EXAMEN

Semestre : 1 [ ] 2 [X]

Session : **Principale** [X]

Module : **Architecture SI II**
Enseignant(s) : **Equipe Spring**
Classe(s) : **4INFINI / 4DS**

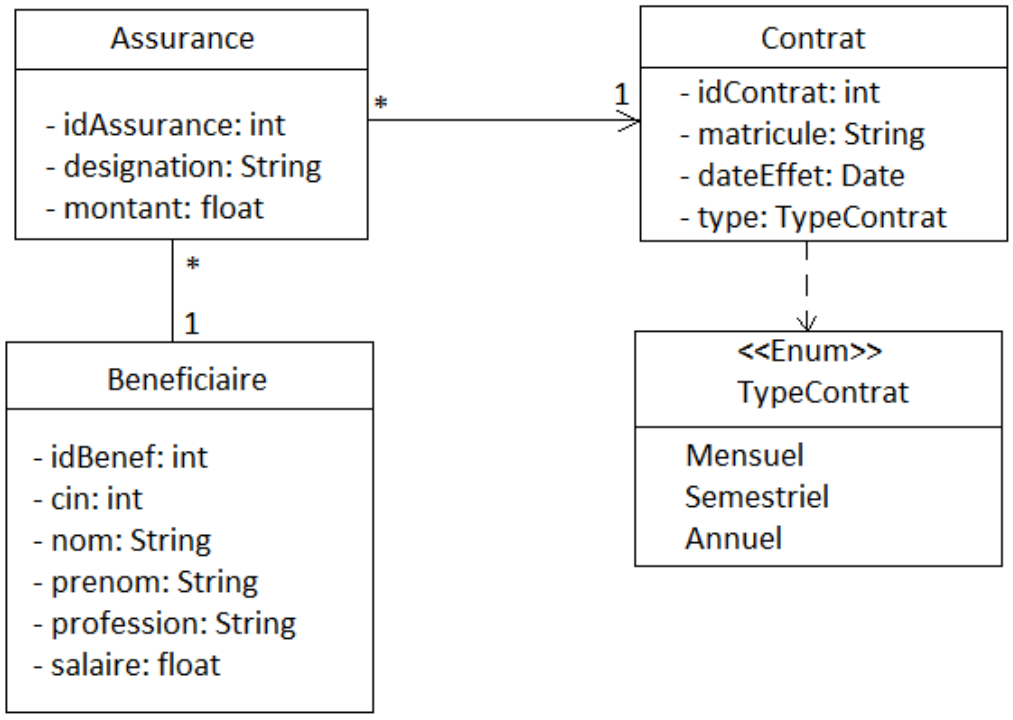
Documents autorisés : **OUI** [X] NON [ ] Nombre de pages : 3

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

Date : 19/05/2022 **Heure 9h00** **Durée : 1h30**

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

On vous propose d'implémenter une application simplifiée de gestion d'une assurance en ligne, suivant le diagramme de classes ci-dessous :



Partie I (5 points) :

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

- Notre base de données est MySQL. Les identifiants sont auto-générés par la stratégie IDENTITY.
- L'association bidirectionnelle Beneficiaire-Assurance indique qu'un bénéficiaire a plusieurs assurances et qu'une assurance ne peut appartenir qu'à un seul bénéficiaire.
- L'association unidirectionnelle Assurance-Contrat indique que plusieurs assurances peuvent être affectées à un seul contrat.
- L'énumération doit être stockée en tant que chaîne de caractères dans la base de données.

Partie II (15 points) :

- Développer les services nécessaires dans des beans Spring @Service, et exposer les entant que Web Services dans des beans de type @RestController.
- Vous pouvez tester les méthodes à travers **Swagger ou Postman**.

1) Ajouter 2 bénéficiaires suivant les détails ci-dessous en respectant la signature suivante :

public Beneficiaire ajouterBeneficiaire (Beneficiaire bf) (1.5 pts)

cin	nom	prenom	profession	salaire
654321	Sahli	Ahmed	Enseignant	1500
123456	Trablsi	Sarra	Directrice	3000

2) Ajouter 3 contrats suivant les détails ci-dessous :

matricule	dateEffet	type
V0945	2020-05-01	Semestriel
M0102	2018-09-01	Mensuel
V0734	2022-01-07	Annuel

En respectant la signature suivante :

public Contrat ajouterContrat (Contrat c) (1 pt)

3) Ajouter 3 assurances en les affectant aux contrats et aux bénéficiaires correspondants en respectant la signature suivante :

public Assurance ajouterAssurance (Assurance a, int cinBf, String matricule) (2 pts)

Assurance		Contrat	Beneficiaire
designation	**montant**	**matricule**	**cin**
Assurance vie	110	V0945	654321
Assurance materiel	50	M0102	123456
Assurance vie	200	V0734	123456

4) Afficher le contrat le plus ancien d'un bénéficiaire donné en respectant la signature suivante :
public Contrat getContratBf (int idBf) (2 pts)

5) Lister tous les bénéficiaires selon un type de contrat donné.
La signature de la méthode est la suivante :

public Set<Beneficiaire> getBeneficairesByType (TypeContrat typeContrat) (2 pts)

6) Afficher le montant annuel des assurances par bénéficiaire, en respectant la signature suivante :

public float getMontantBf (int cinBf) (2.5 pts)

N.B: Vous devez faire le calcul nécessaire pour les types de contact semestriel et mensuel : (semestriel x 2, mensuel x 12)

7) En utilisant **SpringScheduler**, proposer une méthode qui se déclenche toutes les 60 secondes et qui affiche le nombre des assurances pour chaque bénéficiaire, ordonné par ordre décroissant, en respectant la signature suivante :

public void statistiques ( ) (2.5pts)

Indication :
Vous pouvez utiliser une Map ordonnée (TreeMap<K, V>) K : nombre des assurances
V : cin du bénéficiaire correspondant

Pour inverser l'ordre d'une map, vous pouvez utiliser la méthode statique reverseOrder() de la classe utilitaire Collections dans le constructeur de TreeMap:

Déclaration d'une TreeMap :

```
TreeMap<Integer, Integer> myStat = newTreeMap<>(Collections.reverseOrder())
```

Parcours d'une TreeMap :

```
for (Entry<Integer, Integer> entry : myStat.entrySet()) {}
```

8) Créer un aspect permettant d'afficher le message « Bon courage ! » à la fin de l'exécution de chaque méthode du package services et qui commence par ***get***. **(1.5 pts)**

☺ Bon courage ☺