EXAMEN SESSION PRINCIPALE

Semestre : 1

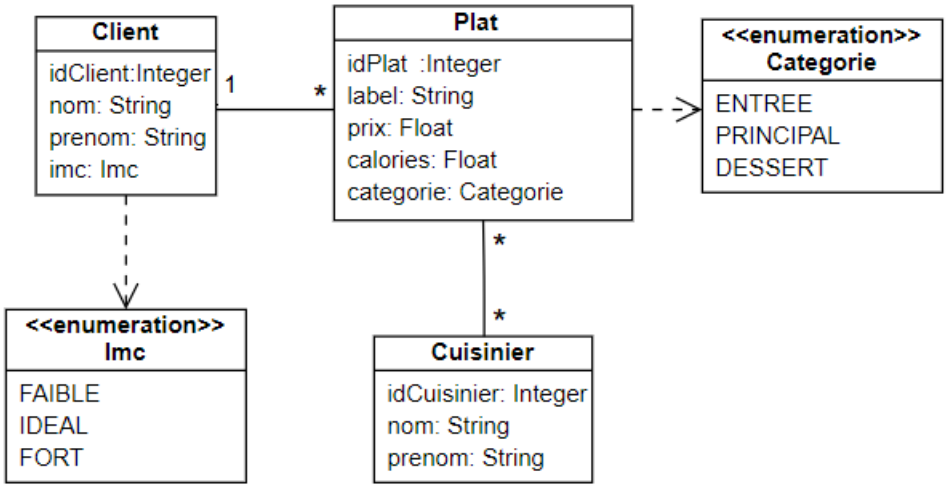
Module : Architecture des Systèmes d'information II — Enseignants : Equipe Spring

Classe(s) : **GAMIX,SE,SLEAM,WIN** — Documents autorisés : OUI — Internet autorisée: NON

Date : 05/01/2023 Heure : 09H00 Durée : 1H30 minutes — Nombre de pages : 3

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 de gestion d'un Traiteur.
L'application aura **le diagramme des classes suivant :**



Partie I (4.5 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 de toutes les entités sont auto-générés avec la stratégie « **IDENTITY** »
Les relations bidirectionnelles :

- La relation bidirectionnelle Plat-Cuisinier indique qu'un cuisinier peut préparer plusieurs plats et qu'un plat peut être préparé par plusieurs cuisiniers (Cuisinier est le fils)

- Plat-Client indique qu'un plat est servi pour un seul client donné el le client peut commander plusieurs plats

Les énumérations doivent être stockées en tant que chaines de caractères dans la base de données (type d'énumération « String »).

Partie II (15.5 points) :

Développer le code nécessaire dans une classe annotée par **@RestController** qui fait appel aux différents services. (Exposition des services avec Spring REST MVC).

Toutes les méthodes seront testées à travers **Swagger** ou bien **Postman**.

a) Ajouter 1 Client ayant les détails ci-dessous en respectant la signature suivante **(1pt)** :

public Client ajouterClient(Client client)

nom	prenom	imc
Paga	Neuron	IDEAL

b) En utilisant la signature suivante, ajouter les trois cuisiniers ci-dessous **(1pt) :**

public void ajouterCuisinier (Cuisinier cuisinier)

nom	prenom
Julien	Tanti
Benjamin	Samat
Laura	Lempika

c) Ajouter 4 plats avec les détails ci-dessous et les affecter chacun à son propre cuisinier (**Entrée pour Julien / 2 Plats Principal pour Benjamin / Dessert pour Laura**) et au client Paga Neuron sachant que le client ne peut pas commander plus que deux plats de type Principal, en respectant la signature suivante **(2.5pts)** :

public void ajouterPlatAffecterClientEtCuisinier (Plat plat, Integer idClient, Integer idCuisinier)

label	categorie	prix	calories	Cuisinier
Foie Gras	ENTREE	7.5	470	**Julien Tanti**
LumLum Crevettes	PRINCIPAL	42	850	**Benjamin Samat**
Gratin dauphinois	PRINCIPAL	42	1980	**Benjamin Samat**
Tiramisu	DESSERT	9.5	220	**Laura Lempika**

d) En utilisant la signature suivante, afficher la liste des plats du Client Paga Neuron (**2pts**).

List <Plat> AfficherListePlatsParClient (String nom, String prenom)

e) Calculer le montant à payer par le client Paga Neuron**,** en respectant la signature suivante **(2.5pts)**:

public float MontantApayerParClient(Integer idClient)

f) Créer un service qui permet d'ajuster l'imc (**Indice de masse corporelle**) du client Paga Neuron tout en basant sur le nombre de calories déjà pris en respectant la signature suivante**(3pts)** :

public void ModifierImc(Integer idclient)

NB:

Total Calories plats = **Σ** nbCalories par plat < 2000 ---> IMC=Faible
Total Calories plats = **Σ** nbCalories par plat = 2000 ---> IMC=Ideal
Total Calories plats = **Σ** nbCalories par plat > 2000 ---> IMC=Fort

g) Créer un service planifié automatiquement toutes les 15 secondes qui permet **d'afficher la liste des cuisiniers ayant préparé au minimum deux plats de catégorie Principal** en respectant la signature suivante **(2pts)** :

public void AfficherListeCuisinier()

h) Créer un Aspect qui permet d'afficher le message "**montant facture calculé**" après le bon déroulement de la méthode MontantApayerParClient au sein de la couche service. **(1.5pt)**

Bon Courage