1);
e3.noter(m1, 18.0);
e3.noter(m2, 5.0);
e3.noter(m3, 14.0);
```

Question 1 : écrire une fonction `afficheSi` qui prend en paramètre une chaîne de caractère en-tête, un prédicat portant sur un étudiant et une année et qui affiche l'en-tête suivi de tous les étudiants pour lesquels le prédicat est vrai. Utiliser dans un premier temps une boucle `for` puis dans un second temps un `forEach` avec un consommateur. Illustrez en utilisant cette fonction pour afficher tous les étudiants.

```
**TOUS LES ETUDIANTS

39001 Alice Merveille
UE2
MAT3 (1) : 10.0
UE1
MAT1 (2) : 12.0
MAT2 (2) : 14.0

39003 Charles Chaplin
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 18.0
MAT2 (2) : 5.0

39002 Bob Eponge
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 14.0
MAT2 (2) : DEF
```

Question 2 : écrire un prédicat `aDEF` qui permet de savoir si un étudiant est DEFaillant (pas de notre pour une matière ou +). Utiliser `afficheSi` pour afficher les étudiants concernés.

```
39002 Bob Eponge
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 14.0
MAT2 (2) : DEF
```

***Question 3 :** écrire un prédicat `aNoteEliminatoire` qui permet de savoir si un étudiant a une note éliminatoire (sous un plancher de 6/20). Utiliser `afficheSi` pour afficher les étudiants concernés.

```
**ETUDIANTS AVEC NOTE ELIMINATOIRE

39003 Charles Chaplin
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 18.0
MAT2 (2) : 5.0
```

Question 4 : écrire une fonction `moyenne` qui calcule la moyenne d'un étudiant. La règle est la suivante :

$$\text{moyenne}(e) = \frac{\sum_{u \in \text{ues}(\text{annee}(e))} \sum_{(m,\ k) \in \text{ects}(u)} \text{note}(e,\ m) \times k}{\sum_{u \in \text{ues}(\text{annee}(e))} \sum_{(m,\ k) \in \text{ects}(u)} k}$$

On ne peut pas calculer de moyenne si l'étudiant est défaillant. Utiliser `aDEF` et retourner `null` dans ce cas.

Question 5 : définir un prédicat `naPasLaMoyennev1` qui permet de savoir si un étudiant n'a pas la moyenne. Se contenter de la comparer à 10. Utiliser `afficheSi` pour afficher les étudiants concernés. Que se passe-t-il quand on utilise ce prédicat sur un étudiant défaillant ?

Question 6 : définir une seconde version de ce prédicat, `naPasLaMoyennev2` qui prennent en compte le cas des étudiants défaillants. Utiliser `afficheSi` pour afficher les étudiants n'ayant pas la moyenne.

```
**ETUDIANTS SOUS LA MOYENNE (v2)

39002 Bob Eponge
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 14.0
MAT2 (2) : DEF
```

Question 7 : définir un prédicat composé `session2v1` à partir des prédicats précédents permettant de savoir si un étudiant va en session 2. Un étudiant va en session 2 s'il n'a pas la moyenne (utiliser `naPasLaMoyennev1` pas `naPasLaMoyennev2`), s'il a une note éliminatoire, ou s'il est défaillant. Qu'observe-t-on pour différents ordres dans la disjonction logique des prédicats ?

```
**ETUDIANTS EN SESSION 2 (v2)

39003 Charles Chaplin
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 18.0
MAT2 (2) : 5.0

39002 Bob Eponge
UE2
MAT3 (1) : 14.0
UE1
MAT1 (2) : 14.0
MAT2 (2) : DEF
```

Question 8 : écrire une fonction `afficheSiv2` qui améliore `afficheSi` en permettant de passer en plus une fonction de représentation d'étudiant qui est utilisée par `afficheSiv2` pour afficher chaque étudiant. Utiliser cette nouvelle fonction `afficheSiv2` pour arriver au même résultat qu'`afficheSi` (utiliser la référence à la méthode qui permet d'afficher un étudiant). Utiliser ensuite à nouveau `afficheSiv2` pour afficher l'ensemble des étudiants avec leur moyenne (définir une fonction ad-hoc anonyme qui pour un étudiant donne son prénom, nom et moyenne, et la passer à `afficheSiv2`).

```
**TOUS LES ETUDIANTS

Alice Merveille : 12,40
Charles Chaplin : 12,00
Bob Eponge : défaillant
```

Question 9 : écrire une fonction `moyenneIndicative` où les notes non indiquées (DEF) sont traitées comme des 0/20. Utiliser cette fonction avec `afficheSiv2`.

```
Alice Merveille : 12,40
Charles Chaplin : 12,00
Bob Eponge : 8,40
```

Question 10 : généraliser `naPasLaMoyennev2` en une fonction `naPasLaMoyenneGeneralise` qui permet de choisir la fonction de moyenne à utiliser. Utiliser cette nouvelle fonction avec `afficheSiv2`.

```
**TOUS LES ETUDIANTS SOUS LA MOYENNE INDICATIVE

Bob Eponge : 8,40
```

En remplaçant les deux 14/20 de `Bob` par des 20/20, il a plus que la moyenne et il n'apparaît plus.

```
**TOUS LES ETUDIANTS SOUS LA MOYENNE INDICATIVE
(rien)
```