CAS RAFE

L'eau potable a toujours été l'un des premiers objets de coopération intercommunale. La sécurité de l'alimentation face à une ressource rare, difficile à mobiliser ou de mauvaise qualité, a poussé les municipalités à regrouper leurs moyens pour obtenir une distribution de qualité. Le Syndicat des Eaux de Gévaudan (SEG) s'est ainsi donné pour mission le captage, le traitement et la distribution de l'eau potable pour satisfaire les usagers répartis sur le territoire des communes regroupées au sein d'un syndicat de communes.

Le service public d'eau est géré en régie : son organisation et son fonctionnement sont assurés directement par le syndicat de communes, qui conserve ainsi une maîtrise complète de sa gestion avec ses propres moyens matériels, humains et financiers. Les communes ont la responsabilité complète des investissements, du fonctionnement des services des eaux, des relations avec les usagers, comme l'émission des factures d'eau et leur recouvrement.

GESTION DE LA CONSOMMATION D'EAU

À utiliser : annexes A, B, C

L'eau est un bien de consommation courante et son utilisation doit être contrôlée pour éviter tout gaspillage. La communauté de communes a donc décidé de mettre en œuvre un projet permettant une gestion dynamique et économe de cette ressource, afin d'en limiter les pertes, qui s'expliquent par plusieurs facteurs : débordement des réservoirs, volumes détournés, utilisations non comptabilisées (réseau incendie, eaux de lavage du domaine public, purges des réseaux après travaux, etc.), fuites dues à l'état des canalisations, etc.

Le service des eaux doit donc disposer de moyens pour connaître, puis localiser les origines des pertes. La sectorisation des réseaux est un outil d'aide à la gestion optimale des pertes. Chaque commune est donc divisée en secteurs géographiques (quartiers) qui sont alimentés en eau par des vannes reliées au réseau général. Ces vannes, équipées de compteurs, permettent de mesurer le volume d'eau utilisé dans chaque quartier. L'eau est ensuite distribuée aux usagers via le réseau de canalisations du quartier.

Une fois dans l'année, au mois de mars, les agents communaux relèvent les consommations d'eau de chaque secteur en consultant le compteur des vannes et les compteurs individuels de chaque abonné du quartier.

Sur chaque compteur (de vanne ou d'abonné) on relève le nouvel index (en m^3) ; la consommation est calculée en y soustrayant l'index précédent.

Pour un quartier, le relevé des index de consommation (vanne et compteurs individuels) s'effectue sur une demi-journée. *On considère comme nulle la consommation d'eau des foyers durant l'opération.*

Les annexes présentent le diagramme de classes (A), les classes métier (B) et la classe Collection (C).

TRAVAIL À FAIRE	
1	Écrire le constructeur de la classe *Secteur*.
2	Écrire la méthode *ajouterUnSecteur()* de la classe *Commune.*
3	Écrire la méthode *secteurEV()* de la classe *Commune*.
4	Écrire la méthode *anomalie()* de la classe *Commune*.

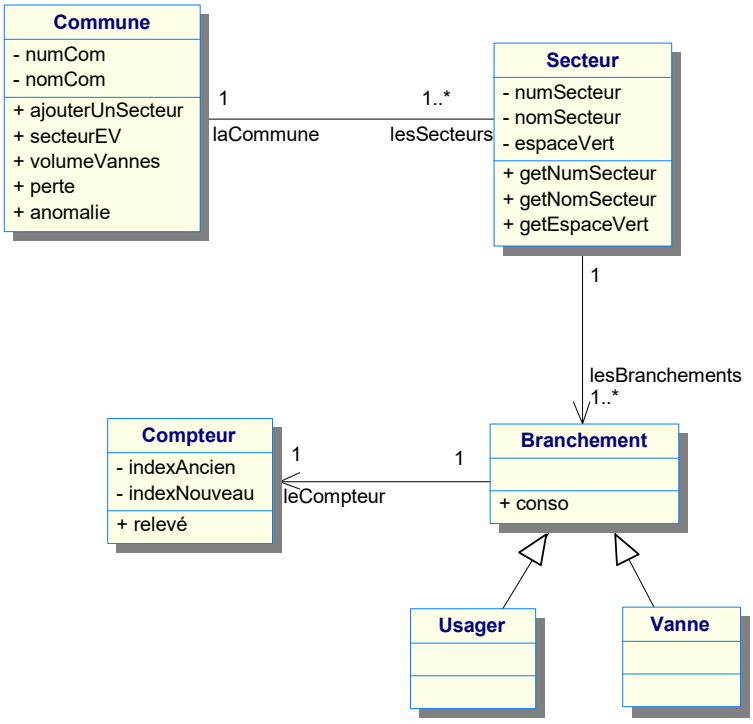
Il reste de nombreuses méthodes à écrire. Dans un premier temps on s'intéresse à la méthode *perte()* de la classe *Commune*. Il faut pour cela concevoir les autres méthodes utiles à la réalisation de cette méthode. Vous pourrez utiliser la fonction *typeDe(Objet)* qui retourne la classe d'une instance. Par exemple :

```
Si typeDe(lesBranchements.extraire(3)) = "Vanne" alors
        Afficher "C'est une vanne"
Fin Si
```

TRAVAIL À FAIRE	
5	Écrire la méthode *perte()* de la classe *Commune*. *Le candidat est libre d'ajouter et d'écrire toutes les méthodes (non citées dans les annexes) qui lui seront nécessaires*.

ANNEXE A : DIAGRAMME DE CLASSES PARTIEL

Les classes sont simplifiées.
Les constructeurs ne figurent pas sur le schéma par souci de lisibilité.
Les paramètres des méthodes ne sont pas présentés.



ANNEXE B : CLASSES MÉTIER

Classe Commune
Privé
numCom : Chaîne
nomCom : Chaîne
lesSecteurs : Collection de Secteur
Public
Commune(numéro : Chaîne, nom : Chaîne)
procédure ajouterUnSecteur(unNuméroSecteur : Entier,
unNomSecteur : Chaîne,
unEspaceVert : Booléen)
// Création du secteur, et ajout dans la collection lesSecteurs
fonction secteurEV() : Collection de Secteur
// Retourne les secteurs qui possèdent un espace vert
fonction volumeVannes() : Entier
// Retourne le volume total distribué par les vannes de la commune
fonction perte() : Entier
// Retourne la différence entre le volume total distribué par les vannes
// et la consommation des usagers
fonction anomalie() : Entier
// Calcule le pourcentage des pertes par rapport au volume distribué
// par les vannes et retourne :
// 1 si ce pourcentage est inférieur à 10 %
// 2 s'il est entre 10 et 15 % inclus
// 3 au dessus de 15 %
…
Fin Classe Commune

Classe Secteur
Privé
numSecteur : Entier
nomSecteur : Chaîne *// nom du quartier*
espaceVert : Booléen
// Indique la présence ou non dans le quartier d'un espace vert municipal à arroser
laCommune : Commune
lesBranchements : Collection de Branchement
Public
Secteur(unNuméroSecteur : Entier, unNomSecteur : Chaîne, unEspaceVert : Booléen, uneCommune : Commune)
fonction getNumSecteur() : Entier
fonction getNomSecteur() : Chaîne
fonction getEspaceVert() : Booléen
…
Fin Classe Secteur

Classe Branchement
Privé
leCompteur : Compteur
Public
Fonction conso() : Entier
Début
retourner leCompteur.relevé()
Fin
…
Fin Classe Branchement

Classe Compteur
Privé
indexAncien : Entier
indexNouveau : Entier
Public
Fonction relevé() : Entier
Début
retourner (indexNouveau – indexAncien)
Fin
…
Fin Classe Compteur

Classe Usager hérite de Branchement
…
Fin Classe Usager

Classe Vanne hérite de Branchement
…
Fin Classe Vanne

ANNEXE C : CLASSE COLLECTION

Classe Collection de TypeElément

// où TypeElément peut être un type simple ou une classe.

Privé

…

Public

…

fonction cardinal() : Entier *// Retourne le nombre d'éléments de la collection.*

fonction extraire(unIndex : Entier) : TypeElément

// Retourne l'objet d'index unIndex. Le premier élément est à l'index 1.

procédure ajouter(unObjet : TypeElément) *// Ajoute un objet à la collection.*

procédure enlever(unIndex : Entier)

// Supprime l'objet d'index unIndex de la collection.

procédure vider() *// Vide le contenu de la collection.*

Fin Classe Collection

Utilisation :

```
uneCol : Collection de Date
uneDate : Date
uneCol ← new Collection de Date
…
uneCol.ajouter(uneDate)
```

Parcours par itération les éléments d'un objet Collection :

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