Université Claude-Bernard Lyon 1
Master CCI

Classes, Tableaux et l'héritage simple

1 Exercice 1

Le but de la classe *Mat2D* est de manipuler un tableau à deux dimensions. Chaque méthode sera appelée dans une fonction main.

1. Déclarer la classe *Mat2D* et son attribut un tableau de double à deux dimensions.
2. Écrire un constructeur qui prend en paramètre un tableau de double, " Créer un objet dans une fonction main à l'aide de ce constructeur". Cet objet aura pour attribut **un tableau à deux dimensions qui n'est pas une matrice (toutes les lignes n'ont pas le même nombre de colonnes)**.
3. Écrire une méthode permettant l'affichage de l'attribut.
4. Écrire une méthode qui retourne la somme des éléments de la ième ligne de l'attribut. L'indice i sera passé en paramètre.
5. Écrire une méthode qui compte le nombre d'éléments du tableau.
6. En utilisant les méthodes précédentes écrire une méthode qui retourne la somme de tous les éléments.
7. Écrire une méthode qui retourne la moyenne des éléments de la table.
8. Écrire une méthode qui retourne l'élément le plus grand dans un tableau.
9. Écrire une méthode qui indique si le double passé en paramètre est présent dans le tableau.
10. Écrire une fonction main permettant de lancer les méthodes écrites.

```
Exemple pour le test: double [][] tableau = {{7,3,1,5},{1,2},{55},{4,8,9}};
```

2 Exercice 2

2.1 La classe Complexe

On considère la représentation algébrique des nombres complexes :z= x+ iy, x est la partie réelle et y la partie imaginaire.

zc= x-iy est le conjugué du complexe z est —x + iy—. La norme de z= $(x^2 + y^2)^{1/2}$. Le but de cette première partie est d'écrire une classe qui permet de travailler avec des nombres complexes.

— Créez une classe **Complexe** qui aura deux attributs privés : un réel x "représentant la partie réelle" et un réel y "représentant la partie imaginaire" de ce complexe.
— Écrire un constructeur qui prend en paramètre les valeurs de x et de y.
— Écrire un constructeur qui gènere aléatoirement un nombre complexe.
— Écrire une méthode qui affiche le complexe sous forme algébrique.
— Écrire une méthode de type **Complexe** qui renvoie le conjugué du complexe courant.
— Écrire une méthode ajoute permettant d'ajouter au complexe un autre complexe.
— Écrire une méthode booléenne qui compare deux complexes. Le complexe courant et le complexe passé en paramètre dans la méthode. La méthode retourne vrai ou faux

2.2 Tableau de complexes

Dans cette partie on va écrire une classe **TableauComplexes** qui permet de gérer un tableau de complexe.

— Déclarer un attribut t qui est un tableau contenant des complexes.
— Écrire un constructeur qui permet de créer un tableau dont la taille est rentrée en paramètre. Les cases seront remplies par des complexes générés aléatoirement.
— Écrire une méthode qui permet d'afficher le tableau t.
— Écrire une méthode qui indique si un complexe passé en paramètre se trouve dans le tableau. La méthode renvoie vraie ou faux.
— Écrire une méthode qui échange deux cases du tableau dont les indices sont donnés en paramètre.
— Écrire une méthode qui supprime une case dont l'indices est donné en paramètre.
— Écrire une méthode qui ajoute un complexe dans le tableau. Le complexe sera passé en paramètre.

3 Exercice Etudian et Promo

3.1 La classe Etudiant

Voici la classe suivante :

```
class Etudiant{

    /** Attributs */

    String nom;
    String prenom;
    int numero;  // numero de l'etudiant
    static int nbretucrees=0;  // nombre d'etudiants crees
    String [] courssuivis;  // tableau contenant les cours suivis
```

```
    int notes[]; // tableau des notes pour chaque cours suivis.
}
```

1. Écrire un constructeur qui prend en paramètre le nom, le prenom et le tableau contenant les cours suivis. Le numéro de l'étudiant sera son rang de creation. Si l'étudiant est le troisième crée son numéro étudiant sera 3. Le tableau de note sera initialisé avec des zéros et sa taille correspondra au nombre de cours suivis.
2. Écrire un constructeur de copie. Le constructeur prend en paramètre un étudiant. Attention il ne faut pas augmenter le nombre d'étudiants crées. Les attributs seront copiés un à un.
3. Écrire une méthode qui compare l'Etudiant courant avec un étudiant passé en paramètre. Cette méthode renvoie vrai si ce sont les mêmes et faux sinon.
4. Écrire une méthode qui transforme un Étudiant en String pour l'affichage.
5. Écrire une méthode qui affiche un Étudiant sans utiliser la méthode précédente.

3.2 La classe Principale

Cette classe a pour but de tester la classe Etudiant précédente.

1. Créer un Etudiant,
2. Afficher l'Etudiant en utilisant la méthode le transformant en String,
3. Afficher l'Etudiant en utilisant la méthode affichage(),
4. Créer un nouvel Etudiant
5. Comparer ce nouvel Etudiant au premier,
6. Créer un nouvel Etudiant par copie et vérifier que la comparaison renvoie vrai.
7. Afficher ce nouvel étudiant en vérifiant que son numéro est bon.

3.3 La classe Promo

On travaille dans le context de la classe suivante :

```
class Promo{
/** tableau d'etudiants */

    Etudiant tab[];
```

1. Écrire un constructeur qui initialise le tableau.
2. Écrire une méthode qui ajoute un Etudiant dans le tableau en utilisant le constructeur de copie.
3. Écrire une méthode qui ajoute un Etudiant dans le tableau en utilisant l'affectation.
4. Écrire une méthode qui affiche le tableau d'étudiants.

3.4 La classe Principale

Tester les méthodes de la classe Promo dans la classe principale.

4 Exercice : heritage simple

4.1 Classe Personne

Créer une classe publique, nommée Personne, contenant 3 champs :

- un champ nom de type String,
- un champ prenom de type String,
- un champ age de type int.
- Créer un constructeur initialisant tous les champs, de signature Personne(String leNom, String leprenom, int lage) initialisant les champs de la classe.
- Créer un constructeur de recopie, de signature Personne(Personne p) qui initialise les champs de la classe à ceux de p.
- Créer la méthode afficher() qui affiche à l'écran le nom, le prenom et l'age au format décrit ci-après :
  Nom : . . . ..
  Prenom : . . . ..
  Age : . . . ..

4.2 Classes Prof et étudiants (Héritage)

- On désire gérer des personnes de type prof qui vont avoir un attribut supplémentaire, la matière qu'elles enseignent. La méthode afficher sera redéfinie, car elle doit permettre d'afficher en plus de tous les renseignements sur un prof, la matière qu'il enseigne.
- Créez une classe Etudiant qui doit hériter de personne, mais qui comporte comme attributs supplémentaires la section, le nombre de jour d'absence.
- Créez une classe Prof qui doit hériter de personne, mais qui comporte comme attribut supplémentaire les matières qu'il enseigne.
- redéfinir la methode afficher()dans chaque classe qui permettre l'affichage des attributs supplémentaires

Tester les méthodes des différentes classes dans une classe Principale (main).

5 Exercice : Recettes de cuisine

On déclare la classe **Recette** avec les attributs suivants :

```
public String nom;
public int numero;
public String[] ingredients;
public int[] proportions;
static int nbrecettes=0;
```

Compléter cette classe avec :

- un constructeur qui prend en paramètre le nom de la recette, un tableau de `String` représentant les ingrédients, et un tableau de `int` représentant les proportions. Le constructeur devra créer un nouveau tableau pour chacun, et recopier le tableau correspondant donné en argument. Le numéro sera attribué automatiquement en fonction du nombre de recettes créées jusqu'ici.
- une méthode **affiche** qui affiche la recette dans la console, en annonçant d'abord le nom et le numéro de la recette, puis sur chaque ligne on trouvera le nom de l'ingrédient et sa proportion.
- une méthode **ajoute** qui prend en argument le nom d'un ingrédient et sa proportion, et les ajoute à la recette. Il faudra penser à agrandir d'une case (et donc recopier) les tableaux existants.

5.1 Menu

On déclare la classe **Menu** avec les attributs suivants :

```
Recette[] plats;
int nbPlatsAjoutes;
```

L'attributs `nbPlatsAjoutes` sert à compter combien de recettes ont déjà été réellement ajoutées au menu, et n'est donc pas forcément égal à la taille du tableau `plats`. Compléter cette classe avec :

- un constructeur qui prend en paramètre le nombre de plats maximum voulus dans le menu, ce qui permet d'allouer la mémoire pour le tableau `plats`. Les recettes du menu seront ajoutées dans une autre méthode. Attention, pour l'instant, le nombre de plats qui ont réellement été ajoutés est zéro.
- une méthode **ajoutPlat** qui prend comme argument une recette. Si le menu n'est pas encore complet, il faut ajouter la recette au menu, et incrémenter le nombre de plats ajoutés. Sinon, il faut afficher un message d'erreur.
- une méthode **affiche** qui affiche les recettes qui ont été ajoutées au menu.
- une méthode **totalIngredient** qui prend en argument le nom d'un ingrédient, et calcul la somme des proportions de cet ingredient dans les recettes du menu.

Vous testerez votre programme sur le `main` suivant :

```
public static void main(String[] args){
       String[] ingredients1= {"farine", "oeufs", "lait"};
       int[] proportions1={500, 6, 1000};
       Recette r1=new Recette("Crepes", ingredients1, proportions1);

       String[] ingredients2= {"sucre", "citron", "beurre", "oeufs"};
```

```
        int[] proportions2={200, 2, 80, 2};
        Recette r2=new Recette("Creme de citron", ingredients2, proportions2);

        Menu m=new Menu(2);
        m.ajoutPlat(r1);
        m.ajoutPlat(r2);
        m.affiche();

        System.out.println("Nb d'oeufs? "+m.totalIngredient("oeufs"));
        r1.ajoute("fleur d'oranger", 5);
        m.affiche();

        m.ajoutPlat(r1);
    }
```

Il doit s'afficher dans la console :

```
Mon menu:
Recette: Crepes numero: 1
farine 500
oeufs 6
lait 1000
Recette: Creme de citron numero: 2
sucre 200
citron 2
beurre 80
oeufs 2

Nb d'oeufs? 8
Nouvel ingredient fleur d'oranger ajouté

Mon menu:
Recette: Crepes numero: 1
farine 500
oeufs 6
lait 1000
fleur d'oranger 5
Recette: Creme de citron numero: 2
sucre 200
citron 2
beurre 80
oeufs 2

Le menu est déjà complet!
```