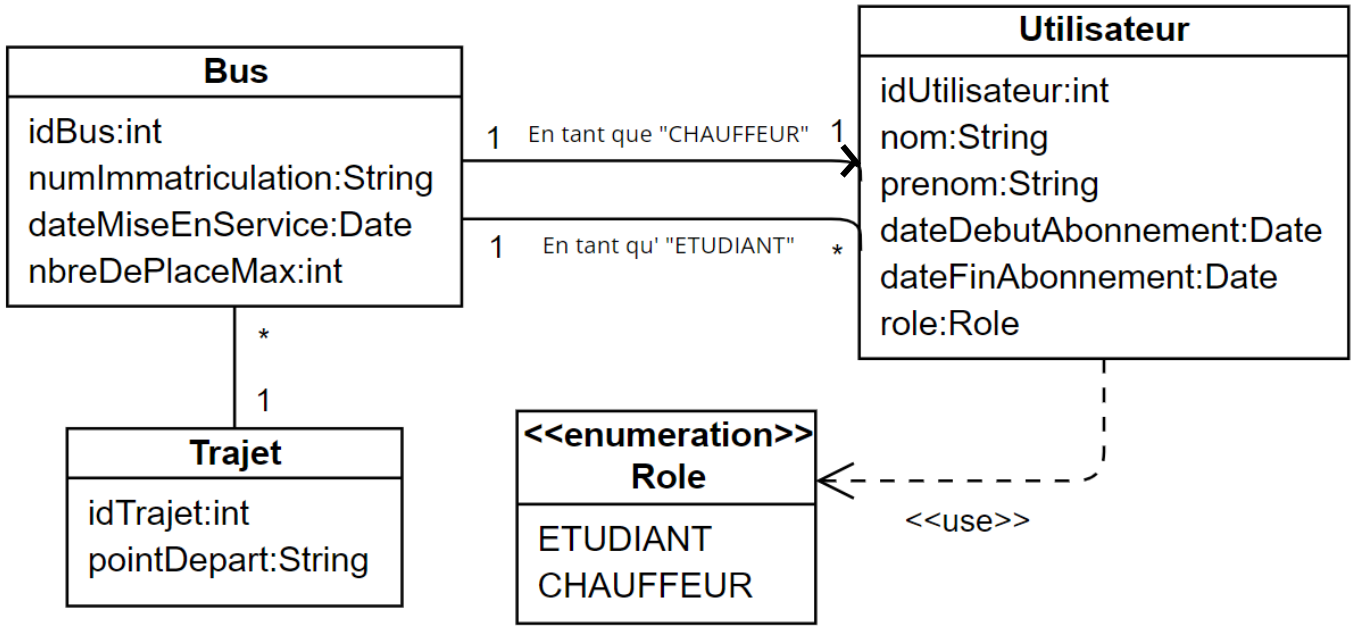
esprit Se former autrement HONORIS UNITED UNIVERSITIES	EXAMEN PRATIQUE Session de rattrapage

Module : **Architecture des SI II (Framework Spring)**

Enseignants : **Équipe Spring**	Classe : **4ème année**	Nombre de pages : **4**
Documents autorisés : **OUI**	Calculatrice autorisée : **NON**	Internet autorisés : **NON**
Date **: 26/06/2023**	Heure **: 11h00**	Durée **: 01h30**

La validation de l'épreuve est appliquée sur la base d'un code source exécutable. Aucun code source non fonctionnel n'est comptabilisé lors de la validation.

On vous propose donc d'implémenter une application simplifiée de gestion des réservations des bus pour le compte d'un établissement scolaire. **Ci-dessous le diagramme de classes :**



I.1. (06 points)

Implémenter les entités qui permettent de générer le schéma de la base de données comme illustré dans le diagramme de classes sachant que :

- Les identifiants sont auto-générés avec la stratégie « **IDENTITY** ».
- L'énumération doit être stockée en tant que chaines de caractères dans la base de données.
- La relation bidirectionnelle **Bus – Trajet** modélise le fait qu'un bus emprunte un seul trajet et qu'un trajet concerne plusieurs bus.

- La **première** relation unidirectionnelle **Utilisateur – Bus** modélise le fait qu'un **chauffeur** peut être affecté à un seul **bus** et qu'un **bus** peut être conduit par un seul **chauffeur**.
- La **deuxième** relation bidirectionnelle **Utilisateur – Bus** modélise le fait qu'un **étudiant** peut être affecté à un seul **bus** et qu'un **bus** peut être affecté à plusieurs **étudiants**.

I.2. (14 points)

Développer le code nécessaire dans une classe annotée par @RestController qui fait appel aux différents services. **(Exposition des services avec Spring REST MVC et Tests avec Postman ou Swagger)**. Voici les Services demandés :

A) En respectant la signature de la méthode suivante, ajouter les deux trajets ci-dessous (1.5 pt) :

```
Trajet ajouterTrajet(Trajet trajet);
```

Trajet
pointDepart
Tunis
Ariana

B) En respectant la signature de la méthode suivante, ajouter les étudiants ci-dessous (1.5 pts) :

```
Utilisateur ajouterEtudiant(Utilisateur etudiant);
```

Utilisateur				
nom	prenom	role	dateDebutAbonnement	dateFinAbonnement
Fedi	Ahmed	ETUDIANT	2022-07-02	2023-07-02
Ben Ali	Ridha		2022-08-04	2023-08-04
Hmem	Sonia		2022-06-26	2023-06-26
Saleh	Oumayma		2022-06-26	2023-06-26

C) En respectant la signature de la méthode ci-dessous, ajouter les bus et leurs chauffeurs correspondants en même temps (Cascade) suivants (2 pts) :

```
Bus ajouterBusEtChauffeur(Bus bus);
```

Bus			**Utilisateur**	
numImmatriculation	dateMiseEnService	nbreDePlaceMax	nom	prenom
2222TUN140	2009-03-09	2	Saidi	Mahmoud
5454TUN131	2008-02-06	2	Fettah	Abbes

NB : Pour l'ajout du chauffeur, vous devez saisir uniquement les champs « nom » et « prenom » et affecter le champs « role » à « CHAUFFEUR » automatiquement dans le code.

D) En respectant la signature de la méthode suivante, affecter les trajets aux bus comme montre le tableau ci-dessous (2 pts) :

```
Bus affecterTrajetABus(int idBus, int idTrajet);
```

Trajet	**Bus**
pointDepart	numImmatriculation
Tunis	2222TUN140
Ariana	5454TUN131

E) En respectant la signature de la méthode suivante, affecter les étudiants aux différents bus comme montre le tableau ci-dessous (2 pts) :

```
String affecterEtudiantABus(String numImma, String nom, String prenom);
```

NB :

- S'il y a assez de places dans un bus, l'affectation sera effectuée et la méthode retourne **« L'affectation de l'étudiant est effectuée avec succès »**.
- S'il n'y a pas assez de places dans un bus, la méthode retourne « **Le bus est complet** ».

Bus	**Utilisateur**	
numImmatriculation	nom	prenom
2222TUN140	Ben Ali	Ridha
	Saleh	Oumayma
5454TUN131	Fedi	Ahmed
	Hmem	Sonia

F) Créer un Advice qui permet d'afficher « C'est une méthode d'ajout » après la bonne exécution des méthodes d'ajout (commençant par ajouter) et « C'est une méthode d'affectation » après la bonne exécution des méthodes d'affectation (commençant par affecter) de la couche « Service » (1 pt).

G) En respectant la signature de la méthode suivante, afficher la liste des étudiants d'un trajet qui ont un abonnement valide entre les deux dates passées en paramètre (2 pts) :

```
List<Utilisateur> afficherEtudiantsAvecTrajet(int idTrajet, Date
                  dateInf, Date dateSup);
```

H) En respectant la signature suivante et en utilisant **Spring Scheduler**, implémenter une méthode planifiée exécutée chaque 60 secondes qui arrête automatiquement les abonnements des étudiants dont la date de fin de l'abonnement est inférieure ou égale à la date courante (date système). Cette méthode affiche sous forme de message sur la console (en utilisant un logger) le nom et le prénom des étudiants dont leurs abonnements sont annulés par ce service (2 pts).

NB :

- L'annulation de l'abonnement se fait comme suit : **désaffecter** l'étudiant du bus affecté et mettre la date début et la date fin de l'abonnement **null**.

```
void arreterAbonnement();
```

Bon courage ☺