TD9 : Structures simples

Le but de ce TD est de créer et manipuler les **structures**. Les structures permettent de définir de nouveaux types, qu'on appelle *type composé* car ils sont définis à partir de types plus simples (ex `String`, `int`, `double[]` etc). Elles permettent de rassembler des données ayant des types différents contrairement à un tableau dans lequel tous les éléments ont le même type.

Attention : pour bien faire ce TD, vous *devez* avoir regardé et travaillé le cours magistral CM9. Les paragraphes suivants donnent un rappel de cours, avec des petits morceaux de code Java qui devraient vous aider pour le TD.

Rappels de cours

Champ d'une structure On appelle les différentes données d'une structure des **champs**, et le moyen de lire la valeur d'un champ s'effectue avec la notation `objet.champ` (un objet instance de cette structure, un point, suivi du nom du champ). C'est la notation que vous avez déjà beaucoup utilisé, par exemple pour les tableaux : la longueur d'un `int[] tab` est dans un champ appelé `length`, auquel on accède avec `tab.length`. Par défaut, un champ est modifiable, par la notation `objet.champ = nouvelleValeur;` (ce n'est pas le cas de `tab.length` mais c'est une exception).

Par exemple, la structure `Etudiant` définie dans le CM9 s'écrit comme cela :

```
1 public class Etudiant {
2   // une liste de lignes de la forme : public type nomDuChamp;
3   public String nom;
4   public String prenom;
5   public int note;
6 }
```

Une fois définie, cette structure s'utilise comme cela, avec la construction `Structure nom = new Structure();` déjà vue pour les chaînes `String`, les tableaux `int[]` (ou autres) ou les `Scanner`. Il faut ensuite remplir les champs de la structure, un à un, pour s'en servir :

```
1 Etudiant xi_zing = new Etudiant(); // un nouvel objet, instance de cette structure
2 xi_zing.nom = "Zing"; // ce champ est modifiable
3 xi_zing.prenom = "Xi";
4 xi_zing.note = 15; // les autres aussi
```

Quand on renvoie une structure dans une fonction, on renvoie une référence (comme pour les tableaux, cf Examen 1 exercice 2.1). Par exemple si on veut écrire une fonction qui augmente d'un point la note d'un ou d'une étudiant-e (instance de la structure `Etudiant`), on peut définir cette fonction comme ceci :

```java
public static Etudiant augmenteNoteDeUnPoint(Etudiant etu) {
  if (etu.note <= 19)   etu.note = etu.note + 1;
  // renvoie une RÉFÉRENCE de etu, et PAS une copie !
  return etu;
}
```

Mais ce n'est vraiment pas une bonne idée d'écrire une telle fonction, qui fait croire qu'elle renvoie un nouvel objet instance de `Etudiant` alors qu'elle modifie l'objet donné en entrée ! C'est une très mauvaise pratique, il faut écrire une fonction qui modifie l'objet donné en argument, sans en renvoyer une référence :

```java
public static void augmenteNoteDeUnPoint2(Etudiant etu) {
  if (etu.note <= 19)  etu.note = etu.note + 1;
}
```

Cette fonction modifie l'objet `etu` passé en argument, et par exemple dans le `main` de votre programme vous pouvez faire ceci :

```java
System.out.println("Pour l'eleve \"" + marie_dupont.prenom + " " + marie_dupont.
    nom + "\"");
System.out.println("Avant le bonus, sa note etait " + marie_dupont.note);
// augmente sa note en modifiant directement l'objet, et renvoie une référence, qu'on utilise
    pour écraser marie_ dupont
marie_dupont = augmenteNoteDeUnPoint(marie_dupont);
System.out.println("Apres le bonus, sa note est " + marie_dupont.note);
// augmente sa note en modifiant directement l'objet
augmenteNoteDeUnPoint2(marie_dupont);
System.out.println("Apres le deuxieme bonus, sa note est " + marie_dupont.note);
```

et cela va afficher

```
Pour l'eleve "Marie Dupont"
Avant le bonus, sa note etait de 15
Apres le bonus, sa note est de 16
Après le deuxième bonus, sa note est de 17
```

Objets plus compliqués : l'an prochain L'année prochaine, vous verrez les objets qui ressemblent aux structures en ajoutant des fonctions pour manipuler les membres de la structure (des méthodes). Vous en avez déjà manipulées dans le cours CM7, le TD8 et le TP8 sur le type `String`, où vous avez utilisé la notation `objet.fonction(arguments)` (un objet instance de cette classe, un point, suivi d'une fonction et éventuellement ses arguments). Par exemple, `machaine.length()` donne la longueur d'une chaîne `String machaine`. Nous n'irons pas jusque là dans ce TD.

Exercices du TD

Dans ce TD, vous allez définir différents types (des structures), afin de gérer une petite application au sein d'un lycée pour gérer des étudiants et étudiantes (ex. « Marie Dupont » ou « Mohamed Ali »), et des classes (ex. la « terminale A », ou la « seconde 6 »). Chaque élève aura des notes dans différentes matières, et nous calculerons des statistiques sur leurs notes (moyennes, etc).

Après chaque définition d'une nouvelle structure, vous allez définir des fonctions pour manipuler ces structures, les afficher, etc.

Conseils de travail : pour ce TD9 à faire de votre côté, en distanciel, nous vous conseillons de travailler avec Eclipse (ou l'éditeur de votre choix) et un seul fichier Java (créez un nouveau projet appelé **TD9**, et un fichier appelé `TD9.java`), sur votre ordinateur. Vous pouvez utiliser le fichier `TD9.java` proposé sur Moodle ou Discord. Vous verrez que dans son `main` il y a tous les exemples utilisés dans ce sujet, dans des commentaires multi-lignes commençant par `/*` et terminant par `*/`. Les exemples sont rangés question par question. Au fur et à mesure des questions, vous pourrez supprimer ces lignes ouvrant/fermant les commentaires (`/*`, `*/`) dans le `main`, et tester les exemples d'utilisation de chaque structures et fonctions définies, afin de vérifier que vos fonctions font ce qui est attendu, avant de passer à la question suivante.

Structure `Matiere` : notes d'un-e élève dans une certaine matière

Question 1) Définir la structure `Matiere` composée d'un champ `nom` de type `String`, et d'un champ `note` qui sera un tableau de `double`.
Par exemple, une fois que la structure est bien définie, on peut s'en servir (dans la fonction `main`) comme cela :

```
// on crée un nouvel objet instance de la structure Matiere
Matiere info_Marie_Dupont = new Matiere();
// on remplit ces deux champs, nom de la matière
info_Marie_Dupont.nom = "Informatique"; // String
// et les notes de "Marie Dupont" dans cette matière
info_Marie_Dupont.notes = new double[] {10.0, 15.0, 14.5}; // double[]
```

Cette structure `Matiere` sera utilisée pour représenter les notes d'un-e élève à cette matière, par exemple ici ces trois notes `10.0, 15.0, 14.5` sont être celles obtenues par l'élève « Marie Dupont ».

De même que pour les tableaux, si on affiche avec `System.out.println` un objet instance d'une structure, cela affichera son **adresse** et **pas** les valeurs de ses champs :

```
System.out.println(info_Marie_Dupont); // cela affichera, par exemple
Matiere@6ff3c5b5
```

Question 2) Écrire une fonction `afficheMatiere`, qui affiche une structure `Matiere`. Par exemple, cela donnera :

```
// on affiche cette matière
afficheMatiere(info_Marie_Dupont); // cela affichera
```

```
Informatique : 10.0, 15.0, 14.5
```

Question 3) Écrire une fonction `moyenneNotes` qui calcule la moyenne des notes d'une matière (cf TP6 Ex1.a]). Par exemple, une fois que la fonction est bien définie, on peut s'en servir (dans la fonction `main`) comme cela :

```java
// on affiche la moyenne des notes de cette élève dans la matière informatique
double moyenne_informatique = moyenneNotes(info_Marie_Dupont);
System.out.println("Moyenne de cette eleve en " + info_Marie_Dupont.nom +
    " = " + moyenne_informatique);
```

```
Moyenne de cette eleve en Informatique = 13.166666666666666
```

(Bonus) mettre à jour la fonction `afficheMatiere` pour aussi afficher la moyenne.

```
Informatique : 10.0, 15.0, 14.5 (moyenne = 13.166666666666666)
```

(Challenge) tronquer l'affichage de la moyenne à deux chiffres après la virgule.

```
Informatique : 10.0, 15.0, 14.5 (moyenne = 13.17)
```

Structure `Eleve` : identité et notes d'un-e élève

Question 4) Définir la structure `Eleve` composée d'un champ `nom` et un champ `prenom` chacun de type `String`, d'un champ `age` de type entier, et d'un tableau de `Matiere`. Il s'agit des informations utilisées pour afficher un bulletin de notes (on calculera la moyenne dans chaque matière plus tard, question 12).
Par exemple, une élève appelée « Marie Dupont » et qui suit juste le cours d'informatique, avec quelques notes dans ce cours, sera créée ainsi :

```java
Eleve marie_dupont = new Eleve();
marie_dupont.nom = "Dupont";
marie_dupont.prenom = "Marie";
marie_dupont.age = 19;
// pour l'exemple, juste une matière
marie_dupont.matieres = new Matiere[] { info_Marie_Dupont };
```

Pour un deuxième exemple, pour un élève appelé « Jules Ferry » :

```java
Eleve jules_ferry = new Eleve();
jules_ferry.nom = "Ferry";
jules_ferry.prenom = "Jules";
jules_ferry.age = 18;

// pour l'exemple, juste une matière
Matiere[] matieres = new Matiere[1]; // tableau de taille = 1
matieres[0] = new Matiere();
matieres[0].nom = "Informatique";
matieres[0].notes = new double[] {10.0, 5.0, 14.5};
jules_ferry.matieres = matieres;
```

Question 5) Écrire une fonction `afficheEleve`, qui prend en entrée un objet instance de la structure `Eleve`, et qui l'affiche. Par exemple, cela pourra donner cet affichage :

```
1 afficheEleve(jules_ferry);
```

Qui affichera :

```
Jules Ferry (âge = 18)
Informatique : 10.0, 5.0, 14.5
```

Structure `MatiereStat`

Question 6) Définir la structure `MatiereStat` stockant trois informations statistiques sur une matière (pour des élèves d'une même classe) : la moyenne la plus haute, la moyenne la plus basse et la moyenne générale des étudiants d'une classe. On stocke ces informations avec un champ de type `String` (nom de la matière), et 3 champs de types `double`.

Une instance de `MatiereStat` pourra être créée manuellement comme ceci (dans votre `main`) :

```
1 MatiereStat info_terminale_A = new MatiereStat();
2 info_terminale_A.nom = "Informatique";
3 // trois moyennes, la plus basse, la plus haut, et la moyenne générale de la classe
4 info_terminale_A.mini = 9.833333333333334; // moyenne de Jules Ferry
5 info_terminale_A.maxi = 13.166666666666666; // moyenne de Marie Dupont
6 info_terminale_A.moyenne = 11.5; // moyenne de leurs moyennes
7 System.out.println(info_terminale_A); // ceci affichera juste l'adresse !
```

```
MatiereStat@52cc8049
```

Attention : on ne demande pas encore de calculer automatiquement ces statistiques, il faut juste définir la structure. Vous écrirez en question 12) une fonction pour calculer ces statistiques pour une classe

Question 7) Écrire une fonction `afficheMatiereStat`, qui prend en entrée une structure `MatiereStat` et qui affiche les informations contenues dans cette structure. Par exemple

```
1 afficherMatiereStat(info_terminale_A);
```

```
Pour la matière Informatique
Pire moyenne = 9.833333333333334
Meilleure moyenne = 13.166666666666666
Moyenne générale = 11.5
```

Question 8) (optionnel) Écrire une fonction `copieMatiereStat`, qui prend en entrée une structure `MatiereStat` et qui renvoie une copie de cette structure (et PAS une référence, mais bien une copie, ie une nouvelle instance de la structure, contenant les mêmes valeurs dans chacun des champs).

Structure `Classe`

Question 9) Définir la structure `Classe` qui sera composée d'un champ `nom` de type `String`, d'un tableau contenant des `Eleve`, et d'un tableau de `MatiereStat`. Par exemple, une instance de cette structure aura pour `nom` `"Terminale A"`, suivi du tableau des étudiants et du tableau contenant les informations statistiques de chaque matière `MatiereStat`.

Sans chercher à être cohérent sur les valeurs des moyennes ou le nombre d'élèves, une instance de `Classe` pourra être créée comme ceci (dans votre `main`) :

```
1 Classe terminale_A = new Classe();
2 terminale_A.nom = "Terminale A";
3 terminale_A.eleves = new Eleve[] { marie_dupont, jules_ferry };
4 terminale_A.stats = new MatiereStat[] { info_terminale_A };
5 System.out.println(terminale_A); // ceci affichera juste l'adresse !
```

```
Classe@27973e9b
```

Fonctions pour construire des objets

Question 10) Écrire une fonction `construireEleve`, qui prend en entrée un nom et un prénom (sous la forme de `String`) et qui crée et renvoie un (nouvel) objet de type `Eleve`, avec ce nom et prénom. Le tableau de matières ne sera **pas** initialisé au début d'année (ou alors initialisé comme un tableau vide). On ne se soucie **pas** ici du champ `age`.

Par exemple,

```
1 Eleve mohamed_ali = construireEleve("Ali", "Mohamed");
2 System.out.println(mohamed_ali); // ceci affichera juste l'adresse !
3 afficheEleve(mohamed_ali);
```

```
Eleve@312b1dae
Mohamed Ali (âge = 0)
```

Si votre fonction `afficheEleve` échoue sur cet exemple, ce n'est pas grave. Vous avez sûrement observé une erreur comme

```
1 Exception in thread "main" java.lang.NullPointerException
2     at TD9_correction.afficheEleve(TD9_correction.java:120)
3     at TD9_correction.main(TD9_correction.java:355)
```

parce que la fonction `afficheEleve` accède à un tableau `mohamed_ali.matieres` qui n'a pas été créé (ce n'est pas comme un tableau vide).

Question 11) (optionnel) Écrire une fonction `afficheMoyennes`, qui prend en entrée une `Classe` et qui, pour chaque matière, affiche la moyenne de chaque élève (de type `Eleve`). On suppose que chaque élève a des notes dans les mêmes matières.

```
1 afficheMoyennes(terminale_A);
```

```
Moyennes pour la classe Terminale A
Moyennes en Informatique :
Marie Dupont : 13.166666666666666
Jules Ferry : 9.833333333333334
```

Question 12) (plus difficile) Écrire une fonction `calculeStatistiques`, qui prend en entrée une `Classe` et qui construit et remplit le tableau contenant les éléments `MatiereStat` en calculant la moyenne la plus basse, la plus élevée et la moyenne générale. On supposera que tous les étudiants d'une même classe suivent les mêmes matières et que les matières sont dans le même ordre pour chaque étudiant. Remarque : la moyenne générale d'une classe (à une matière donnée) est simplement la moyenne des moyennes de chaque élève dans cette matière.

Pour compléter l'exemple des questions précédentes, on peut rajouter des notes pour une deuxième matière (« Anglais ») aux deux élèves de cette classe « Terminale A » (`marie_dupont` et `jules_ferry`), puis appeler `calculeStatistiques` pour calculer les statistiques pour les deux matières, et enfin afficher ces statistiques :

```
1  // pour Marie Dupont, on récupère ses notes d'informatique
2  Matiere[] matieres_MD = new Matiere[2];
3  matieres_MD[0] = new Matiere();
4  matieres_MD[0].nom = "Informatique";
5  matieres_MD[0].notes = marie_dupont.matieres[0].notes;
6
7  // et on rajoute des notes d'anglais
8  matieres_MD[1] = new Matiere();
9  matieres_MD[1].nom = "Anglais";
10 matieres_MD[1].notes = new double[] {6, 11, 9.5, 12};
11 marie_dupont.matieres = matieres_MD;
12
13 // pour Jules Ferry, on récupère ses notes d'informatique
14 Matiere[] matieres_JF = new Matiere[2];
15 matieres_JF[0] = new Matiere();
16 matieres_JF[0].nom = "Informatique";
17 matieres_JF[0].notes = jules_ferry.matieres[0].notes;
18
19 // et on rajoute des notes d'anglais
20 matieres_JF[1] = new Matiere();
21 matieres_JF[1].nom = "Anglais";
22 matieres_JF[1].notes = new double[] {20, 17, 14, 13.5};
23 jules_ferry.matieres = matieres_JF;
24
25 calculeStatistiques(terminale_A);
26 afficheMoyennes(terminale_A);
```

```
Moyennes pour la classe Terminale A
Moyennes en Informatique :
  Marie Dupont : 13.166666666666666
  Jules Ferry : 9.833333333333334
Moyennes en Anglais :
  Marie Dupont : 9.625
  Jules Ferry : 16.125
```

Calcul de l'élève ayant la meilleure moyenne d'une matière

Question 13) Écrire une fonction `meilleurMoyenne`, qui prend en entrée une `Classe` et le nom d'une matière sous la forme d'un `String` et renvoie la structure `Eleve` qui a la meilleure moyenne dans la matière.

Une fois que cela est fait, on peut par exemple s'en servir pour afficher les meilleur-e-s élèves dans chaque matière :

```
String[] nomsMatieres = new String[] { "Informatique", "Anglais" };
// Pour chaque matière, afficher le ou la meilleur-e étudiant-e
for (int i = 0; i < nomsMatieres.length; i++) {
  Eleve e = meilleurMoyenne(terminale_A, nomsMatieres[i]);
  System.out.println("Le ou la meilleur-e etudiant-e en " + nomsMatieres[i]
      + " est \"" + e.prenom + " " + e.nom + "\"");
}
```

```
Le ou la meilleur-e etudiant-e en Informatique est "Marie Dupont"
Le ou la meilleur-e etudiant-e en Anglais est "Jules Ferry"
```

Challenge : calculer l'âge en fonction de la date de naissance

Question 14) (challenge) On veut maintenant automatiquement remplir le champ `age` de la structure `Eleve` au moment où on crée un objet de cette structure. On suppose qu'on a à notre disposition une structure `Date` qui a 3 champs `jour`, `mois` et `annee` de type `int`. Quand on appelle le constructeur de cette structure `Date()`, les champs de cette structure sont remplis par le système et contiennent les informations actuelles. Compléter la question 9 pour obtenir la fonction qui remplit le champ `age` d'un objet de type `Eleve` en ajoutant en entrée de cette fonction un objet de type `Date` contenant la date de naissance de l'étudiant.

Vous pouvez utiliser ce morceau de code, pour faire cette fonction :

```
// Une structure qui représente une date
class Date {
  public int jour; // avec un jour (entier), entre 1 et 31
  public int mois; // avec un mois (entier), entre 1 et 12
  public int annee; // avec une année (entier)

  // calcule la date d'aujourd'hui, fourni par l'enseignant
  public Date() {
    java.time.LocalDate today = java.time.LocalDate.now();
    this.jour = today.getDayOfMonth();
    this.mois = today.getMonthValue();
    this.annee = today.getYear();
  }
}
```

a) Écrire d'abord une fonction `int calculeAge(Date naissance)` qui calcule l'âge (nombre d'année) en fonction de la `Date` de naissance (en utilisant `Date aujourdhui`

`= new Date()`) et des calculs sur `naissance.annee` et `aujourdhui.annee` (attention pour le faire correctement ce n'est pas si rapide qu'une simple différence !).

b) Vous devez ensuite écrire une nouvelle fonction, complétant celle de la question 9), de signature `Eleve nouvelEleve(String nom, String prenom, Date naissance)`.

Challenge : des constructeurs propres pour chaque structure

Question 15) En vous inspirant de la syntaxe montrée ci-dessus pour la méthode `Date()` dans la structure `Date` (méthode dite *constructeur* de la structure), écrivez des constructeurs pour chacune des structures précédemment définies. Cela pourra permettre d'écrire du code plus concis comme ces exemples :

```
1 Matiere info_Marie_Dupont = new Matiere( "Informatique", new double[]
      {10.0, 5.0, 14.5} );
2
3 Eleve marie_dupont = new Eleve( "Dupont", "Marie", 18, new Matiere[] {
      info_Marie_Dupont } );
4
5 Classe groupe_1 = new Classe( "Groupe 1", new Eleve[] { marie_dupont } );
```