EXAMEN BLANC

Semestre : Session : Principale

Module : Architecture des SI II(Spring)

Enseignant (s) : Équipe Spring

Documents autorisés : OUI
Calculatrice autorisée : NON

Nombre de pages : 4 pages

Internet autorisées : NON

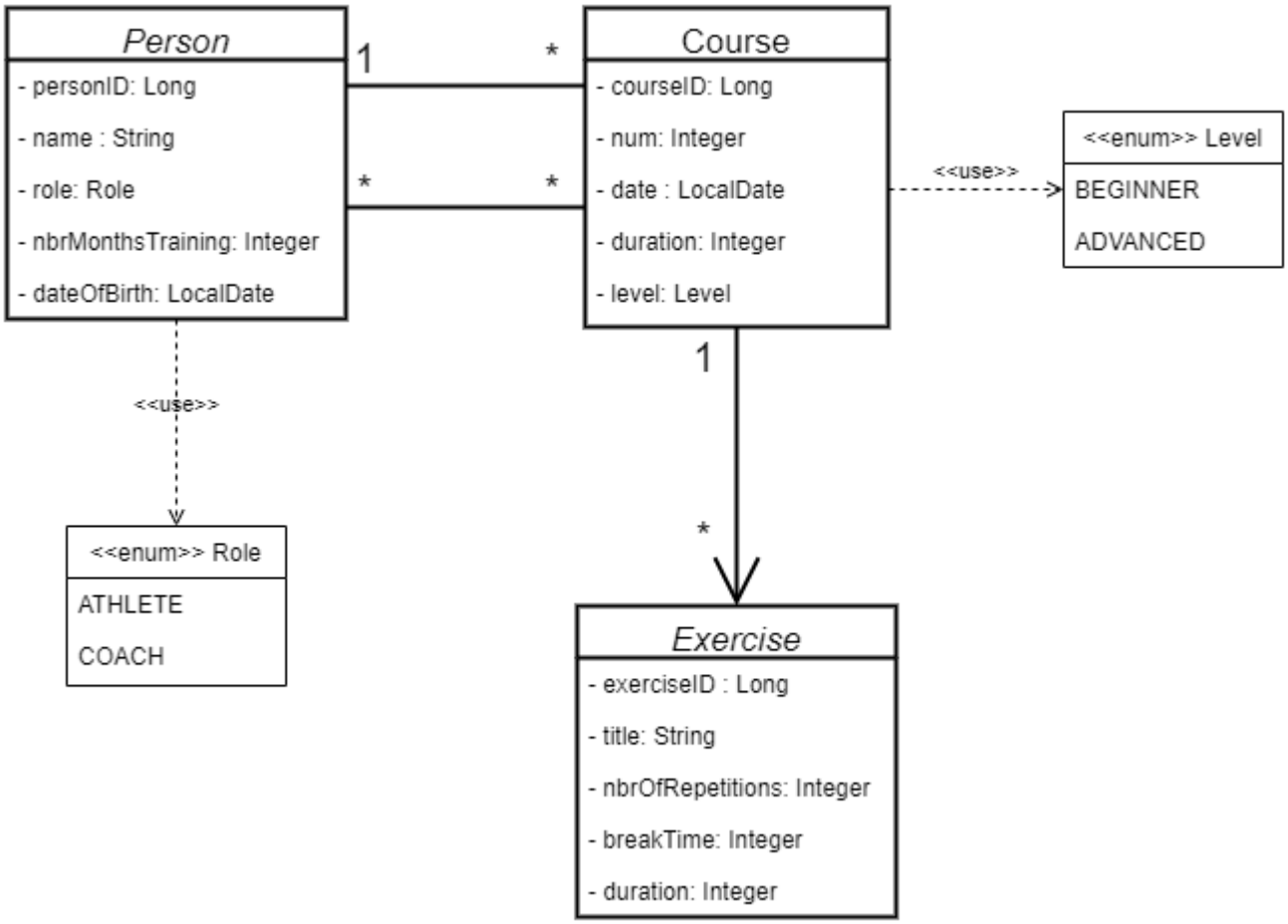
Durée : 01h30

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 d'implémenter une application simplifiée dédiée à la gestion du coaching privé en ligne.

Ci-dessous le diagramme de classes :



- ❖ Chaque séance du cours est composée de plusieurs exercices, et un exercice particulier ne peut être attribué qu'à un seul cours.
- ❖ Une personne peut jouer le rôle de coach ou d'athlète.
- ❖ Chaque séance du cours est dirigée par un seul coach, qui a la possibilité d'animer plusieurs cours. De plus, chaque athlète a la flexibilité de participer à plusieurs séances du cours.

I. Entités/associations (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 des entités sont auto-générés avec la stratégie « **IDENTITY** ».
- Dans l'association bidirectionnelle **Person-Course**, l'entité Person est le **Child**.
- Les énumérations doivent être stockées en tant que **chaînes de caractères** dans la base de données.

II. Services (15 points) :

Élaborez le code requis au sein d'une classe annotée par @RestController, sollicitant les divers services. Exposez ces services grâce à Spring REST MVC, et effectuez des tests avec Postman ou Swagger.

Les services requis sont les suivants :

1. En respectant la signature de la méthode suivante, ajouter les personnes mentionnées ci-dessous (1pt) :

public Person addPerson(Person person)

Person			
name	**role**	**nbrMonthsTraining**	**dateOfBirth**
Yasmine	COACH	-	1993-07-22
Ines	ATHLETE	13 mois	1991-11-05
Majd	ATHLETE	18 mois	2000-04-12
Firas	ATHLETE	5 mois	2005-10-16

2. Implémenter une méthode permettant d'ajouter **à la fois (en cascade)** les cours avec leurs exercices, tout en les attribuant à **coach Yasmine**, en respectant la signature suivante (2pts) :

public Course addCourseAssignCoach(Course course, Long personID)

NB : Lors de l'ajout d'un cours, il faut créer en même temps ses exercices et vérifier le rôle de la personne, qui doit être un coach.

Course				Exercise			
num	date	duration	level	title	nbrOfRepetitions	breakTime	duration
1	2024-06-01	0	BEGINNER	Abdos	4	2 min	15 min
2	2024-06-01	0	ADVANCED	Mollets	2	1 min	10 min
				Abdos	3	1 min	15 min
3	2023-11-20	0	ADVANCED	Dorsaux	5	2 min	10 min

3. En utilisant **Spring AOP**, créer un **Aspect** qui permet d'afficher le message suivant :

" Fin méthode « nom de la méthode » avec succès "

Cet aspect s'exécute **après la bonne exécution** des méthodes d'**add** de la couche service (1.5 pt).

4. Affichez les séances du cours **passées** dirigées par le **coach Yasmine**, avec un niveau de difficulté **ADVANCED**, en utilisant les **KEYWORD** et en respectant la signature suivante (2 pts) :

public List<Course> retrieveCourse(Level level , String name)

5. Développez une méthode qui assigne les athlètes aux séances du cours spécifiées, tout en prenant en compte le fait qu'un nouvel athlète ne peut pas être ajouté à une séance qui a déjà eu lieu.

Il est important de noter que l'athlète peut participer à une séance du cours de niveau avancée uniquement s'il a accumulé plus d'un an d'expérience d'entraînement.

Tout en respectant la signature suivante (2pts) :

public Course assignAthleteCourse(Long personID, Integer num)

Person	Course
name	**num**
Ines	2
Majd	2
Firas	1

6. Afficher la liste des athlètes participant à un exercice spécialisé **d'Abdos**, dont **l'âge est inférieur à 20 ans**, en respectant la signature suivante (2 pts) :

public List<Person> retrieveAthlete(String title)

7. Proposer une méthode visant à afficher les exercices les plus rigide durant la période du “2023-01-01” au “2023-12-31” en utilisant **JPQL**. Un exercice est considéré comme rigide s'il répond aux critères suivants :
    - Il appartient à une séance du cours classée avec un niveau de difficulté "ADVANCED".
    - Il nécessite un temps de repos égal ou supérieur à 2 minutes.

   La méthode doit respecter la signature de suivante (2.5 pts) :

   public List<Exercise> retrieveRigidEx(LocalDate startDate, LocalDate endDate)

8. Proposez une méthode respectant la signature ci-dessous, qui met à jour la durée totale d'un cours toutes les 20 secondes. La durée se calcule en respectant la formule suivante (2pts) :

public void updateNbRept()

Durée cours = Σ((durée exercice + temps repos) × nombre répétition)

Bon courage 😊