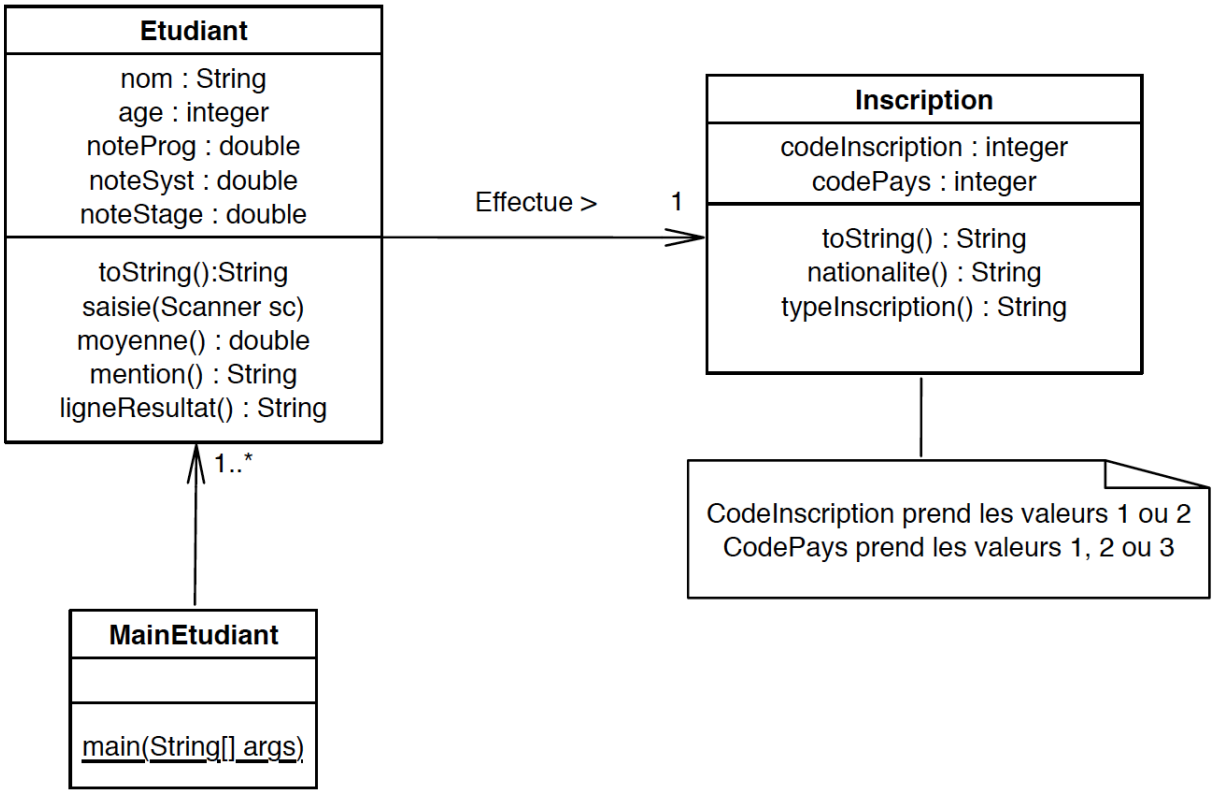
TP2 Etudiant Partie 1: Définition et utilisation d'une classe dans un programme

1 Structure du programme

Voici le schéma UML qui représente le programme que vous allez mettre en oeuvre dans ces deux TPs.



Vous allez donc définir trois classes, toutes appartenant à un même paquetage (par ex. nommé `tp2`) :

- Une classe `Etudiant` (sans méthode `main`) qui décrira ce qu'est un objet étudiant et quelles opérations peuvent être appliquées à un étudiant.
- Une classe `MainEtudiant` (avec une méthode `main` donc) qui permettra de créer des objets étudiants et d'appeler les différentes opérations.
- Une classe `Inscription` (sans main) qui décrira les différentes possibilités d'inscription des étudiants.

Nous allons effectuer ce TP en deux parties :

- Le TP 2.1 est consacré uniquement à la création de la classe `Etudiant` et de `MainEtudiant` (pour pouvoir créer des objets et tester vos méthodes). Pour une meilleure compréhension des concepts, des fichiers `Etudiant.java` et `MainEtudiant.java` sont fournis et préremplis.
- Dans le TP 2.2, nous ajoutons des contraintes sur les attributs de la classe `Etudiant`, nous créons la classe `Inscription` et nous faisons le lien entre `Inscription` et `Etudiant`.

2 La classe `Etudiant`

2.1 Structure de la classe `Etudiant`

Dans cette première partie, nous définissons la partie structurelle de la classe, c'est-à-dire ses attributs. La classe est définie dans le fichier `Etudiant.java` qui est pré-rempli et annoté.
Vous choisirez les types des attributs en fonction des informations fournies par le modèle UML. Les attributs devront être privés (leur type sera précédé du mot clef `private`). Un étudiant devra être décrit par :

- un nom (code déjà écrit pour servir d'exemple),
- un âge (à insérer),
- trois notes : une note de programmation, une note de système et une note de stage (à insérer).

2.2 Constructeurs de la classe `Etudiant`

Pour pouvoir définir des objets de la classe `Etudiant`, vous devez au préalable spécifier au moins un constructeur. Le constructeur par défaut d'une classe est sans paramètre et donne si besoin des valeurs par défaut cohérentes aux différents attributs. Cependant, vous pouvez définir autant de constructeurs que vous le souhaitez, chaque constructeur ayant plus ou moins de paramètres en fonction du besoin de l'application (chaque paramètre servira à renseigner une valeur d'attribut). Dans le fichier `Etudiant.java`, nous avons écrit le constructeur par défaut et le constructeur permettant de créer un nouvel étudiant dont on ne connaît que le nom, ceci afin que vous ayez un modèle. Vous devez écrire trois autres constructeurs :

- Le constructeur permettant de créer un nouvel étudiant en ne renseignant que son nom et son âge.
- Le constructeur permettant de créer un nouvel étudiant dont vous connaissez tous les attributs, mais pas l'âge.
- Le constructeur permettant de remplir tous les attributs.

2.3 Accesseurs de la classe `Etudiant`

Pour chaque attribut que vous avez déclaré privé, vous pouvez définir des accesseurs de type `get` et `set` afin de pouvoir accéder et/ou modifier les valeurs des attributs (principe de l'encapsulation).
Un exemple est fourni dans le fichier `Etudiant.java` pour l'attribut nom. En vous inspirant de cet exemple, vous devez, pour chaque attribut déclaré dans la partie 2.1 du TP, écrire les accesseurs correspondant (`get` et `set`).

3 La classe-programme `MainEtudiant`

Le fichier nommé `MainEtudiant.java` contient une classe possédant une méthode `main`. C'est cette méthode qui va permettre d'exécuter le programme. Dans cette classe, vous pouvez créer des étudiants grâce aux constructeurs que vous avez définis, modifier leurs informations grâce aux accesseurs de type `set`, obtenir des informations grâce aux accesseurs de type `get`, et utiliser des méthodes que vous aurez préalablement définies dans la classe `Etudiant`.

3.1 Création des étudiants grâce aux constructeurs

Dans le fichier `MainEtudiant.java`, l'étudiant etud1 a déjà été créé et peut vous servir de modèle pour instancier les autres étudiants dont les caractéristiques sont données ci-dessous :

- Créez etud2 : Jean, âgé de 24 ans, sans notes connues,
- Créez etud3 : Abdoulkhader, âgé de 23 ans, sans notes connues,
- Créez etud4 : Astrid, âgée de 26 ans, sans notes connues,
- Créez etud5 : Paolo, âgé de 27 ans, sans notes connues,
- Créez etud6 : Zoé, âgée de 26 ans, avec les notes 12 (programmation), 14 (système), 17 (stage).

3.2 Affichage des informations grâce aux accesseurs de type `get`

Nous allons ici utiliser l'accesseur de type `get` associé à l'instruction `System.out.println()` afin d'afficher les informations des étudiants. Un exemple vous est fourni pour l'étudiant `etud1` dans le fichier `MainEtudiant.java` .
Vous devez :

- Afficher toutes les informations relatives à chaque étudiant.
- Vérifier pour chaque étudiant que les informations saisies correspondent bien à celles données dans la partie 3.1 du TP.

3.3 Modification des étudiants grâce aux accesseurs de type `set`

Nous allons ici utiliser l'accesseur de type `set` pour ajouter ou modifier une valeur d'attribut d'un étudiant. Un exemple vous est fourni pour l'étudiant `etud1` dans le fichier `MainEtudiant.java`.
Vous devez :

- Modifier l'âge de Jean en mettant la valeur 25.
- Modifier la note de programmation de Zoé qui devient 15.
- Donner à Paul-Henri les notes 16 (programmation), 15 (système), 14 (stage).
- Donner à Jean les notes 8 (programmation), 7 (système), 11 (stage).
- Donner à Adoulkhader les notes 10 (programmation), 14 (système), 11 (stage).
- Donner à Astrid les notes 12 (programmation), 5 (système), 18 (stage).
- Donner à Paolo les notes 2 (programmation), 10 (système), 11 (stage).
- Pour chaque étudiant, vérifier que les modifications ont bien été prises en compte en affichant les informations grâce à l'accesseur `get` et à l'instruction `System.out.println()`

4 Définition des méthodes dans la classe `Etudiant` et utilisation dans `MainEtudiant`

4.1 Méthode `toString`

La méthode `toString` est une méthode prédéfinie dans Java qui renvoie par défaut (si on ne la rédéfinit pas dans la classe) le nom de la classe de l'objet concerné suivi de l'adresse de cet objet. Elle s'utilise en appelant directement l'objet dans un `System.out.println`.

- Tester cette méthode en faisant `System.out.println(etud1)` dans votre classe MainEtudiant.

Il est possible de redéfinir cette méthode afin de donner une description plus détaillée des objets de la classe. Dans ce cas, on fait en sorte qu'elle renvoie une chaîne de caractères contenant les informations permettant de décrire l'objet concerné.

- Dans la classe `Etudiant`, décommenter la méthode `toString` et ajouter les informations entre les "" afin de retourner une chaîne de caractères contenant le nom et l'âge d'un étudiant. Exécutez de nouveau votre programme pour vérifier que la méthode fonctionne correctement.

— Modifier de nouveau la méthode `toString` afin que celle-ci affiche non seulement le nom et l'âge, mais également les notes d'un étudiant.
— Dans `MainEtudiant`, utiliser la méthode `toString` pour afficher les informations de chaque étudiant.

4.2 Méthode de saisie

Nous allons ici reprendre les notions vues dans le TP1 et ajouter une méthode qui permettra de saisir au clavier les informations pour un étudiant.

— Ajoutez à la classe `Etudiant` une méthode `saisie` qui prend un scanner en paramètre et récupère sur ce scanner toutes les informations concernant un étudiant afin d'initialiser celui-ci.
— Dans `MainEtudiant`, ajouter un nouvel étudiant et appeler la méthode `saisie` sur cet étudiant.
— Afficher ensuite les informations de cet étudiant grâce à la méthode `toString()`

4.3 Méthode `moyenne`

— Dans la classe `Etudiant`, décommentez la méthode `moyenne()` et ajouter le code afin que celle-ci retourne la moyenne d'un étudiant.
— En vous inspirant de l'exemple fourni dans `MainEtudiant`, appelez `moyenne()` pour les autres étudiants afin de vérifier son bon fonctionnement.

4.4 Méthode `mention`

— Ajoutez à la classe `Etudiant` une méthode `mention()` qui retourne une chaîne de caractères correspondant à la mention d'un étudiant. Les mentions possibles pour un étudiant sont : `ajourné` si l'étudiant n'a pas obtenu la moyenne, `passable` s'il a obtenu la moyenne mais a moins de 12, `assez bien` s'il a 12 ou plus mais moins de 14, `bien` s'il a 14 ou plus mais moins de 16, `très bien` s'il a 16 ou plus.
— Dans `MainEtudiant`, appelez la méthode `mention` sur vos étudiants pour vérifier son bon fonctionnement.

4.5 Méthode `ligneResultats`

— Ajoutez à la classe `Etudiant` une méthode `ligneResultats` qui retourne une chaîne de caractères d'une ligne précisant le nom, la moyenne et la mention, et, seulement s'il est ajourné, les modules obtenus (c'est-à-dire ceux où il a obtenu la moyenne).
— Dans `MainEtudiant`, appelez `ligneResultats` sur vos étudiants pour vérifier son bon fonctionnement.