EXAMEN

Semestre : 1 2 [X]

Session : **Principale** [X] Rattrapage [ ]

Module :..... **Architecture des SI II** .....
Enseignant(s) :..... **Spring Team** .....
Classe(s) :..... .....

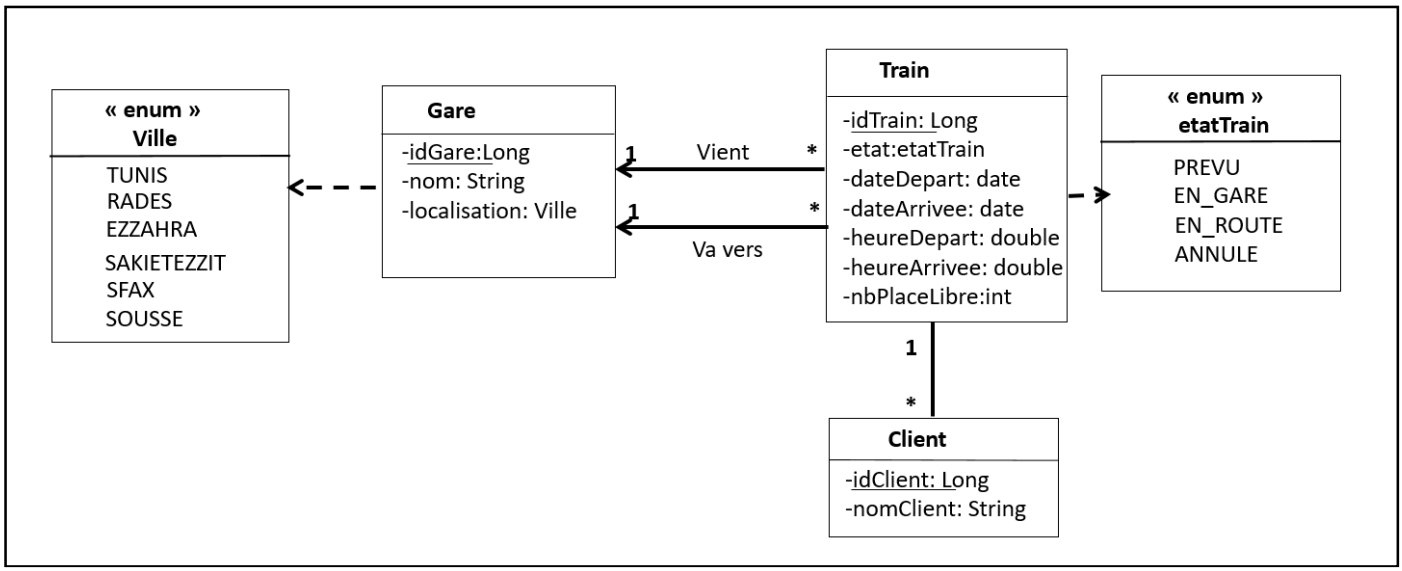
Documents autorisés : **OUI** [X] NON [ ] Nombre de pages : **3**

Calculatrice autorisée : OUI [ ] **NON** [X] Internet autorisée : OUI [ ] **NON** [X]

Date : ……..….. **Heure .....** **Durée : 1h30.....**

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 se propose de mettre en place une application de gestion des trains de la société nationale des chemins de fer. Ci-dessous, le diagramme de classes.



I.1 (5 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'association **unidirectionnelle** Train-Gare indique qu'un train « vient » d'arriver d'une gare et a comme destination « va vers » une autre gare.

- L'association **bidirectionnelle** Client-Train indique qu'un client peut rejoindre un seul train et qu'un train peut être accueillir plusieurs clients à une date donnée.
- L'**énumération** doit être stockée en tant que chaîne de caractères dans la base.

I.2 (15 points)

Pour chacune des questions suivantes, développer le code nécessaire dans une classe annotée par **@RestController** qui fait appel aux différents services.

N.B : Chaque question doit être testée sur Postman ou Swagger et l'affichage sur console doit être fait avec Logging. Afficher un message à l'entrée de chaque service.

a) Ajouter **5 gares** ayant les détails ci-dessous en respectant la signature suivante **(/1.5)** :

```
public void ajouterGare(Gare g);
```

nom	localisation
TUNIS	TUNIS
RADES LYCEE	RADES
EZZAHRA LYCEE	EZZAHRA
SAKIET EZZIT	SFAX
SOUSSE	SOUSSE

b) Ajouter **4 trains** ayant les détails ci-dessous en respectant la signature suivante **(/1.5)** :

```
public void ajouterTrain(Train train);
```

Train	dateDepart	dateArrivee	heureDepart	heureArrivee	nbPlaceLibre	etat
1	20-03-2022	20-03-2022	7.45	8.30	1	EN_GARE
2	21-04-2022	21-04-2022	6.15	9.00	20	EN_GARE
3	20-03-2022	20-03-2022	7.00	7.45	14	EN_GARE
4	21-04-2022	21-04-2022	9.00	10.00	20	EN_GARE

c) Affecter les trains aux gares suivants tout en s'assurant que la gare de départ est différente de la gare destination en respectant la signature suivante **(/1.5)** :

```
public void affecterTrainAGare(Long idTrain, Long idGareDepart, Long
idGareArrivee);
```

Train	Nom GareDepart	Nom GareArrivee
1	EZZAHRA LYCEE	RADES LYCEE
2	TUNIS	SOUSSE
3	TUNIS	RADES LYCEE
4	SOUSSE	SAKIET EZZIT

d) Ajouter les 2 clients ayant les détails ci-dessous en respectant la signature suivante **(/1)** :

```
public void ajouterClient(Client c);
```

nomClient
Ali
Mohamed

e) Affecter un train (départ : de EZZAHRA LYCEE à 07h45, arrivée : RADES LYCEE), pour lequel on a des places disponibles, **aux deux clients**. Après affectation, mettre à jour les places disponibles sinon afficher un message de non disponibilité (si la capacité est atteinte).

Utiliser la signature suivante **(/2)**.

```
public void affecterTainAClient(Long idClient, Long idGareDepart);
```

f) Afficher **en JPQL** le nombre **moyen** de places libres dans tous les trains qui ont comme Gare de départ « TUNIS » tout en respectant la signature suivante **(/1.5)**.

```
public int TrainPlacesLibres(Long idGareDepart);
```

g) Nous souhaitons aller à « SAKIET EZZIT » à partir de TUNIS mais il n'y a pas de train direct sur cet itinéraire. Lister tous les trains indirects à avoir pour arriver à « SAKIET EZZIT » tout en respectant la signature suivante **(/1.5)**. Eviter d'avoir **une boucle d'affichage sur Postman/Swagger**.

```
public List<Train> ListerTrainsIndirects(Long idGareDepart, Long
idGareArrivee);
```

h) Le train (départ : de EZZAHRA LYCEE à 07h45), est bien arrivé. Désaffecter **le/les** clients et mettre à jour le nombre de places disponibles et l'état à «PREVU» en respectant la signature suivante **(/1.5)**.

```
public void DesaffecterClientsTrain(Long idGareDepart,double heureDepart);
```

i) Créer **un aspect** qui permet de calculer et afficher dans les logs la durée d'exécution de chaque méthode qui a comme type de retour « int » **(/1)**.

j) Nous souhaitons créer un service programmé automatiquement (scheduled) permettant d'afficher tous les trains dont la date d'arrivée est inférieure à la date système et mettre leurs états à «En GARE».
Créer le service qui permet d'afficher les trains concernés et faire la mise à jours de leurs états toutes les 30 secondes en respectant la signature suivante **(2pts)** :

```
public void TrainsEnGare()
```

Indication : La méthode before() de la classe Java Date teste si la date est antérieure ou non à la date spécifiée.

Bon travail.