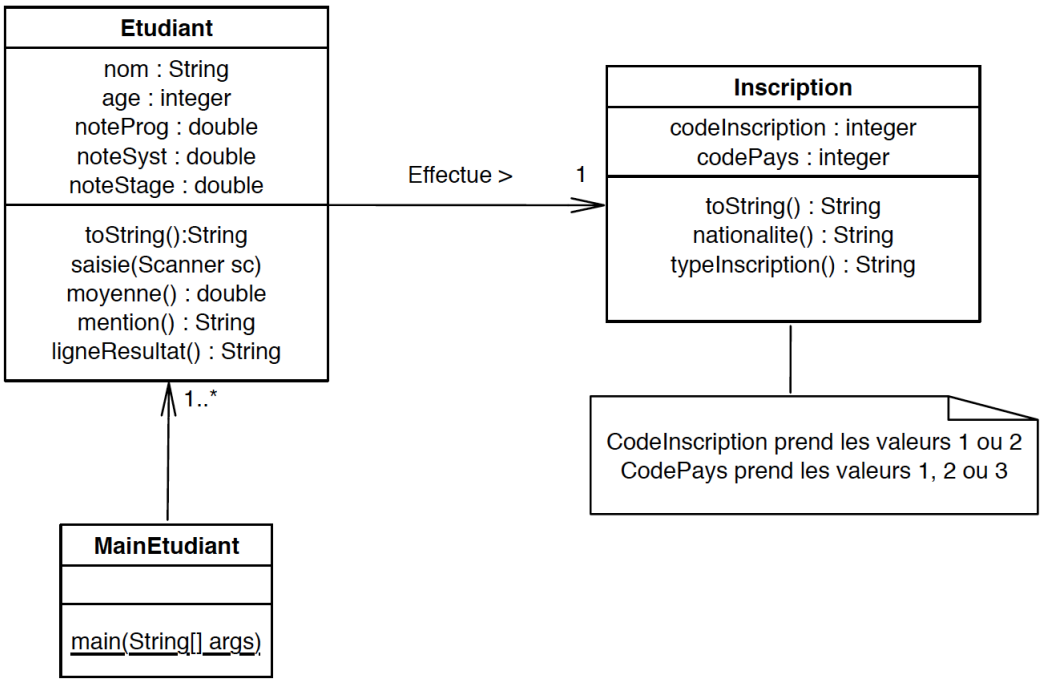
TP2 Etudiant Partie 2: Association 1-1 entre classes et contraintes

1 Structure du programme

Nous remettons le schéma UML qui représente le programme que vous allez mettre en œuvre.



Vous avez normalement défini deux classes dans le paquetage `tp2` :

- Une classe `Etudiant` 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 méthodes.

Nous allons dans ce deuxième TP :

- Créer la classe `Inscription` avec ses attributs, constructeurs, accesseurs et méthodes.
- Mettre en place l'association 1-1 entre la classe `Inscription` et la classe `Etudiant`.
- Ajouter des contraintes afin de vérifier que les valeurs saisies pour les attributs correspondent bien à celles du domaine de valeur. Par exemple, une note doit être comprise entre 0 et 20.

2 Ajout de la classe `Inscription`

Dans cette partie, nous allons ajouter une nouvelle classe, appelée `Inscription`. Pour cela, vous devez :

- Dans Eclipse, créer une nouvelle classe `Inscription.java` appartenant au package `tp2`.
- Ajouter les attributs, les constructeurs et les accesseurs de cette classe en suivant le schéma UML.

— Ajouter une méthode `nationalite()` qui retourne un texte spécifiant si l'étudiant est français (lorsque le codePays vaut 1), étranger francophone (lorsque le codePays vaut 2), ou étranger non francophone (lorsque le codePays vaut 3)
— Ajouter une méthode `typeInscription()` qui renvoie un texte précisant si c'est une première inscription (code Inscription égal à 1) ou une réinscription (code Inscription égal à 2)
— Ajouter une méthode `toString()` permettant d'afficher les informations relatives à l'inscription. Cette méthode utilise les méthodes `nationalite()` et `typeInscription()` précédemment définies.

3 Association 1-1 entre la classe `Inscription` et la classe `Etudiant`

Dans cette partie, nous allons mettre en place l'association 1-1 entre la classe `Inscription` et la classe `Etudiant`. Pour cela, il faut regarder le type et la cardinalité de la relation dans le modèle UML. Dans ce TP, l'association entre l'étudiant et son inscription est unidirectionnelle et se lit dans le sens "un étudiant effectue une et une seule inscription". Nous allons donc insérer une référence à `Inscription` dans la classe `Etudiant`. En pratique, cela revient à ajouter/modifier plusieurs éléments de notre classe `Etudiant` :

3.1 Ajouter un nouvel attribut

Pour mettre en place l'association 1-1 entre la classe `Etudiant` et la classe `Inscription`, vous devez, dans la section attribut de la classe `Etudiant`, ajouter un nouvel attribut `inscription` qui prendra pour type celui de la classe référencée, à savoir `Inscription`.

3.2 Ajouter et/ou modifier les constructeurs

Dans cette partie, nous allons voir comment ajouter ou modifier les constructeurs afin qu'ils prennent en compte l'attribut nouvellement créé. Pour cela, vous devez :

— Ajouter un constructeur qui prend en paramètre le nom, l'âge, le code d'inscription et le code pays, selon le modèle suivant :

```
public Etudiant(String nom, int age, int codeInsc, int codePays) {
    this.nom=nom;
    this.age=age;
    this.inscription = new Inscription(codeInsc, codePays);
}
```

Vous remarquerez que dans cet exemple, l'attribut `inscription` de la classe `Etudiant` est initialisé via un nouvel objet de la classe `Inscription`. On utilise ici le constructeur de la classe **Inscription** permettant de saisir les attributs `codeInsc` et `codePays`.

— En vous inspirant du modèle ci-dessus, ajouter un constructeur qui prend en paramètre des valeurs pour tous les attributs d'un étudiant, augmenté du code d'inscription et du code Pays. Sa signature sera la suivante :

```
public Etudiant(String nom, int age,
                double noteProg, double noteSyst, double noteStage,
                int codeInsc, int codePays) {....}
```

— Ajouter un constructeur qui prend en paramètre des valeurs pour tous les attributs d'un étudiant, y compris l'attribut `inscription` mais pas les attributs codeIns et codePays). Sa signature sera la suivante :

```
public Etudiant(String nom, int age,
                double noteProg, double noteSyst, double noteStage,
                Inscription insc) {....}
```

3.3 Ajouter les accesseurs

— Ajouter les accesseurs `get` et `set` pour ce nouvel attribut `inscription`.

3.4 Modifier les méthodes

— Modifier la méthode `toString` afin de retourner une chaîne de caractères comprenant les informations concernant l'attribut `inscription`. Vous devez utiliser ici la méthode `toString` définie dans la classe `Inscription`, pour que les informations concernant la nationalité et le type d'inscription soient visibles.
— Modifier la méthode `saisie` afin qu'un utilisateur puisse insérer ces nouvelles informations. Pour cela vous devez créer un nouvel objet `inscription` en utilisant le constructeur de `Inscription` avec des paramètres `codeInsc` et `codePays` préalablement saisis par l'utilisateur.

3.5 Test du programme avec `MainEtudiant`

Dans la classe `MainEtudiant`, vous allez vérifier que les nouvelles informations ont bien été prises en compte :

— Créer un nouvel étudiant `etud8` grâce au constructeur que vous avez créé, et lui donner les informations suivantes : Pierre, 24 ans, sans notes connues, ayant pour `codeInscription` 1 et pour `codePays` 2. Afficher ses informations pour vérifier grâce à la méthode `toString` de `Etudiant`.
— Modifier `etud2` (jean), afin de préciser que celui-ci est un étudiant québecois francophone, qui s'inscrit pour la première fois et afficher ses informations.
— Modifier `etud3` (Abdoulkhader), afin de préciser que celui-ci est un étudiant tunisien francophone, qui se réinscrit, et afficher ses informations.
— Modifier `etud4` (Astrid), afin de préciser que celle-ci est une étudiante finlandaise non francophone, qui s'inscrit pour la première fois, et afficher ses informations.
— Modifier `etud5` (Paolo), afin de préciser que celui-ci est un étudiant brésilien non francophone, qui s'inscrit pour la première fois,et afficher ses informations.
— Modifier `etud6` (Zoé), afin de préciser que celle-ci est une étudiante française, qui s'inscrit pour la première fois, et afficher ses informations.
— Créer un nouvel étudiant grâce à la méthode `saisie` (vous lui donnez les valeurs que vous souhaitez).

4 Amélioration des accesseurs et des constructeurs

Nous allons ajouter des contraintes sur les accesseurs `set` afin de restreindre le domaine de valeurs de ceux-ci. Dans les classes correspondantes, modifiez les accesseurs afin d'ajouter les conditions suivantes :

— l'âge doit être compris entre 0 et 140.
— les notes saisies doivent être comprises entre 0 et 20, sinon un message d'erreur est affiché à l'écran.
— le code d'inscription ne peut prendre que les valeurs 1 ou 2, sinon un message d'erreur est affiché à l'écran.
— le code pays ne peut prendre que les valeurs 1, 2 ou 3, sinon un message d'erreur est affiché à l'écran.

Nous allons ensuite modifier les constructeurs afin que la contrainte soit également appliquée en cas d'ajout de nouvel utilisateur :

— En vous inspirant du code ci-dessous, utilisez les accesseurs dans vos constructeurs afin de contrôler la bonne initialisation des objets.

```
public Etudiant(String nom, int age) {
    this.setNom(nom);
    this.setAge(age);
}
```

Enfin, nous allons tester avec trois cas différents pour vérifier le bon fonctionnement des contraintes :

— Dans `MainEtudiant`, testez la possibilité de changer la note de système de Zoé avec 22. Vous devez obtenir un message d'erreur.

— Dans `MainEtudiant`, testez la possibilité de créer un nouvel étudiant avec un code d'inscription égal à 3. Vous devez obtenir un message d'erreur.

— Dans `MainEtudiant`, testez la possibilité de créer un nouvel étudiant avec un code pays égal à 4. Vous devez obtenir un message d'erreur.