M1 : Ingénierie du Logiciel - UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen Réparti 2eme partie

17 Janvier 2018 (2 heures avec documents : tous SAUF ANNALES CORRIGEES). Barème indicatif sur 22 points (donne le poids relatif des questions) (max 20/20).

Questions de cours [3,5 Pts]

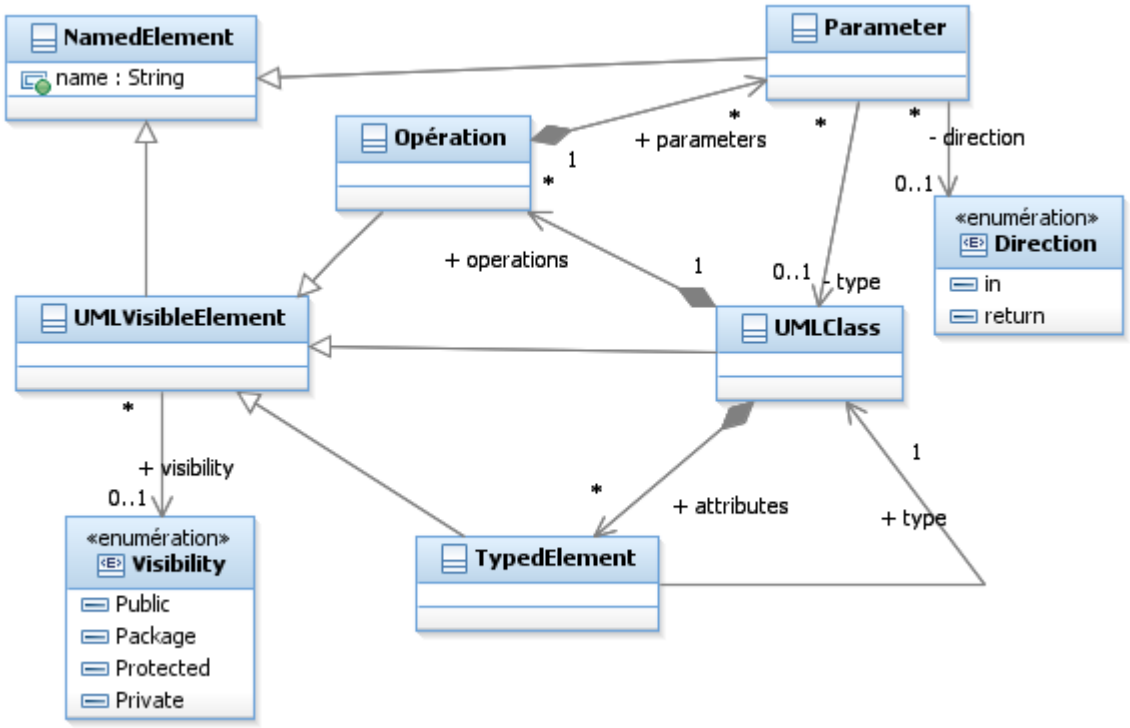
Répondez de façon précise et concise aux questions.

Question Cours (QC):

QC1) (1 point) On suppose que l'on a défini le métamodèle d'un nouveau langage. Quels outils peuvent utiliser ce métamodèle comme entrée, et dans quel but ? Citez deux exemples.

QC2) (1 point) Expliquez pourquoi dans un test dans le cas général on ne peut se limiter à une invocation isolée, mais qu'on a recours à une **séquence** d'invocations ?

QC3) (1,5 point) Soit le méta-modèle simplifié du diagramme de classe UML tel que présenté en TD.



D'après ce méta-modèle, est-il possible de modéliser en UML

1. L'héritage entre deux classes
2. L'héritage multiple (une classe a plusieurs parents)
3. Une opération avec plusieurs paramètres de retour
4. Du polymorphisme : une classe avec deux opérations de même nom mais de signature différente
5. Les attributs de type « simple » comme en Java : int, char…

2. Problème: Conception eServer [Barème sur 18,5 Pts]

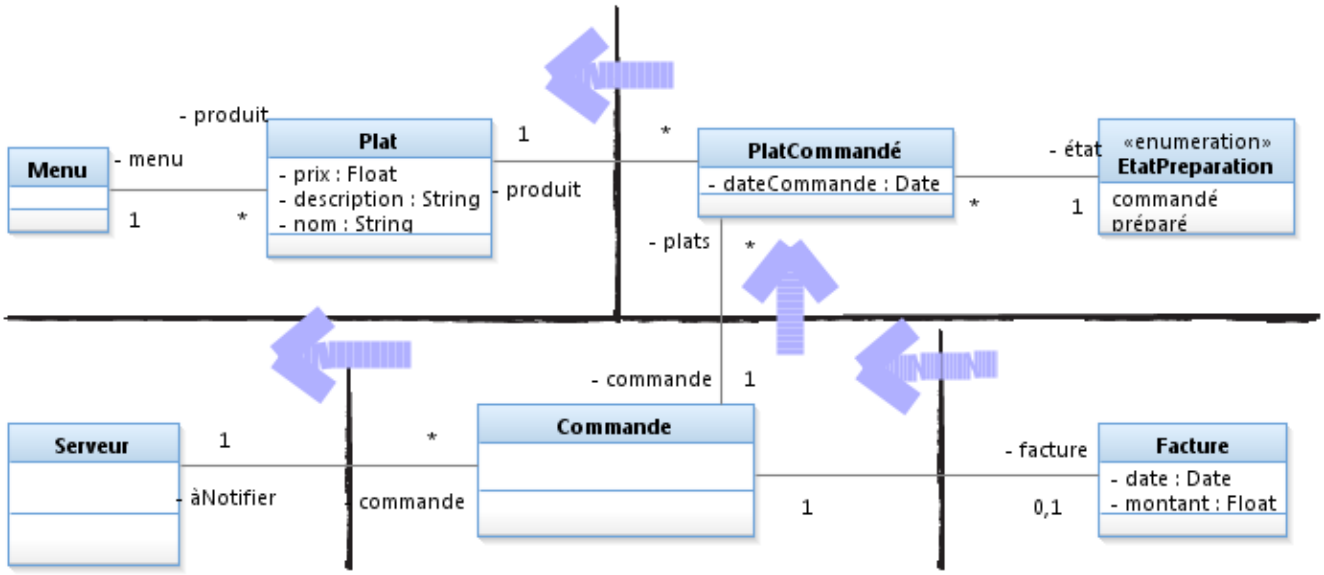
Rappels d'analyse : Cet énoncé se base sur une version simplifiée de l'application eServer dont on a fait l'analyse en Novembre. Nous considérons l'application eServer, permettant de gérer un restaurant.

- Les serveurs sont équipés d'un smartphone ; ils saisissent dessus les plats commandés par les clients (qui sont automatiquement transmis aux cuisines) tout au long du service. Les serveurs sont notifiés quand les plats commandés sont prêts.
- Les cuisines sont équipées d'une interface tactile simple où les plats commandés par les clients peuvent être visualisées par ordre chronologique. Le cuisinier prépare les plats, puis peut signaler quand les plats sont prêts à l'application, ce qui notifie le serveur concerné (vibration du téléphone).

Le menu du restaurant est composé de plats. Chaque plat est qualifié par

- Son nom e.g. « Crêpe Grand-Père »
- Sa description, e.g. « chocolat noir, amandes »
- Son prix, e.g. 6 eu

On a obtenu le diagramme des classes métier suivant, qu'on se propose de découper comme indiqué sur la figure pour obtenir des composants.



I. *Composant CMenu (4 points)*

Ce composant bout de chaîne stocke les informations relatives aux plats que l'on sert dans le restaurant.

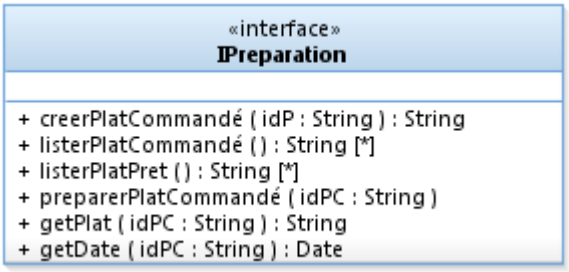
1. (1 point) Définir une interface *IMenu* (opérations et signature) que *CMenu* offre. On demande de **fournir une seule interface**, qui fonctionne avec des **identifiants** et qui permette :
   - De créer des plats
   - De lister les plats existants
   - De lister les détails d'un plat donné à partir de son nom (supposé unique) : description et prix
   - De supprimer un plat du menu

On ne demande donc pas d'API en modification (il faut supprimer et recréer), et les noms sont confondus avec les identifiants des plats.

2. (0,5 point) Sur un diagramme de composant, représenter *CMenu* (interfaces requises et offertes).
3. (1,5 point) Sur un diagramme de séquence contenant une ligne de vie pour une IHM et une ligne de vie pour une instance de **CMenu** initialement vide, modélisez la séquence où l'on crée un plat « kebab » de description « sandwich » et de prix 5eu. On doit ensuite modéliser un affichage par l'IHM de tous les plats du menu, avec tous les détails (prix et description).
4. (1 point) Sur un diagramme de classes, donner une conception détaillée possible de ce composant *CMenu*.

II. CPréparation, la dynamique des Plats (2 points)

Le composant **CPreparation** est responsable de la gestion des plats commandés et de leur état. On s'est limité à deux états possibles (commandé ou préparé) dans cet énoncé. Un plat nouvellement ajouté par le serveur est à l'état « commandé », puis il bascule à l'état « préparé » quand le cuisinier notifie l'application. On propose l'interface **IPréparation** pour encapsuler ce comportement :



Les paramètres nommés *idP* sont des identifiants de Plats, et permettent d'interagir avec **CMenu** défini en question I. La date de la commande est obtenue en interrogeant l'horloge système. Les paramètres *idPC* sont des identifiants de plats commandés, i.e. avec un état particulier et une date de commande. Les opérations lister et créer rendent des *idPC*.

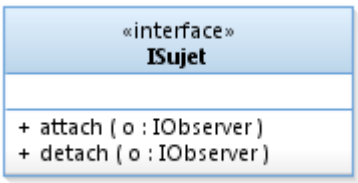
1. (0,5 point) Sur un diagramme de composant, représenter *CPréparation* (interfaces requises et offertes).
2. (1,5 point) Sur un diagramme de classes, donner une conception détaillée possible de ce composant *CPréparation*.

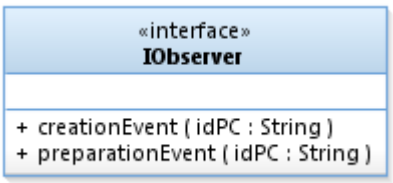
III. Une décoration observable (6,5 points)

eServer nécessite la mise en place d'un système de notifications, pour que :

- Les cuisines soient notifiées des nouvelles commandes de plat
- Les serveurs soient notifiés quand les plats sont prêts.

Pour répondre à ce besoin, on introduit deux nouvelles interfaces, qui matérialisent le DP Observer.





Le *sujet* est une source d'événements asynchrones. L'*observer* est intéressé par les mises à jour du *sujet*. On commence par abonner l'observer au sujet (avec *attach).* Puis quand le sujet est mis à jour, les observer(s) seront notifiés de la nature de l'événement qui s'est produit. On ne considère ici que les deux événements qui nous intéressent : un plat a été crée (*creationEvent) ou* un plat a fini d'être préparé (*préparationEvent)*. L'observateur est notifié de la nature de l'événement et de l'identifiant du plat commandé concerné par l'événement.

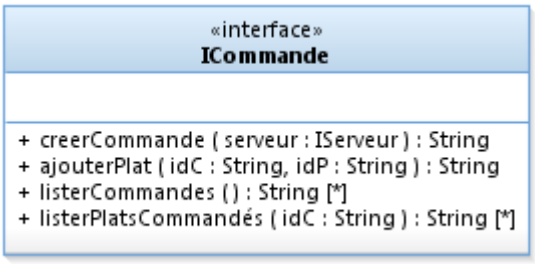
On souhaite définir un composant **CPréparationObservable** qui offre **IPreparation** et qui soit observable. Ce composant s'appuie par délégation sur une instance de **CPreparation** pour réaliser les opérations déclarées dans **IPreparation**. Les opérations de modification de **IPréparation** seront décorées de manière à réaliser le traitement par délégation, puis notifier tous les observateurs abonnés de la mise à jour.

1. (1 point) Sur un diagramme de composant, représenter **CPréparationObservable** (interfaces requises et offertes).
2. (1,5 point) Sur un diagramme de classes, donner une conception détaillée possible de ce composant *CPréparationObservable*.
3. (3 points) On introduit le composant **CCuisine**, qui représente l'interface tactile au sein des cuisines. Ce composant souhaite recevoir les notifications concernant les nouveaux plats commandés. Quand il y a un nouveau plat commandé, **CCuisine** doit afficher le nom du plat commandé (e.g. « kebab ») et sa description (e.g. «sandwich »).
   Sur un diagramme de séquence contenant une instance de chacun des composants **CMenu, CIHM, CPreparation, CPreparationObservable, CCuisine** modélisez une séquence où le composant d'IHM crée une commande pour un « kebab ». On supposera que la partie abonnement (*attach*) a déjà été réalisée en amont ; on ne modélisera donc pas cette phase.

4. (1 point) Sur un diagramme de composant, représentez le composant **CCuisine**, avec les interfaces offertes et requises *que l'on peut déduire du précédent diagramme*.

IV. Gestion des Commandes (3,5 points)

Une commande correspond à l'ensemble des plats qu'a commandé un groupe de clients. Chaque commande est associée dès sa création à un serveur, dont il faut faire *vibrer* le téléphone quand les plats commandés qu'elle contient sont prêts. Le composant **CCommandes** gère l'ensemble des commandes ; il est abonné auprès du composant **CPreparationObservable** et aiguille les notifications concernant les plats prêts vers le bon serveur (celui qui est responsable de cette commande). On donne les signatures suivantes :



L'opération de création de commande rend un identifiant *idC* unique pour cette commande.

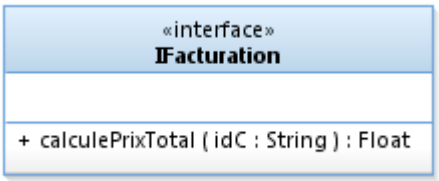
Les plats sont ajoutés un par un à la commande en donnant l'identifiant *idC* de la commande concernée et l'identifiant *idP* du plat à commander. Cette opération rend l'*idPC* généré par l'opération *creerPlatCommandé* de **IPreparation**.

Les opérations *lister* offrent une API en lecture permettant respectivement d'accéder à tous les *idC* des commandes, ou pour une commande donnée d'obtenir tous les *idPC* des plats commandés qu'elle contient.

1. (1 point) Sur un diagramme de composant, représenter **CCommandes** (interfaces requises et offertes).
2. (1,5 point) Sur un diagramme de classes, donner une conception détaillée possible de ce composant *CCommandes*.
3. (1 point) En cohérence avec cette conception détaillée, expliquez informellement (en texte, 3 à 5 lignes devraient suffire) comment traiter la réception d'une invocation à « *preparationEvent* » de **IObserver** au sein de **CCommandes**.

V. Facturation (2,5 points)

Le composant CFacturation est responsable de calculer le montant total d'une commande. Il offre une unique opération qui calcule le montant total d'une commande donnée à partir de son identifiant *idC*.



1. (1,5 point) Sur un diagramme de séquence, modéliser une séquence où l'IHM demande à une instance de CFacturation de calculer le montant total de la commande « *c01* » qui contient un « kebab ».
2. (1 point) Sur un diagramme de composant, en déduire les interfaces offertes et requises de CFacturation.