Exercice 1

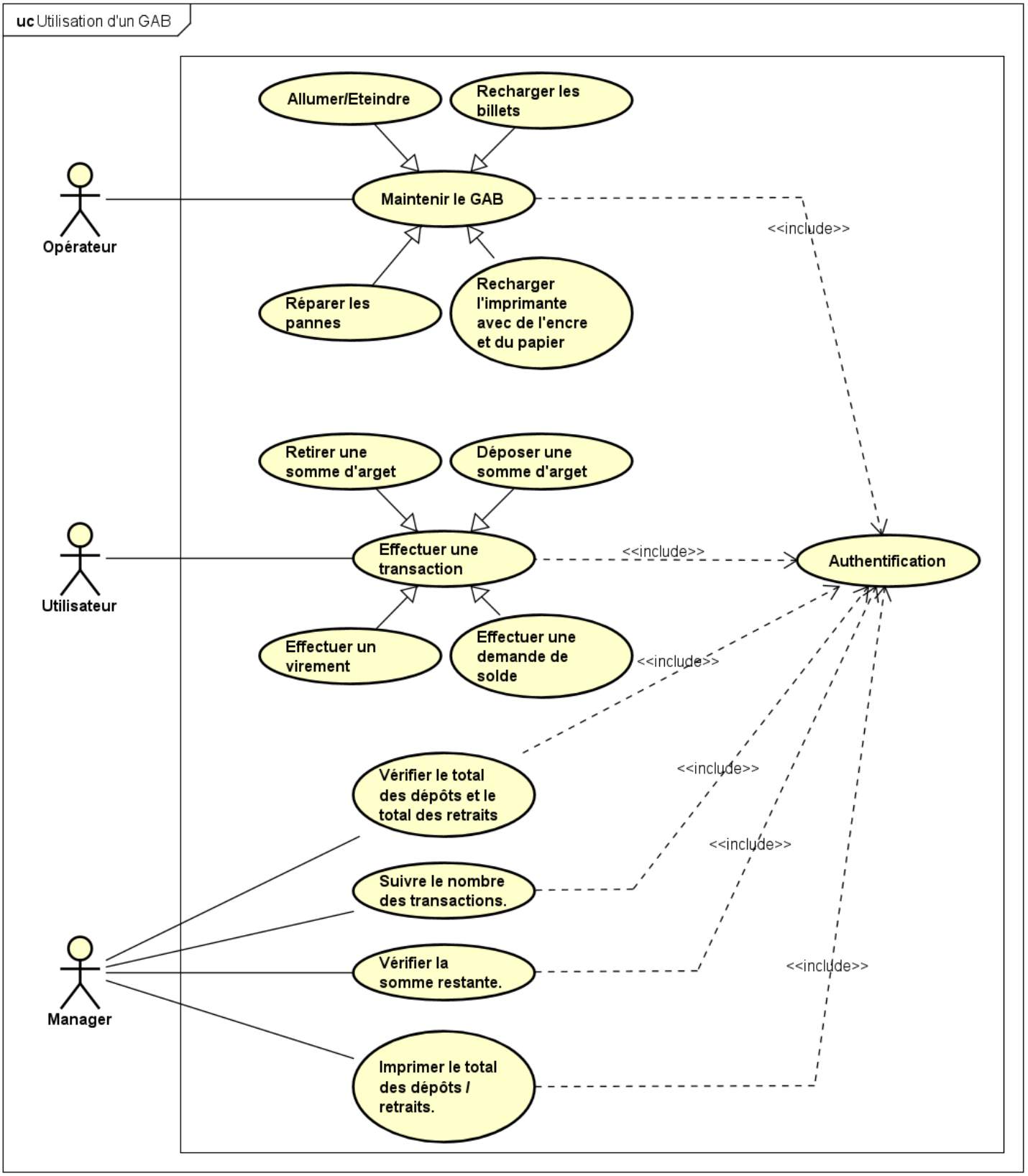


Figure 1 *: Diagramme de cas d'utilisation de la gestion d'un GAB.*

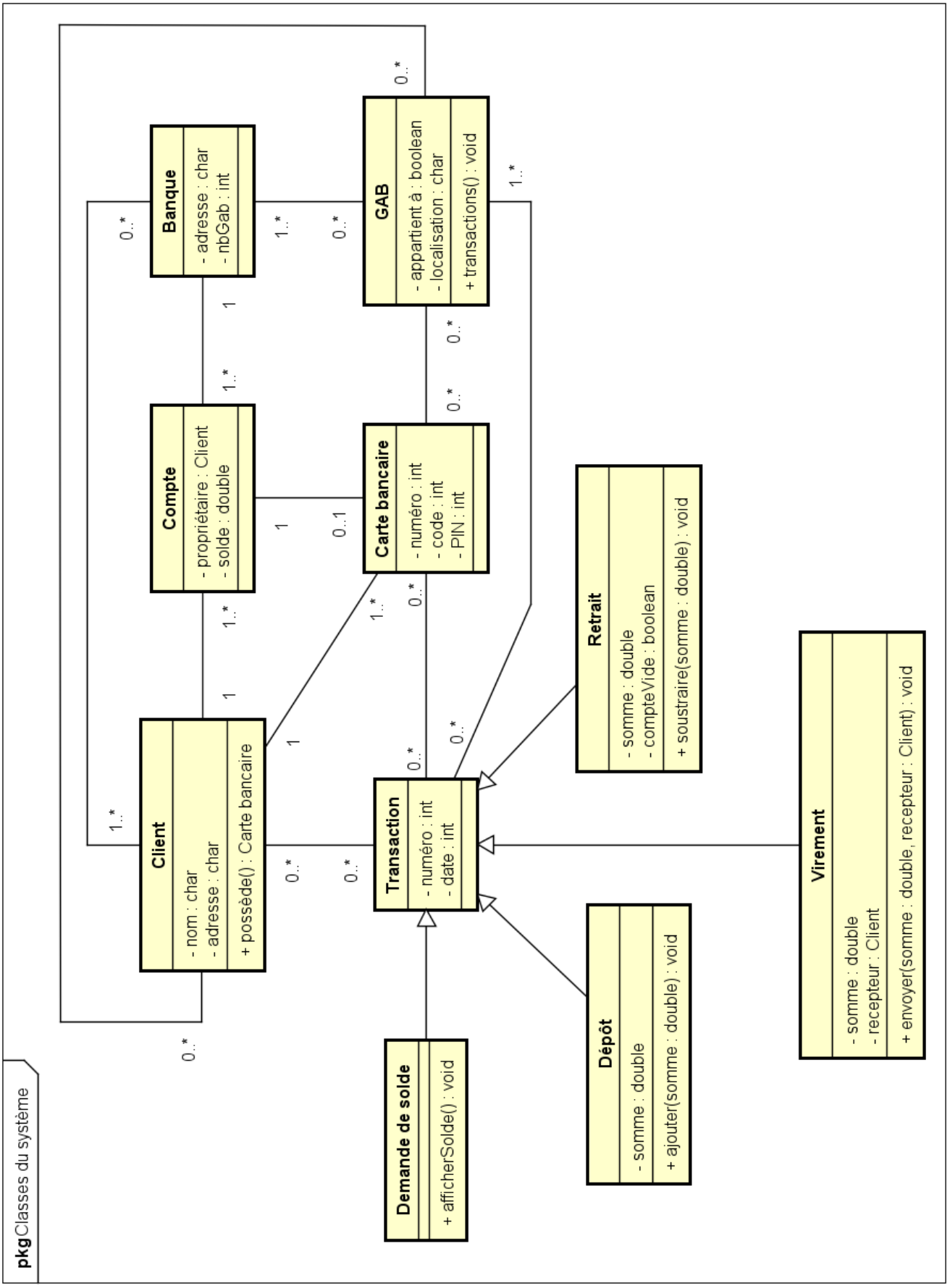


Figure 2 *: Diagramme de classes du système de GAB.*

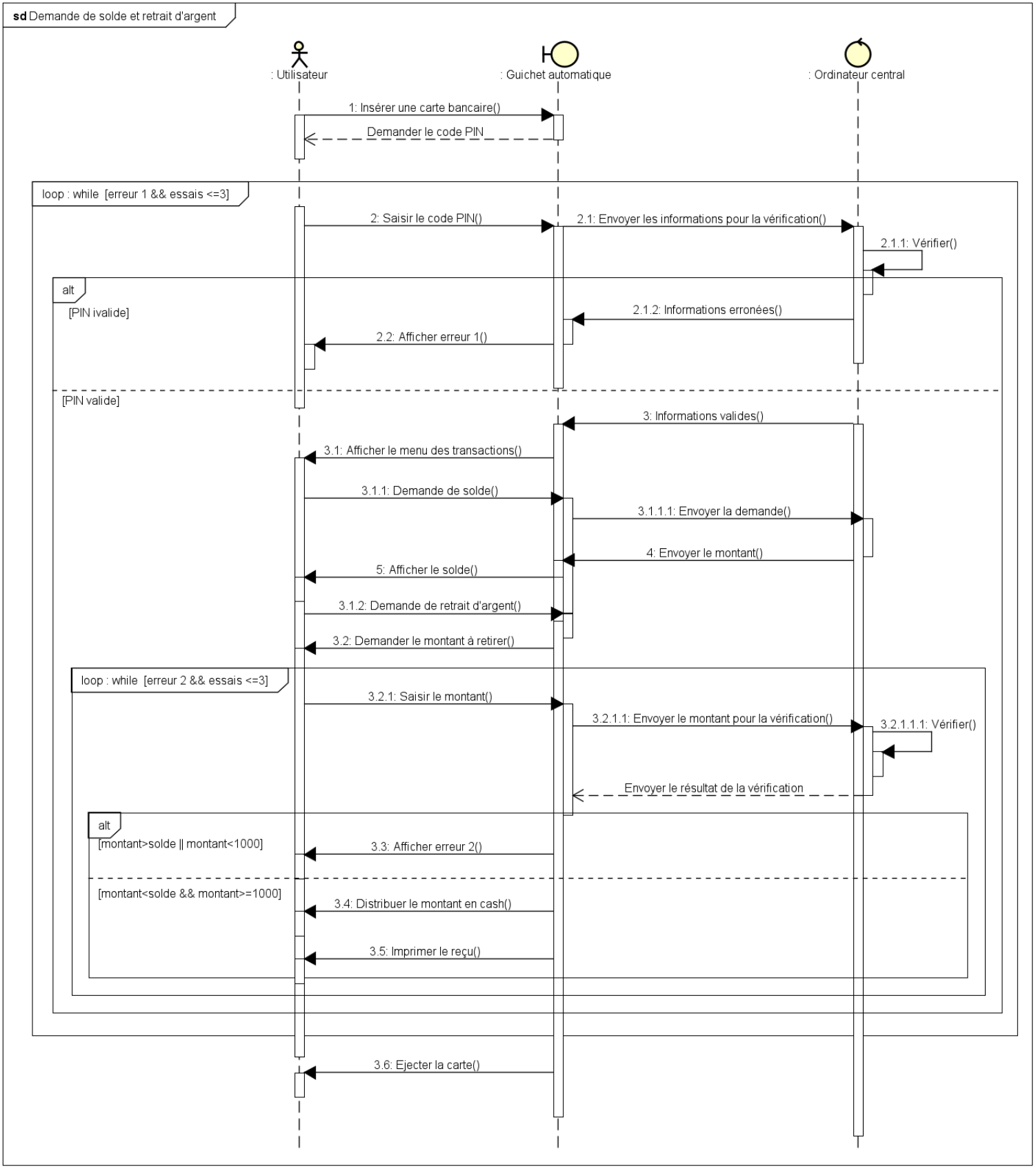


Figure 3 *: Diagramme de séquence pour la demande de solde et le retrait d'argent.*

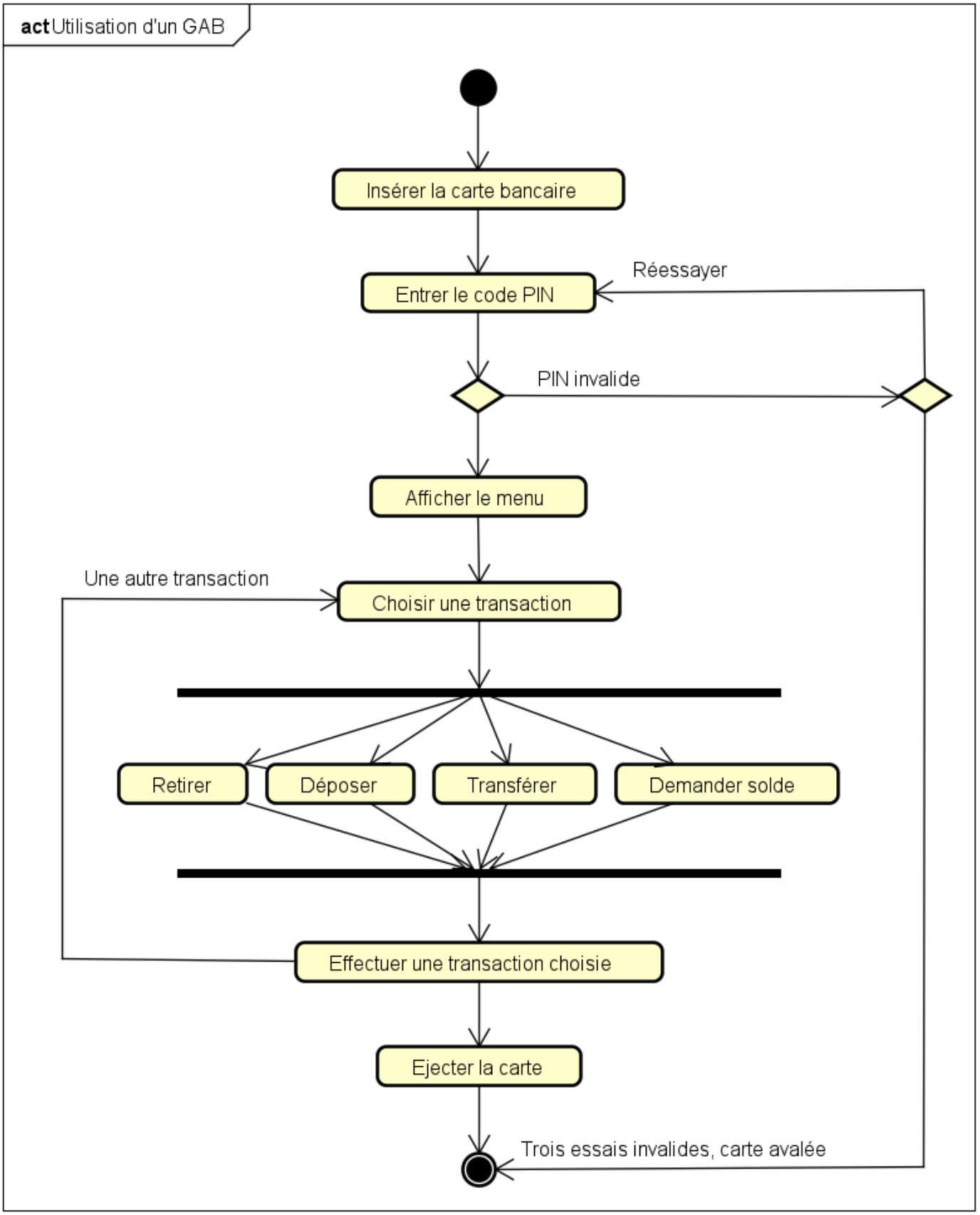


Figure 4 *: Diagramme d'activité d'utilisation d'un GAB.*

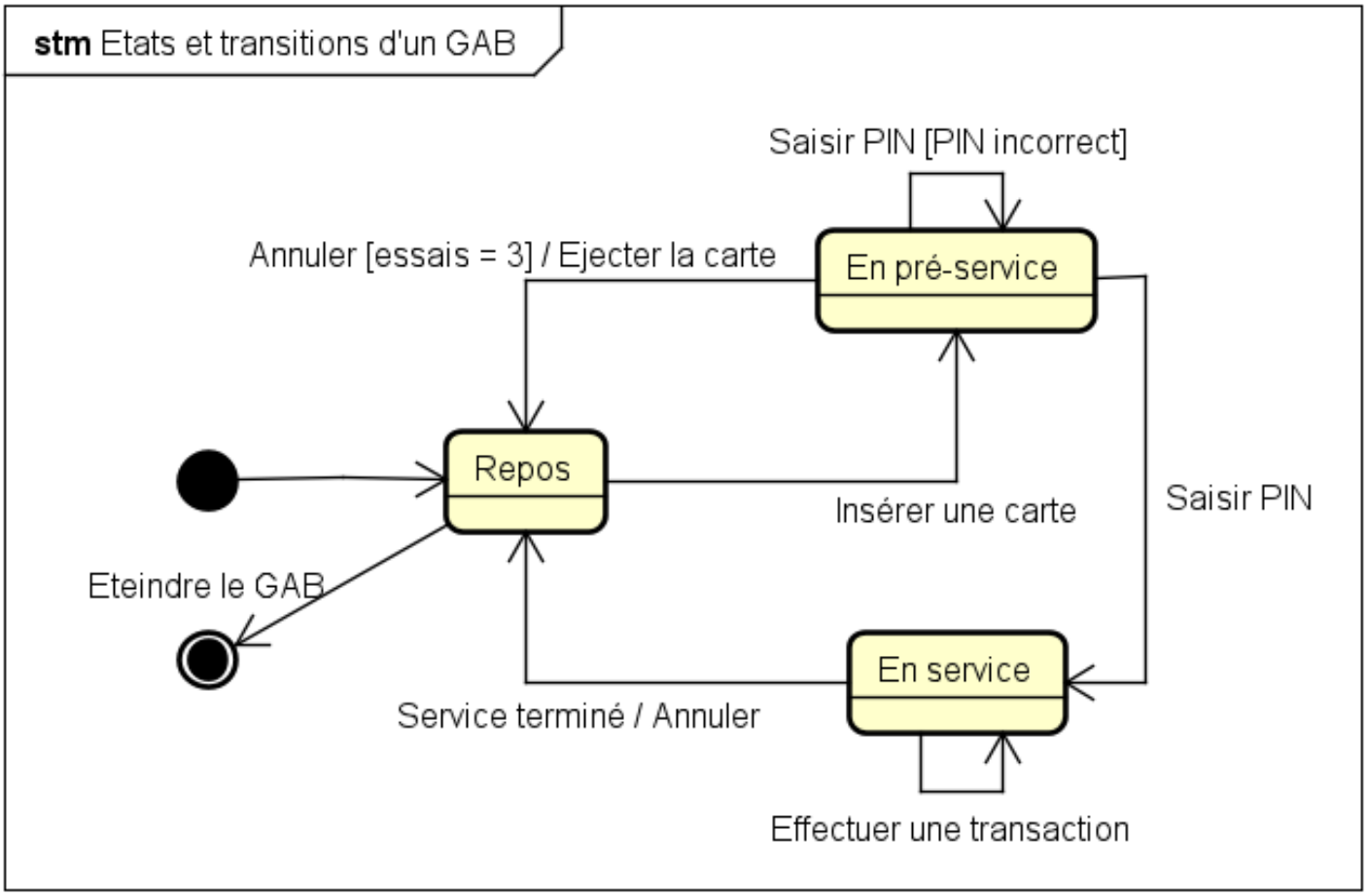


Figure 5 *: Diagramme d'états et transitions d'un GAB.*

Exercice 2

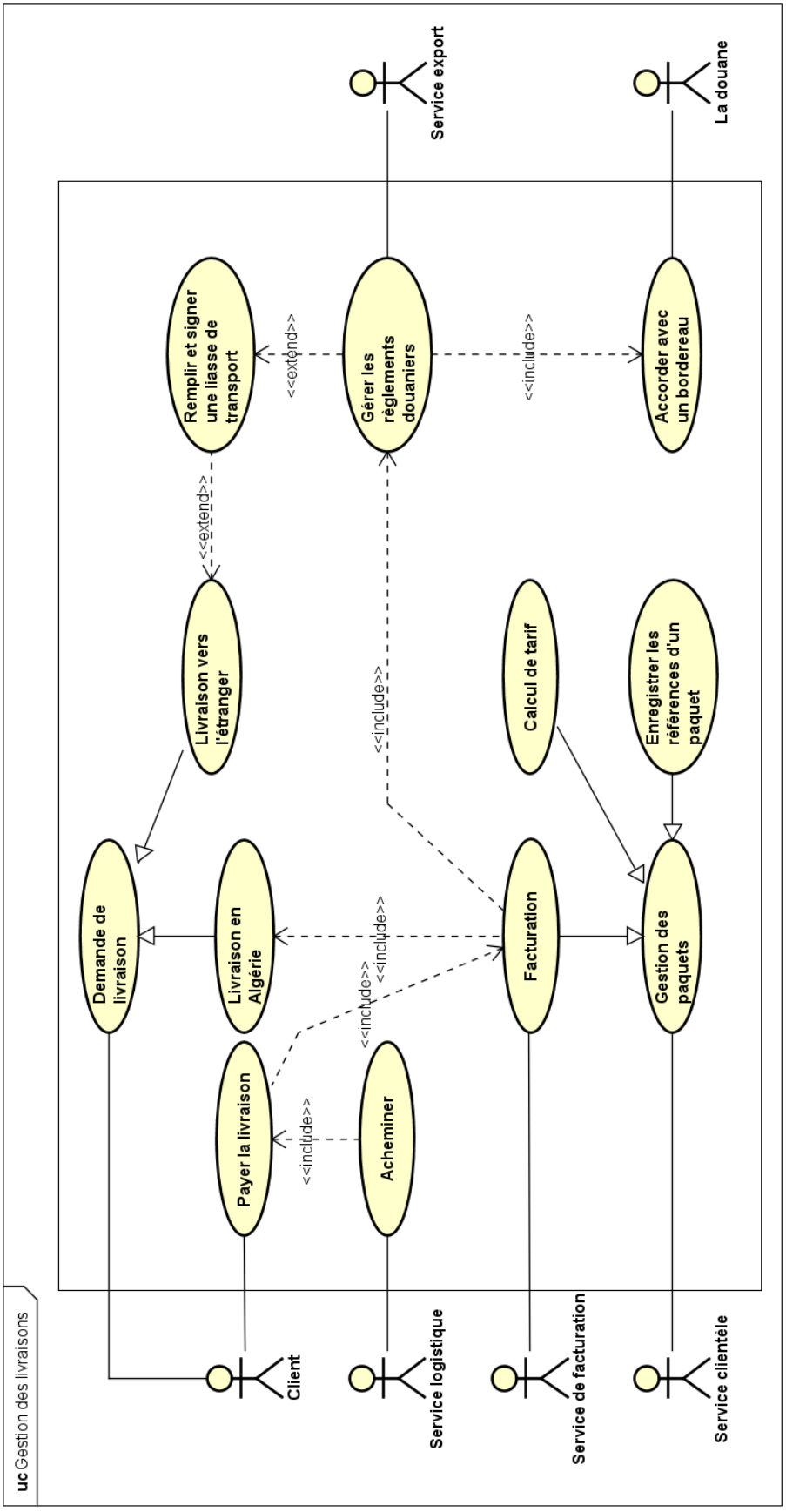


Figure 1 *: Diagramme de cas d'utilisation de la gestion des livraisons.*

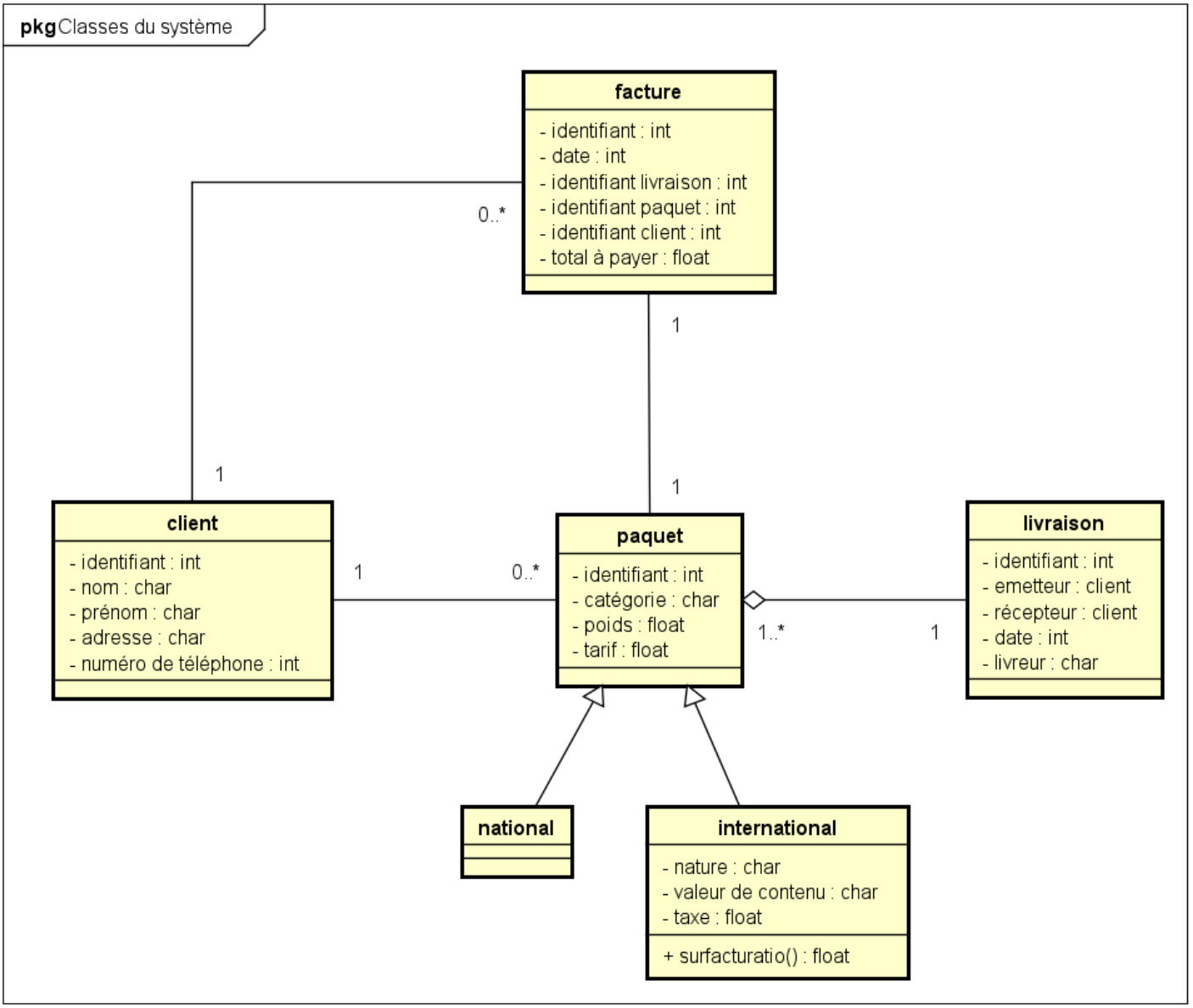


Figure 2 *: Diagramme de classes du système*

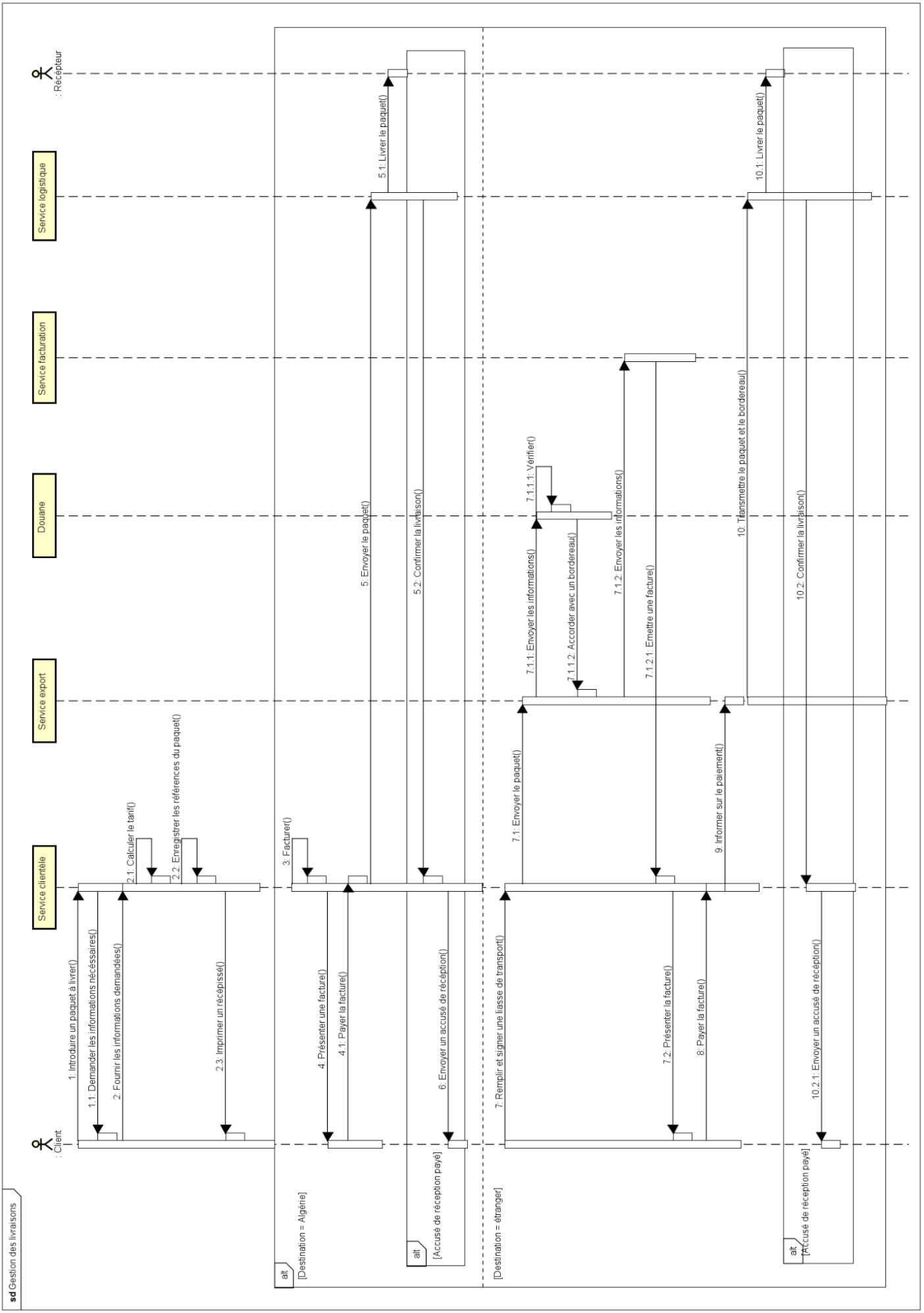


Figure 3 *: Diagramme de séquence de la gestion des livraisons.*

Exercice 3

1. Cas d'utilisation :

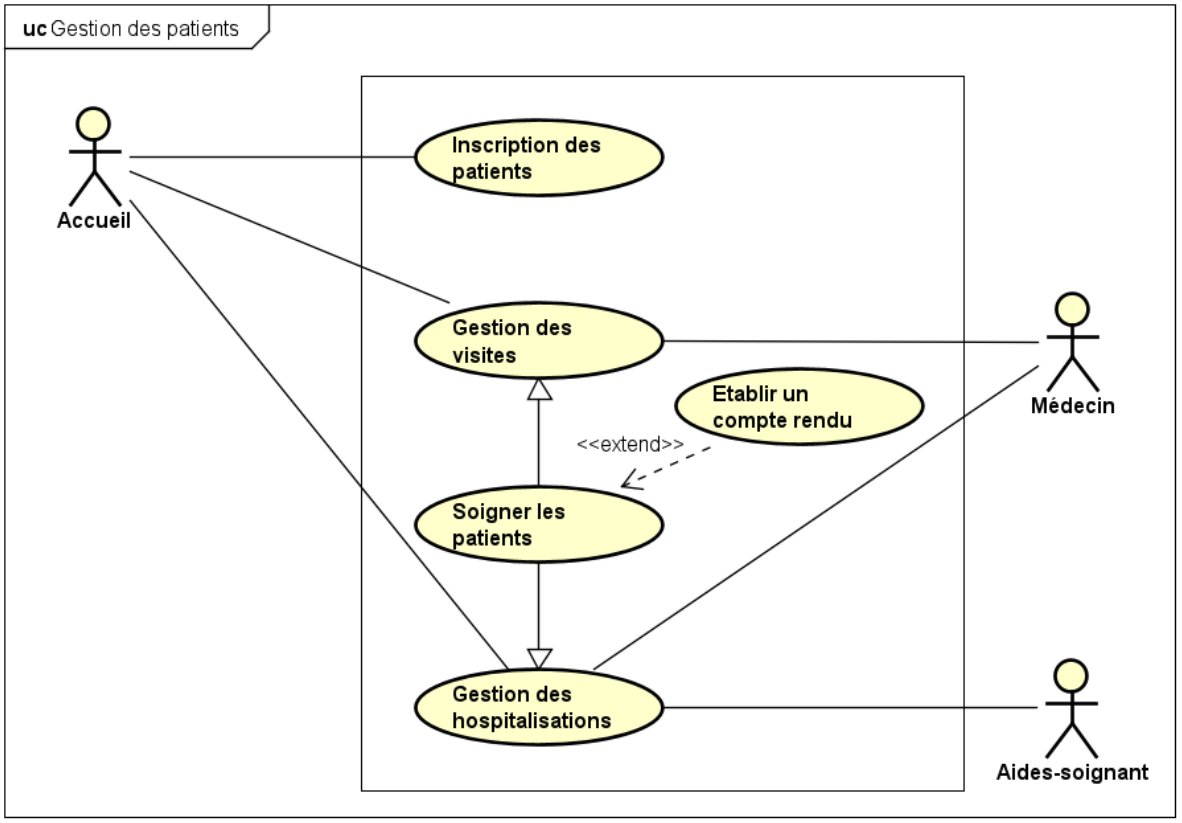


***Figure 1** : Diagramme de cas d'utilisation de la gestion des patients.*

2. Les scénarios majeurs des cas d'utilisation proposés :

CU	Inscription des patients
Description	Le patient s'inscrit au niveau de l'accueil
Acteur primaire	Accueil de l'hôpital
Pré-conditions	• Informations du patient • Le patient n'a pas de compte
Enchainement	• Le patient vient pour la première fois • Il fournit ses coordonnées à l'accueil • L'accueil lui affecte un identifiant et lui crée un compte
Post-conditions	Compte créé

CU	Gestion des visites
Description	Le patient vient pour une consultation
Acteur primaire	Accueil de l'hôpital, médecin
Pré-conditions	Le patient est inscrit
Enchainement	● L'accueil note le médecin que le patient vient consulter. ● Le médecin soigne le patient. ● Le médecin reporte dans le dossier patient l'ensemble des actes pratiqués sur le patient. ● Le médecin établit un compte rendu pour chaque acte.
Post-conditions	Compte du patient mis à jour

CU	Gestion des hospitalisations
Description	Le patient vient pour une hospitalisation
Acteur primaire	Accueil de l'hôpital, médecin
Pré-conditions	● Le patient est inscrit ● L'hôpital dispose des places disponible
Enchainement	● L'accueil note la date d'entrée du patient, la chambre dans laquelle le patient est hospitalisé, ainsi que le motif du séjour à l'hôpital et une date prévisionnelle de sortie. ● Le personnel soignant prend soin du patient tout au long de son séjour. ● Le personnel soignant reporte dans le dossier patient l'ensemble des actes pratiqués sur le patient ainsi que la date et l'heure de l'acte. ● Une fois l'acte terminé sur le patient, le médecin ayant pratiqué cet acte doit enregistrer son compte rendu dans le dossier patient. ● Une fois l'hospitalisation terminée, l'accueil enregistre la date de sortie du patient.
Post-conditions	Compte du patient mis à jour

3. Classes du système

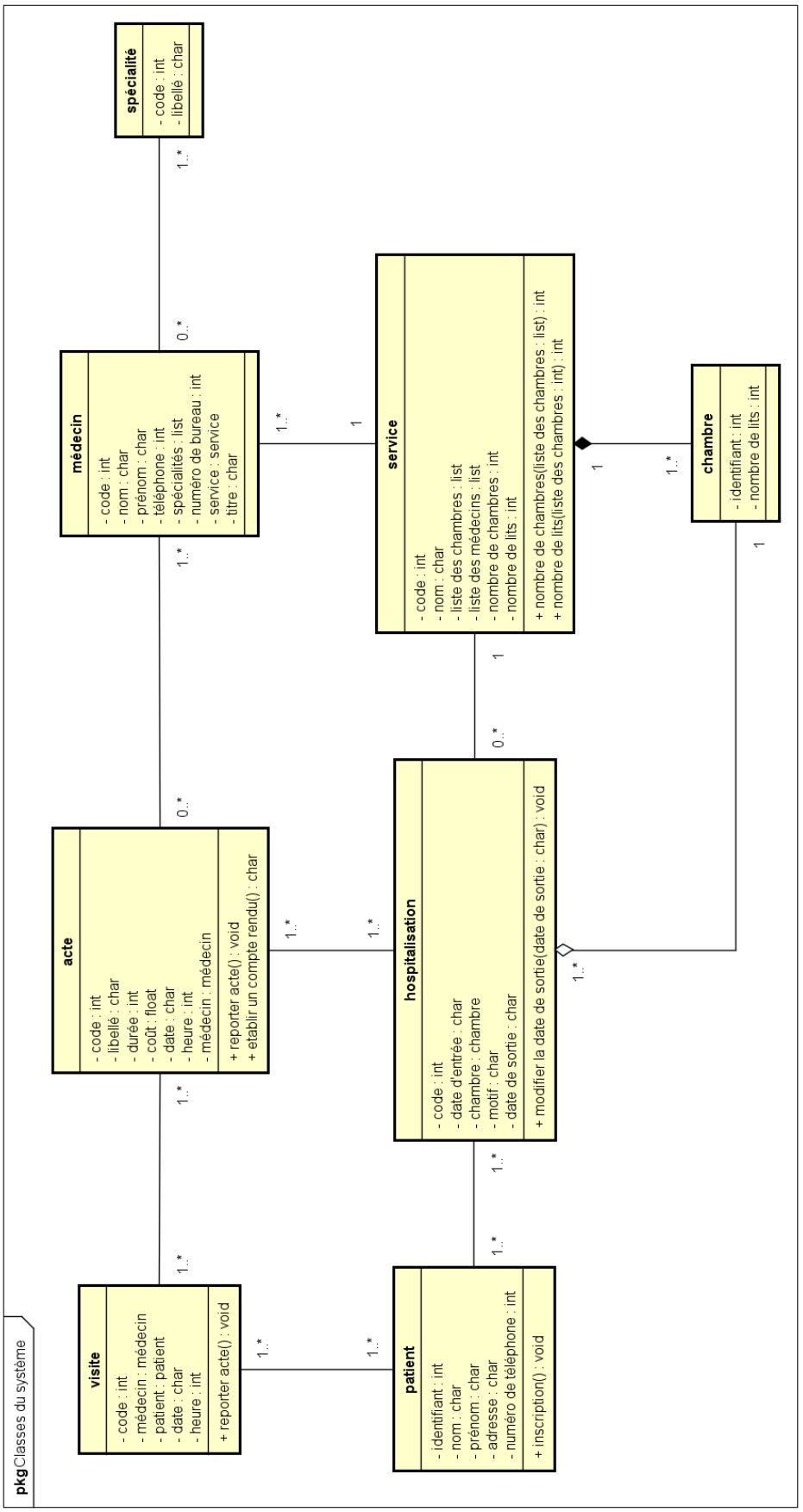


Figure 2 *: Diagramme de classes du système.*

4. **Etats et transitions :**
   La classe ayant la dynamique la plus importante dans cette gestion est celle du patient sans lequel tout le système ne peut fonctionner.

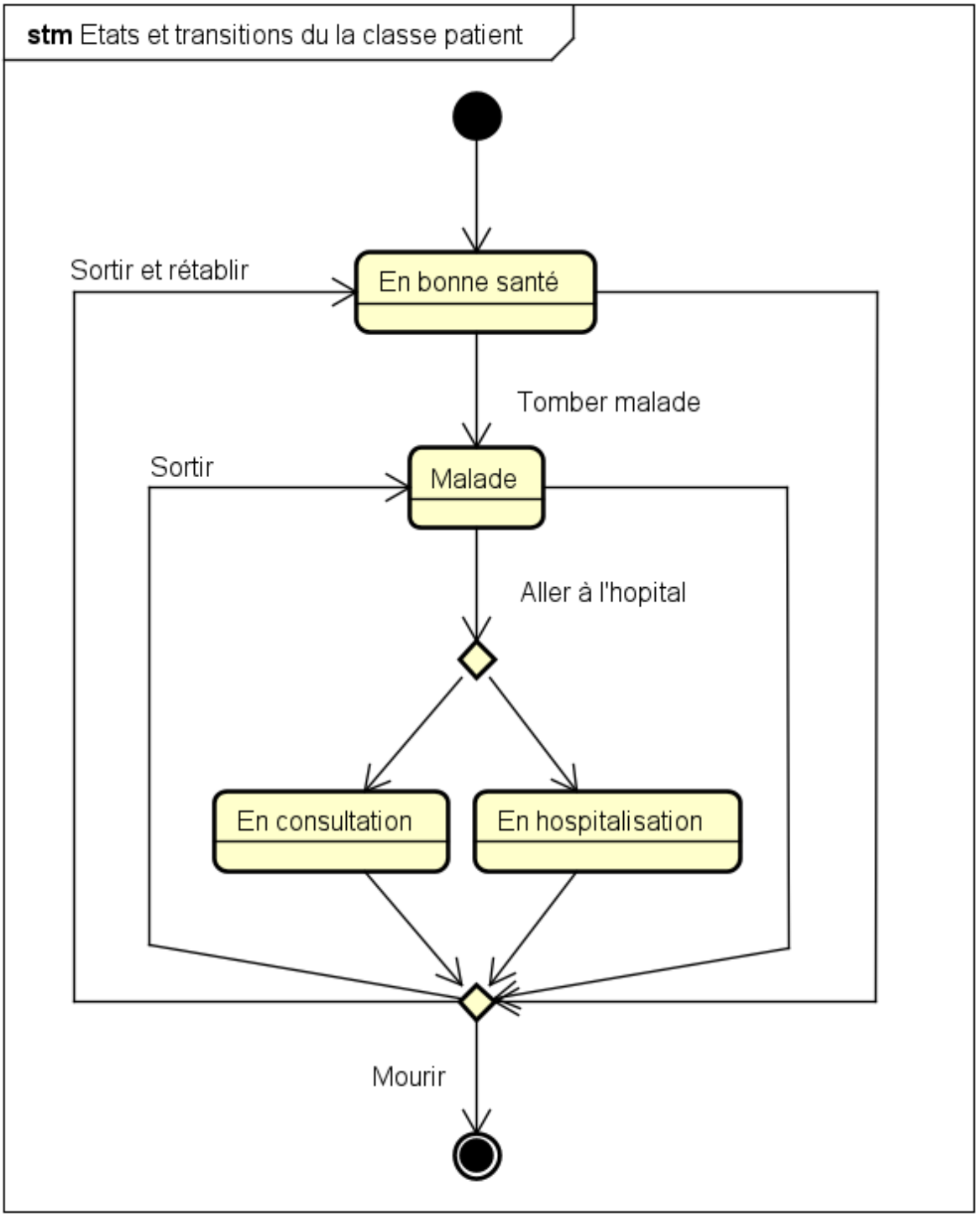


***Figure 3** : Diagramme d"états et transitions de la classe patient.*

Exercice 4

1. **Diagramme de classes :**

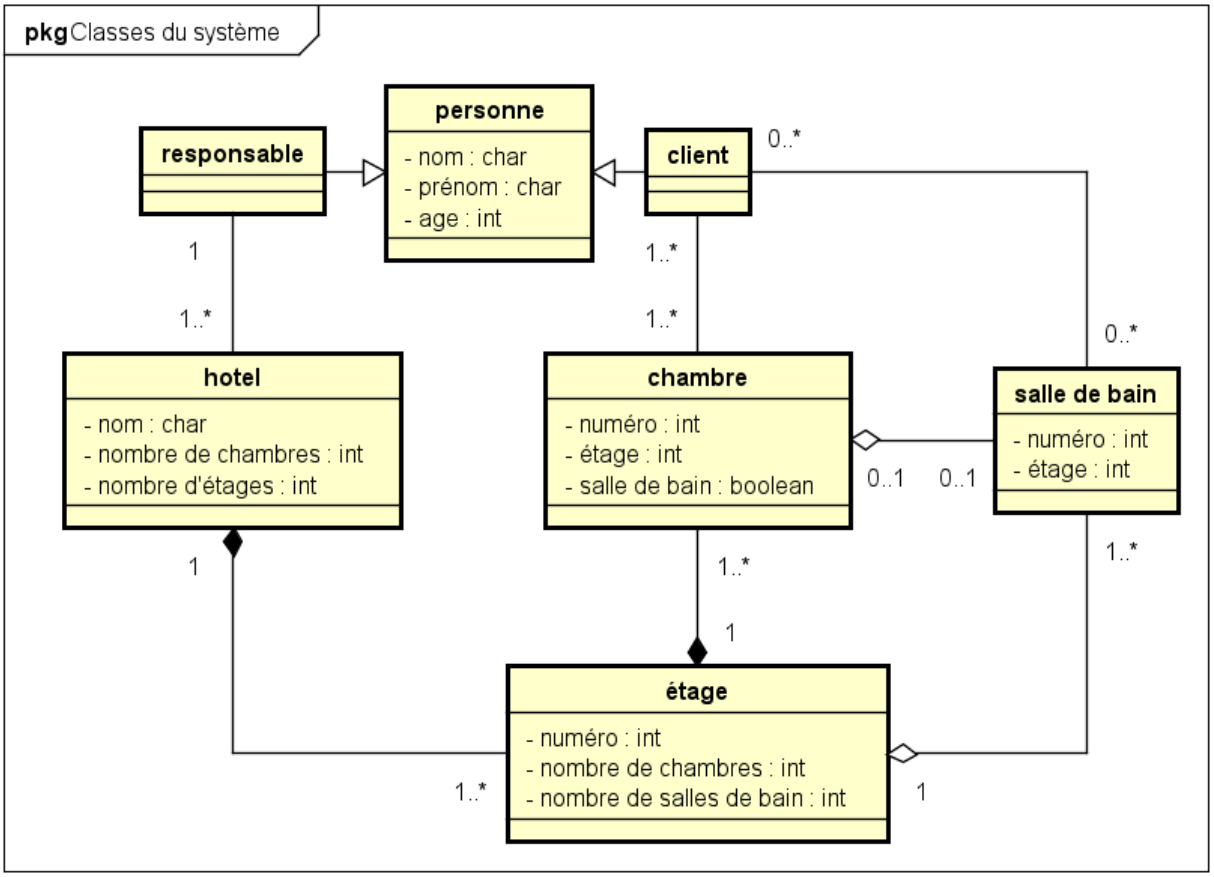


***Figure 1** : Diagramme de classes du système.*

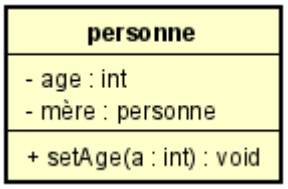
2. **Formulation en langage naturel :**
   a. L'étage 13ème ne se compose ni de chambres ni de salles de bain (n'existe pas selon la question 3).
   b. Le nombre de clients dans la chambre ne doit pas dépasser le nombre de lits si le dépassement ne concerne pas des enfants de moins de 4 ans.

3. **Expression OCL :**
   (Chaque étage possède au moins une chambre)

   Context e : étage
   inv : e.numéro < 13 implies e.nombre_de_chambres >= 1

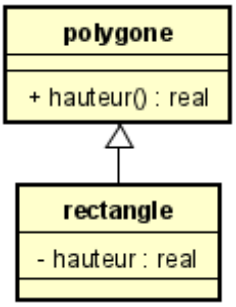
Exercice 5

Question 1 **:**



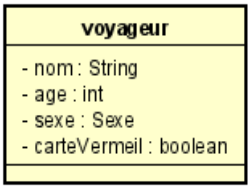
context personne
inv : self <> self.mère and self.age < self.mère.age

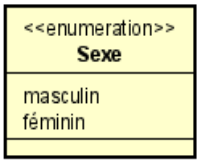
Question 2 :



context polygone :: hauteur() : Real
post : if self.oclIsKindOf(rectangle) then
result = self.oclAsType(rectangle).hauteur
else
result = 0

Question 3 :





context voyageur
if ((self.sexe = Sexe :: féminin and self.age > 60) or
(self.sexe = Sexe :: masculin and self.age > 65))
then
self.carteVermeil = true
else
self.carteVermeil = false

Question 4 :

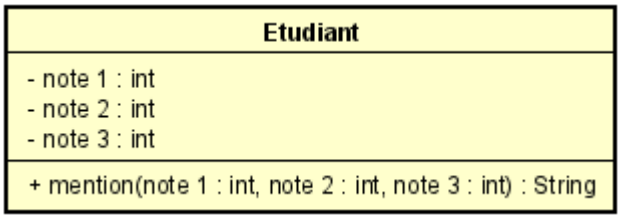
- **Le salaire d'un agent de secrétariat est inférieur à celui de son responsable**
  context encadrement
  inv : self.agentSecrétariat.contrat.salaire < self.responsable.contrat.salaire

- **Un agent de secrétariat a un type de contrat "agent Administratif"**
  context encadrement
  inv : self.agentSecrétariat.contrat.typeContrat = 'agent Administratif'

- **Un agent de secrétariat a une date d'embauche antérieure à la date de début de l'encadrement**
  context encadrement
  inv : self.dateDébut > self.agentSecrétariat.contrat.dateEmbauche

- **La dernière contrainte dans le contexte Personne**
  context personne
  inv : self.encadrement.dateDébut > self.agentSecrétariat.contrat.dateEmbauche

Question 5 :



context Etudiant :: mention(n1 : Integer, n2 : Integer, n3 : Integer) : String
let moyenne : Real = (n1 + n2 + n3) / 3 in
if (moyenne = 20) then
result = 'Excellent'
else if (moyenne >= 16) then
result = 'Très bien'
else if (moyenne >= 14) then
result = 'Bien'
else if (moyenne >= 12) then
result = 'Assez bien'
else if (moyenne >= 10) then
result = 'Passable'
else if (moyenne >= 5) then
result = 'Insuffisant'
else
result = 'Médiocre'

Question 6 :
En accédant à un attribut multi-valué on obtient une collection (classe abstraite Collection), ça peut être :

- Set : pas de doublons, pas d'ordre.
- OrderedSet : pas de doublons, ordonné.
- Bag : doublons possibles, pas d'ordre.
- Sequence : doublons possibles, ordonné.

Question 7 :

- **size() dans le contexte de la classe Collection**
  context Collection :: size() : Integer
  post : self —> iterate(element : Type; i : Integer = 0 | i+1)

- **forAll dans le contexte de la classe Collection**
  context Collection :: forAll(boolean) : Integer
  post : self —> iterate(element : Type; acc : Boolean = true | acc and boolean)

Question 8 :
Les employés ordonnés suivant leur salaire avec la méthode salariésTriés dans le contexte de la classe Entreprise :

context Entreprise :: salariésTriés() : OrderedSet(Personne)
post : result = self —> sortedBy(p : personne | p.contrat.salaire)

Exercice 6 :

1. **Le nombre d'employés travaillant dans un département doit être supérieur ou égal au nombre de projets contrôlés par le département.**
   context Departement
   inv : self.Employee —>size() >= self.Project —>size()

2. **Les employés obtiennent un salaire plus élevé lorsqu'ils travaillent sur plus de projets.**
   context Employee
   inv : self.allInstances->forAll(e1, e2 | e1.Project —>size() > e2.Project —>size()
   implies e1.salary > e2.salary)

3. **Le budget d'un projet ne doit pas dépasser le budget du département de contrôle.**
   context Project
   inv : self.budget <= self.department.budget

4. **Les employés travaillant sur un projet doivent également travailler dans le département de contrôle.**
   context Department
   inv : self.Project.Employee —>includesAll(self.Employee)

Exercice 7

1. **Diagramme de classes**

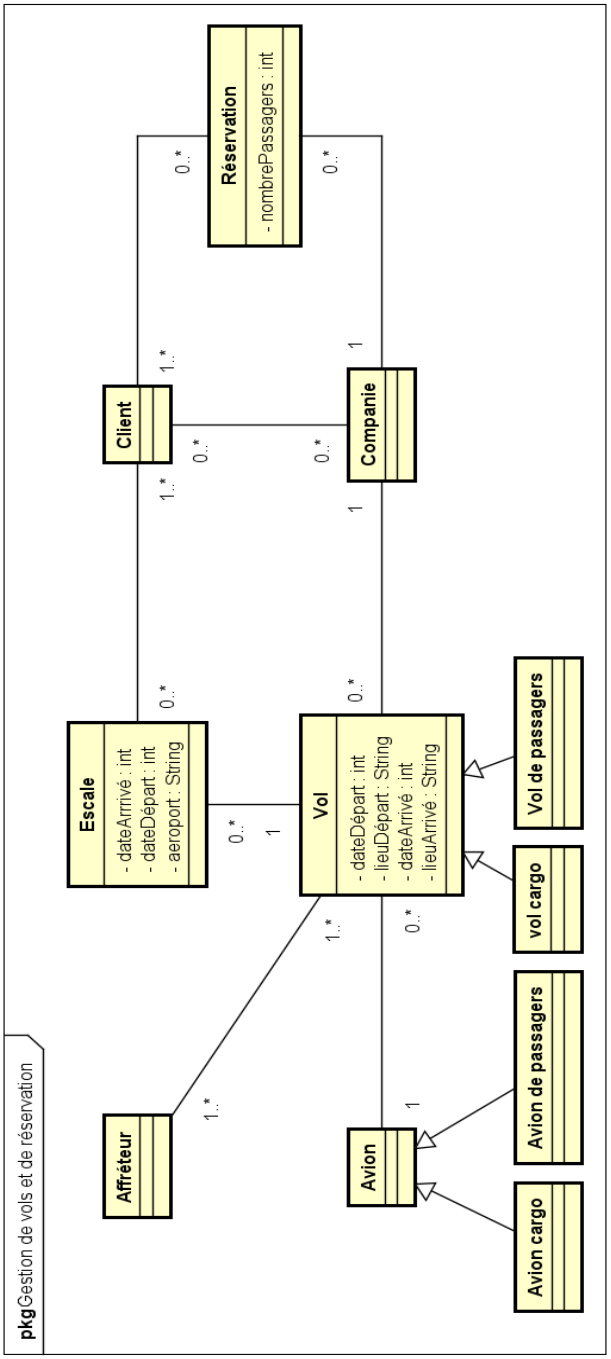


Figure 1 *: Diagramme de classes du système de gestion de vols et de réservations.*

- **Contrainte OCL :**
  Un vol cargo est assuré par un avion-cargo et un vol de passagers est assuré par un avion de passagers :

  context Vol
  inv self.oclIsKindOf(Vol cargo) implies self.avion.oclAsType(Avion cargo)
  inv self.oclIsKindOf(Vol de passagers) implies self.avion.oclAsType(Avion de passagers)

2. Diagramme de cas d'utilisation relatif à chaque compagnie aérienne

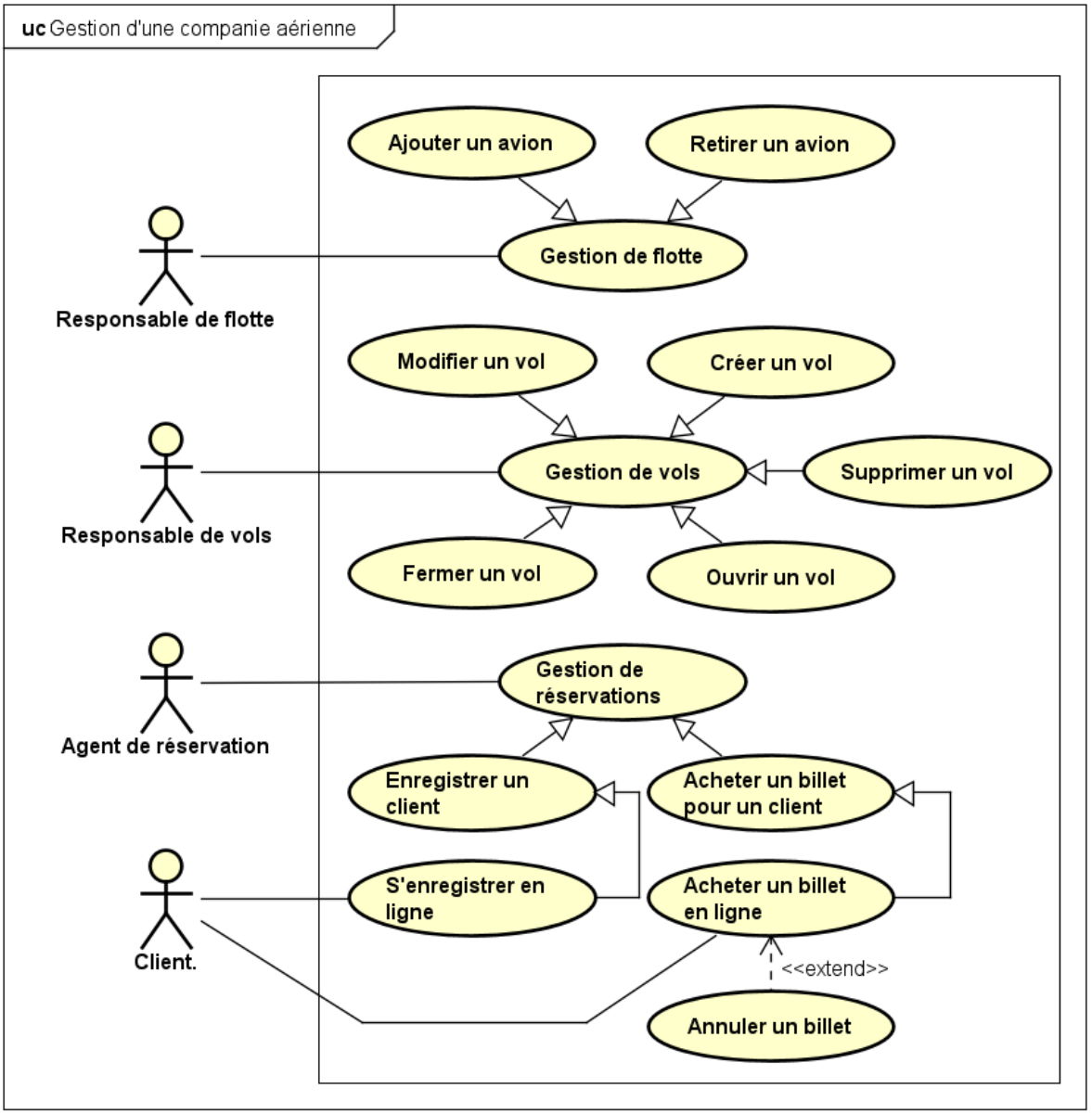


Figure 2 *: Diagramme de cas d'utilisation de la gestion d'une compagnie aérienne*

3. Diagramme d'activité pour la gestion des réservations

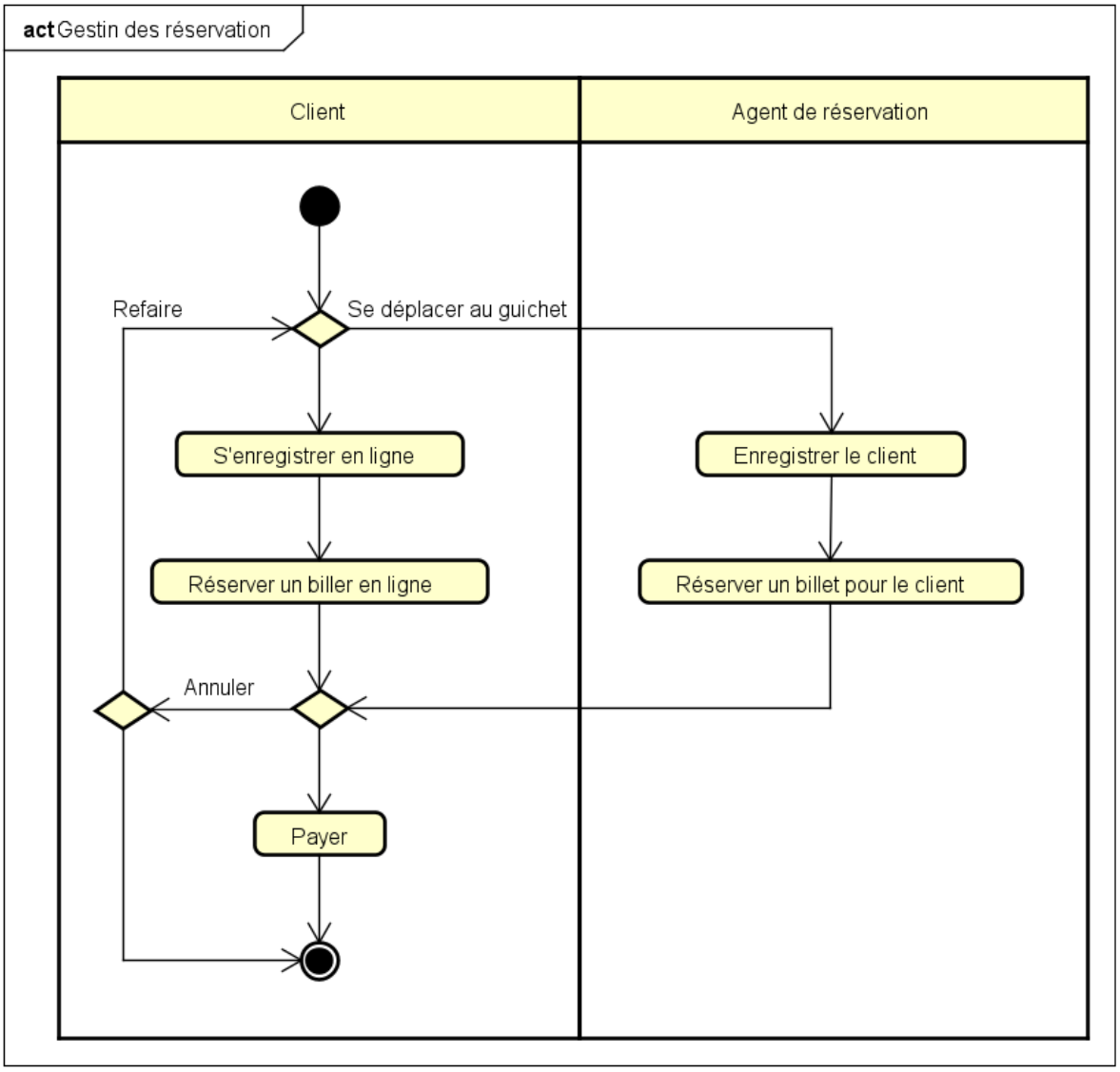


Figure 3 *: Diagramme d'activités de la gestion des réservations.*

Exercice 8

1. **Diagramme de classes du système**

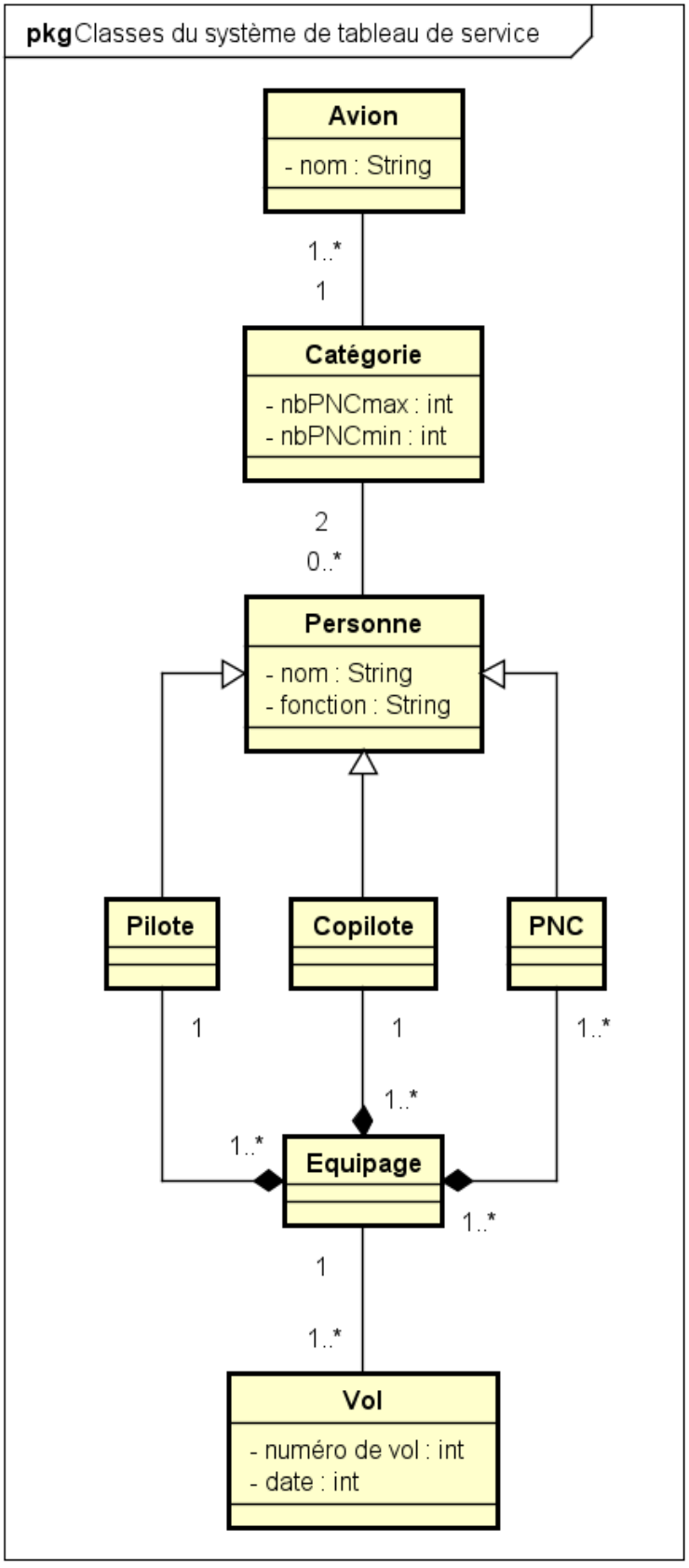


***Figure 1** : Diagramme de classes du système de tableau de service.*

2. **Contrainte en OCL**

- Chaque catégorie d'avions requiert un nombre de PNC dans son équipage oscillant entre un minimum et un maximum :

context Catégorie
inv : self.Personne.oclAsType(PNC) —> size() >= self.nbPNCmin
and self.Personne.oclAsType(PNC) —> size() <= self.nbPNCmax

3. **Diagramme de cas d'utilisation**

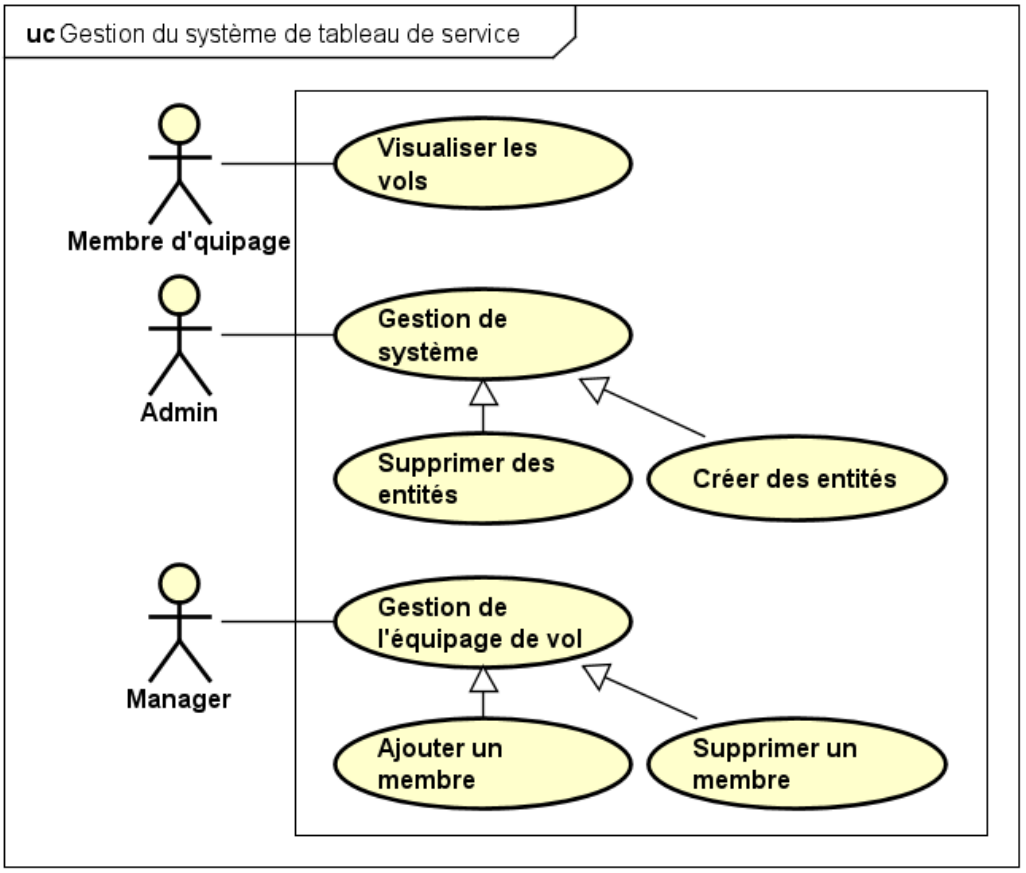


***Figure 2** : Diagramme de cas d'utilisation du système de tableau de service.*

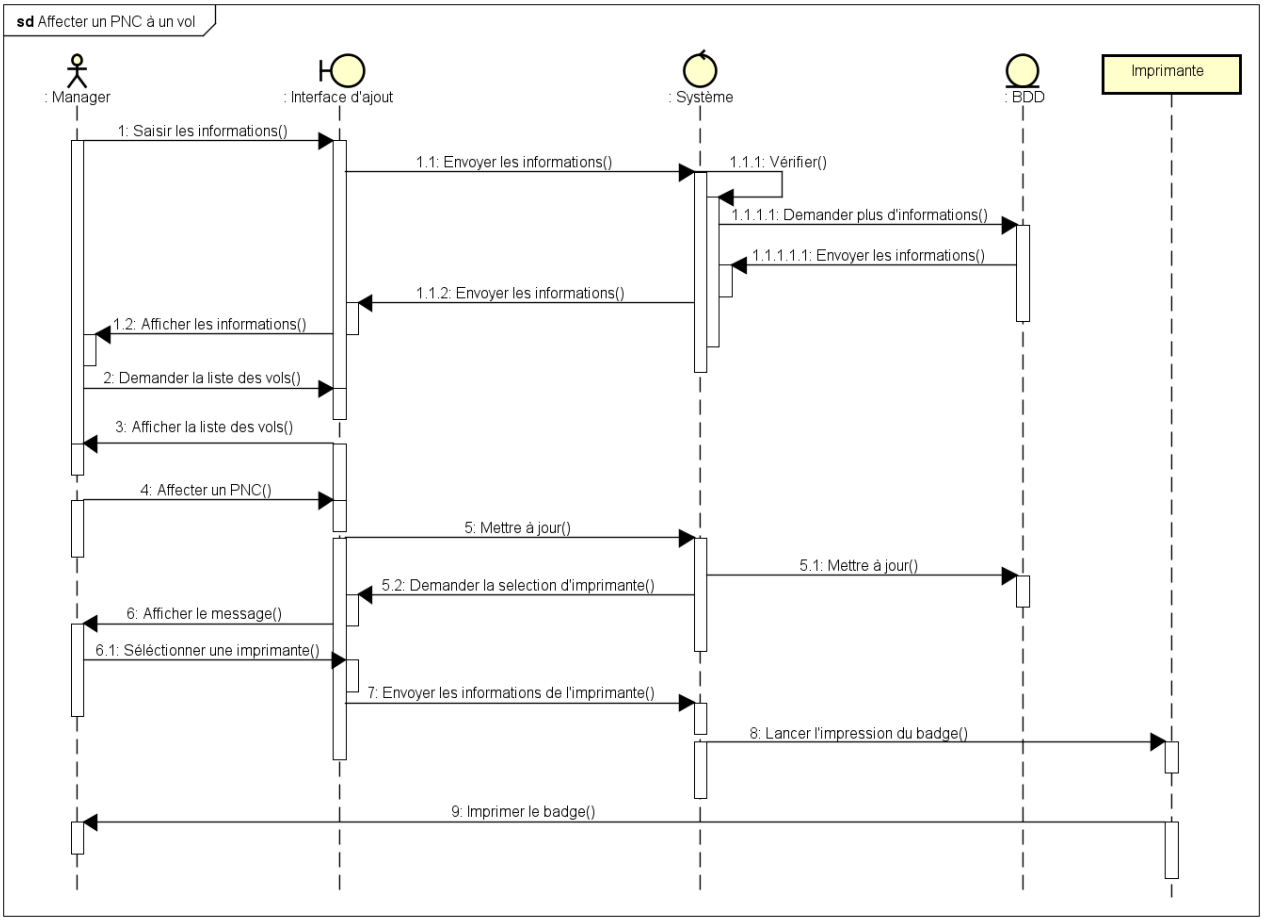
4. **Description détaillée du cas d'utilisation "Affecter un PNC à un vol"**

Le cas d'utilisation "Affecter un PNC à un vol" est inclus dans le cas d'utilisation "Ajouter un membre".

Fonction	Ajout d'un PNC à un vol.
Description	Un manager ajoute un PNC dans un équipage pour un vol donné.
Entrées	Coordonnées du PNC.
Source	Page d'informations du compte du PNC.
Sortie	Les informations sur le vol dont le PNC est affecté.

Destination	Impression du badge.
Action	Les coordonnées du PNC sont transformées dans une requête qui est transmise à la base de données. Ensuite, le résultat est transmis au système et traité (catégories d'opérations du PNC, nombre de vols, disponibilité, etc.). L'affichage doit prévoir un moyen pour sélectionner l'imprimante en vue de l'impression.
Requis	Imprimante de destination.
Pré-condition	PNC disponible et imprimante opérationnelle.
Post-condition	Informations modifiées sur le compte du PNC en termes de disponibilité, nombre de vols etc.
Effets de bord	Opération enregistrée. PNC ajouté à l'équipage du vol. Badge imprimé.

5. Diagramme de séquence représentatif pour ce cas d'utilisation



Exercice 9

1. Diagramme de classes

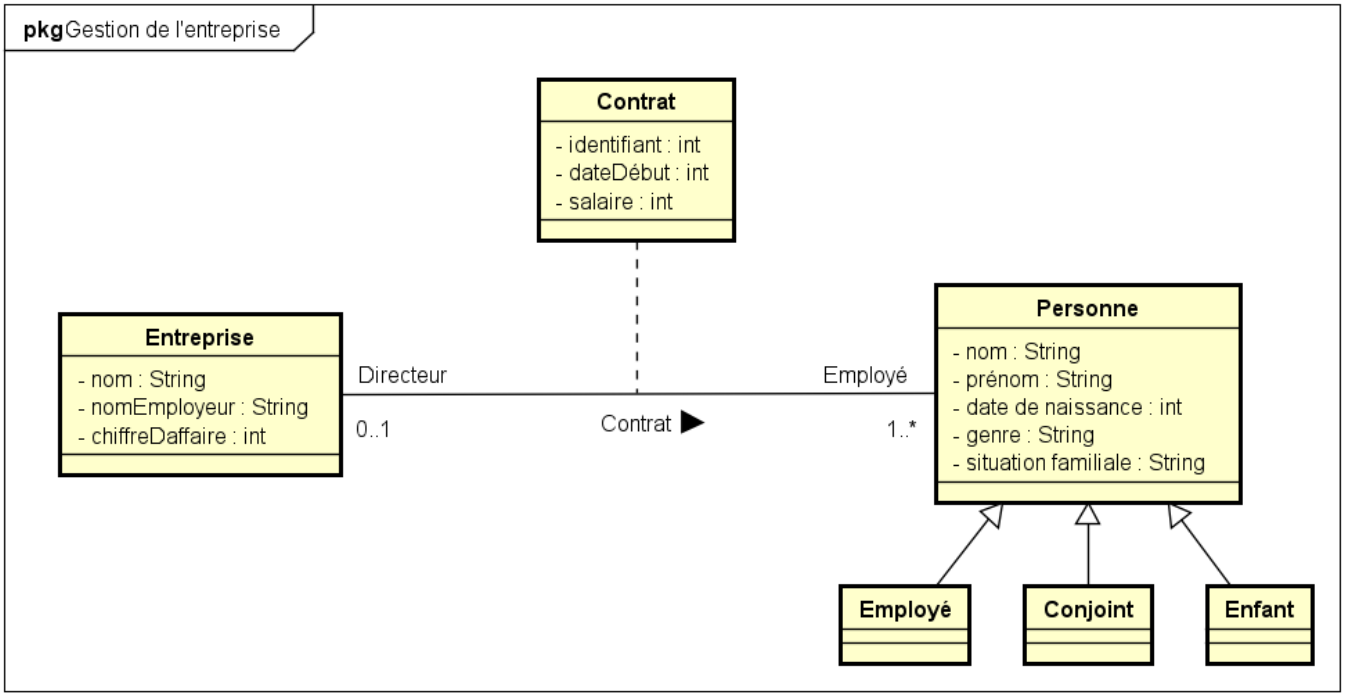


Figure 1 *: Diagramme de classes de la gestion de l'entreprise.*

2. Les contraintes OCL

- **L'entreprise est dirigée par un employé de l'entreprise**

  context Entreprise
  inv : self.Employé —> includes(self.Directeur)

- **Initialement le salaire d'un employé est de 40 000 DA**

  context Personne
  init : self.oclAsType(Employé).Contrat.salaire = 40000

- **Le directeur de l'entreprise doit avoir plus de 40 ans**

  context Entreprise
  let dateDembauceDirecteur : Integer in
  inv dateDembauceDirecteur - self.Directeur.dateDeNaissance >= 40

- **Si le chiffre d'affaires d'une entreprise est supérieur à 100 million DA, elle doit avoir plus que 10 employés**

  context Entreprise
  Inv : self.chiffreDaffaire > 100000000 implies self.Employé —> size() >= 10

- **Un homme est marié avec une femme et une femme avec un homme**

  context Personne
  def conj : Personne = self.oclAsType(Conjoint)
  inv : self.situationFamiliale = 'Marié' and self.genre = 'Homme'
  implies self.situationFamiliale = 'Marié' and self.conj.genre = 'Femme'
  inv : self.situationFamiliale = 'Marié' and self.genre = 'Femme'
  implies self.situationFamiliale = 'Marié' and self.conj.genre = 'Homme'

- **Pour être marié, il faut avoir plus de 18 ans**

  context Personne
  let dateMariage : Integer in
  inv : self.situationFamiliale = 'Marié'
  implies dateMariage - self.dateDeNaissance >= 18