Compléments de Programmation
Licence 1 UPEC 2022/2023
TM 10 : Structures

1 Questions de compréhension

Exercice 1: à faire sur feuille

1. Dans quelle partie du code est déclarée la structure ?
2. Est ce qu'on peut parcourir une structure à l'aide d'une boucle for ?
3. Est ce que la taille de la structure dépend de la taille des données ?
4. Est ce qu'une structure peut contenir des données de type structure ?
5. Est ce que les noms des cases peuvent être définies à l'interieur du main ?
6. Tableaux versus structures :
   On souhaite modéliser les points dans un espace à 3 dimensions. On peut, au choix, modéliser un point soit par un tableau de 3 éléments de type `double` : `double [] p = new double[3];` , soit par une structure, de la manière suivante :

   ```
   class Point{
       double x;
       double y;
       double z;
   }
   ```

   que l'on utilise dans le corps de la manière suivante : `Point p = new Point();`
   (a) Quels sont les avantages et les inconvénients de chaque approche ?
   (b) On souhaite maintenant stocker plusieurs points (pour par exemple faire un dessin). Comment procédez-vous, pour chaque modélisation ?
7. Le constructeur d'une classe (ie. l'appel `new Point()` dans l'exemple précédent) sert-il à déclarer ou bien initialiser une variable (ou les deux) ?

2 Programmation

Exercice 2: Un exemple de structure simple

On souhaite implémenter une structure `Personne` dont les champs sont le nom, le prénom, l'âge et la taille d'une personne.

1. Ecrivez la structure avec ses champs. Choisissez bien les types.
2. Testez votre structure en saisissant et affichant les différents champs dans un main (attention, ici on ne demande pas encore d'écrire des fonctions).
3. Écrivez une procédure `saisirFichePerso` qui demande à l'utilisateur de saisir les champs d'une personne passée en paramètre.
4. Écrivez une procédure `afficherFichePerso` qui affiche de façon jolie une personne passée en paramètre.
5. Réécrivez le `main` en déclarant une personne, puis en saisissant ses champs à l'aide de la fonction `saisirFichePerso` et enfin en affichant ses champs à l'aide de la fonction `afficherFichePerso`.

Exercice 3: Des fonctionnalités plus avancées

On travaille toujours avec la structure `Personne` de l'exercice précedent.

1. Écrivez une procédure `anniversaire` qui vieillit une personne d'une année.
2. Écrivez une fonction `plusGrand` qui prend en paramètres deux personnes et qui renvoie la plus grande des deux.
3. Écrivez une fonction `estPlusJeune` qui prend en paramètres deux personnes et qui renvoie `true` si la première personne est plus jeune que la seconde et `false` sinon.
4. Maintenant créez un `main` dans lequel vous devez déclarer plusieurs personnes, puis les créer à l'aide des fonctions de l'exo 2, puis vieillir certaines et afficher le résultat, puis afficher laquelle est plus la grande et la plus jeune.

5. Écrivez une fonction `agesPersonnes` qui prend en paramètre un tableaux de personnes, et qui calcule la moyenne de leur ages.
6. Écrivez une fonction `trouvePersonne` qui prend en paramètre un tableaux de personnes et un nom, et qui renvoie la première personne du tableau qui a ce nom, sinon `null`.
7. Enfin, modifiez le `main` en créant un tableau de personnes, en l'initialisant en utilisant les fonctions de l'exo 2, puis afficher la moyenne de leurs ages en utilisant `agesPersonnes`, et puis demande à l'utilisateur un nom et, à l'aide de la fonction `trouvePersonne`, vérifie si ce nom apparaît dans le tableau de personnes que vous avez créé.

Exercice 4: Les articles

Dans les prochains exercices, on désire modéliser un système de « panier » (comme sur vos sites internet d'achat préférés). Ces paniers contiendront les articles que l'utilisateur aura décidé d'acheter. On utilisera pour cela deux structures, Article et Panier.

La classe Article représente les différents articles qu'un utilisateur peut décider d'acheter et d'ajouter à son panier. Chaque article sera représenté par deux champs : son nom et son prix.

```
class Article{
  String nom;
  double prix;
}
```

1. Écrivez une fonction `saisirArticle` qui prend en paramètre une chaîne de caractères `nom` et un réel `prix` et qui déclare et renvoie un article dont les champs correspondent aux valeurs passées en paramètre.
2. Écrivez une procédure `afficherArticle` qui affiche les informations de l'article passée en paramètre. Par exemple si `a` est l'article "Brosse à dents" et coûte 2 euros, la procédure `afficherArticle` appliquée à `a` devra afficher

   ```
   L'article Brosse à dents coûte 2 euros.
   ```

Exercice 5: Le panier

La classe Panier représente le panier contenant les différents achats de l'utilisateur. Elle contient donc deux champs : un tableau d'Article (de capacité maximale `MAX`, une constante du programme), et le nombre d'éléments dans le panier.

```
class Panier{
  Article [] article;
  int nbArticles ;
}
```

1. Écrivez une fonction `initialiserPanier` qui crée un panier avec son tableaux d'articles et nbArticles intialisé à zéro.
2. Écrivez une procédure `afficherPanier` qui affiche les informations de chaque article du panier (pensez à utiliser les fonctions déjà programmées de la classe Article). Par exemple, si le panier contient un seul article Brosse à dents, la procédure `afficherPanier` appliquée au panier devra afficher

   ```
   Votre panier contient 1 article(s) :
     L'article Brosse à dents coûte 2 euros.
   ```
3. Écrivez une fonction `prixTotal` qui renvoie la somme des prix de chaque article du panier passé en paramètre, c'est à dire le prix que l'utilsateur devra payer pour son panier.
4. Écrivez une fonction `panierPlein` qui renvoie `true` si le panier est plein, `false` sinon.
5. Écrivez une fonction `ajouterArticle` qui prend en paramètre un article et un panier, et qui ajoute l'article dans le panier s'il reste de la place, et qui renvoie un booléen qui permet de dire si l'ajout a pu se faire ou non (par exemple par manque de place). On pensera par ailleurs à mettre à jour le nombre d'articles et à ne pas dépasser la taille maximale.
6. Écrivez un programme qui demande à l'utilisateur de saisir un panier, puis qui affiche son contenu ainsi que son prix total. Si le panier n'est pas plein, demander à l'utilisateur s'il veut y ajouter un article supplémentaire, et le faire si c'est le cas.

3 Trace d'execution

Exercice 6: A faire sur feuille

Pour chaque bout de code, décrire les affichages.

```
1  void main() {
2    Rectangle r = new Rectangle();
3    afficheRectangle(r);
4    saisieRectangle(r, 3, 4);
5    afficheRectangle(r);
6    println("L'aire du rectangle est " + aire(r) + ".");
7  }
8
9
10 class Rectangle{
11   int longueur;
12   int largeur;
13 }
14
15 void saisieRectangle(Rectangle r, int longueur, int largeur) {
16   if (longueur < largeur) {
17     r.longueur = largeur;
18     r.largeur = longueur;
19   } else {
20     r.longueur = longueur;
21     r.largeur = largeur;
22   }
23 }
24 void afficheRectangle(Rectangle r) {
25   println("Rectangle de " + r.largeur + " fois " + r.longueur + ".");
26 }
27 int aire(Rectangle r) {
28   return  r.longueur * r.largeur;
29 }
```

```
1  void main() {
2    Point p1 = new Point();
3    affichePoint(p1);
4    Point p2 = creePoint(1,2,3);
5    affichePoint(p2);
6    saisiePoint(p1, 2,4,6);
7    affichePoint(p1);
8
9    affichePoint(somme(p1, p2));
10 }
11
12 class Point{
13    double x, y, z;
14 }
15
16 Point creePoint(double x, double y, double z) {
17   Point p = new Point();
18   p.x = x;
19   p.y = y;
20   p.z = z;
21   return p;
22 }
23
24 void saisiePoint(Point p, double x, double y, double z){
25   p.x = x;
26   p.y = y;
27   p.z = z;
28 }
29
30 void affichePoint(Point p) {
31    println("Point de coordonnees: ( " + p.x + ", " + p.y + ", " + p.z + ").");
32 }
33
34 Point somme(Point p1, Point p2) {
35   Point p = creePoint(p1.x + p2.x, p1.y + p2.y, p1.z + p2.z);
36   return p;
37 }
```