M1 : Ingénierie du Logiciel – 4I502 – SORBONNE UNIVERSITE

Examen 2eme Session

20 Juin 2018 (2 heures avec documents : tous SAUF ANNALES CORRIGEES).
Barème indicatif sur 20,5 points (donne le poids relatif des questions) (max 20/20).

Questions de cours [4 Pts]

Répondez de façon précise et concise aux questions.

Barème : VALABLE sur toutes les questions de cours : -25 à -50% si la réponse inclut la bonne idée, mais qu'elle est noyée dans des infos ou autres réponses fausses/inappropriées.

Question Cours (QC):

QC1) Définissez une métrique permettant d'évaluer la pertinence et la complétude d'un jeu de **test de validation** et une métrique permettant d'évaluer la pertinence et la complétude d'un jeu de **test unitaires**.

* 50% validation : e.g. ratio de couverture des séquences nominales/alternatives, nombre de test par use case, nombre de tests (tout court) …
* 50% unitaire : la couverture des lignes de code est la principale métrique utilisée.

QC2) On suppose deux composants **CA** et **CB**. **CA** offre **IA** et requiert **IB. CB** offre **IB** et requiert **IA.** Peut-on dire qu'on a un cycle de dépendances structurelles entre les composants **CA** et **CB** ? Justifiez votre réponse.

50% non

50% justification : on peut avoir **CC** qui offre aussi **IA** ou **IB.** Les composants ne dépendent jamais directement les uns des autres, mais via des dépendances fonctionnelles vers des **interfaces**, pas d'autres composants.

QC3) En conception architecturale, on réalise un diagramme de séquence où une ligne de vie représentant une instance du composant **CX** invoque une opération **m()** sur une ligne de vie représentant une instance d'un composant **CY.** Dessinez un diagramme de composant représentant **CX** et **CY**.

* 30% on a défini une interface, a priori IY
* 35% CX l'utilise
* 35% CY l'offre

QC4) Dans une approche dirigée par les modèles et leurs transformations, donnez un exemple de transformation T2M, M2M et M2T.

* 35% T2M un parser quelconque, reverse engineering, …
* 35% M2M : e.g. UML vers E/R, UML vers Java (MétaModèle), refactoring endogène, …
* 35% M2T : sérialiseur, eg. UML vers Java traité en TD.

Borné à 100%.

2. Problème: Analyse VPBP : VosPlusBellesPhotos [9,5 Pts]

Votre entreprise vient de décider de construire une application web permettant à tout un chacun de stocker dans le cloud ses photos numériques et d'en faciliter la recherche ainsi que la présentation.

L'application, nommée VosPlusBellesPhotos, est une application Web. Les utilisateurs non abonnés peuvent utiliser l'application uniquement pour visionner les photos des utilisateurs abonnés. Les utilisateurs

abonnés peuvent utiliser l'application pour stocker leurs photos ainsi qu'effectuer plusieurs traitements graphiques sur celles-ci. La création d'un compte abonné nécessite un login, un password, un email et doit être accompagnée d'une acceptation de la licence de l'application (cette licence demande en particulier de ne pas stocker des photos illicites). Toute demande de création de compte abonné doit être validée par un administrateur de l'application. Un album initialement vide destiné à contenir toutes les photos de l'utilisateur est alors crée pour l'utilisateur.

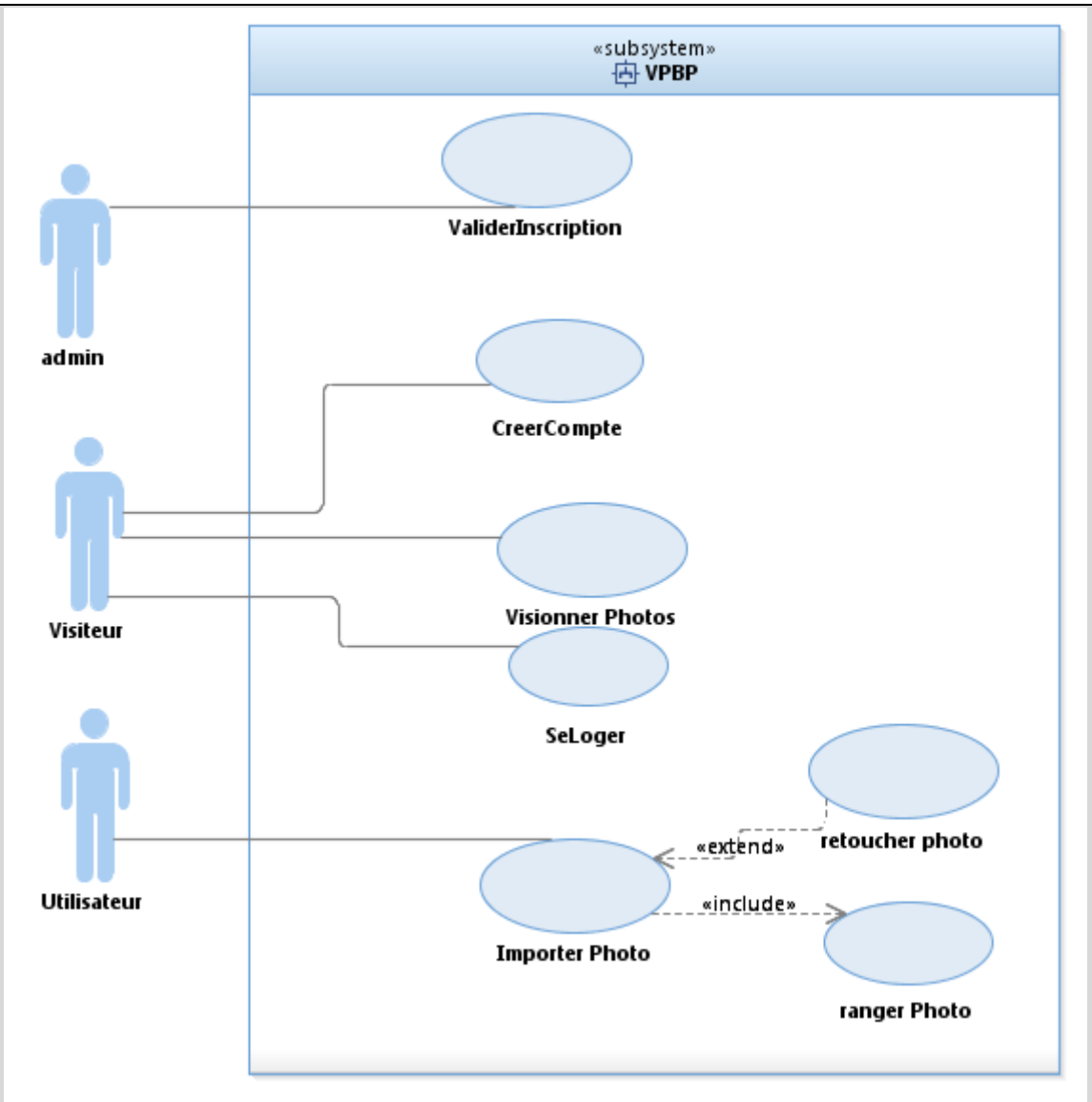
L'accès à l'application VosPlusBellesPhotos doit se faire avec un navigateur Web. L'URL de l'application débouche sur la page d'accueil de l'application. Celle-ci permet à n'importe qui (y compris un utilisateur non abonné) de visiter les albums photos des membres abonnés. Cette page d'accueil permet aussi aux utilisateurs abonnés de s'authentifier afin d'entrer dans leur espace personnel.

VosPlusBellesPhotos doit permettre à l'utilisateur d'importer ses photos numériques à partir de n'importe quel format photo numérique (JPG, PNG…). Les photos nouvellement importées sont stockées dans l'espace de travail de l'application. Chaque photo importée a un nom et une date de création.

L'application permet ensuite d'effectuer différents travaux sur les photos présentes dans l'espace de travail de l'utilisateur. Elle permet en particulier d'effectuer un recadrage de la photo, de gommer les yeux rouges et d'effectuer des corrections de luminosité. Une fois que l'utilisateur a effectué les différents travaux qu'il souhaite, il peut choisir de ranger ses photos. Pour ce faire, il doit associer aux photos une ou plusieurs étiquettes. Par défaut, l'application propose les étiquettes : « Famille », « Vacances », « Travail », « Ami », « Fête ». L'utilisateur peut aussi créer ses propres étiquettes. Une fois que la photo a été rangée, c'est-à-dire que l'utilisateur lui a associé une ou plusieurs étiquettes, l'application propose à l'utilisateur de déplacer la photo de l'espace de travail vers son album virtuel dans l'espace de stockage. Il n'est alors plus possible de travailler sur la photo. Par contre, seules les photos présentes dans les albums de l'espace de stockage peuvent être visionnées.

L'application permet à n'importe qui de visionner les photos des albums présents dans l'espace de stockage. Il est possible de parcourir l'ensemble des photos d'un album ou bien d'en sélectionner seulement une partie en fonction des étiquettes qu'elles portent. Il est aussi possible de sélectionner les photos en fonction de leur date de création.

Question 2.1 : (3 pts) Réalisez le diagramme de cas d'utilisation de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme.

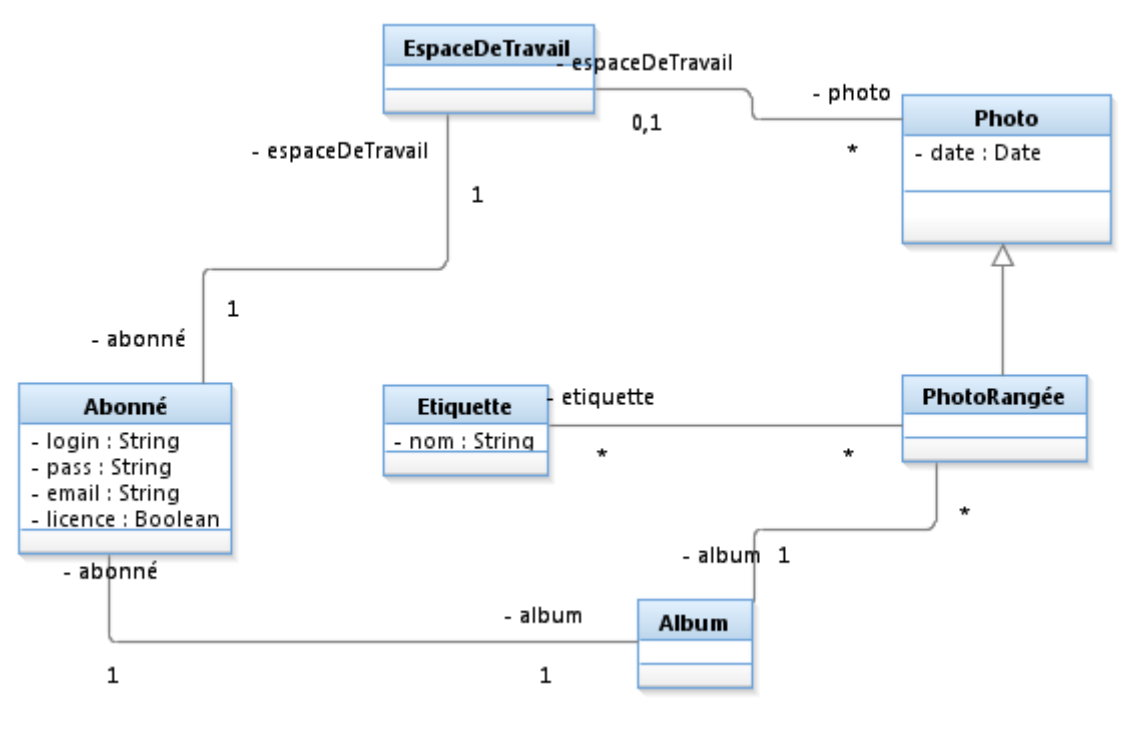


Barème :

20% par use case principal correctement associé au bon acteur.

-10% par include ou extend injustifiable

Question 2.2 : (3 points) Réalisez le diagramme de classes métier de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme. On ne représentera pas la classe représentant le « Système », introduite dans l'approche en V du module.



Barème :

Abonné 20%

Photo + Etiquette 40%

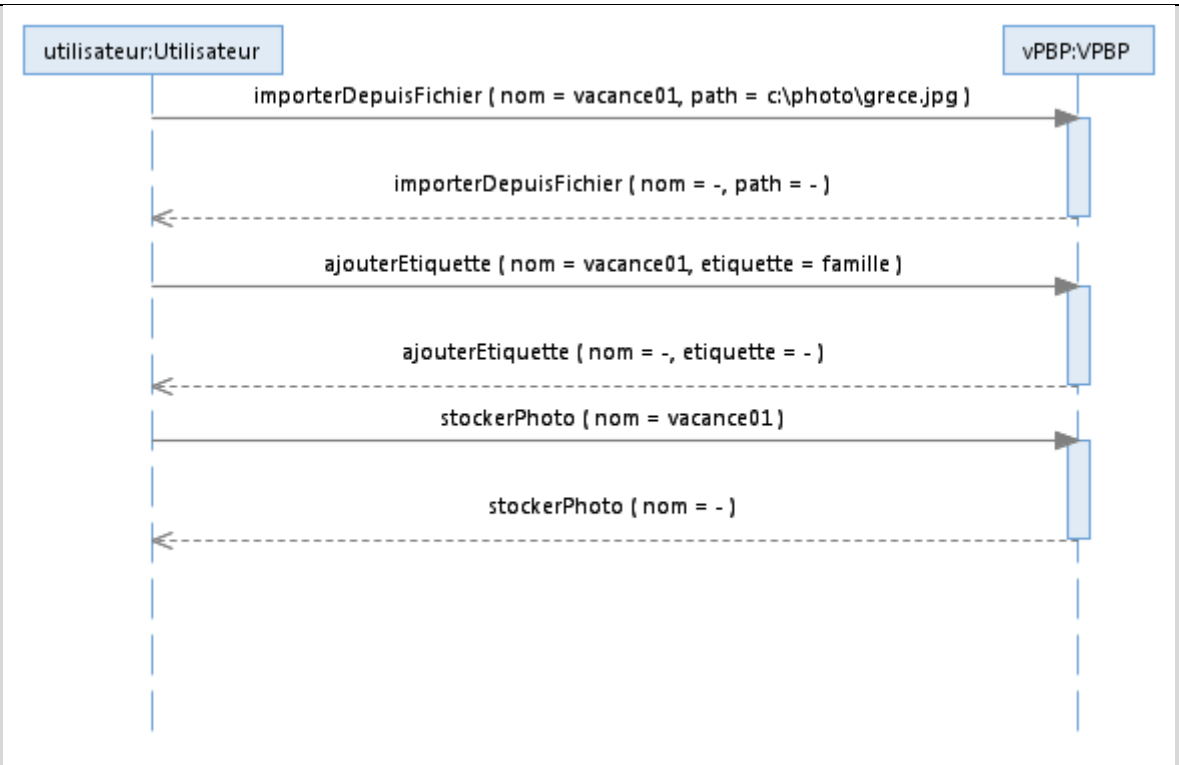
Lien abonné photos 20%

Date sur photo : 10%

Distinctions espace stockage/album : 10%

-10% par faute grossière, méthodes, modélisation d'acteurs…

Question 2.3 : (2 pts) Dans un diagramme de séquence de niveau analyse, modélisez l'importation dans l'application d'une photo (sans la retoucher) classifiée comme « Famille ».



On attend 3 invocations distinctes : 60%.

On doit clairement voir circuler le nom de l'étiquette et un chemin/fichier image de l'acteur vers le système : 40%

Question 2.4 : (1,5 pts) Ecrivez un test de validation couvrant le visionnage des photos de « Famille ».

* contexte : on a deux photos P1 et P2 qui ont le tag « Famille »

* entrée : tag « Famille »

* Scenario :

1. connection au site
2. saisie de étiquette « Famille » parmi les étiquettes visibles
3. sélection de la vignette de P1

R.A. : la photo P1 est affichée en plein écran (on y voit un couple générique avec un enfant).

MV: visuel

Barème :

Contexte **40%** : sans ça pas de test possible

Entrée **10%**: champ utilisé correctement.

Scenario **20%** : 10% cohérent avec l'objectif, étapes du testeur seulement…, 10% précision suffisante pour la reproductibilité (e.g. le testeur choisit une photo = imprécis)

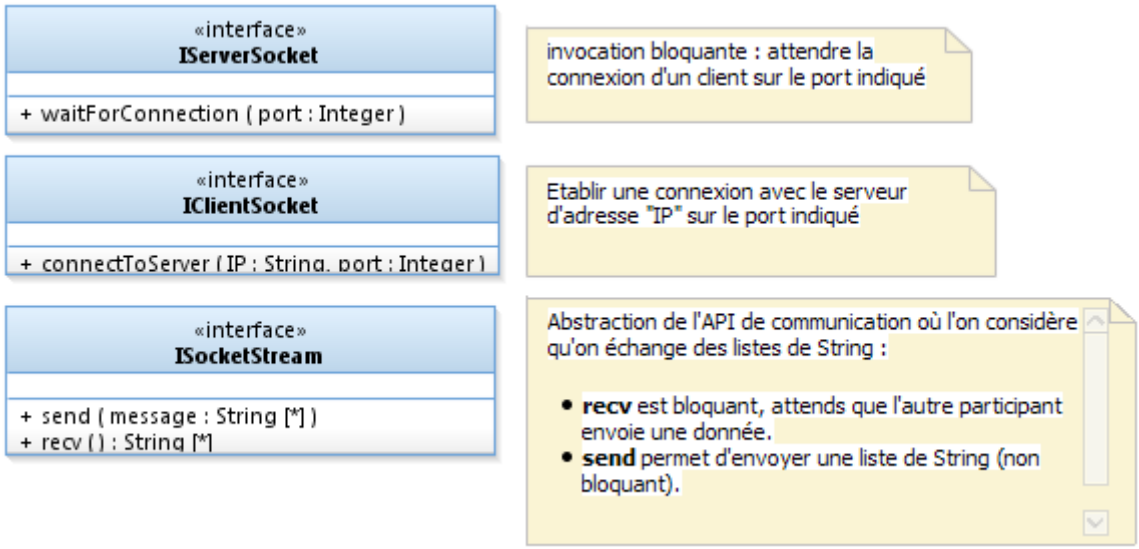
R.A **20%** on spécifie quelle photo on est censé voir.

M.V. **10% :** visuel

-10% à -20% aberrations énorme (alternatives dans le scenario, …)

3. Problème: Proxy Distant [Barème sur 7 Pts]

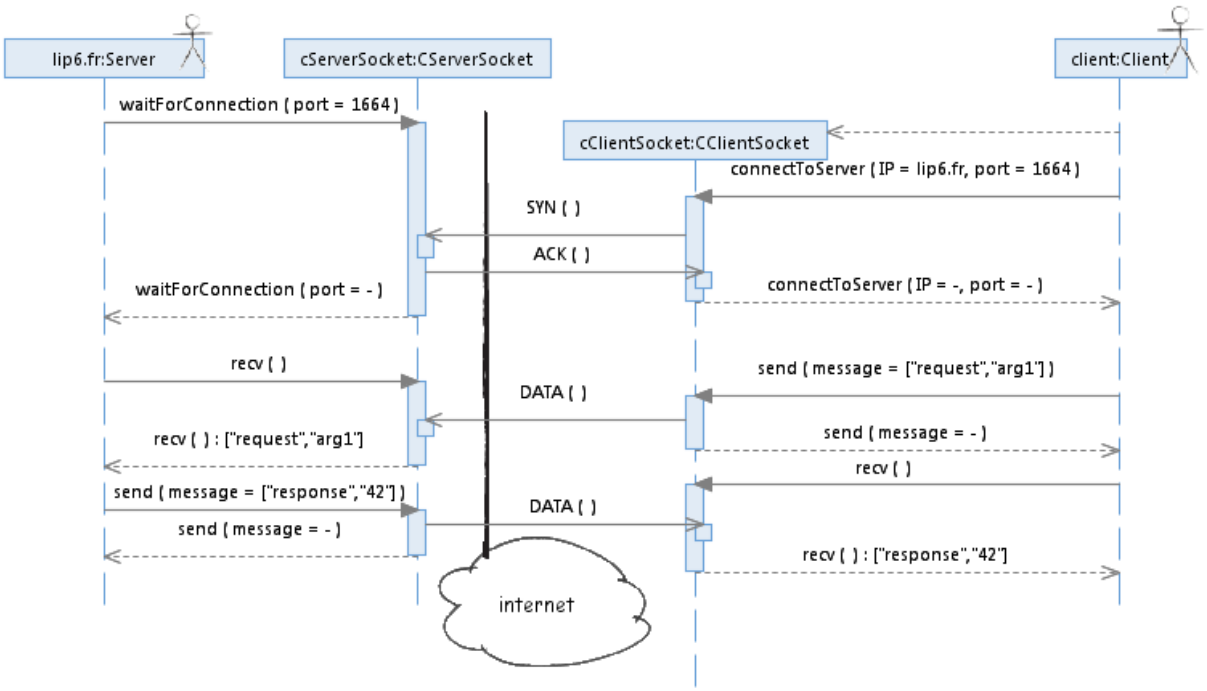
Les composants CClientSocket et CServerSocket représentent des extrémités de points de communication.



L'utilisation de ces composants a deux phases :

- l'établissement de connexion est asymétrique : le serveur attends des connexions de clients en fixant un port (l'IP est implicitement celle du serveur). Le client qui connait le couple IP :port du serveur s'y connecte directement.
- La communication est ensuite symétrique, le serveur comme le client peuvent utiliser l'API décrite par ISocketStream pour envoyer ou recevoir des données de l'autre participant.

Le diagramme de séquence suivant montre l'établissement d'une connexion et la simulation d'une requête. Les échanges réseau sont modélisés par des messages asynchrones (SYN/ACK, DATA) qu'on ne cherchera pas à détailler plus ni à faire figurer dans des interfaces.



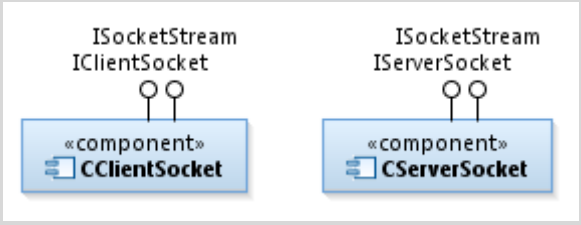
Question 3.1 : (1 pts)

Représentez sur un diagramme de composant les composants CServerSocket et CClientSocket.

a)

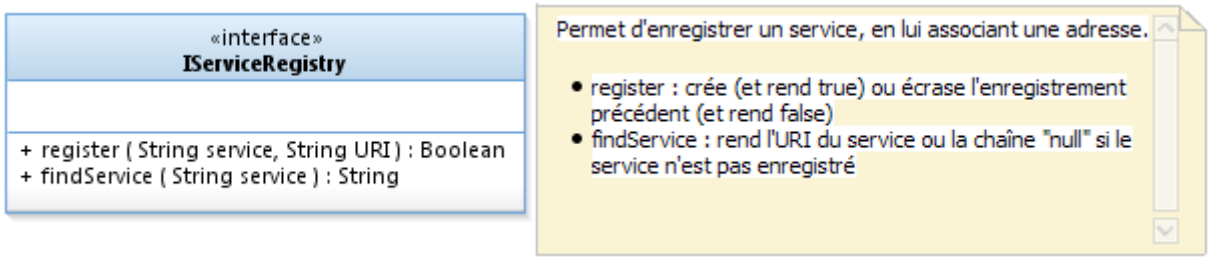
50% par composant, binaire (rien en trop, rien en moins)

max 100%



Question 3.2 : (1,5 pts)

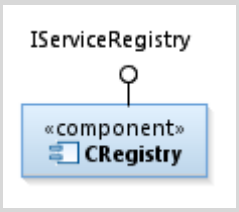
On introduit une interface **IServiceRegistry** modélisant un service de nommage, permettant d'associer à des noms de services (les clés) une adresse unique permettant de contacter le service. Le comportement est celui d'une table associative.



On considère une réalisation simple de ce comportement dans un composant **CRegistry.**

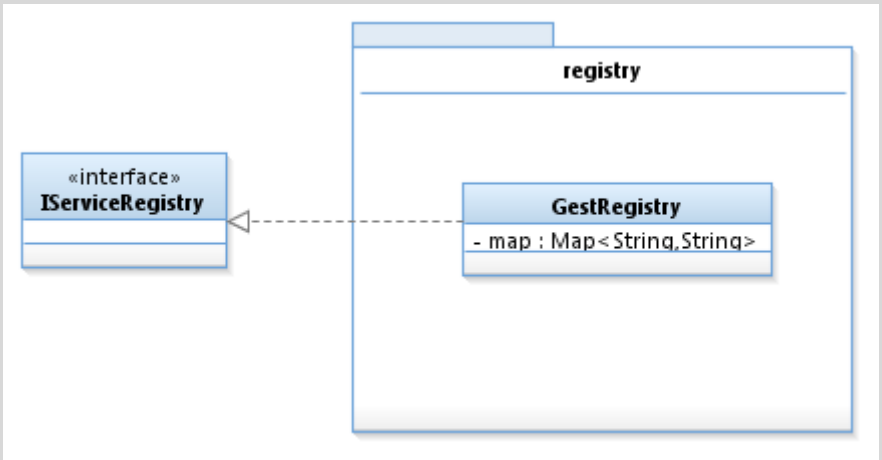
a) Représentez sur un diagramme de composant ce composant CRegistry
b) Proposez à l'aide d'un diagramme de classe une conception détaillée possible de ce composant.

a)



30%

b)



35% implements

35% attribut pertinent

Question 3.3 : Stub Serveur (4,5 pts)

On souhaite permettre à des clients distants d'interagir avec le service de nommage.

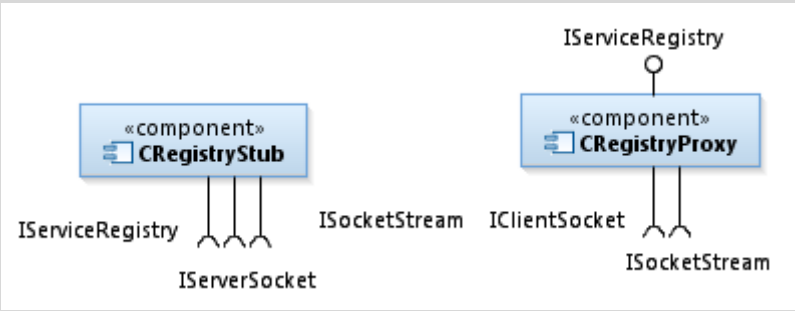
Le composant **CRegistryStub** est instancié sur le serveur, qui contient aussi l'instance locale de CRegistry qu'on souhaite exposer au réseau. Ce composant instancie une CServerSocket et attends les connexions de clients sur le port 42. Quand une connexion est établie, il reçoit et décode la requête du client (passée sous la forme d'une liste de String), et invoque la méthode adaptée du registre local (soit register, soit findService). Il renvoie alors le résultat de cette requête (sous la forme d'une liste de String) au client.

Le composant **CRegistryProxy** est instancié sur la machine cliente. Son rôle est d'offrir le service de nommage sur cette machine, comme si le service était local à la machine (d'où son nom « proxy »). A la construction, il instancie un socket cliente, qu'il connecte au serveur (on supposera « name.lip6.fr » sur le port 42). Il se trouve alors connecté au CRegistryStub. Quand des demandes sont faites sur le proxy, il envoie une liste de String exprimant la requête au serveur, puis attends sa réponse (une liste de String), l'interprète et rend le résultat à l'appelant.

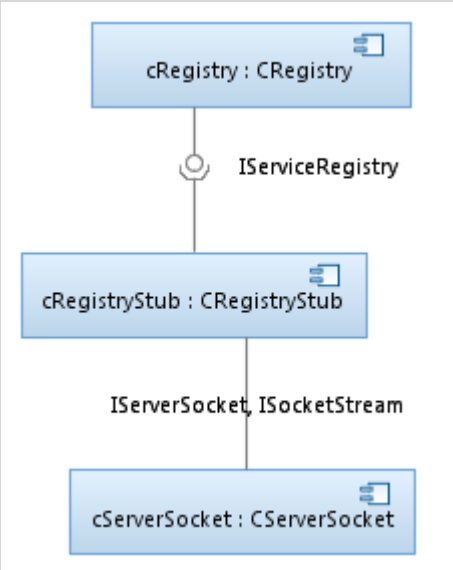
Représentez sur un diagramme de composant ce composant CRegistryStub.

a) (0,5) Modéliser sur un diagramme de composant ces deux composants stub et proxy.
b) (1 pt) Sur un diagramme de structure interne modéliser les composants instanciés sur la machine du serveur après la connexion d'un client.
c) (1 pt) Sur un diagramme de structure interne modéliser les composants instanciés sur la machine du client après la connection au serveur. On supposera l'existence un composant bout de chaîne CClient qui utilise l'API de IRegistry.
d) (2 pts) Représentez sur un diagramme de séquence de niveau architecture mettant en jeu les instances modélisées en b) et c) les interactions nécessaires pour obtenir depuis la machine cliente l'adresse du service « football2018 » stockée sur le serveur (la réponse est « Russie »). On supposera que la phase d'établissement de connexion est déjà terminée. On s'inspirera du diagramme de séquence fourni pour modéliser les échanges réseau.

a)

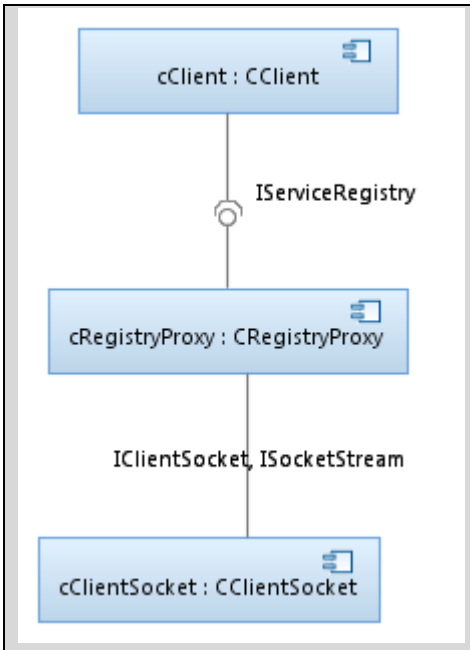


b)



(deps orientée de Stub vers ServerSocket)

c)



Deps orientées de Proxy vers Socket

d)