Les listes et les tables :

TP1 :Modéliser un élève

Un élève sera ici modélisé par la classe `Eleve` d'un paquetage nommé `gestionEleves`, de la façon suivante.

La classe `Eleve` posséde trois attributs privés :

- son nom, nommé `nom`, de type `String`,
- un ensemble de notes, nommé `listeNotes`, qui sont des entiers rangés dans un `ArrayList<Integer>`
- une moyenne de type **`double`**, nommée `moyenne`, qui doit toujours être égale à la moyenne des notes contenues dans l'attribut `listeNotes`. Un élève sans aucune note sera considéré comme ayant une moyenne nulle.

La classe `Eleve` possède un constructeur permettant uniquement d'inialiser le nom de l'élève.

La classe `Eleve` possède aussi cinq méthodes publiques :

- Un getter pour la moyenne de l'élève c'est-à-dire une méthode d'en-tête
  ```
  public double getMoyenne()
  ```
  renvoie la valeur de l'attribut moyenne ;
- Un getter pour le nom de l'élève c'est-à-dire une méthode d'en-tête
  ```
  public String getNom()
  ```
  renvoie le nom de l'élève ;
- Un getter pour la liste des notes de l'élève c'est-à-dire une méthode d'en-tête
  ```
  public ArrayList<Integer> getListeNotes()
  ```
  renvoie la liste des notes de l'élève ;
- La méthode d'en-tête
  ```
  public void ajouterNote(int note)
  ```
  ajoute la note reçue en paramètre à `listeNotes` ; si la note reçue en paramètre est négative, la note introduite est 0 ; si la note reçue en paramètre est supérieure à 20, la note introduite est 20 ; la méthode actualise en conséquence l'attribut moyenne ; l'actualisation est faite à temps constant, et non pas en un temps proportionnel au nombre de notes déjà enregistrées.
- La méthode d'en-tête
  ```
  public String toString()
  ```
  qui retourne une description de l'élève considéré (par exemple : "Sophie (12.25)").

Indication : si note est une variable de type `int`, l'instruction :

```
listeNotes.add(note);
```

ajoute un `Integer` contenant la valeur `note` à `listeNotes`.

Après avoir terminé la classe `Eleve`, écrire un programme qui teste cette classe.
Indication (détail) : si la méthode `toString` décrite ci-dessus a été définie dans la classe `Eleve`, si `eleve` est de type `Eleve`, l'instruction :

```
System.out.println(eleve);
```

est équivalente à l'instruction :

```
System.out.println(eleve.toString());.
```

Modéliser un groupe d'élèves

Cet exercice fait suite à l'exercice , qu'il faut donc faire avant celui-ci.

Un groupe d'`Eleve`(s) (instances de la classe `gestionEleves.Eleve` précédemment définie) sera ici modélisé par la classe `GroupeEleves` du paquetage `gestionEleves` de la façon suivante.

La classe `GroupeEleves` posséde un attribut privé : une collection d'`Eleve`(s) nommée `listeEleves`, de type `ArrayList<Eleve>`.

La classe `GroupeEleves` ne possède pas de constructeur explicite.

La classe `GroupeEleves` possède aussi cinq méthodes publiques :

- La méthode d'en-tête

  ```
  public int nombre()
  ```

  renvoie le nombre d'`Eleve`(s) contenus dans `listeEleves`
- La méthode d'en-tête

  ```
  public ArrayList<Eleve> getListe()
  ```

  renvoie `listeEleves`.
- La méthode d'en-tête

  ```
  public void ajouterEleve(Eleve eleve)
  ```

  ajoute l'`Eleve` reçu en paramètre à `listeEleves`.
- La méthode d'en-tête

  ```
  public Eleve chercher(String nom)
  ```

  renvoie l'`Eleve` dont le nom est indiqué par le paramètre ; si plusieurs `Eleve`(s) ont même nom, la méthode renvoie le premier `Eleve` ayant ce nom contenu dans `listeEleves` ; si aucun `Eleve` n'a le nom indiqué, la méthode retourne `null` . On pourra utiliser la méthode `equals` de la classe `String` pour comparer une chaîne de caractères à une autre.

- La méthode d'en-tête

  ```
  public void lister()
  ```

  écrit à l'écran la liste des `Eleve`(s). Elle utilise une ligne par `Eleve` ; elle utilise la méthode `toString` de la classe `Eleve`.

  Après avoir terminé la classe `GroupeEleves`, écrire un programme qui teste cette classe.