CAS SETETE & DOS

La société de services informatiques SETETE propose des développements informatiques sur mesure à une clientèle d'entreprises ou d'organisations n'ayant pas trouvé de solutions satisfaisantes dans les progiciels standards du marché. La société SETETE vient d'être saisie d'une demande du comité d'établissement (CE) d'une entreprise régionale.

Celui-ci souhaite améliorer sa gestion en accroîssant l'automatisation du traitement de ses activités. Il propose un large éventail de prestations concernant les loisirs et activités extraprofessionnelles des employés et de leurs familles. Les différents types de prestations proposées vont du prêt de livres à la vente de billets de cinéma à tarif réduit, en passant par la proposition de séjours de vacances et la vente de jouets à l'occasion des fêtes de fin d'année.

M. Baratin, responsable du CE, est l'interlocuteur de SETETE dans l'entreprise.

Pour préciser sa demande, il a souhaité dans un premier temps faire état des réalisations et réflexions déjà menées en interne, puis de souhaits particuliers.

SETETE élaborera une proposition commerciale intégrant tous les éléments du projet.

Offre de jouets pour Noël

Annexe à utiliser.

Chaque année, le CE organise un achat groupé de jouets pour les enfants du personnel. Le catalogue des jouets proposés est disponible sur l'intranet de la société ; chaque salarié peut commander un jouet pour chacun de ses enfants âgés de moins de 15 ans.

Les jouets sont répartis d'une part par tranche d'âge (exemple : 5 à 7 ans), et d'autre part par catégorie de jouet (exemple : éducatif, jeu de société, etc.).

Une application, nommée DOS, est en cours de développement. Son objectif est d'optimiser l'offre de jouets en fonction des commandes des années précédentes. Par exemple, il est inutile de proposer un éventail important de jeux de société pour les enfants de 5 à 7 ans si ce type de produit a été très peu commandé.

M. Baratin souhaite que la finalisation de cette application soit prise en charge par la société SETETE.

La description des classes utilisées figure en *annexe*.

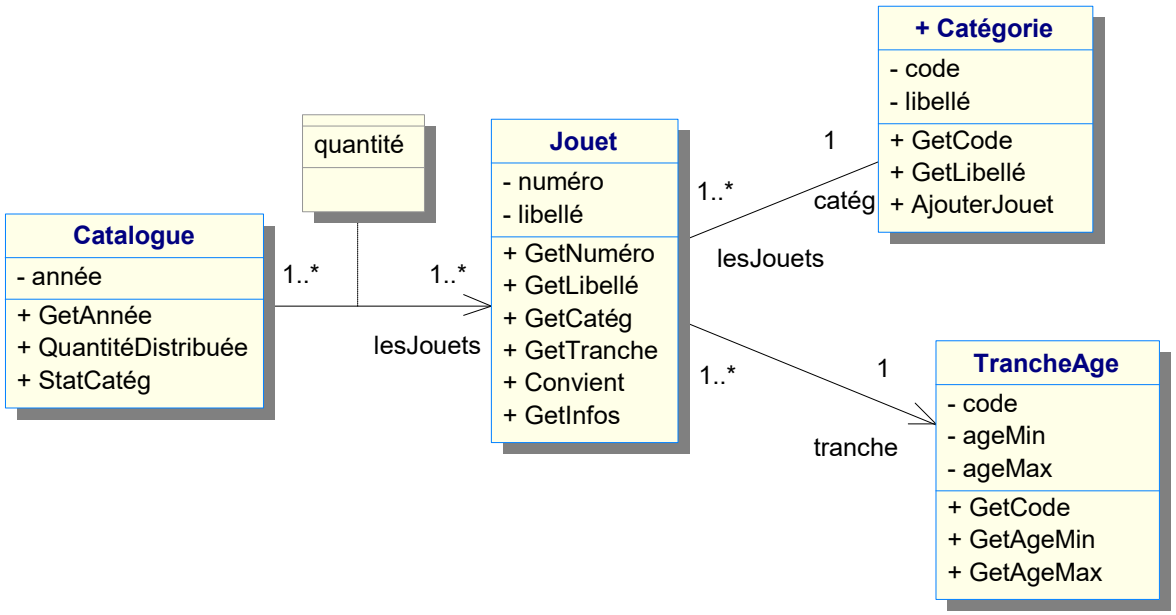
Travail à faire :	
1	Écrire la méthode *Convient()* de la classe Jouet.
2	Écrire la méthode *GetInfos()* de la classe Jouet.
3	Écrire le constructeur de la classe Jouet.
4	Écrire la méthode *QuantitéDistribuée()* de la classe Catalogue.
5	Écrire la méthode *StatCatég()* de la classe Catalogue.

ANNEXE : Description des classes utilisées (extrait)

Remarques :

- Chaque objet de la classe *Catalogue* correspond à une année.
- Un jouet proposé plusieurs années de suite est présent dans les catalogues correspondants.

Diagramme de classes :



La classe association a été implémentée par un dictionnaire (*lesJouets*) dans la classe Catalogue.
Le diagramme de classes ne mentionne ni les paramètres des méthodes, ni les constructeurs des classes.

Classe Catalogue
privé
année : Chaîne
lesJouets : Dictionnaire de <Jouet, Entier>
// Contient pour chaque jouet du catalogue :
// - en clé, l'objet de la classe Jouet
// - en valeur, la quantité de ce jouet distribuée pour ce catalogue
public
Constructeur Catalogue (uneAnnée : Chaîne)
Fonction GetAnnée ( ) : Chaîne
Fonction QuantitéDistribuée () : Entier
// Retourne la quantité totale de jouets distribués pour ce catalogue.
Fonction StatCatég () : Dictionnaire de <Catégorie, Entier>
// Retourne un dictionnaire contenant pour chaque catégorie de ce catalogue :
// - en clé, l'objet de la classe Catégorie
// - en valeur, la quantité de jouets distribués pour cette catégorie.
Fin classe

Classe Jouet
privé
numéro : Entier
libellé : Chaîne
catég : Catégorie
tranche : TrancheAge
public
Constructeur Jouet (unNumero : Entier, unLibellé : Chaîne, uneCatégorie : Catégorie, uneTranche : TrancheAge)
// Instancie un objet Jouet et l'ajoute dans la collection des jouets de sa catégorie
Fonction GetNuméro ( ) : Entier
Fonction GetLibellé ( ) : Chaîne
Fonction GetCatég () : Catégorie
Fonction GetTranche () : TrancheAge
Fonction Convient (unAge : Entier) : Booléen
// Retourne vrai si le jouet convient à l'âge passé en paramètre.
Fonction GetInfos () : Chaîne
// Retourne une chaîne contenant : le libellé du jouet, le libellé de sa catégorie,
// les âges minimum et maximum de la tranche d'âge lui correspondant.
// Les informations sont séparées par des points-virgules.
Fin classe

Classe Catégorie
privé
code : Entier
libellé : Chaîne
lesJouets : Collection de <Jouet>
// ensemble des jouets de cette catégorie
public
Constructeur Catégorie (unCode : Entier, unLibellé : Chaîne)
Fonction GetCode ( ) : Entier
Fonction GetLibellé ( ) : Chaîne
Procédure AjouterJouet (unJouet : Jouet)
// Ajoute le jouet passé en paramètre à la collection lesJouets
Fin classe

Classe TrancheAge
privé
code : Entier
ageMin : Entier
ageMax : Entier
public
Constructeur TrancheAge (unCode : Entier, unAgeMin : Entier, unAgeMax : Entier)
Fonction GetCode ( ) : Entier
Fonction GetAgeMin ( ) : Entier
Fonction GetAgeMax ( ) : Entier
Fin classe

Classe Collection de <nom de la classe>

public

Fonction Cardinal () : Entier
// Renvoie le nombre d'objets de la collection
Fonction ObtenirObjet (e unIndex : Entier) : Objet de la classe
// Retourne l'objet d'index unIndex
Procédure Ajouter (e unObjet : Objet de la classe)
// Ajoute un objet à la collection

Fin classe

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 Dictionnaire de <*TypeClé, TypeValeur*>
// collection d'éléments (clé, valeur) permettant d'extraire une valeur (de type TypeValeur)
// à partir de sa clé (de type TypeClé) ; à une clé présente dans le dictionnaire correspond
// une et une seule valeur

public

procédure Ajouter (clé : TypeClé, valeur : TypeValeur)
// ajoute un élément (clé, valeur) dont le premier paramètre est la clé et le second
// paramètre la valeur.
procédure Modifier (clé : TypeClé, valeur : TypeValeur)
// modifie la valeur de l'élément dont la clé est le premier paramètre, la nouvelle valeur
// étant le second paramètre.
fonction DonnerValeur (clé : TypeClé) : TypeValeur
// retourne la valeur correspondant à la clé passée en paramètre
fonction DonnerToutesLesClés () : Collection de <TypeClé>
// retourne une collection contenant toutes les clés du dictionnaire
procédure Retirer (clé : TypeClé)
// retire du dictionnaire un élément dont la clé est fournie en paramètre ; ne fait rien si
// cette clé n'est pas présente
fonction Existe (clé : TypeClé) : Booléen
// retourne vrai si l'élément dont la clé est passée en paramètre est présent dans le
// dictionnaire

Fin classe

Exemple d'utilisation du dictionnaire :

```
unDico : Dictionnaire de <Date,Entier>
nbHeures : Entier
dateJ : Date
unDico ←new Dictionnaire de <Date,Entier>
…
nbHeures ← unDico.DonnerValeur(dateJ)
```

Remarque : Pour parcourir un dictionnaire, il faut :

- Récupérer la collection des clés : fonction *DonnerToutesLesClés() ;*
- Parcourir ensuite la collection de clés.