M1 : Ingénierie du Logiciel MI-017

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen Réparti 2eme partie

5 Janvier 2012 (2 heures avec documents : tous SAUF ANNALES CORRIGEES) Barème indicatif sur 21 points.

Problème: Conception de M@npower [21 Points]

Rappels simplifiés du cahier des charges :

Une entreprise d'intérim souhaite mettre en place une gestion informatisée de son activité. Elle propose un certain nombre de prestations (ménage, petits travaux, cours,…) et gère une base de personnel qu'elle envoie en mission chez des clients.

La description du contrat est réalisée en ligne par le client : celui-ci doit créer un compte, puis il peut définir diverses prestations à intégrer dans un contrat.

Une prestation est décrite par une ou plusieurs dates où l'intervention peut avoir lieu et une description de la compétence nécessaire pour cette prestation. On considère dans cet énoncé que chaque Prestation nécessite exactement une journée de travail. Par exemple une prestation :

« réparer évier salle de bain », plombier, 20 ou 21 Janvier 2012.

L'agent de la DRH cherche ensuite à affecter du personnel pour satisfaire la demande. La compagnie possède une base de personnel d'intérim et leurs compétences. Chaque employé possède une ou plusieurs compétences, avec pour chacune un certain niveau de compétence évalué entre 1 basique et 30 expert. Pour chaque employé on dispose également d'un agenda qui donne ses disponibilités pour l'entreprise.

Pour chaque prestation, la procédure consiste à allouer un employé compétent et disponible pour réaliser la tâche demandée. Le logiciel devra assister l'agent dans ce travail en proposant une liste du personnel adapté en termes de compétences et disponibilité pour chaque tâche.

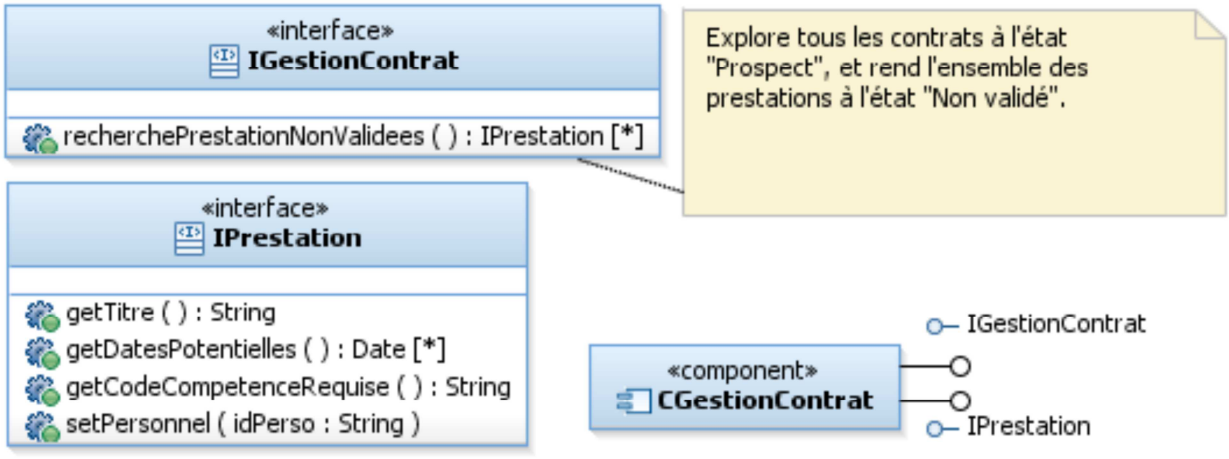
C'est cette procédure d'affectation des personnels aux prestations qui sera étudiée dans cet énoncé.

Partie I : Gestion des contrats (3,5 points)

Le composant de gestion des contrats a pour responsabilité de gérer l'établissement et le suivi des contrats par le client. Les contrats sont construits par le client via une page web dédiée. Cette partie traitant la construction des contrats par les clients ne sera pas étudiée dans cet énoncé.

La page web du client, représentée par un composant **CClientIHM** utilise le composant de gestion des contrats **CGestionContrat** pour construire la description du client, des contrats et des prestations.

NB : Les opérations de construction utilisées pour cela ne seront pas détaillées dans cet énoncé. On se concentre ici sur les interfaces offertes par **CGestionContrat** utiles pour gérer l'affectation des personnels aux prestations.



Q1 (1 point): Pourquoi créer un composant **CGestionContrat**, plutôt que de faire interagir directement l'IHM de l'Agent **CAgentIHM** avec le composant gérant la page web des clients **CClientIHM** ?

Pour permettre de faire évoluer indépendamment les deux IHM.

Là on a CAgent et CClient qui dépendent de CContrat, mais pas dépendances (directes) entre CAgent et CClient. Comme ce sont des vues, on suppose qu'elles évoluent féquemment, CGestionContrat est plus un composant de nature Modèle.

Les réponses « pour avoir un composant bout de chaine », pour factoriser le code des deux IHM, pour éviter des cycles de dépendances, seront comptées 100

Vu la formulation de la question, il faut évaluer évaluer la clarté et la pertinence de la réponse.

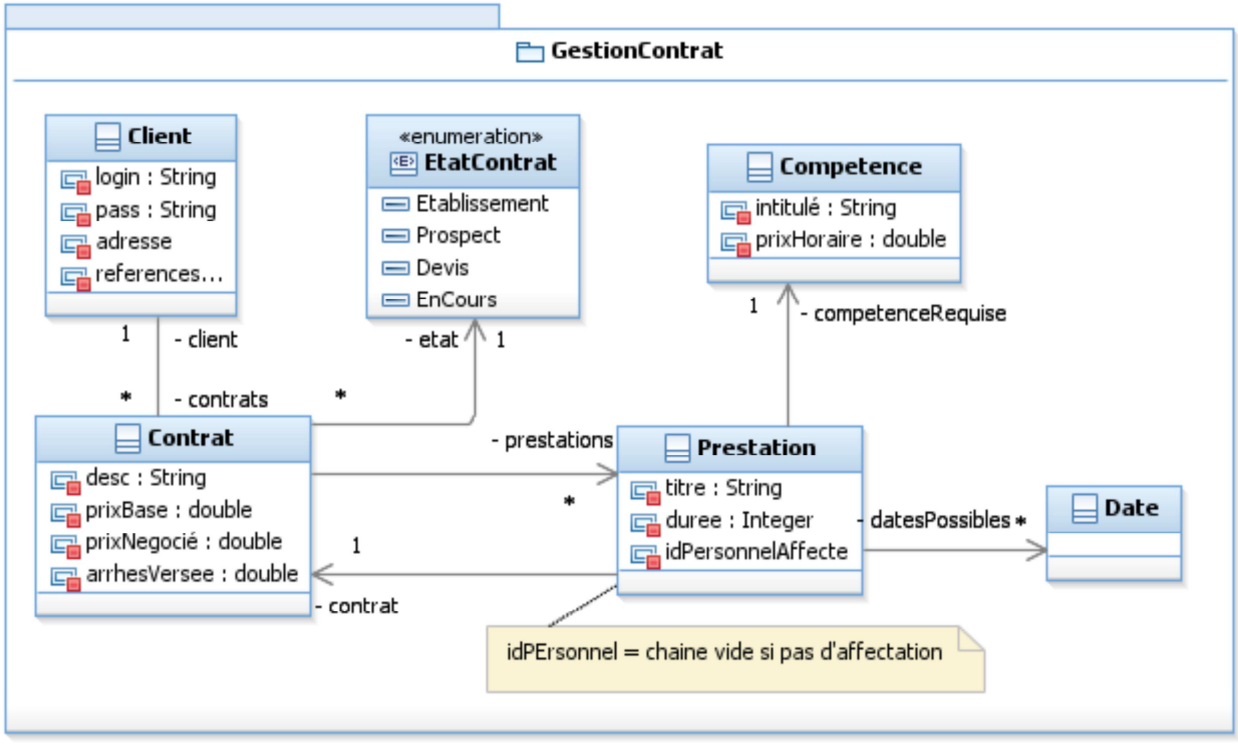
Barème : (sur 0,5 points)

0% hors sujet

50% peu convaincant

100% cohérent et correct

Q2 (2,5 points): On propose d'affecter la responsabilité des classes métier d'analyse suivantes à ce composant de Gestion des contrats :



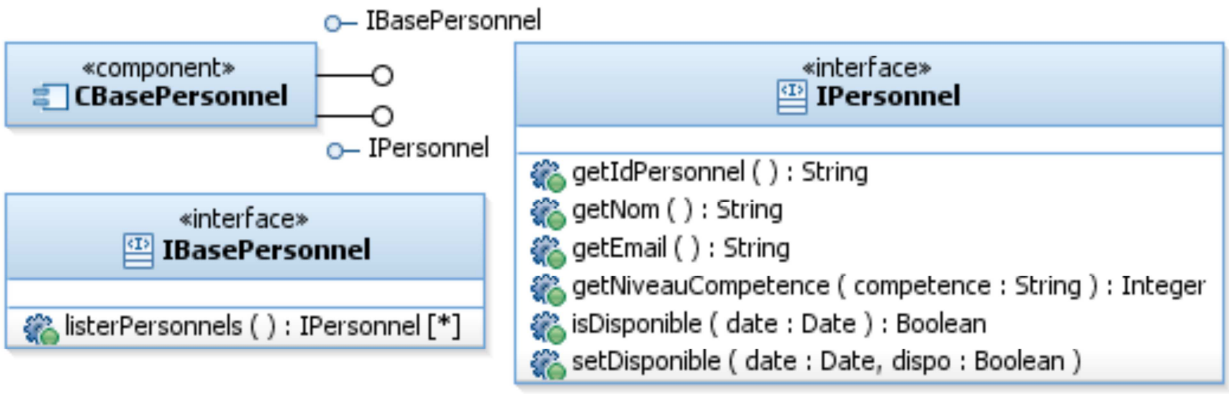
Complétez ce diagramme pour qu'il puisse servir de base à la conception détaillée du composant **CGestionContrat** en précisant les liens avec les interfaces offertes de **CGestionContrat**. On ne demande pas dans cette question de spécifier les opérations des classes.

On veut voir :

- 40% : Prestation implements IPrestation
- 60% : Ajout d'une classe gestionnaire, qui porte les * contrats gérés par le composant et qui implémente IGestContrats. (60%) . On donnera 30% pour une réponse qui ajoute un gestionnaire qui porte * Prestation plutôt que des contrats. 0% pour une réponse où Contrat implements IGestionContrat

Partie II : Recherche de Personnel adapté (6,5 points)

La base de personnel de l'entreprise est accessible via le composant **CBasePersonnel** décrit dans ce diagramme :



On propose de réaliser un nouveau composant **CRecherchePersonnel** qui s'appuie sur la base de personnel pour servir les besoins spécifiques à la recherche de personnel pour une prestation. A cette fin, on souhaite réaliser un composant flexible supportant la recherche de

personnel selon divers critères. Ce composant doit s'appuyer sur la base de personnel pour réaliser l'interface suivante :

Le critère de recherche est une chaîne de caractère composée de connecteurs booléens (AND, OR, NOT), de parenthèses pour les priorités, et de prédicats sur les personnels prenant la forme : « Disponible (date) », « NiveauCompetence (compétence) >= entier », ou « Nom = xxx ».

Par exemple, la requête :

(Disponible (20/01/2012) OR Disponible (21/01/2012))

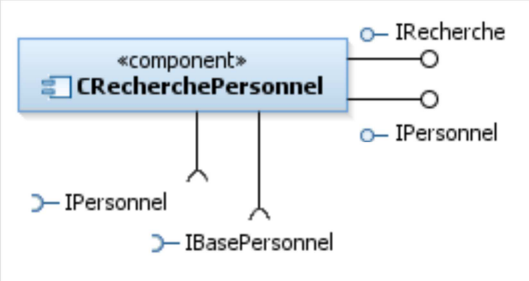
AND (NiveauCompetence (Maçonnerie) >= 5)

AND (NOT Nom=Toto)

Permet de trouver les personnels compétents en maçonnerie et disponibles le 20 ou 21 janvier, qui ne soient pas le personnel portant le nom « Toto ».

Q3. (1 point) Modélisez ce composant **CRecherchePersonnel** (interfaces requises et offertes) sur un diagramme de composants.

On veut voir :



30% offre IRecherche

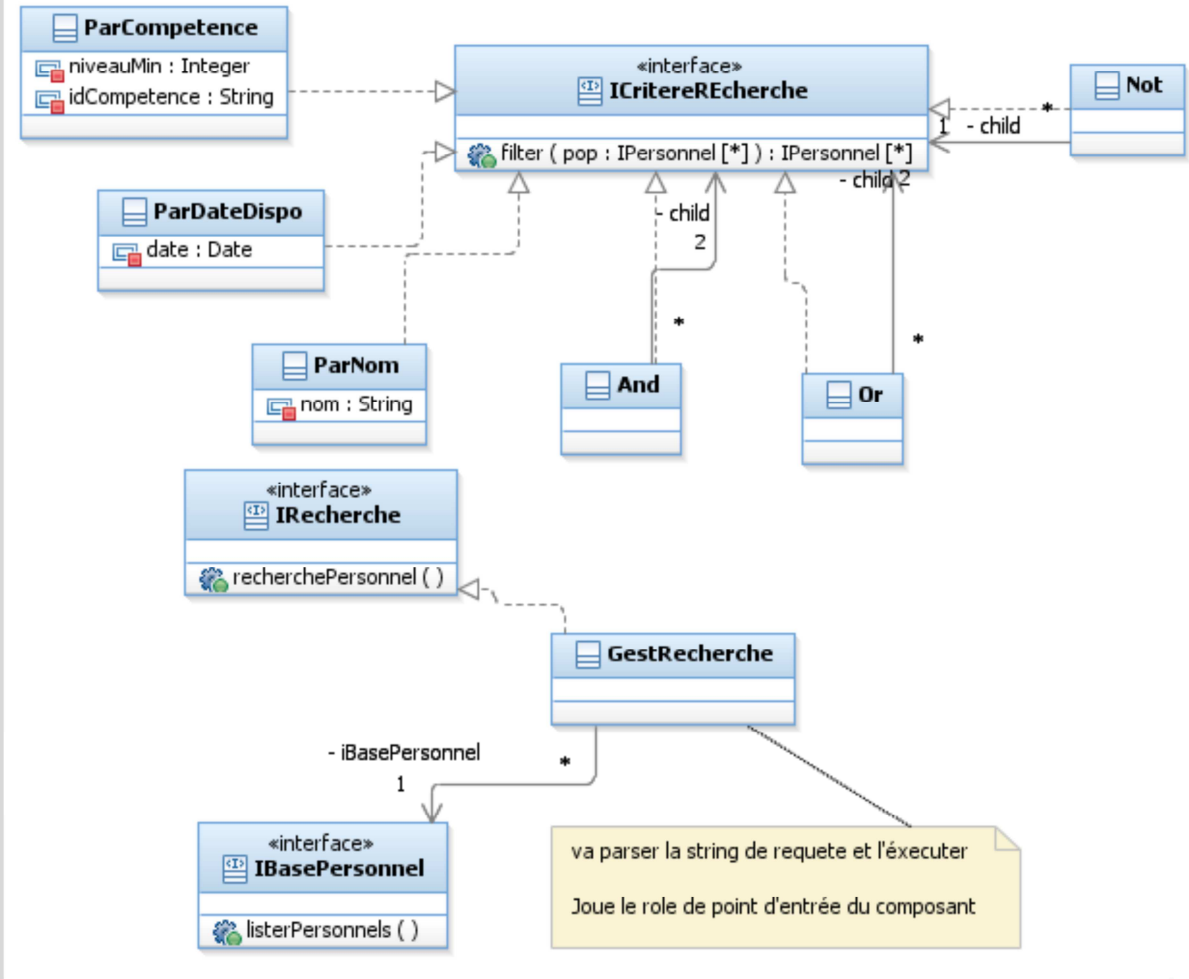
30% requiert IPersonnel

30% requiert IBasePerso

10% offre IPersonnel (ce n'est pas évident de le modéliser)

Q4. **(3 points)** Proposez une conception détaillée pour ce composant, à travers un diagramme de classe. On pourra s'appuyer un DP Composite pour la représentation orientée objet des requêtes.

On veut voir :



25% définition de GestRecherche, 15% implements IRecherche, 10% un lien sur la base de personnel

10 % par connecteur booléen (And, Or, Not) avec composite (i.e. implements une interface qui est aussi référencée)

15% par classe de critère, dont 5% sur l'implements de l'abstraction, et 10% sur les attributs définissant le critère.

On ne sanctionnera pas si c'est IRecherche qui est utilisé comme abstraction dans le DP composite, même si ce n'est pas ce qu'il faut faire. En général la sanction se fera sentir sur la question suivante.

Q5. **(2,5 points)** Annotez le diagramme précédent en précisant à l'aide de pseudo-code (ou de code Java) le corps des opérations critiques pour réaliser la recherche. On pourra définir une opération « filtrer(IPersonnel[*]) : IPersonnel[*] » qui fonctionne comme un filtre, c'est-à-dire qui élimine de l'entrée les éléments qui ne correspondent pas au critère de recherche courant.

On veut voir : filter(pop) :

20% : And : return child[1].filter(child[0].filter(pop)) (ou avec une intersection)

20% : Or : return child[0].filter(pop) union child[1].filter(pop)

15% : Not : return pop privé de child.filter(pop)

15% : ParCompétence :

IPerso [] result ;

for (IPerso p : pop) { if (p.getNiveauComp(this.idComp) > = this.niveauMin) result.add(p) }

return result

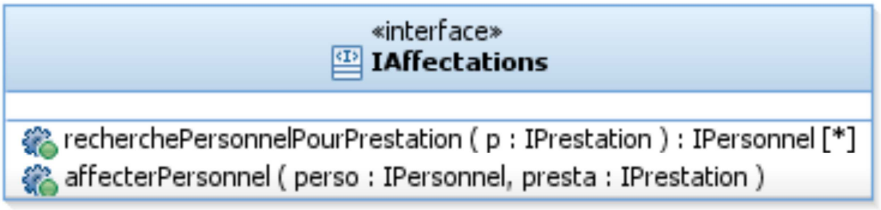
15% : Par nom : meme chose avec critère p.getNom() == this.nom

15% : Par date : même chose avec critère : p.isDispo(this.date())

On accordera des points dès que le principe est saisi, mais il faut formuler en pseudo-code ou java pour avoir 100%.

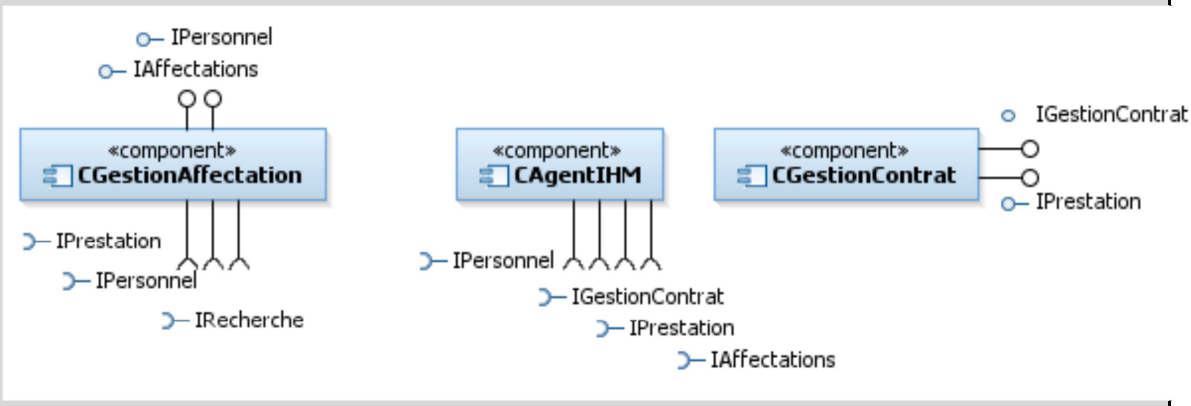
Partie III : Gestion des affectations (7 points)

L'IHM de l'agent **CAgentIHM** est un composant qui s'appuie sur le composant de gestion des contrats **CGestionContrat** (cf. Partie I) et sur un composant jouant un rôle de contrôleur **CGestionAffectation**, qui réalise l'interface suivante :



Q6. (1,5 points) Réalisez un diagramme de composant représentant ces deux composants **CAgentIHM** et **CGestionAffectation**. Le composant **CGestionAffectation** s'appuie sur le composant **CRecherchePersonnel** défini en partie II.

On veut voir :



NB : CGestioncontrat est fourni en Q1.

Barème (sur 110%)

CAgentIHM (sur 50%) :

- 15% requiert IAffectations
- 15% requiert IGestContrats
- 10% requiert IPersonnel
- 10% requiert IPrestation

CGestionAffectations (sur 60%) :

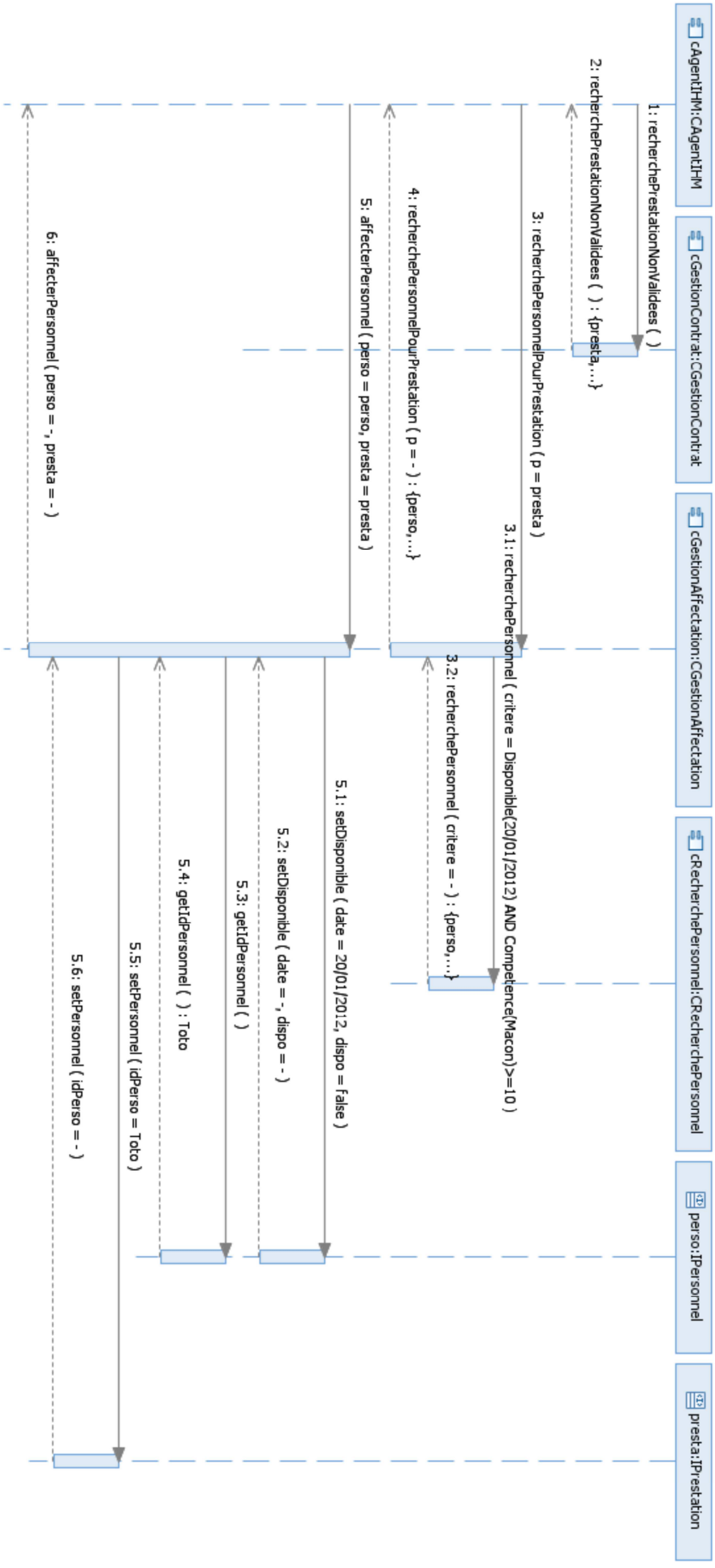
- 15% requiert IRecherche
- 15% offre IAffectations
- 10% requiert IPrestation
- 10% requiert IPersonnel
- 10% (c'est du bonus) offre IPersonnel

Q7. (4 points) Réalisez un diagramme de séquence montrant la séquence nominale correspondant à l'affectation d'un employé « Toto » à une prestation non satisfaite qui porte sur une intervention le 20/01/2012 nécessitant un plombier (niveau compétence >= 5).

On représentera les lignes de vie des composants **CAgentIHM**, **CGestionContrat**, **CGestionAffectation**, **CRecherchePersonnel** plus éventuellement des occurrences de **IPrestation** et **IPersonnel**.

Les invocations entre **CRecherchePersonnel** et **CBasePersonnel** ne seront donc pas modélisées (elles sont déjà largement décrites en partie II).

On fera explicitement figurer les valeurs des paramètres et les valeurs de retour sur les messages.


cAgentIHM:CAgentIHM
cGestionContrat:CGestionContrat
cGestionAffectation:CGestionAffectation
cRechercherPersonnel:CRechercherPersonnel
perso:IPersonnel
presta:IPrestation
1: rechercherPrestationNonValidees ( )
2: rechercherPrestationNonValidees ( ) : {presta,...}
3: rechercherPersonnelPourPrestation (p = presta)
3.1: rechercherPersonnel (critere = Disponible(20/01/2012) AND Competence(Macon)>=10)
3.2: rechercherPersonnel (critere = -) : {perso,...}
4: rechercherPersonnelPourPrestation (p = -) : {perso,...}
5: affecterPersonnel (perso = perso, presta = presta)
5.1: setDisponible (date = 20/01/2012, dispo = false)
5.2: setDisponible (date = -, dispo = -)
5.3: getIdPersonnel ( )
5.4: getIdPersonnel ( ) : Toto
5.5: setPersonnel (idPerso = Toto)
5.6: setPersonnel (idPerso = -)
6: affecterPersonnel (perso = -, presta = -)

10% rechPresta de IHM sur GestContrat

10% rechPersoPourPresta de IHM sur CGestAffect

20% recherche avec la chaîne critère appropriée, de GestAffect sur CRecherchePersonnel

15% affecterPersonnel de IHM sur CGestAffect

15% màj de la dispo de l'employé

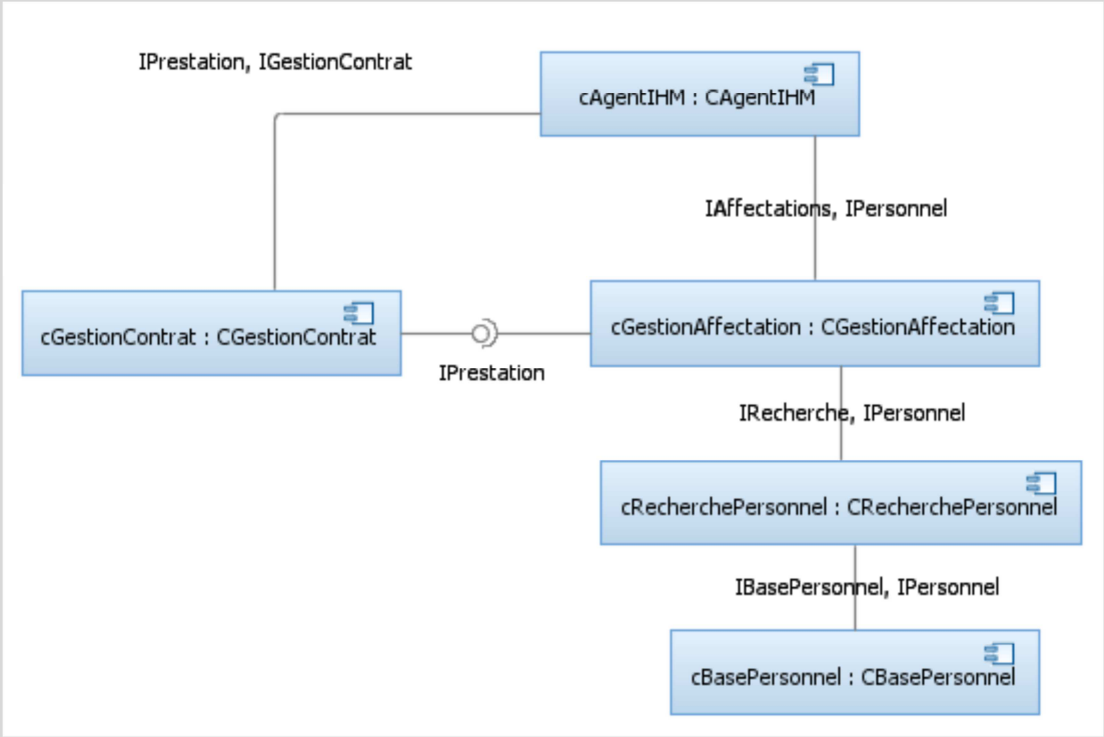
15% invoquer getIDPersonnel

15% màj de la prestation avec cet ID

Il faut voir les paramètre et valeurs de retour pour avoir tous les points. On sera indulgent sur la syntaxe pour les listes retournées, etc…

Q8. (1,5 points) Réalisez un diagramme de structure interne décrivant l'assemblage de tous les composants définis dans cet énoncé (**CAgentIHM**, **CGestionContrat**, **CGestionAffectation**, **CRecherchePersonnel, et CBasePersonnel**) dans la configuration nominale du système. Critiquez brièvement l'architecture proposée (qualités et défauts).

On veut voir :



/ !\ je ne sais pas pourquoi mais je n'arrive pas à représenter les deux connecteurs séparément.

Tous les liens sont orientés de haut en bas (composants en haut dépendent de ceux en dessous)

Sur 80% : 20% par lien orienté correctement :

- de IHM sur GestContrat (avec les deux connecteurs, IPresta et IGestContrat)
- de IHM sur GestAffect (dès qu'on a un connecteur)

- de GestAffect sur RecherchePersonnel (dès qu'on a un connecteur)
- de RecherchePersonnel sur BasePersonnel (avec deux connecteurs, IPersonnel et IBasePersonnel)

pénalité -20% si ce ne sont pas clairement des instances des composants qui sont représentées.

On accordera un bonus de 10% pour une dépendance matérialisée de GestAffectaion sur GestionContrat.

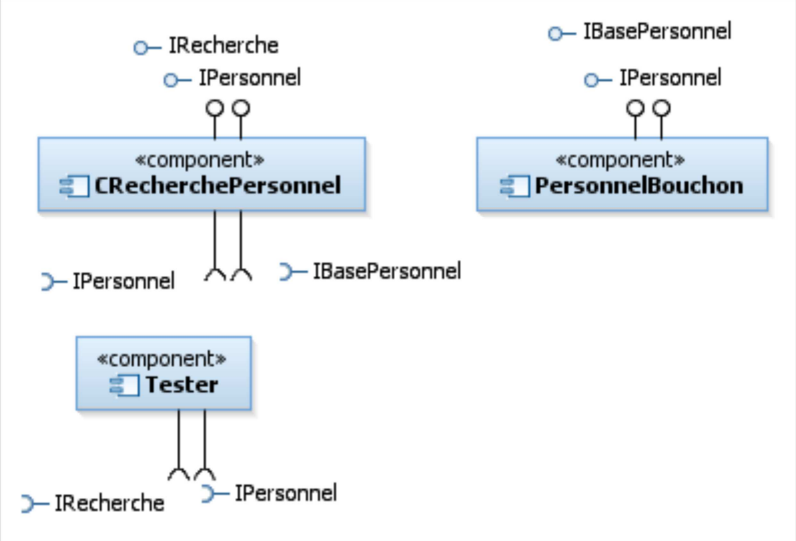
Sur 20% : Critique : (barème binaire, on donne les 20% dès que la critique est présente et que les commentaires formulés sont pertinents)

- Très bonne architecture, en couches, qui assure que l'on puisse substituer des couches basses sans le dire aux autres composants.
- L'appli complète est construite par assemblage de briques simples, c'est bien
- Pas de cycles de dépendances, c'est bien
- Je ne trouve pas beaucoup de défauts à cette archi, mais c'est la mienne :) .

Partie IV : Tests d'intégration pour CRecherchePersonnel (4 points)

Q9. (1 points) Définissez les composants utiles pour permettre de tester le composant **CRecherchePersonnel** (cf. Partie II) en isolation.

On veut voir :



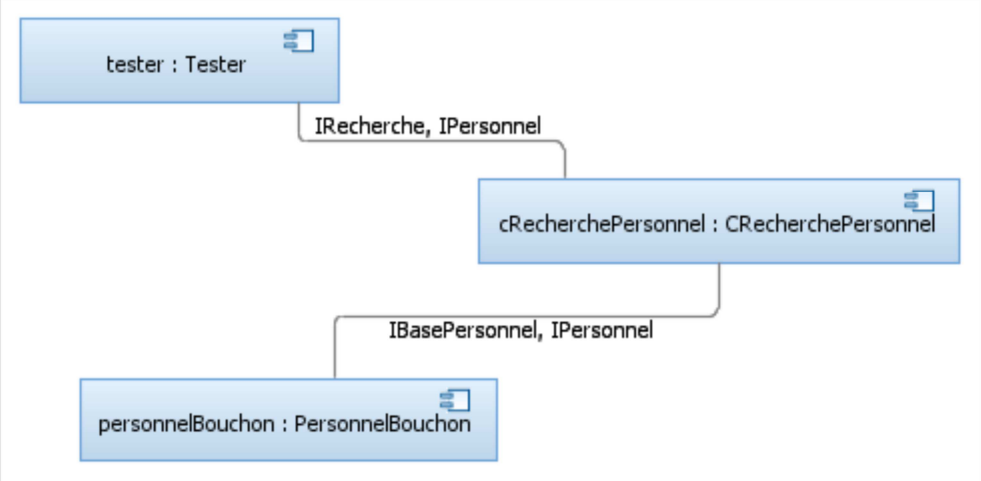
La question ne précise pas (malheureusement) qu'on attends un diagramme de composants.

70% il faut définir un ou des bouchons pour IPersonnel/IBasePersonnel

30% il faut un testeur

Q10. (1 point) A l'aide d'un diagramme de structure interne, représentez une configuration de ces composants permettant de tester **CRecherchePersonnel.**

On veut voir :



20% le testeur

40% le composant à tester

40% le(s) bouchon

Maximum 50% si ce n'est pas clairement un diagramme de structure interne.

Q11. (1,5 points) Ecrivez un test d'intégration pour ce composant **CRecherchePersonnel**. Précisez les données qu'il faut embarquer dans les éventuels composants/classes bouchon pour faire fonctionner ce test.

On veut voir :

IPerso[] propose = cRech.recherche (« Nom=Toto »)

Assert (propose.size() == 1 && propose[0].getNom() == Toto)

Il faut donc que le bouchon embarque un Personnel Toto.

40% une invocation à l'opération : recherche (critère) avec une string bien formée

40% présence d'un résultat attendu

20% données du bouchon cohérentes avec le test