TP2 Etudiant Partie 3:
Relations entre classes, ArrayList, répétitives

1 Structure du programme

La figure 1 introduit le schéma UML qui représente le programme que vous allez mettre en œuvre.

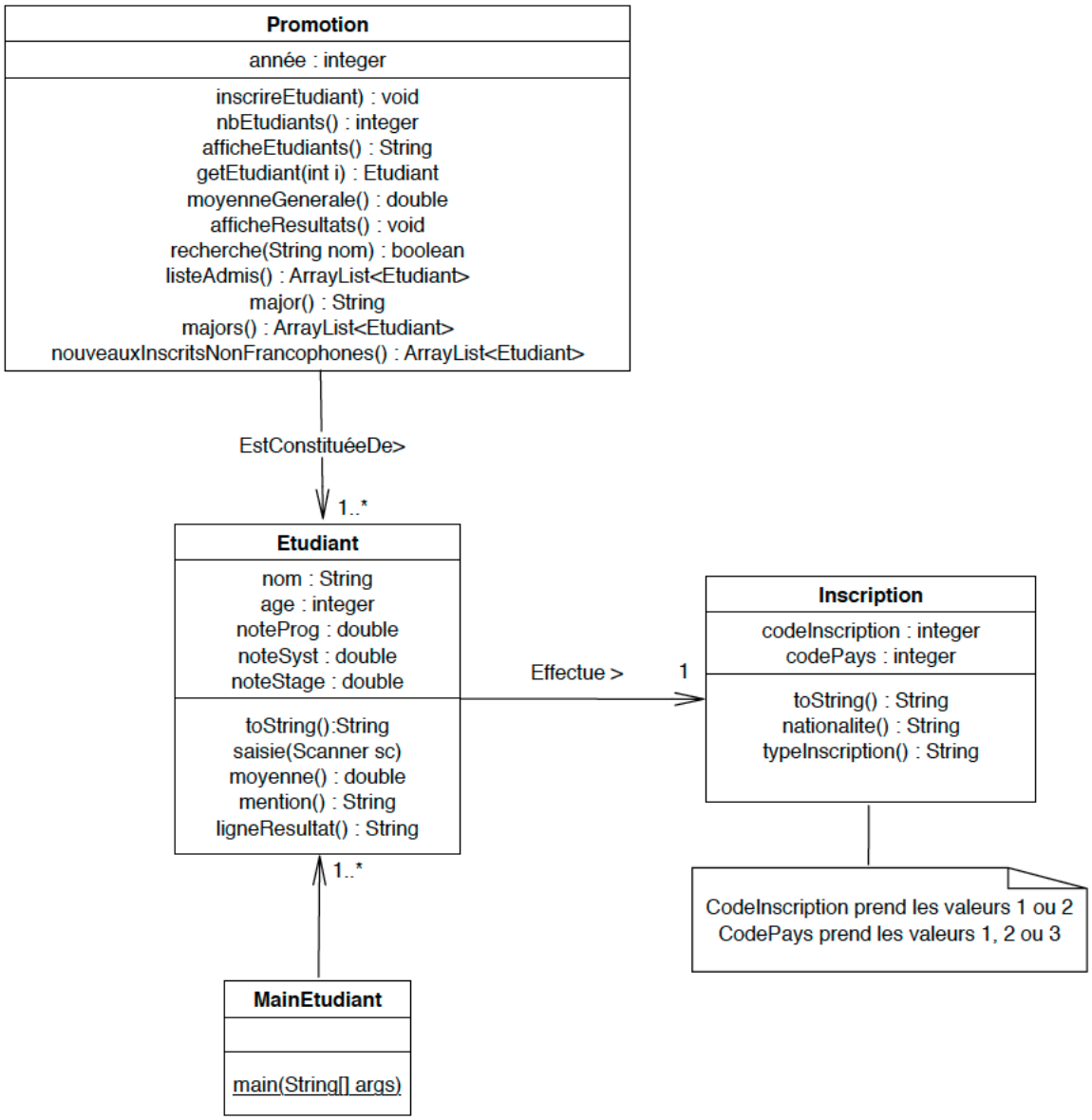

FIGURE 1 – Schéma UML des classes à mettre en place. Nota : la syntaxe pour les types de retour est celle d'UML. La syntaxe pour les paramètres est celle de Java.

Vous avez normalement défini trois classes dans le TP précédent (supposons que votre paquetage s'appelle `tp2`) :

- Une classe `Etudiant` (sans `main`) qui décrit ce qu'est un objet étudiant et quelles opérations peuvent être appliquées sur celui-ci.
- Une classe `MainEtudiant` (avec un `main`) qui permet de créer des objets étudiants et d'appeler les différentes opérations.
- Une classe `Inscription` qui décrit les modalités d'inscription des étudiants.

Nous allons dans ce troisième TP :

- Créer la classe `Promotion` avec ses attributs, constructeurs, accesseurs et méthodes.
- Faire le lien entre la classe `Promotion` et la classe `Etudiant`.
- Tester les méthodes de la classe `Promotion` dans la classe `MainEtudiant`.

2 Classe Promotion

Créer une classe `Promotion` dans le package `tp2`. La classe `Promotion` représente une promotion d'étudiants et doit disposer des caractéristiques décrites dans les sections suivantes (voir diagramme UML) :

2.1 Les attributs

- Une promotion est composée de plusieurs étudiants (qui sont stockés dans une liste). Voir l'API Java pour plus d'informations sur la classe `ArrayList` et ses méthodes associées).
- On veut également connaître l'année de la promotion.

2.2 Les constructeurs

- Créer un constructeur vide qui crée une nouvelle liste vide d'étudiants et initialise l'année à 0.
- Créer un constructeur qui prend en paramètre une année permettant d'initialiser l'attribut `année` de la promotion, et qui crée une nouvelle liste vide d'étudiants.

2.3 Les accesseurs

- Créer les accesseurs en accès (lecture) et modification (écriture) pour l'attribut `année`.

2.4 Utilisation des méthodes prédéfinies dans la classe `ArrayList`

Vous trouverez toutes les méthodes disponibles pour la classe `ArrayList`, ici

- Créer une méthode `void inscrire(Etudiant etud)`, qui permet d'inscrire (d'ajouter) un étudiant dans la promotion, après vérification qu'il n'y soit pas déjà.
- Créer une méthode `int nbEtudiants()` qui retourne le nombre d'étudiants de la promotion.
- Créer une méthode `String afficheEtudiants()`, qui, après avoir vérifié que la promotion n'est pas vide, retourne sous forme de chaîne de caractères, la liste des étudiants de celle-ci.
- Créer une méthode `Etudiant getEtudiant(int i)`, qui retourne l'étudiant de rang i de la liste.
- Améliorer la méthode `getEtudiant` afin de vérifier que le paramètre i correspond à un indice existant de la liste (Utiliser la méthode `nbEtudiant()` définie précédemment). Retourner `null` si ce n'est pas le cas, sinon retourner l'étudiant rangé à cet indice i.

2.5 Test des méthodes dans la classe `MainEtudiant`

- Créer une nouvelle promotion pour l'année 2021.
- Inscrire, dans cette promotion, les étudiants que vous avez créés dans le TP2_1 et TP2_2.
- Afficher la liste des étudiants.
- Afficher le nombre d'étudiants de la promotion.
- Afficher le troisième étudiant inscrit dans la promotion.

2.6 Lien entre les classes `Etudiant`, `Inscription` et `Promotion`

- Écrire une méthode `double moyenneGénérale()`, qui retourne la moyenne générale de la promotion (vous devez utiliser la méthode `moyenne` de la classe `Etudiant`). Tester cette méthode dans `MainEtudiant`.
- Ajouter une condition à la méthode `moyenneGénérale()` afin de vérifier que la promotion contient des étudiants. Si la liste est vide, retourner la valeur 0 (on veut ici éviter la division par 0), sinon retourner la moyenne générale de la promotion.
- Ecrire une méthode `void afficheResultats()` qui, pour chaque étudiant, affiche une ligne contenant l'ensemble de ses résultats (vous devez utiliser la méthode `ligneResultat` de la classe `Etudiant`). Tester cette méthode dans `MainEtudiant`.
- En utilisant la boucle `while`, créer une méthode `boolean recherche(String nom)`, qui retourne vrai si un étudiant est inscrit dans la promotion (on suppose qu'il n'y a pas d'homonymes), faux sinon. Tester cette méthode dans `MainEtudiant`.
- Ecrire une méthode `ArrayList<Etudiant> listeAdmis()`, qui stocke dans une nouvelle liste les étudiants admis. Dans `MainEtudiant`, utiliser cette méthode pour retourner uniquement le nom et la moyenne des étudiants admis.
- Ecrire une méthode `String major()`, qui retourne le nom du major de la promotion (on suppose dans cette première version un peu simpliste qu'il n'y a pas deux moyennes identiques). Tester cette méthode dans `MainEtudiant`.
- Ecrire une méthode `ArrayList<Etudiant> majors()`, qui retourne la liste des étudiants qui sont majors de promotion (lorsque l'on suppose qu'il peut exister des moyennes identiques). Tester cette méthode dans la classe `MainEtudiant`, et afficher le ou les noms des majors et leur moyenne.
- Ecrire une méthode `ArrayList<Etudiant> nouveauxInscritsNonFrancophones()`, qui retourne la liste des étudiants francophones dont c'est la première inscription. Tester cette méthode dans `MainEtudiant` pour afficher :
  - Le nombre d'étudiant nouvellement inscrits et non francophones.
  - Le nom et les informations concernant l'inscription de ces étudiants (utilisation de la méthode toString de la classe `Inscription`).

3 Uniquement si vous êtes avancés en programmation ...

Complétez la classe Promotion de manière à pouvoir :

— afficher un histogramme des moyennes de la promotion. Par exemple, si des étudiants ont obtenu les moyennes suivantes : Jacques 16,3, Justine 18, Germain 15,7, Hugues 12,2, Sylvia 15,9, Gaston 11,4, Astrid 11, Kim 11,1, on affiche le diagramme suivant (limité entre 10 et 20 ici mais vous devez le faire entre 0 et 20) :

```
[10-11[
[11-12[ ***
[12-13[ *
[13-14[
[14-15[
[15-16[ **
[16-17[ *
[17-18[
[18-19[ *
[19-20]
```

— connaître les moyennes les plus fréquentes,

— en utilisant l'histogramme, déterminer combien de personnes ont une certaine moyenne (arrondie) donnée,

— afficher le même histogramme qu'à la première question mais pivoté comme montré dans le tableau 1.

Table 1 – Histogramme pivoté

[10-11[	[11-12[	[12-13[	[13-14[	[14-15[	[15-16[	[16-17[	[17-18[	[18-19[	[19-20]
	*	*			*	*		*	
	*				*				
	*								