CAS ASDOMI

L'ASDOMI est une association qui gère un service de soins infirmiers à domicile (SSIAD) dans le département de l'Ain. Sa mission est de permettre le maintien à domicile de personnes dépendantes ou âgées.

L'ASDOMI dispose d'une capacité d'accueil de 96 « lits » et reçoit une dotation annuelle globale de la DASS (Direction des Affaires Sanitaires et Sociales) calculée en fonction du nombre de lits (nombre maximal de patients pouvant être pris en charge simultanément).

L'activité de l'ASDOMI est supervisée par la CPAM (Caisse Primaire d'Assurance Maladie) de l'Ain, dite caisse pivot, qui accorde les demandes d'entente préalable pour la prise en charge des soins et effectue des contrôles.

Pour mener à bien sa mission, l'ASDOMI emploie 35 aides-soignants, 3 infirmières coordinatrices, une secrétaire, une comptable et une directrice. Elle peut aussi recourir à des prestataires libéraux (infirmiers, pédicures, etc.) pour des soins spécialisés.

Pour faire face à l'explosion prévue des services à la personne, recentrer les aides-soignants sur leur cœur de métier et répondre aux exigences de qualité définies par la DASS, l'ASDOMI a décidé de refondre son système d'information devenu obsolète et a pour cela fait appel à une société de service et d'ingénierie informatique (SSII).

Vous faites partie de l'équipe de la SSII chargée de cette mission.

Outils d'évaluation des prestations

Documents à utiliser : annexes A, B, C

L'ASDOMI reçoit chaque année une dotation globale basée sur un nombre prévu de jours de soins. Afin d'équilibrer son budget et de satisfaire au mieux les demandes d'admission de plus en plus nombreuses, il est demandé à la SSII de développer une application de type « tableau de bord » proposant différents outils d'évaluation des prestations réalisées depuis le début de l'année, notamment en termes de :

- ratio entre les prestations réalisées par des intervenants externes et celles prises en charge par les aides-soignants salariés de l'ASDOMI,
- nombre de jours de soins déjà consommés.

Cette application sera réalisée dans un langage de programmation objet.

Les *annexes A et B* fournissent une description des classes métier requises pour l'application, notamment :

- La classe *Dossier* permet de conserver les données propres à un dossier ouvert pour un patient, et d'y associer les différentes prestations réalisées en précisant pour chacune la nature, le jour et heure, ainsi que la personne l'ayant pris en charge.
- Les classes *Intervenant* et *IntervenantExterne* permettent de conserver les caractéristiques d'un intervenant et celles, plus spécifiques, d'un intervenant externe à l'ASDOMI.

L'*annexe C* fournit la description textuelle des classes techniques utilisées.

Travail à faire	
1	Écrire la méthode *ajoutePrestation()* de la classe Dossier.
2	Écrire le constructeur de la classe Prestation, ce dernier ayant la responsabilité d'assurer la navigation bidirectionnelle de l'association entre les classes Prestation et Intervenant.

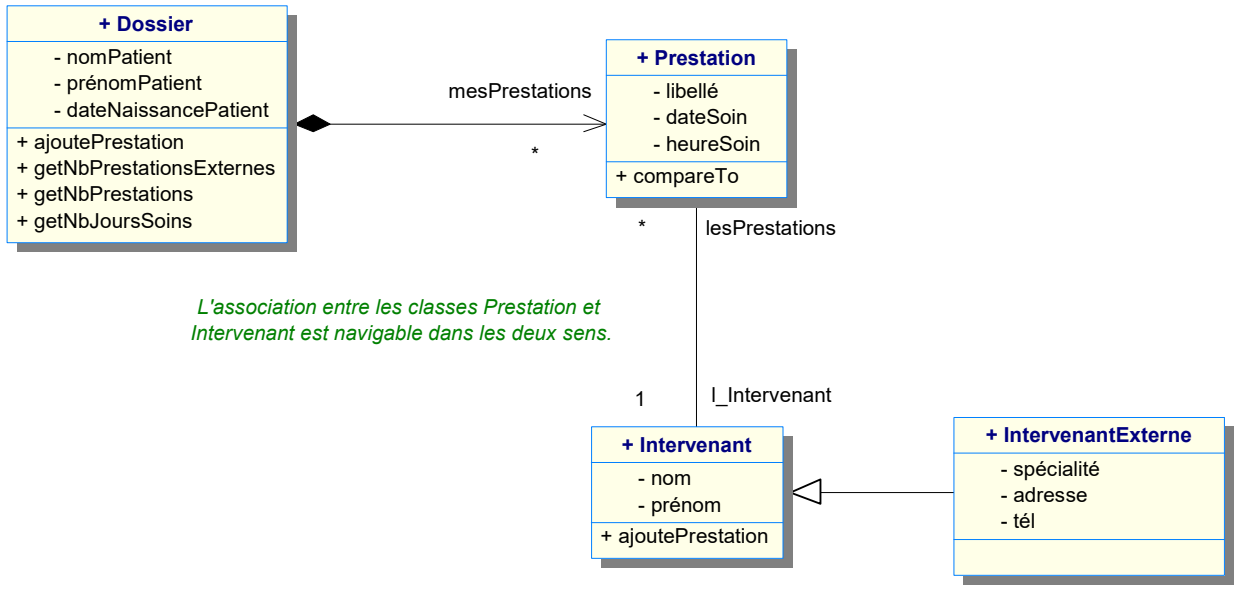
Comme dit précédemment, le rapport du nombre global de prestations externes sur le nombre total de prestations constitue un indicateur d'évaluation de l'activité de l'ASDOMI.

Travail à faire	
3	Écrire la méthode *getNbPrestationsExternes()* de la classe Dossier.

Le nombre de jours de soins consommés sur l'ensemble des dossiers permet à la directrice de l'ASDOMI de prévoir ses possibilités de prestations pour les semaines à venir.

Travail à faire	
4	Écrire la méthode *compareTo()* de la classe Prestation, permettant de comparer une prestation à une autre, en regard de la date de réalisation de cette prestation.
5	Écrire la méthode *getNbJoursSoins()* de la classe Dossier qui retourne le nombre de jours de soins consommés pour le dossier.

Annexe A : Diagramme partiel des classes métier



Commentaires :

Accessibilité :

Symbole – (moins) : membre privé
Symbole + (plus) : membre public

Remarque : Les constructeurs, accesseurs et paramètres des méthodes ne sont pas présentés sur ce diagramme.

Annexe B : Description textuelle des classes métier

Classe Dossier

Attributs privés :

nomPatient : Chaîne
prénomPatient : Chaîne
dateNaissancePatient : Date
mesPrestations : Collection de Prestation
// cette collection est initialement dans un ordre quelconque

Méthodes publiques :

Dossier (unNomPatient : Chaîne, unPrénomPatient : Chaîne, uneDateNaissancePatient : Date)

// constructeur de la classe Dossier

Procédure ajoutePrestation (unLibellé : Chaîne, uneDate : Date, uneHeure : Heure, unIntervenant : Intervenant)

// permet d'ajouter une prestation au dossier

Fonction getNbPrestationsExternes () : Entier

// Retourne le nombre de prestations externes réalisées pour le dossier

Fonction getNbPrestations () : Entier

// Retourne le nombre de prestations réalisées pour le dossier

Fonction getNbJoursSoins () : Entier

// Retourne le nombre de jours de soins comptabilisés pour le dossier. Il ne s'agit pas ici de déterminer le nombre de prestations attachées à un dossier, mais le nombre de jours pour lesquels au moins une prestation a été réalisée. Ainsi, un dossier avec 7 prestations, les 3 premières ayant eu lieu le 10 août 2008, les 4 dernières le 11 août 2008, retournera la valeur 2 sur appel de la méthode getNbJoursSoins().

FinClasse

Classe Prestation

Attributs privés :

libellé : Chaîne
dateSoin : Date
heureSoin : Heure
l_Intervenant : Intervenant

Méthodes publiques :

Prestation (unLibellé : Chaîne, uneDate : Date, uneHeure : Heure, unIntervenant : Intervenant) ***// constructeur de la classe Prestation***

Fonction compareTo (unePrestation : Prestation) : Entier

// Fonction permettant de comparer 2 prestations, la prestation courante et le paramètre unePrestation, la comparaison porte ici sur la date de la prestation. Retourne 0 si la date de la prestation courante est égale à la date de la prestation unePrestation. Retourne 1 si la date de la prestation courante est postérieure à la date de la prestation unePrestation. Retourne -1 si la date de la prestation courante est antérieure à la date de la prestation unePrestation.

Fonction getDateSoin ( ) : Date

// accesseur sur l'attribut dateSoin

Fonction getHeureSoin ( ) : Heure

// accesseur sur l'attribut heureSoin

Fonction getL_Intervenant ( ) : Intervenant

// accesseur sur l'attribut l_Intervenant

FinClasse

Remarque : Le constructeur s'utilise de la manière suivante :

laPrest : Prestation
laPrest ☐ new Prestation(<arguments du constructeur>)

Classe Intervenant

Attributs privés :

nom : Chaîne
prénom : Chaîne
lesPrestations : Collection de Prestation

Méthodes publiques :

Intervenant (unNom : Chaîne, unPrénom : Chaîne)
// constructeur de la classe Intervenant
Fonction getNom () : Chaîne *// accesseur sur l'attribut nom*
Fonction getPrénom () : Chaîne *// accesseur sur l'attribut prénom*
Procédure ajoutePrestation (unePrestation : Prestation)
// ajoute unePrestation dans la collection lesPrestations

FinClasse

Classe IntervenantExterne : hérite de Intervenant

Attributs privés :

spécialité : Chaîne *// Spécialité, tel que Infirmier, Kinésithérapeute, Pédicure, etc.*
adresse : Chaîne
tél : Chaîne

Méthodes publiques

IntervenantExterne (unNom : Chaîne, unPrénom : Chaîne, uneSpécialité : Chaîne, uneAdresse : Chaîne, unTél : Chaîne) *// constructeur de la classe IntervenantExterne*
Fonction getSpécialité () : Chaîne *// accesseur sur l'attribut spécialité*
Fonction getAdresse () : Chaîne *// accesseur sur l'attribut adresse*
Fonction getTél () : Chaîne *// accesseur sur l'attribut tél*

FinClasse

Annexe C : Description textuelle des classes techniques

Classe Collection de <nom de la classe>

Méthodes publiques

Fonction cardinal () : Entier

// Renvoie le nombre d'objets de la collection

Fonction obtenirObjet (unIndex : Entier) : Objet de la classe

// Retourne l'objet d'index unIndex, le premier objet de la collection a pour index 1

Procédure ajouter (unObjet : Objet de la classe)

// Ajoute un objet à la collection

Procédure trier ()

// Trie la collection en se référant à la fonction compareTo de la classe des éléments.

FinClasse

Pour instancier une collection :

uneCollection : Collection de <classe>
uneCollection = new Collection() de <classe>

Pour parcourir par itération les éléments d'un objet Collection

Pour chaque <objet> dans <collection> faire
// instructions avec <objet>
FinPour

Classe Date

Méthodes publiques

Fonction jour () : Entier *// renvoie la valeur entière correspondant au jour du mois*
Fonction mois () : Entier *// renvoie la valeur entière correspondant au numéro du mois*
Fonction année () : Entier *// renvoie la valeur entière correspondant à l'année*
Fonction estEgale (uneDate : Date) : Booléen

// renvoie true si la date courante est égale au paramètre uneDate, false sinon

Fonction estSupérieure (uneDate : Date) : Booléen

// renvoie true si la date courante est supérieure au paramètre uneDate, false sinon

Fonction estInférieure (uneDate : Date) : Booléen

// renvoie true si la date courante est inférieure au paramètre uneDate, false sinon

FinClasse

Classe Heure

Méthodes publiques

Fonction heure () : Entier
// renvoie la valeur entière correspondant au nombre d'heures (0 à 23)
Fonction minute () : Entier
// renvoie la valeur entière correspondant au nombre de minutes (0 à 59)
Fonction seconde () : Entier
// renvoie la valeur entière correspondant au nombre de secondes (0 à 59)

FinClasse

Classe Objet *//toute classe dérive implicitement de la super classe Objet*

Méthodes publiques

Fonction getType () : Chaîne

// retourne dans une chaîne de caractères le nom de classe de l'objet courant

// Par exemple : Dossier unDossier = new Dossier()

// unDossier.getType() retournera la chaîne de caractères '' Dossier ''

FinClass