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

Examen Réparti 2eme partie

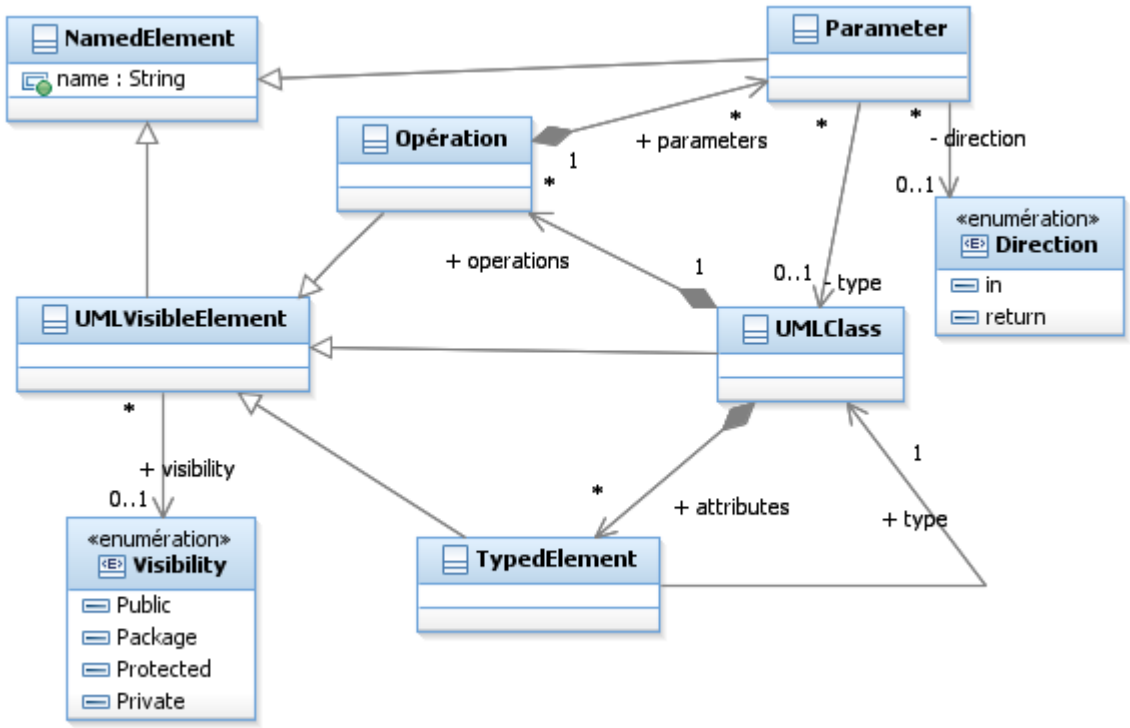
5 Janvier 2016 (2 heures avec documents : tous SAUF ANNALES CORRIGEES).
Barème indicatif sur 21 points (donne le poids relatif des questions) (max 20/20).

Questions de cours [3 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.

Soit le méta-modèle simplifié du diagramme de classe UML tel que présenté en TD.



Question Cours (QC):

a) Représentez sous une forme textuelle arborescente (proche de xmi) le modèle instance suivant constitué de deux classes :

b) Proposez une modification de ce métamodèle permettant de modéliser le concept de classe interne, définie au sein d'une autre classe.

c) Proposez une façon de modéliser un package, c'est-à-dire un objet nommé (NamedElement) qui contient d'autres éléments nommés.

```
<UMLclass name=A vis=public id=#o1>
  <attributes>
```

<TypedElement name=b vis=private type=#o2 />

</attributes>

</UMLClass>

<UMLclass name=B vis=public id=#o2>

<attributes>

<TypedElement name=a vis=private type=#o1 />

</attributes>

</UMLClass>

Barème :

50% on a bien deux classes munies d'un attribut

50% gestion des référenes croisées correcte

b)

Association reflexive de Class sur Class, composition de * classes, nommée « nested » par exemple.

Barème :

80% association 1-*

20% c'est une composition (losange noir)

c)

On extends NamedElement et on compose * NamedElement, la classe s'appelle Namespace techniquement (et pas package, qui est un namespace particulier) dans la norme UML. C'est une réalisation du DP composite.

100% réponse parfaite

50% on propose une simple association reflexive de NamedElement sur soi même

2. Problème: Conception Twitter [Barème sur 18 Pts]

Rappels d'analyse : Nous considérons l'application Twitter, permettant d'échanger des « Tweets » ou messages courts avec le monde entier. Par exemple, l'utilisateur @ytm peut envoyer le tweet « Love #UML and @GradyBooch» qui utilise le tag #UML et cite explicitement l'utilisateur « @GradyBooch ».

Tout utilisateur muni d'un compte est associé à un fil d'actualité, visible par tous, qui porte par ordre chronologique du plus récent au plus ancien tout tweet :

- qu'il a tweeté,
- qu'il a retweeté,
- qui le cite explicitement dans le corps du tweet,
- qu'un utilisateur auquel il est abonné à lui-même tweeté ou reweeté.

On peut également demander à afficher le « fil d'actualité » correspondant à un tag particulier, qui porte tous les tweets qui utilisent ce tag dans le corps du tweet.

Nous allons faire plusieurs hypothèses simplifiant le problème dans cet énoncé :

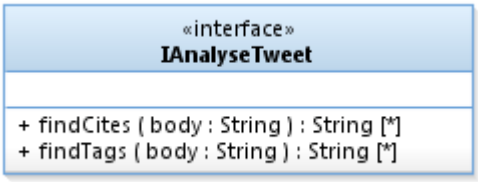
- Les tweets sont du texte pur, on ne peut pas attacher d'images ou fichier
- On ne peut pas rajouter du texte quand on « retweete » un message. Retweeter consiste simplement à ajouter le tweet en question à son propre fil d'actualité ainsi qu'à celui de tous ses abonnés potentiels.
- On ignore la fonctionnalité « aimer le tweet »

- On ne considère que le cas nominal, en particulier tous les noms de compte utilisés comme auteur ou cités existent par hypothèse
- Plus globalement, on ignore l'authentification et la création de compte (profil etc…) dans tout l'énoncé

Ca laisse pas mal de choses quand même, le cœur de notifications de twitter en gros.

I. Composant CAnalyseTweet, analyse du contenu des Tweet (1,5 points)

Le composant bout de chaîne CAnalyseTweet est responsable d'analyser les Tweet, pour déduire les personnes citées dans le tweet ainsi que les tags utilisés par un tweet. Les citations de personnes sont précédées d'une @ comme « @GradyBooch », les tags sont précédés d'un symbole # comme « #UML ». Ce composant offre IAnalyseTweet auquel on passe le corps du message à analyser et qui rend la liste des personnes citées ou des tags identifiés.



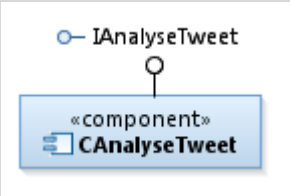
Question I : (1,5 pts)

a) Représentez sur un diagramme de composant le composant CAnalyseTweet (interfaces offertes et requises).
b) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.

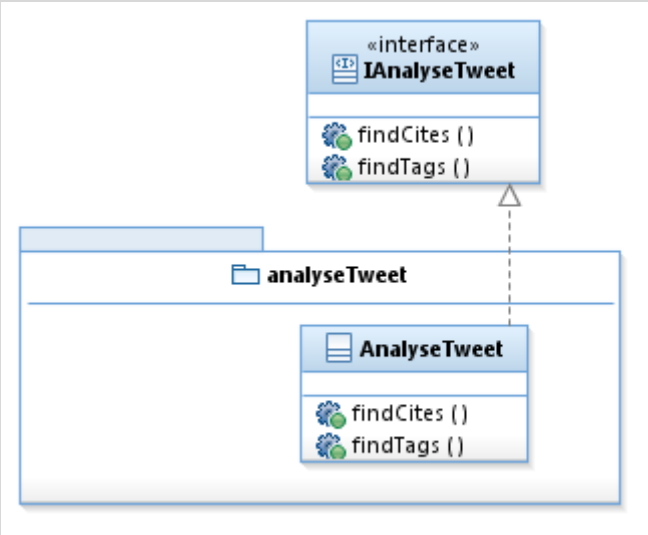
Un composant très simple pour se mettre en route. Il va permettre de se mettre une dépendance dans les pieds pour permettre des questions plus « intéressantes » dans la suite.

a)

binaire 0/40



b)



30% une classe façade (10%), dans son package (20%)

30% elle réalise IAnalyseTweet

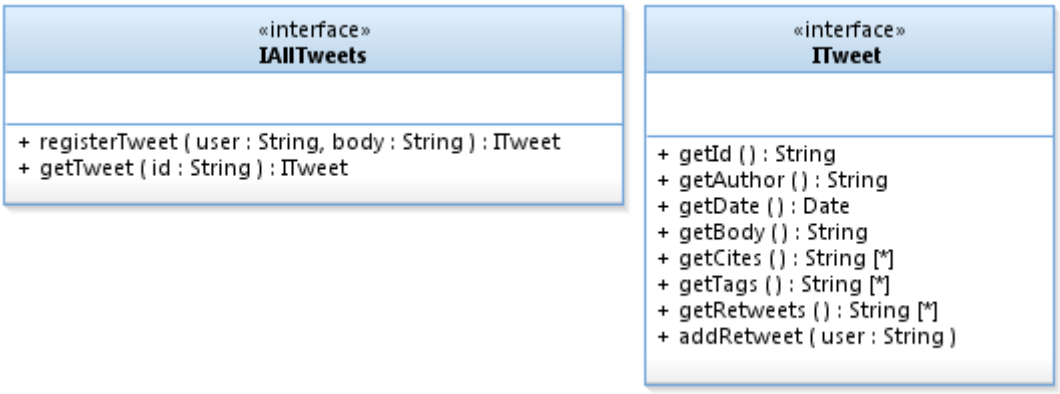
On peut introduire un parser ou autre classe annexe dans le composant.

II. Composant CAllTweets, persistance et accès aux Tweets (6 points)

Ce composant de stockage est responsable d'assurer la persistance et l'accès au contenu des tweets. Pour des raisons d'efficacité et de temps d'accès il est typiquement répliqué en plusieurs instances dont l'état est maintenu synchronisé via des technologies du cloud qu'on n'abordera pas ici. On se contentera de supposer que pour les besoins de l'application c'est un composant singleton, dont l'unique instance est accessible de tout point de l'application.

Les tweets y sont stockés à l'aide de la méthode d'enregistrement, « registerTweet » à laquelle on passe le nom de l'auteur du tweet et le corps du texte tweeté ; la date du tweet est celle à laquelle il est soumis et les autres éléments sont déduits du tweet lui-même, à l'aide du composant CAnalyseTweet décrit en Question I. L'on accède au contenu d'un tweet via un identifiant « id » unique.

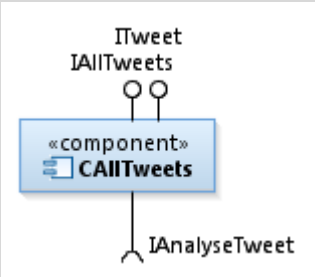
Le tweet lui-même porte en lecture seule le nom de son auteur, le corps de texte, la liste des personnes citées dans ce tweet, la liste des tags présents dans le tweet, et la liste des noms des personnes ayant « retweeté » ce tweet. La seule opération permettant de modifier le tweet est « addRetweet » qui permet d'enregistrer qu'un retweet de ce message par « user » a été fait.



Question II.1: (2,5 pts)

a) Représentez sur un diagramme de composant le composant CAllTweets (interfaces offertes et requises).
b) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.

a)



20% on réalise les deux itf

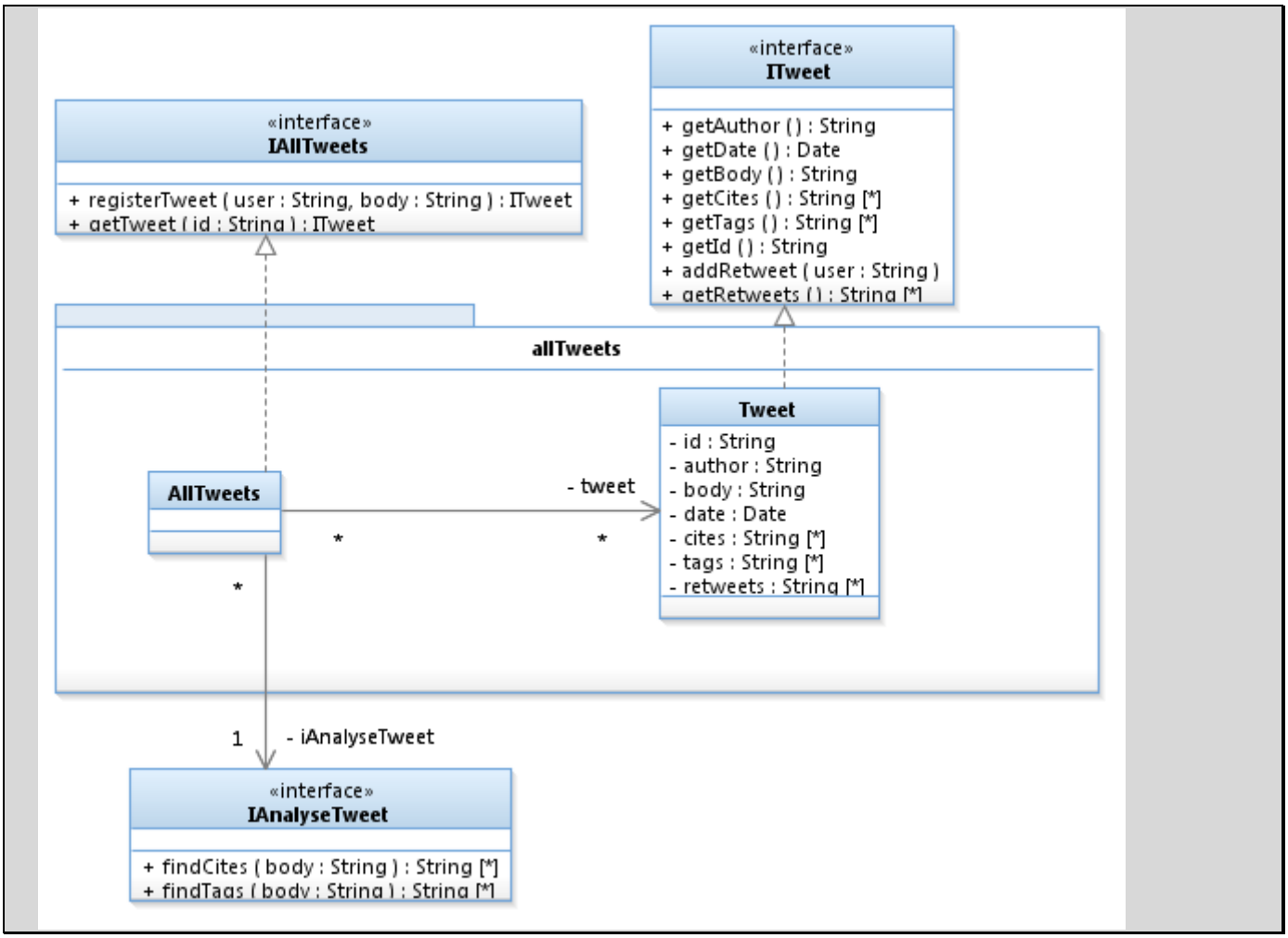
20% dépendance sur analyse tweet

b)

20% classe façade, connait * Tweets, réalise IAllTweets.

20% Tweet, porte les attributs utiles, réalise ITweet.

20% dépendance à AnalyseTweet modélisée, on peut aussi l'associer au tweet (les champs attributs cites et tags sont alors des attributs dérivés, recalculés au besoin).



Question II.2: (3,5 pts)

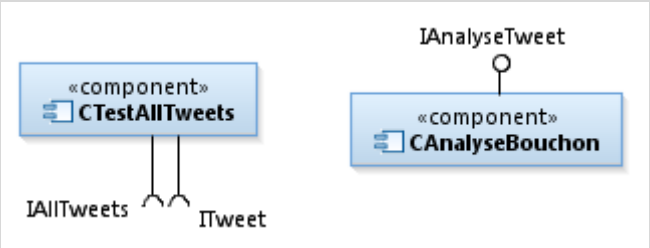
On souhaite mettre en place des tests d'intégration de ce composant validant (entre autres) le scénario suivant : l'utilisateur @ytm tweete « Love #UML».

a) Représentez un ou plusieurs composants (testeur, bouchon(s)) permettant de mettre en place ces tests.
b) Sur un diagramme de structure interne, représentez une configuration des composants pour le test.
c) Sur un diagramme de séquence ou vous utiliserez une ligne de vie par instance identifiée en b), montrez les échanges nécessaires pour enregistrer le tweet donné en exemple puis s'assurer (résultat attendu) qu'il porte bien uniquement le tag #UML.

a)

Bon il faut effectivement un bouchon pour CAnalyseTweet ou plutôt IAnalyseTweet.

On teste à la fois IAllTweets et ITweet de concert visiblement. Ca donne :

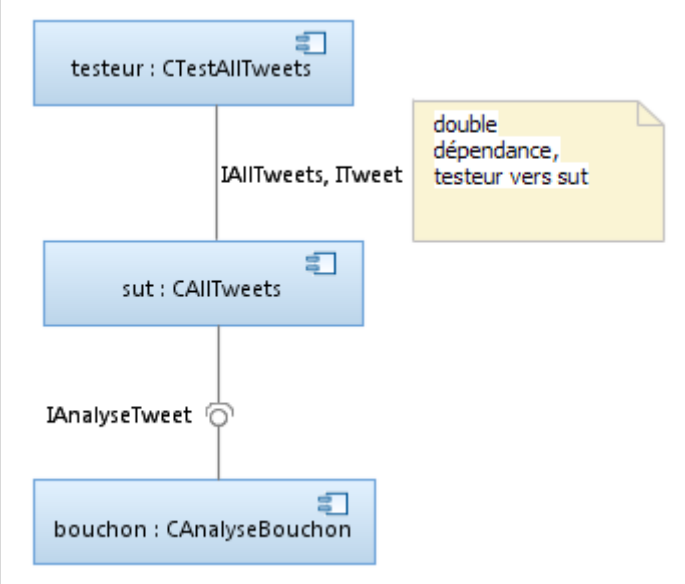


15% définition du bouchon

15% Un composant testeur qui utilise les deux interfaces offertes

b)

Avec des « part » ou instances :



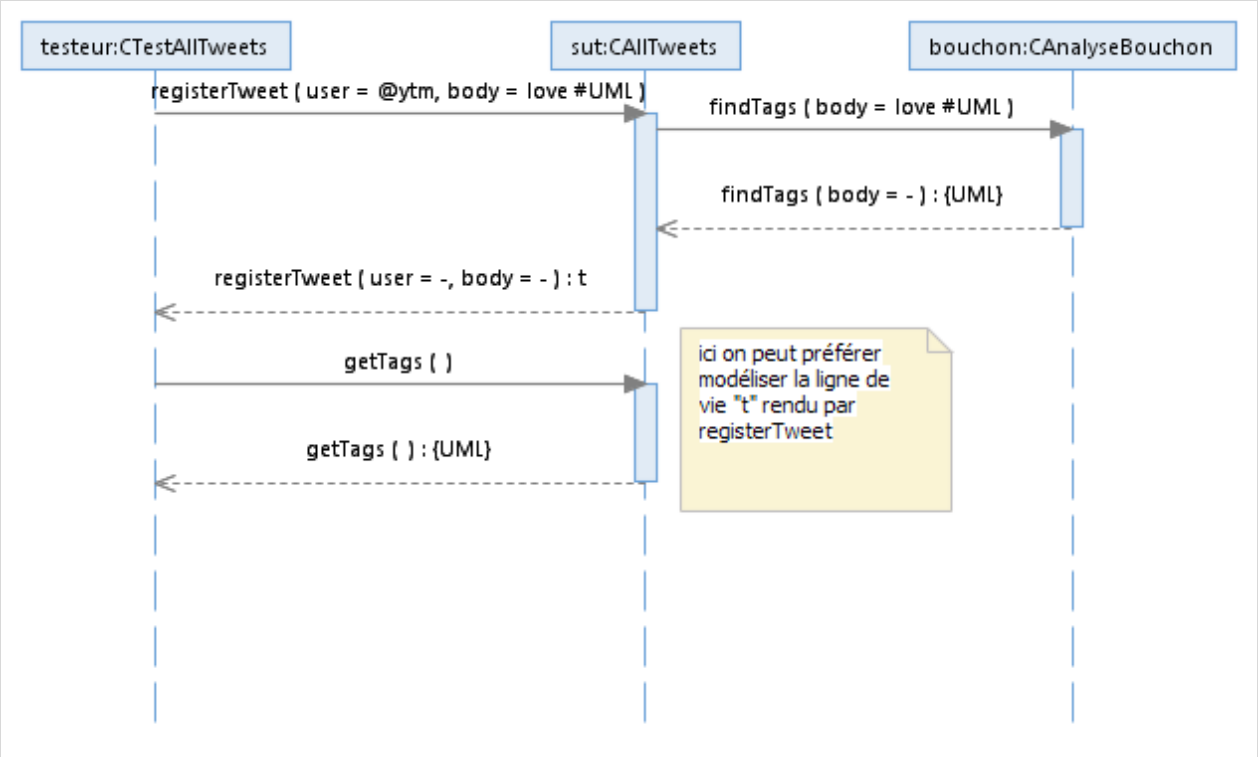
Binaire : 0/20% : le diagramme est du bon type, instancie bien les composants définis an a), et fait figurer les dépendances dans le bon sens pour modéliser une configuration de test.

On peut obtenir les 20% même si la réponse en a) est incomplète/légèrement incorrecte si la cohérence avec a) est bonne, et que c'est bien une config de test.

c)

L'intérêt du diagramme est qu'on voit ce qu'il faut embarquer dans le bouchon.

L'invocation à findTags peut se faire quand on veut. Les lignes de vie du sut et du tweet sont confondues sur ce dia, mais on peut les distinguer (le texte de la question n'incite pas à le faire). Les deux versions sont correctes en UML (le tweet appartient à la structure interne du composant « sut »).



10% on identifie bien (noms+types) les instances définies en b)

10% on voit passer registerTweet (avec données) du testeur vers le sut

15% on identifie que sut invoque findTags sur le bouchon à un moment

15% le testeur invoque getTags() sur le sut pour permettre le test du R.A. On accepte diverses modélisations correctes pour la ligne de vie cible de l'invocation.

III. Composant CListeTweet (2 points)

Ce composant va permettre de matérialiser un ensemble de tweet correspondant soit à un fil d'actualité d'un utilisateur, soit à un mot clé ou tag comme #UML.

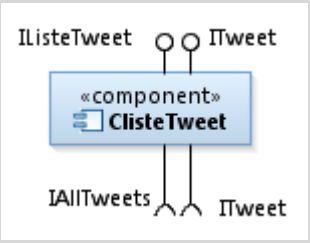
Il offre IListeTweet et doit donc permettre :

- D'ajouter de nouveaux tweets, à travers leur identifiant. Les tweets eux-même ne sont pas stockés par ce composant, ils restent la responsabilité de CAllTweets (cf. Question II).
- De lister les tweets, du plus récent au plus ancien. On passe pour cela un indice entre 0 et (size()-1). Par convention l'indice 0 correspond au plus ancien tweet de ce fil. Le détail des tweets est obtenu en s'appuyant sur le singleton CAllTweets.

Question III. : (2 pts)

a) Représentez sur un diagramme de composant le composant CListeTweet (interfaces offertes et requises).
b) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.

a)



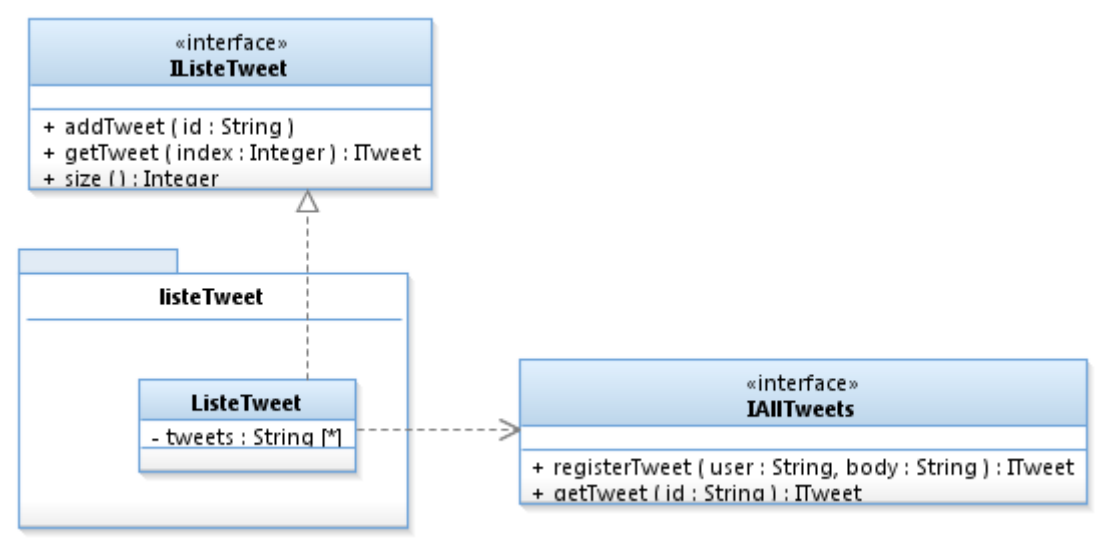
40% offre IListeTweet

40% utilise IAllTweets

20% offre ITweet. Techniquement, la réalisation de ITweet est faite par CAllTweets, mais du point de vue isolation, on est forcé de considérer que cette interface est aussi offerte par CListeTweet (e.g. peut-etre que le contenu des tweets est mis en cache dans ce composant). Les occurrences de ITweet rendues dans l'interface IListeTweet pourraient venir de n'importe où.

0% La dépendance sur ITweet peut être modélisée ou non (en fait ce composant n'invoque pas les opérations de ITweet).

b)



40% classe facade, réalise IListeTweete

30% attribut une liste de String les identifiants de tweets. On ne doit pas avoir de nouvelle implantation de ITweet a priori. On accepte les dépendances à ITweet (texte de commentaire ou dépendance généralisée) mais on ne doit pas associer * ITweet directement (on ne donne pas les 20% si c'est le cas).

30% dépendance sur IAllTweets, j'ai modélisé une simple dépendance généralisée (vu que c'est un singleton on accède à la demande à priori), mais on peut utiliser une association plutôt.

IV. Composant CAbonnements, gestion des abonnements (3 points)

Le composant bout de chaîne CAbonnements est responsable de savoir quels utilisateurs sont abonnés à un compte twitter donné. Un compte peut être abonné à un nombre arbitraire de comptes et réciproquement on peut avoir un nombre arbitraire d'abonnés.

On souhaite pouvoir :

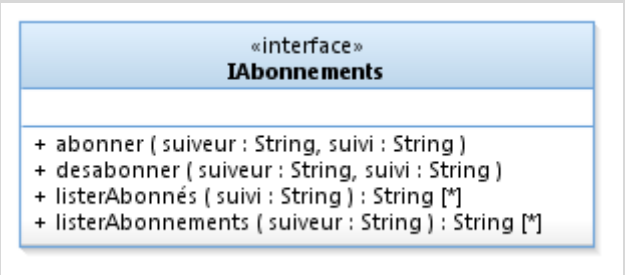
- Abonner un utilisateur à un autre
- Désabonner un utilisateur d'un autre (sans effet si l'abonnement n'existe pas)
- Lister les noms des comptes des abonnés d'un compte donné
- Lister les noms des comptes auquel un utilisateur est abonné

Question IV. : (3 pts)

a) Définissez une interface (avec signatures) supportant ces scenarios
b) Sur un diagramme de séquence, modéliser l'interaction qui permet d'abonner l'utilisateur @bob à @alice. On suppose un composant CIHM représentant l'interface homme machine.
c) Représentez sur un diagramme de composant le composant CAbonnements (interfaces offertes et requises).
d) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.

Questions a et b sur 100

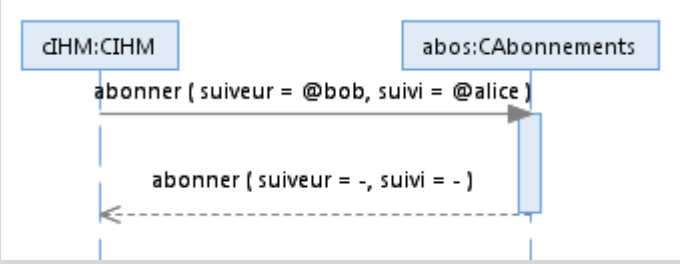
a)



60% = 15% par opération + signature

On peut rendre des bool si on préfère.

b)

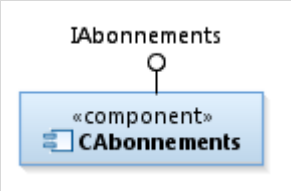


10% les instances/lignes de vie,

30% on voit circuler l'information

Questions c et d sur 100%

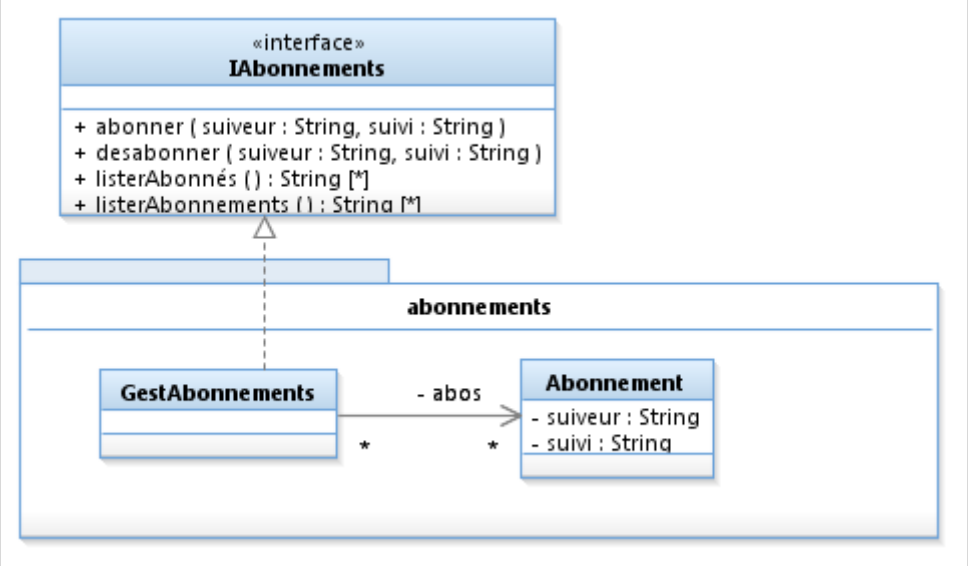
c)



Bout de chaine.

30% binaire

d)



30% réalise IAbos

40% on voit le stockage des paires pertinentes.

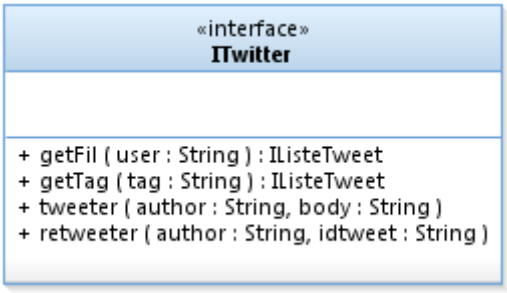
On peut préférer deux Map<String, List<String>> pour la réalisation (c'est même fortement indiqué ici) mais cette version qui stocke une liste de paires de String va bien aussi.

V. Composant CTwitter, façade et contrôleur (5,5 points)

Le composant CTwitter va assembler l'ensemble des fonctionalités. Il instancie et utilise les autres composants définis à ce stade.

Il offre ITwitter portant les opérations permettant d'obtenir les tweets constituant le fil d'actualité d'un compte, de trouver les tweets correspondant à un tag particulier, de tweeter un message (on fournit l'auteur

et le texte du corps du tweet) et de retweeter un message (on fournit l'auteur du retweet et l'identifiant unique du tweet à retweeter).

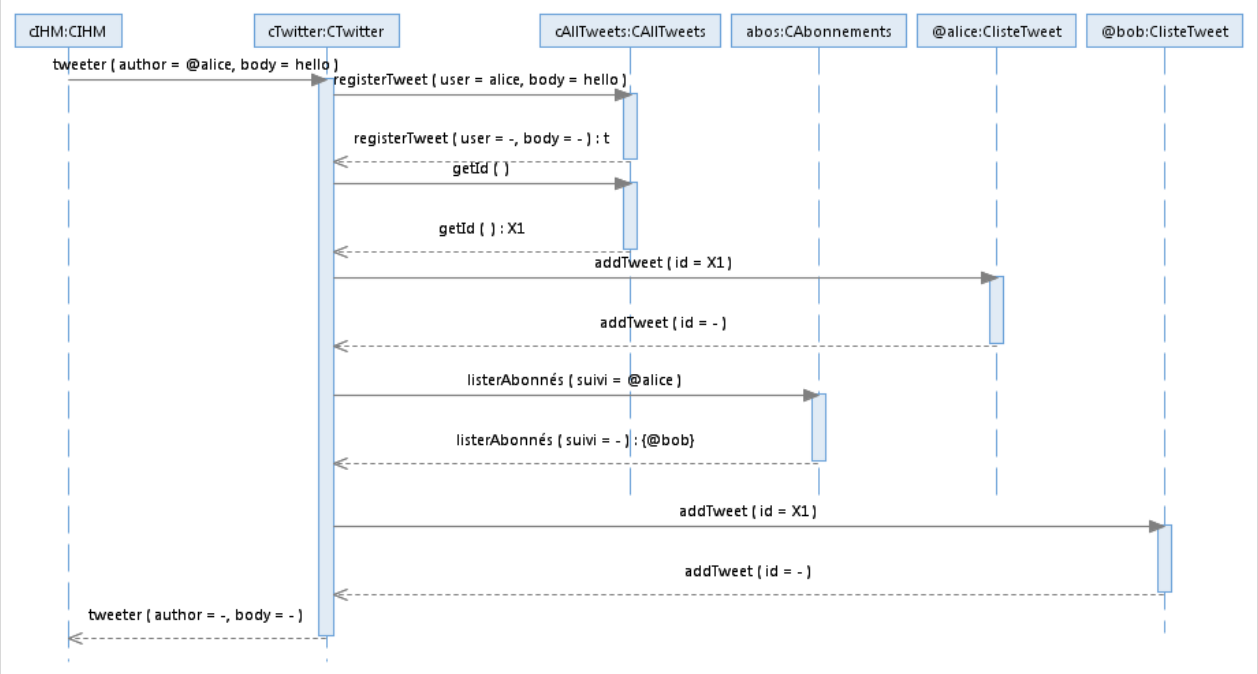


Question V.1 : (4 pts)

On suppose que le composant CTwitter stocke et associe à chaque nom de compte et à chaque tag une occurrence de CListeTweet (listeAlice :CListeTweet, listeBob :CListeTweet…). On suppose aussi que le composant d'abonnement à enregistré que @bob est abonné à @alice (cf. question IV. b).

a) Modéliser sur un diagramme de séquence où les lignes de vie représentent des instances des composants identifiés à ce stade ce qui se passe quand @alice tweete « Hello World ». On ne modélisera que les interactions réellement utiles ; les opérations testées en pratique mais qui rendent des listes vides dans cet exemple ne seront pas modélisées (e.g. il n'y a pas de tags ni de citation).
b) Modéliser sur un nouveau diagramme de séquence le retweete par @charlie de ce même message. Charlie n'a pas d'abonnés mais il faut mettre à jour le tweet et l'ajouter à son propre fil d'actualité.
c) Sur un nouveau diagramme de séquence, modéliser l'affichage du fil de @charlie par le composant CIHM. Il ne contient que le message d'Alice. On n'affichera que l'auteur et le corps du tweet.

a)Même commentaire que précédemment, on peut distinguer la ligne de vie t de cAllTweets.



20% tweeter de IHM vers le controleur englobe toute l'action, paramètres cohérents.

20% on enregistre le tweet

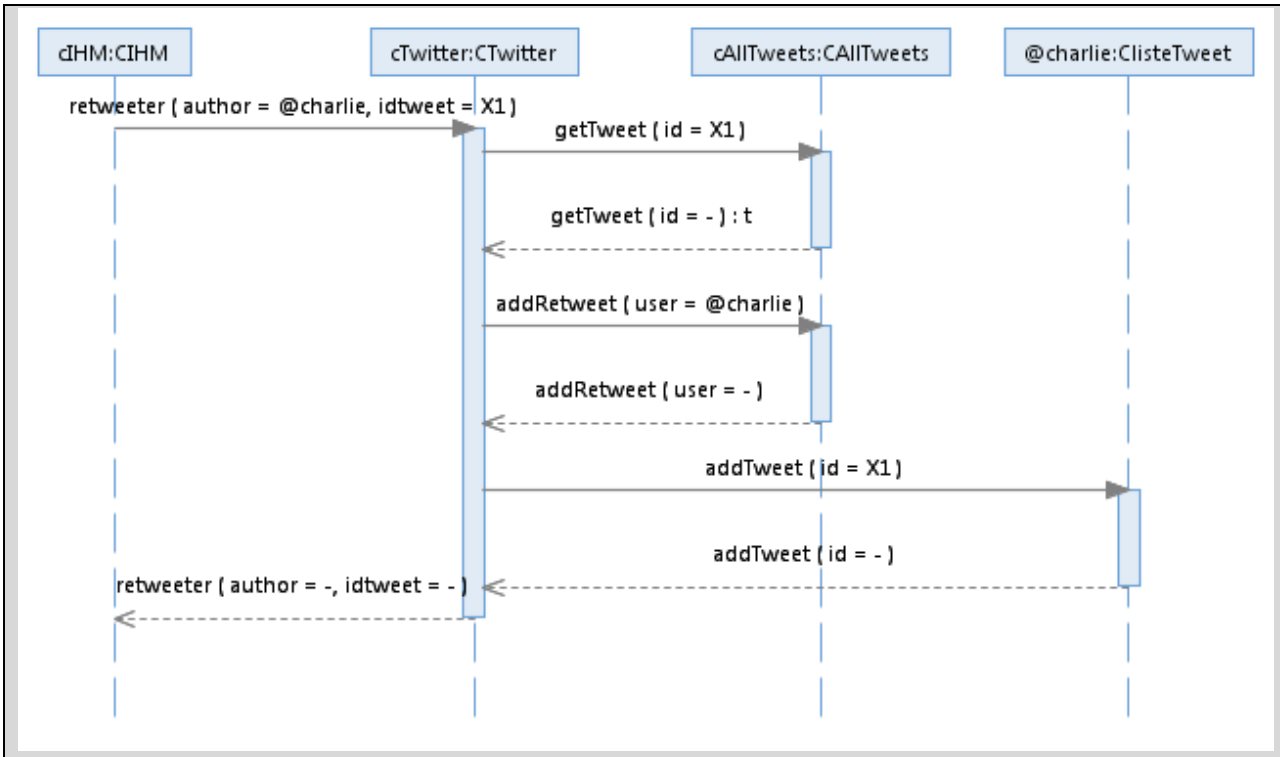
10% on voit la récupération de l'identifiant du tweet/c'est bien cet identifiant qui circule dans la suite.

20% on le diffuse sur @Alice

20% on retrouve bob via les abonnements (cohérent avec l'interface proposée en IV sinon 0)

10% on le diffuse sur @bob

d) Idem sur distinction ligne de vie « t »



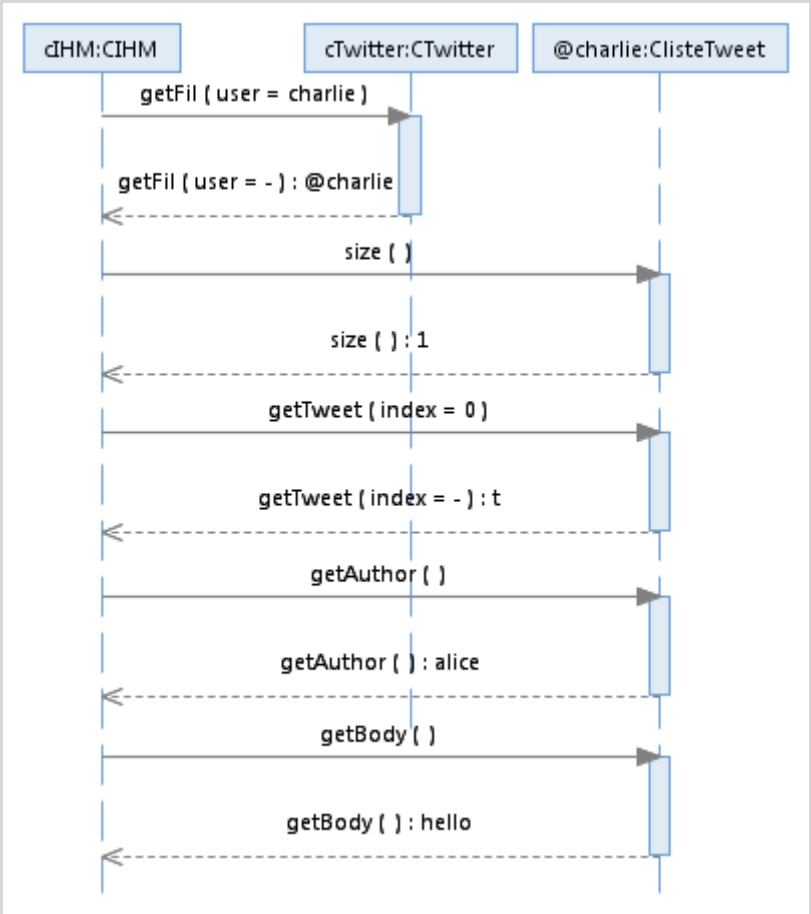
30% reweet englobe tout, paramètres cohérents

20% on récupère le tweet par identifiant sur CAllTweet

20% on pense à mettre à jour le tweet t

30% on diffuse le tweet sur charlie

c) Même commentaire que précédemment, on peut distinguer la ligne de vie t de @Charlie.

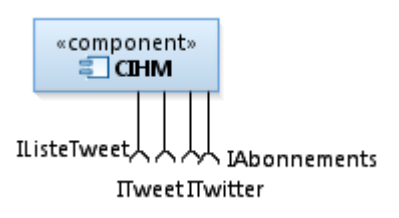


20% par message correct (source, destination, arguments)

Question V.2 : (1,5 pts)

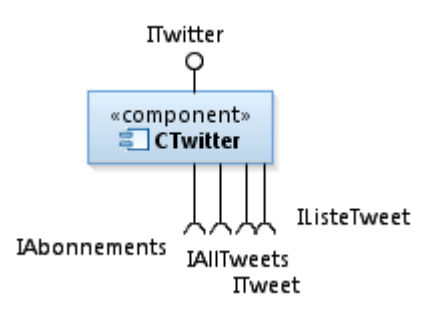
Sur un diagramme de composant en déduire les interfaces requises et offertes de CTwitter et de CIHM.

d)



10% par dépendance correctement identifiée.

-20% si on dépend de IAllTweets



20% offre ITwitter,

40% = 10% par dépendance