

Contrôle Court UML

Pierre Gérard

pierre.gerard@univ-paris13.fr

*DUT Informatique S2
Université de Paris 13*

Résumé

Ce contrôle dure 3 heures. Aucun document n'est autorisé. Observez la plus grande rigueur dans les notations. Si vous êtes amenés à émettre des hypothèses, veuillez les expliciter sur la copie. Le barème est donné à titre indicatif.

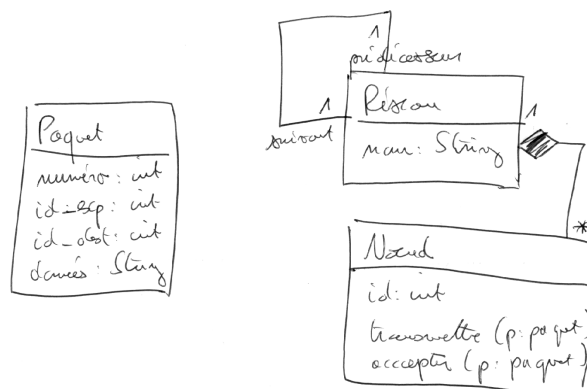
1 Diagramme de classes et d'objets (4 pts)

Lorsque des données sont transmises par un réseau, elles sont découpées en paquets qui contiennent chacun une partie des données. Un paquet comporte :

- le numéro du paquet ;
- l'identifiant de l'expéditeur ;
- l'identifiant du destinataire ;
- les données transmises.

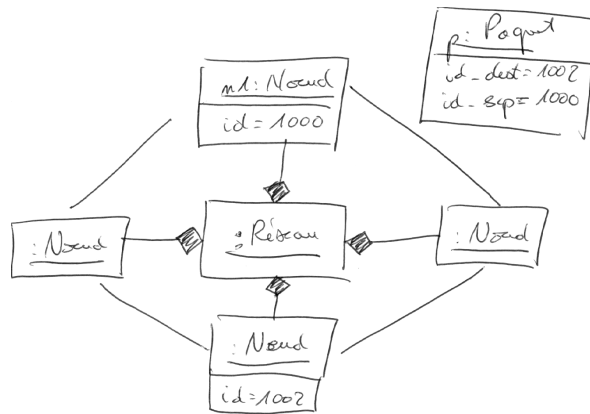
Un réseau en anneau est composé de noeuds reliés les uns aux autres par des relations précédent/suivant : chaque noeud ne connaît que le noeud qui le suit dans la chaîne ainsi celui qui le précède. La chaîne est fermée pour former un anneau. Un réseau est identifié par son nom et chaque noeud possède un identifiant unique dans le réseau. Les paquets sont émis par leur expéditeur et transitent de noeud en noeud suivant jusqu'à arriver à leur destinataire. Chaque noeud peut se voir transmettre un paquet par son prédécesseur dans l'anneau. S'il en est le destinataire, il peut l'accepter.

Question : Donnez un diagramme de classes pour modéliser les réseaux en anneau tels qu'ils sont décrits ci-dessus. Utilisez votre bon sens pour déterminer les multiplicités.



Barème () :

Question : Proposez un diagramme d'objets conforme à la description ci dessus. Ce diagramme devra comporter au moins 6 objets, avec au moins un de chaque classe.



Barème () :

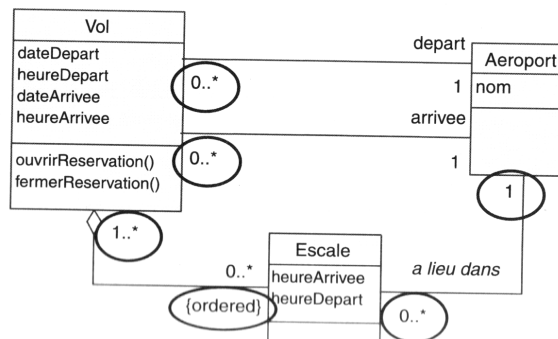
2 Diagramme de classes (3 pts)

Des interviews d'experts métier ont permis de mettre en évidence les éléments suivants :

- Un vol a un aéroport de départ et un aéroport d'arrivée ;
- Un vol a une heure de départ et une heure d'arrivée, ainsi qu'une date de départ et une d'arrivée ;
- Un vol peut comporter des escales dans des aéroports ;
- Les escales interviennent dans un ordre déterminé ;
- Une escale a une heure d'arrivée et une heure de départ ;
- Chaque aéroport a un nom ;
- On peut ouvrir (et fermer) à la réservation chacun des vols.

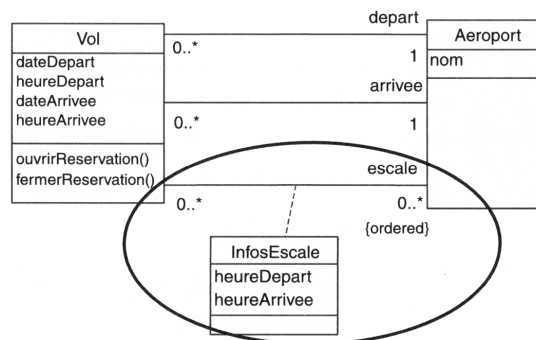
Question : Proposez deux solutions pour modéliser les éléments ci-dessus :

- Un diagramme de classe sans classe-association ;



Barème () :

- Un autre avec une classe association.



Barème () :

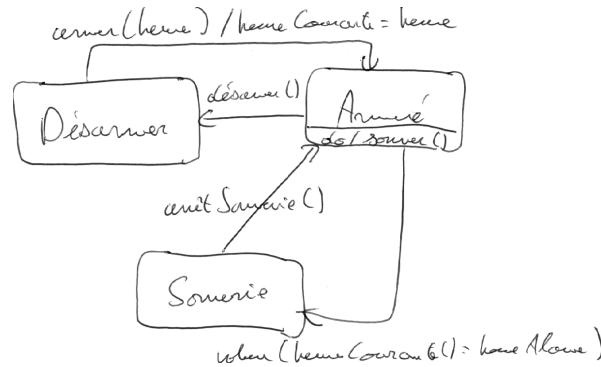
3 Diagramme d'états/transitions (3 pts)

Considérons un réveille-matin simplifié :

- On peut armer ou désarmer l'alarme;
- Quand l'heure courante devient égale à l'heure de l'alarme, le réveille sonne;
- On peut interrompre la sonnerie;
- La sonnerie s'interrompt automatiquement après 30 minutes.

Un réveil matin dispose des opérations armer(heure), désarmer(), arrêtSonnerie(), sonner() et heureCourante(). Il dispose d'un attribut heureAlarme.

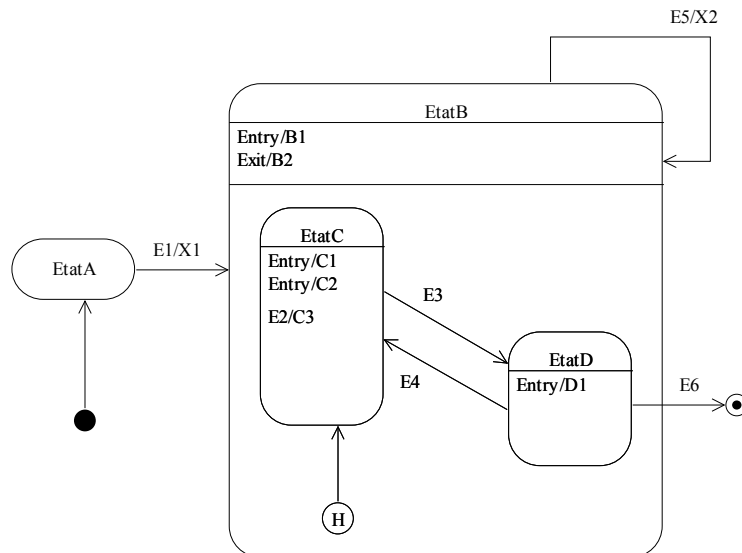
Question : Proposez un diagramme d'états/transitions pour modéliser la dynamique d'un réveil-matin.



Barème () :

4 Diagramme d'états hiérarchique (2 pts)

Considérons le diagramme d'états suivant :



Question : Considérant que l'état courant est « Etat A », quelles activités sont produites par la séquence d'événements E1, E2, E3, E5, E3, E4, E6. Présentez votre réponse dans un tableau semblable à celui ci-dessous.

Evénement	Activités résultantes	Nouvel état
E1		
E2		
...		

Événement	Activités résultantes	Nouvel état
E1	X1, B1, C1, C2	EtatC
E2	C3	EtatC
E3	D1	EtatD
E5	B1, X2, B1, D1	EtatD
E3	rien	EtatD
E4	C1, C2	EtatC
E6	rien	EtatC

Barème () :

5 OCL (4 pts)

Considérons les contraintes OCL suivantes :

```
// contrainte 1
context Compte : débiter(somme : int)
pre: somme > 0
post: solde = solde@pre - somme

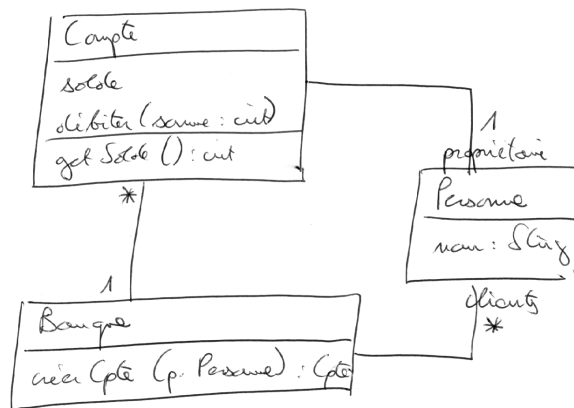
// contrainte 2
context Compte
inv: banque.clients -> includes (propriétaire)

// contrainte 3
context Compte::getSolde() : int
post: result = solde

// contrainte 4
context Banque :: créerCompte(p : Personne) : Compte
post: result.oclIsNew() and
      compte = compte@pre -> including(result) and
      p.compte = p.compte@pre -> including(result)

// contrainte 5
context Personne
inv: Personne.allInstances() -> forAll(p1, p2 |
      p1 <> p2 implies p1.nom <> p2.nom)
```

Question : Proposez un diagramme de classes compatible avec ces contraintes OCL



Barème () :

Question : Que signifient chacune de ces contraintes ?

Question : Ecrire des contraintes OCL pour les énoncés suivants :

– Le solde d'un compte doit toujours être positif ou nul ;

context *Compte inv : solde > 0*

Barème () :

–

– Si une personne possède au moins un compte bancaire, alors elle est cliente d'au moins une banque.

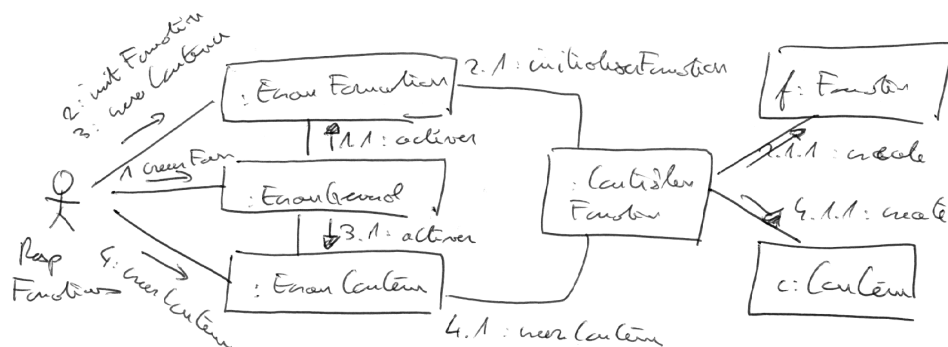
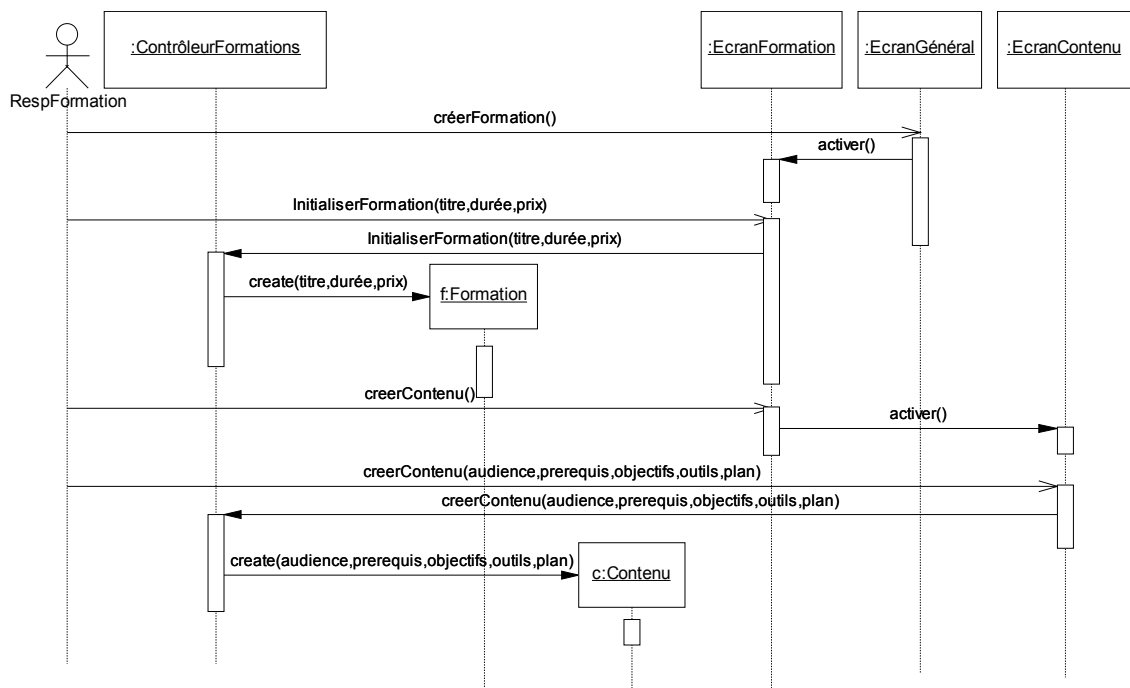
context *Personne inv : compte -> notEmpty() implies banque -> notEmpty()*

Barème () :

–

6 Diagrammes de séquences et de communication (2 pts)

Question : Transformez le diagramme suivant en un diagramme de communication équivalent.



Barème () :

–

7 Questions de cours (2 pts)

I *Barème () :*
—

Question : Suivant la terminologie employée dans la méthode utilisée dans le projet UML (Monoply), décrivez les différents types de classes participantes. Expliquez le principe présidant à cette distinction entre types de classes.

Question : Que sont les Design Patterns ? A quoi sont-ils utiles ?