

Contrôle Long UML

Pierre Gérard

pierre.gerard@univ-paris13.fr

*DUT Informatique S2D
Université de Paris 13*

Résumé

Ce contrôle dure 3 heures. Aucun document n'est autorisé. Si vous êtes amenés à émettre des hypothèses, veuillez les expliciter sur la copie.

1 Questions de cours (3 pts)

La réponse à chaque question devra être claire et concise : en quelques lignes, vous devez expliquer l'essentiel. Les réponses devront être justifiées.

Question : En quoi diffèrent principalement les approches telles que RUP de celles telles que XP ? Qu'ont-elles en commun ?

RUP est un processus complexe adapté aux gros projets nécessitant des équipes de développement importantes. Le développement y est très rationalisé avec une emphase mise sur la documentation. RUP est plutôt adapté aux petites équipes sur des projets plus modestes. Elle met l'accent sur les pratiques plutôt que sur une documentation pléthorique. Elle ont néanmoins en commun l'approche incrémentale du développement. Barème : 1 pt.

Question : Donnez trois exemples possibles d'applications pouvant tirer parti du Design Pattern « Singleton »

Exemples : spooler d'impression, gestionnaire graphique, générateur de nombres aléatoires (à chaque fois, il est nécessaire de garantir l'unicité d'une instance de la classe). Barème : 0.5 pour au moins un exemple s'il est justifié par une ligne ou deux (0.25 sinon), 0.25 pour chaque exemple supplémentaire

Question : Si vous deviez réutiliser un code source développé et maintenu par ailleurs, mais que l'interface des classes de ce code source ne corresponde pas tout à fait à celle que vous escomptiez, comment vous-y prendriez-vous ? Quels sont les avantages principaux que vous escomptez de votre solution ?

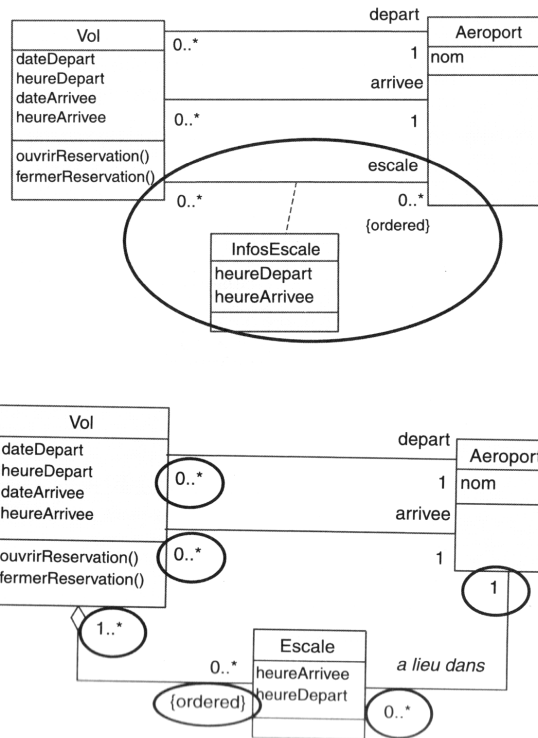
La solution est d'employer le pattern « adapter ». Ainsi, on peut réutiliser du code sans le modifier, de manière à profiter des évolutions ultérieures du code réutilisé. Barème : 1 pt.

2 Diagramme de classes (3 pts)

Une compagnie aérienne assure la desserte d'un certain nombre de destinations. Un vol a un numéro, un horaire de départ et d'arrivée prévus, un aéroport de départ, un aéroport de destination et éventuellement un ensemble d'escales avec les horaires de départ et d'arrivée prévus pour l'escale. Les escales des différents vols ont lieu dans des aéroports. On connaît le code international et le nom des aéroports. En plus des horaires prévus, on garde une trace des conditions réelles de vol. Ces informations comportent les heures de départ et d'arrivée réels, ainsi que le nombre de places adultes et enfants disponibles. On enregistre également les horaires réels des escales.

Question : Etablir le diagramme de classes pour la gestion des vols.

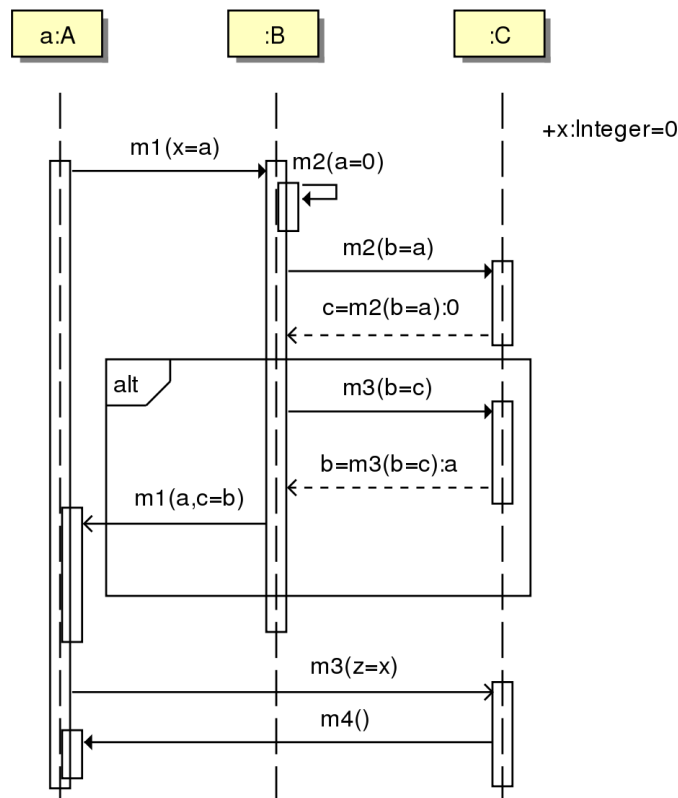
DANS LA CORRECTION, IL MANQUE QUELQUES ATTRIBUTS (VOIR BAREME). 2 solutions sont possibles (avec ou sans classe association) mais on préférera celle avec :



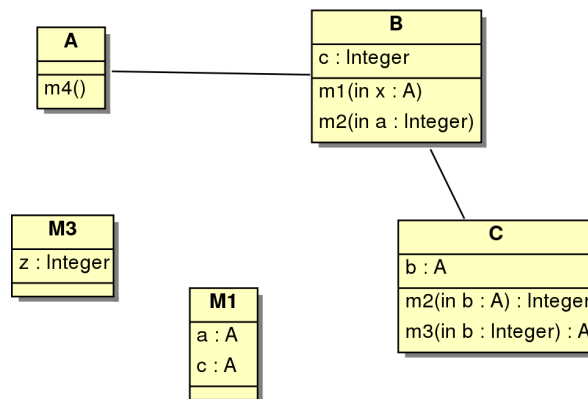
Barème :

- 0.25 Classe Vol
- 0.25 Attributs horaires prévus de la classe vol
- 0.25 Attributs conditions réelles de la classe vol (horaires, nb de places enfants et adultes)
- 0.25 Classe Aéroport avec attributs
- 0.25 Associations départ
- 0.25 Multiplicités de départ
- 0.25 Associations arrivée avec multiplicités
- 0.25 Multiplicités de arrivée
- 0.25 Classe Escale avec attributs (horaires prévus)
- 0.25 Horaires réels des escales
- 0.25 Classe association et assoc escale / ou bien / 2 associations
- 0.25 Multiplicités correspondantes

3 Diagramme de séquences et diagramme de classes (3 pts)



Question : Construisez un diagramme des classes cohérent avec le diagramme de séquences ci-dessus. On veut un diagramme des classes simple mais qui définisse tous les éléments utilisés dans le diagramme de séquence.



ATTENTION ERREUR DANS LA CORRECTION : L'attribut *b* est dans la classe *B* et pas *C*!!!

C'est le même exo que dans le contrôle court (que j'avais corrigé de manière détaillée en cours). On peut donc se permettre d'être un peu exigeant. Barème :

- 0.25 pour la classe signal *M1*
- 0.5 pour les attributs de *M1* (avec leurs types) : 0.25 pour chaque
- 0.25 pour la classe signal *M3*
- 0.25 pour les attributs de *M3* (avec leurs types)
- 0.25 pour *B* : *b* (avec le type correct)
- 0.25 pour *B* : *c* (avec le type correct)
- 0.25 pour *B* : *m1* (avec tout correct)
- 0.25 pour *B* : *m2* (avec tout correct)
- 0.25 pour *C* : *m2* (avec tout correct)
- 0.25 pour *C* : *m3* (avec tout correct)

0.25 pour A : :m4 (avec tout correct)

A chaque fois, tout correct signifie que tous les arguments doivent y être, avec les bon types (sinon : 0)

4 Diagramme d'états-transitions (4 pts)

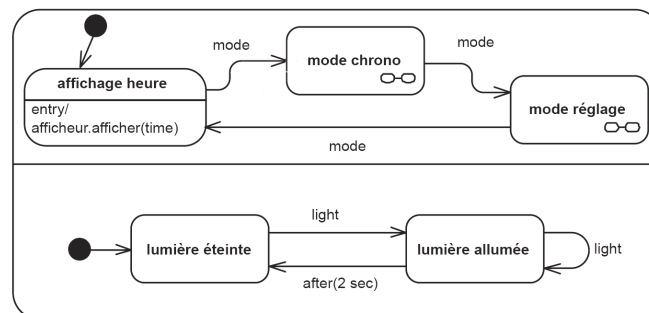
Une montre digitale propose une fonction horloge et une fonction chronomètre. Elle est munie de quatre boutons, l'appui sur chacun des boutons donnant lieu à un événement :

- Le bouton « light », quand il est pressé, allume une lumière. La lumière reste allumée pendant une durée de deux secondes, après quoi elle s'éteint à nouveau. Un appui sur ce bouton alors que la lumière réinitialise la durée l'allumage de la lumière.
- Le bouton « mode » active successivement les trois modes principaux de la montre : l'affichage de l'heure courante, le mode chronomètre et le mode réglage (pour mettre à jour l'heure courante).
- En mode chronomètre, le bouton « start/stop » lance et arrête le chronomètre. En mode réglage, il active successivement les fonctions de réglage de l'heure, des minutes ou des secondes.
- En mode réglage, le bouton « set » incrémente les heures, les minutes ou les secondes. En mode chronomètre, il remet le chronomètre à zéro.

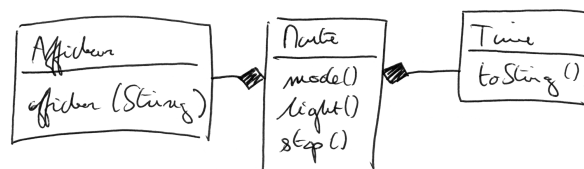
L'affichage est géré par un objet ayant le rôle d'« afficheur » et disposant d'une opération afficher qui permet d'afficher la chaîne de caractères qu'on lui donne en paramètre. Le décompte du temps est géré par un objet ayant le rôle de « time » et disposant d'une opération toString retournant une chaîne de caractères correspondant à l'heure courante. Le décompte du temps chronométré est géré par un objet avec le rôle de « chrono » et qui dispose des opérations stop, start, init et enfin toString retournant une chaîne de caractères. L'afficheur dispose également de méthodes indicateurHeure, indicateurMinute et indicateurSeconde pour afficher l'indicateur adéquat. La classe « Time » dispose de méthodes incrementeHeure, incrementeMinute et initSeconde.

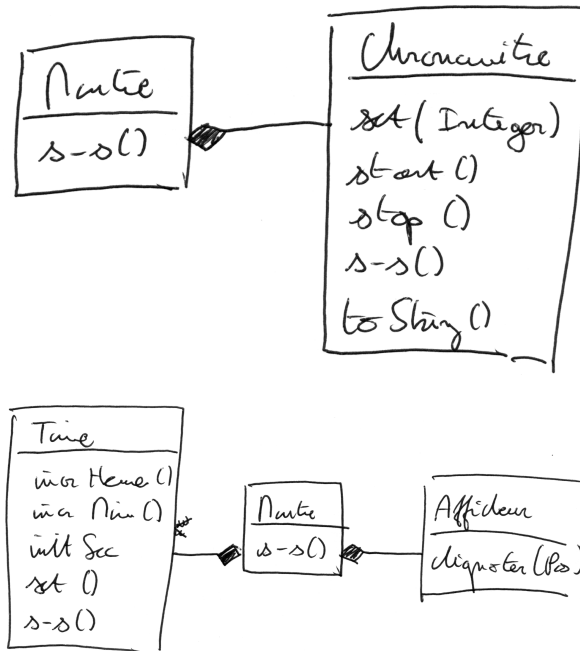
Question : Donnez un diagramme de classes permettant de modéliser une montre fonctionnant comme décrit ci-dessus

On a déjà modélisé les changements d'état de la montre liés à l'emploi des boutons « mode » et « light » (voir figure ci-dessous). Certains états sont composites, comme « Mode Chrono » et « Mode réglage ». On considère que « Mode Chrono » est déjà modélisé mais il reste le « Mode réglage » à détailler.



le solution est un mix entre tout ça, sauf qu'il n'y a pas de s-s autre part que dans Montre :



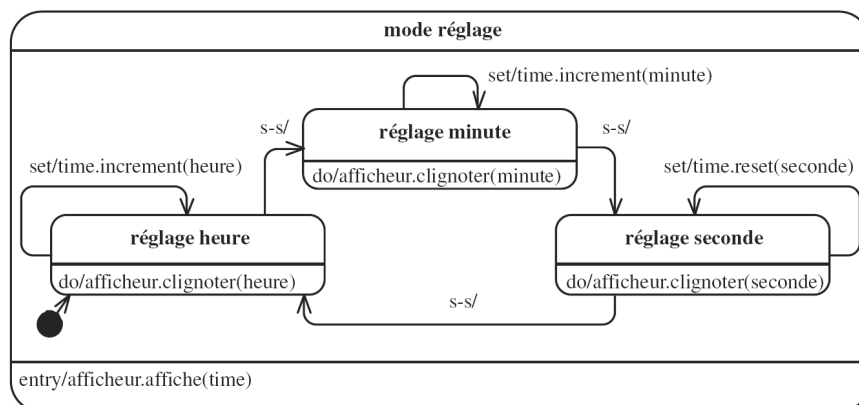


Barème (1.5 pts) :

- 0.25 pour avoir identifié toutes les classes
- 0.25 pour des liens de composition
- 0.25 pour mode, light, set et s-s dans montre
- 0.25 pour les autres méthodes de Chrono
- 0.25 pour les autres méthodes de Time
- 0.25 pour les autres méthodes de Afficheur

En mode réglage, le bouton start-stop active successivement les fonctions de réglage de l'heure, des minutes ou des secondes. Selon la fonction de réglage courante, l'afficheur affiche un indicateur sous les heures, les minutes ou les secondes. Le bouton set incrémente l'heure courante d'une heure en mode réglage de l'heure, d'une minute en mode réglage des minutes, ou remet les secondes à zéro en mode réglage des secondes. L'heure affichée ne change plus dès qu'on entre dans le mode réglage.

Question : Proposez un diagramme de d'états/transitions pour détailler l'état composite « Mode réglage »



Barème (2.5 pts) :

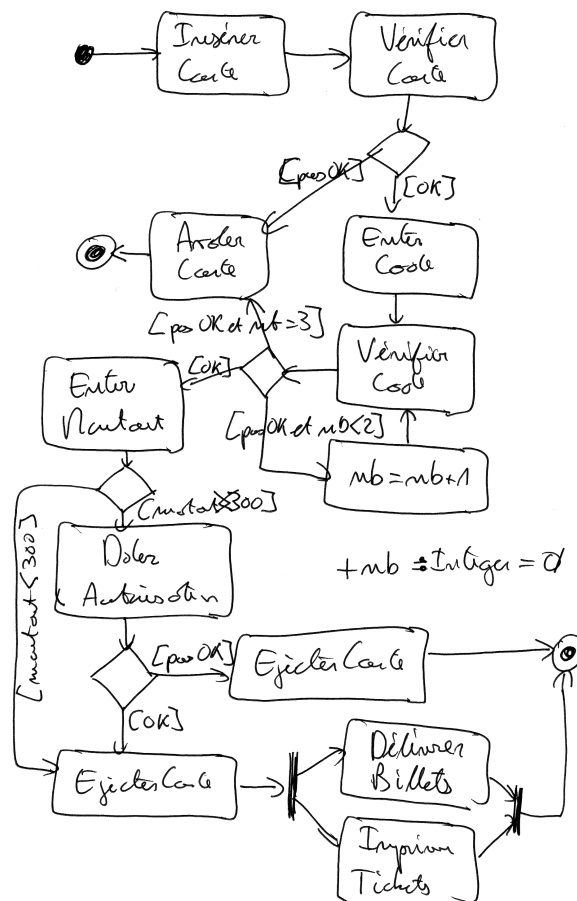
- 0.25 Pour la syntaxe (pas de rectangles ou de cercles)
- 0.25 Pour avoir mis un état de départ
- 0.25 Pour avoir mis 3 sous-états
- 0.25 Pour les transitions de l'un à l'autre (peu importe que ça boucle mais si ça ne boucle pas, il faut un état final)

- 0.25 Pour que ces transitions soient déclenchées par l'événement start-stop
- 0.25 Pour les boucles sur les états
- 0.25 Pour que ces transitions soient déclenchées par l'événement set
- 0.25 Pour avoir mis des activités qui ont le bon effet sur l'heure
- 0.25 Pour pour le entry/afficherheure (on accepte aussi les do/afficher)
- 0.25 Pour les afficherIndicateur()

5 Diagramme d'activités (3 pts)

Dans le cadre de la mise au point d'un Distributeur Automatique de billets (DAB), on cherche à établir l'enchaînement des activités des différents intervenants : le client, le distributeur et la banque. Le client insère sa carte avant d'entrer son code. La vérification du code peut être faite localement sur le DAB puisque la puce de la carte contient les informations disponibles. S'il est autorisé, le client peut demander un montant particulier qui sera délivré par le DAB à condition que son solde soit suffisant. L'opération de vérification du solde ne peut pas être effectuée sur place et requiert l'intervention du serveur de la banque. Avant que la carte ne soit éjectée pour que le client la récupère, on affiche le solde du compte (à moins bien sûr que le code n'ait pas été valide).

Question : Etablissez le diagramme d'activité correspondant au scénario ci-dessus.



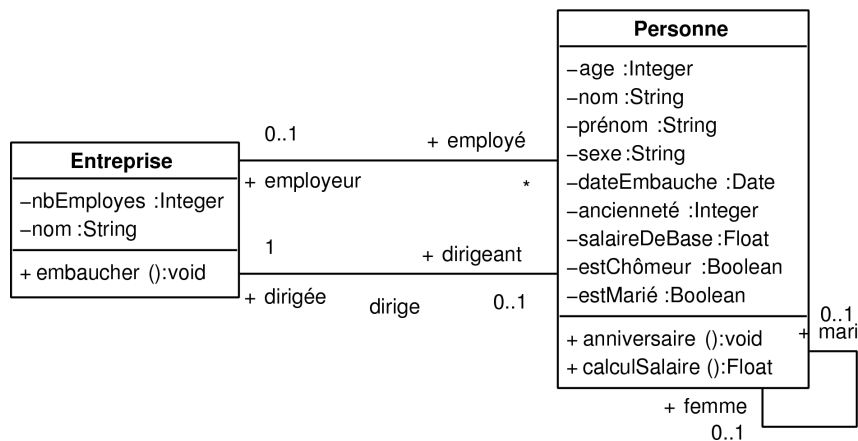
Cette solution est la plus complexe. Comme l'énoncé ne dit rien sur la manière de s'y prendre quand le code est refusé ou quand l'autorisation est refusée, ça peut être plus simple (éjection de la carte et fin). Barème :

- 0.5 Pour le respect des notations (pas de rectangles, pas de cercles etc...)
- 1 Pour les activités : insérer carte; entrer code; verif code; demander montant; verifier solde; afficher solde; ejecter carte
- 0.5 Pour l'enchaînement global des activités
- 0.25 Pour l'alternative du code carte
- 0.25 Pour l'alternative du solde du compte

0.5 Pour avoir mis les activités dans les bons couloirs (ou alors avoir mis l'acteur entre parenthèse dans les activités)

6 OCL (4 pts)

Considérons le diagramme de classes suivant :



Barème : 0.5 pts par question. Si l'interprétation d'une contrainte est trop « mot à mot », ne mettre que 0.25.

Question : Qu'expriment les contraintes suivantes ?

```

context Personne
inv: femme->notEmpty() implies
    (femme.sexe = 'femme' and self.sexe = 'homme')
inv: mari->notEmpty() implies
    (mari.sexe = 'homme' and self.sexe = 'femme')
  
```

Toutes les femmes par le lien du mariage sont de sexe féminin. Et tous les hommes par le même lien sont de sexe masculin.

Question : Qu'exprime la contrainte suivante ?

```

context Entreprise
inv: self.employé->forall(age<=65) and self.employé->forall(age>=16)
  
```

Tous les employés doivent être âgés au moins, de 16 ans, et au plus de 65 ans.

Question : Est-ce que la contrainte suivante est une bonne simplification de la précédente ?

```

context Personne
inv: (age<=65) and (age>=16)
  
```

Non pas du tout car ici, toutes les personnes sont concernées pas seulement celles qui sont employées.

Question : Qu'exprime la contrainte suivante ?

```

context Personne
def: estEmployé: not employeur->isEmpty()
  
```

Cette contrainte définit un attribut (dérivé) dont la valeur dépend du lien de la personne avec une entreprise quelconque.

Question : Qu'exprime la contrainte suivante ?

```

context Personne
inv : femme<>self and homme<>self
  
```

■ *Personne ne peut être marié avec soi-même.*

Question : Donnez une contrainte OCL permettant de définir l'attribut « estMarié » grâce aux relations des personnes entre elles.

■ *On peut prendre modèle sur la contrainte définissant estEmployé (plus haut)*
context Personne def : estMarié : not (femme->isEmpty() and homme->isEmpty())

Question : Que pensez-vous de la contrainte suivante, en regard des contraintes exprimées plus haut ?

```
context Entreprise
inv: employé.includes(dirigeant)
```

■ *Cette contrainte impose aux dirigeants d'être employés (0.25 pt). Ainsi, toutes les contraintes s'appliquant aux employés (age etc) s'appliquent aux dirigeants (0.25 pt)*

Question : L'opération « anniversaire() » est appelée automatiquement chaque année à la date d'embauche de la personne, si celle-ci est employée. Quels devraient être ses effets ? Exprimez le comportement de cette méthode en OCL

■ *A chaque anniversaire d'embauche, il fait augmenter l'ancienneté (0.25 pt) Contrainte (0.25 pt) :*
context Personne : :anniversaire() post : ancienneté = ancienneté + 1