Ruben BLANCUZZI
Noé DOUDOU
Maxime ERNANDEZ
Paul HOPSORE
Marek SEDLACEK

Rapport Projet Java UML

1er Rendu

Introduction et objectifs :

Dans ce projet, nous nous intéressons à un système de gestion et de tri de déchets. Notre objectif premier est de modéliser tous les acteurs de ce système par un diagramme UML. Pour cela, nous devons mettre en évidence les différents objets à manipuler et comprendre comment ils interagissent entre eux.Tout en gardant à l'esprit que cette modélisation servira de base à une implémentation en Java.

Organisation du travail :

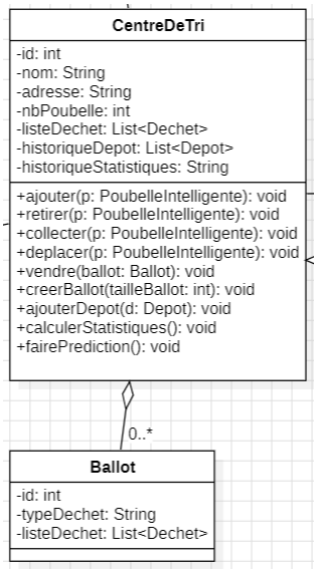
Pour organiser au mieux notre travail, nous avons utilisé un serveur discord dédié, sur lequel chacun partageait ses avancées, et nous avions prévu des réunions de manière régulière afin de faire le point sur l'avancée du rendu. C'est un outil qui s'est révélé très efficace pour travailler en groupe, notamment grâce aux appels et à la fonction de partage d'écran.

Pour la répartition des tâches, une grande partie du travail a été faite en appel discord, un des membres partageait son écran et les autres donnaient leurs idées, ce qui permet de confronter les opinions de chacun et de choisir la meilleure solution. Pour le reste, chacun a choisi une partie du projet afin de la mettre sous la forme UML.

Conception du diagramme :

Description des classes principales :

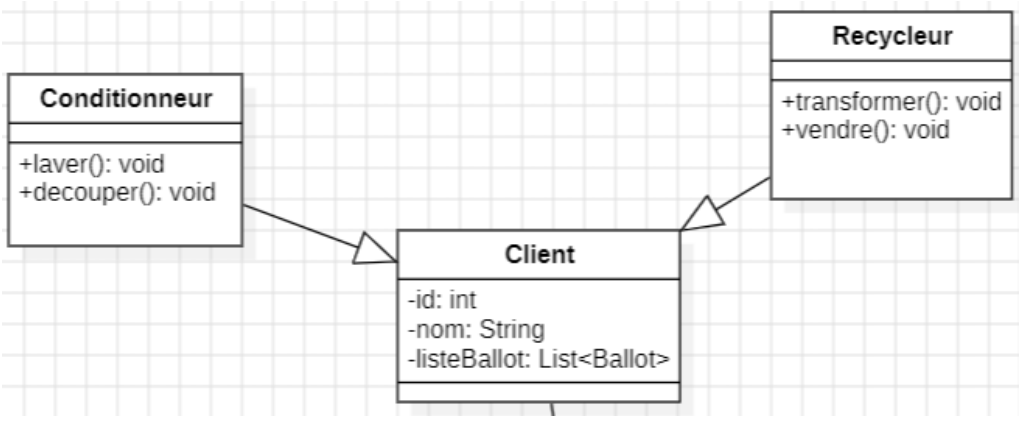
Le socle principal de notre diagramme est la classe **CentreDeTri.** Elle possède les attributs et les méthodes ci-dessous qui permettent notamment d'effectuer plusieurs actions sur les poubelles qui lui sont associées.



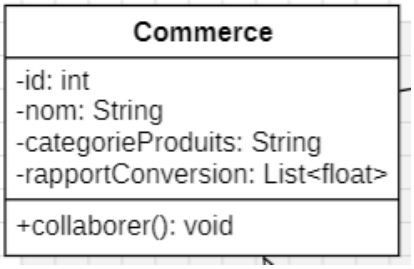
Explication de la méthode creerBallot(tailleBallot : int) :

On choisit un nombre de déchets que l'on veut placer dans le ballot (tailleBallot), et on construit un ensemble de déchets en ne sélectionnant que des déchets d'une même nature.

Le centre de tri peut vendre les déchets qu'il détient à ses clients : des **conditionneurs** et des **recycleurs**

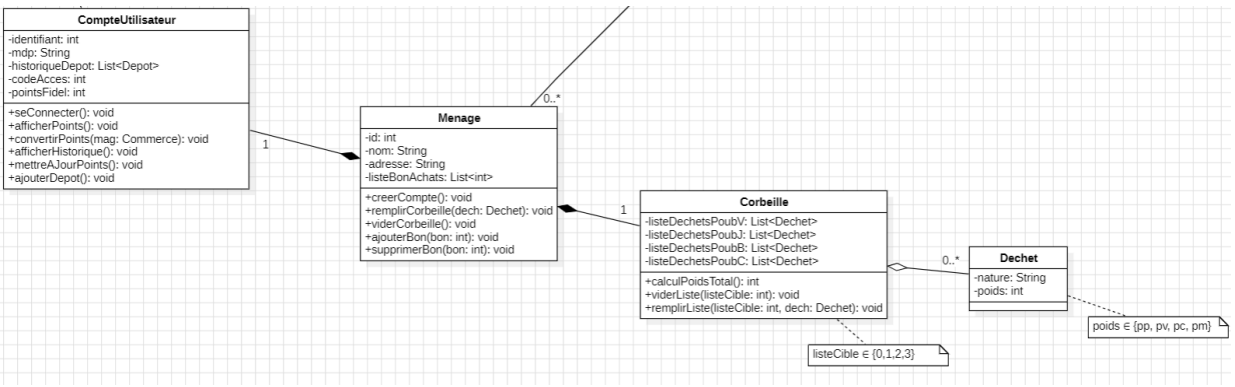


Le centre de tri fait également des partenariats avec des **commerces.**

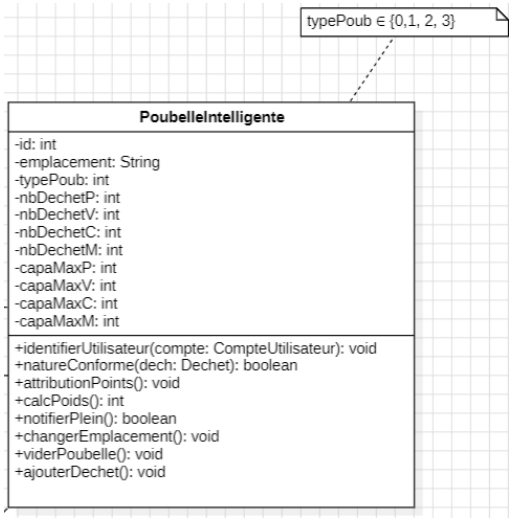


L'attribut rapportConversion ci-dessus donne les correspondances entre nombre de points de fidélité et remise.

On doit aussi représenter les **ménages**, ils ont chacun un **CompteUtilisateur** et une **Corbeille**. La corbeille est composée de différents **déchets**, et on a décidé de partitionner la corbeille de l'utilisateur en 4 sous-corbeilles de déchets : listeDechcetPoubV, J, B et C. Chacune de ces sous-corbeilles correspond au tri effectué par l'utilisateur avant de vider ses déchets dans une poubelle. (Il place par exemple les déchets qu'il prévoit de jeter dans la poubelle verte dans la sous corbeille listeDechetPoubV).



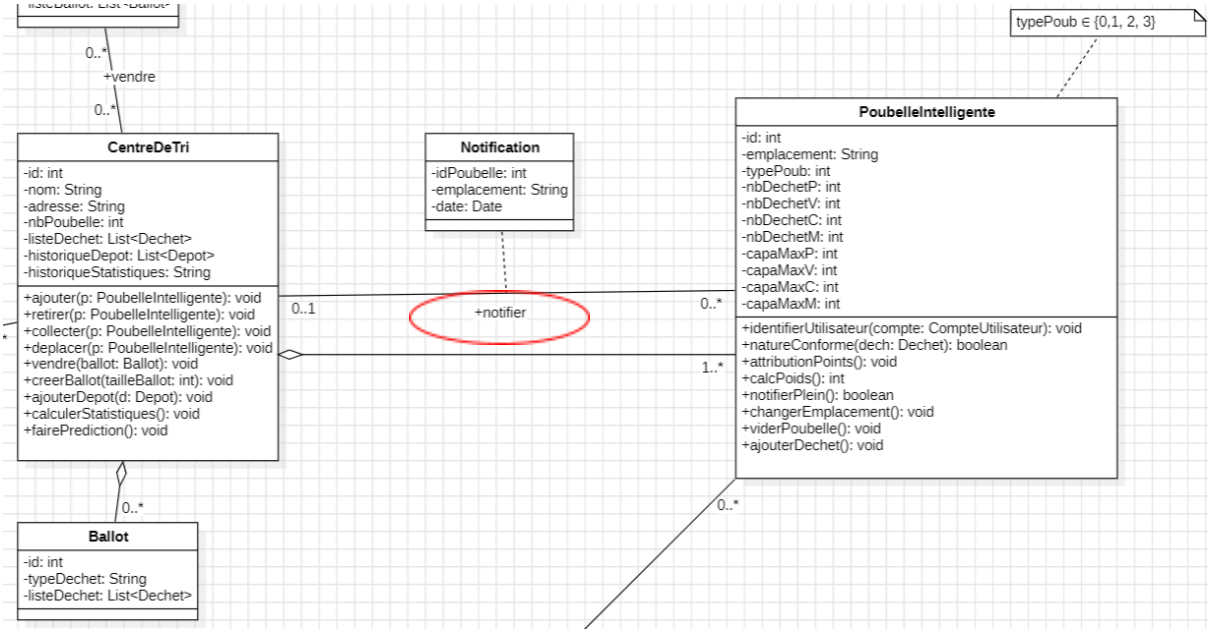
La classe **PoubelleIntelligente** représente les poubelles dans lesquelles sont jetés les déchets par les utilisateurs. Elle peut être de 4 types différents : 0 : Verte, 1 : Jaune, 2 : Bleue ou 3 : Classique.



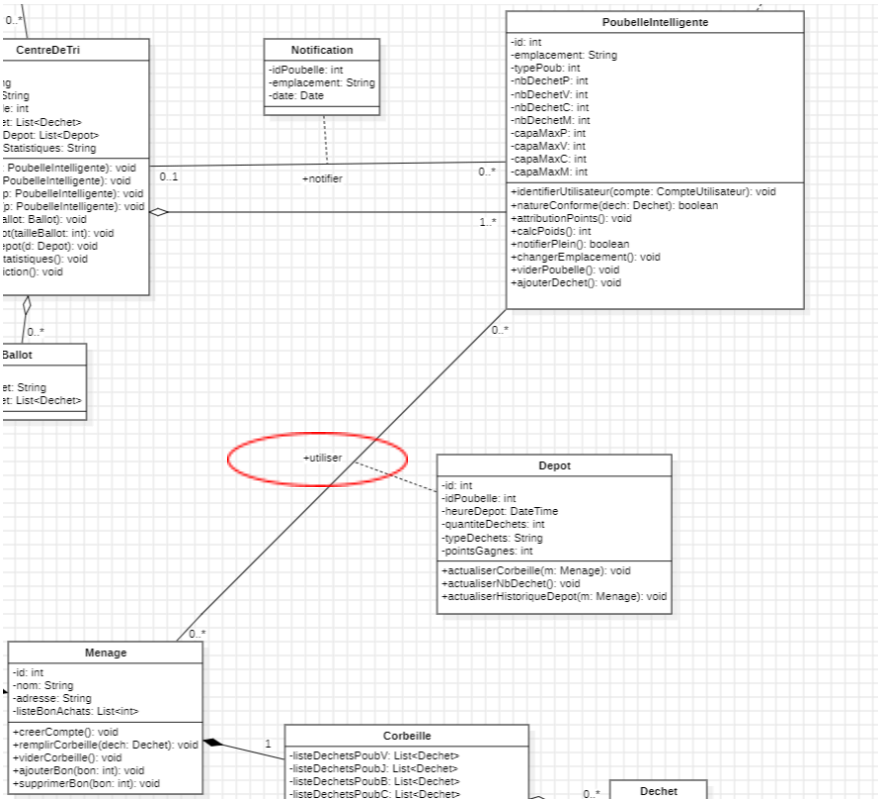
Associations entre classes

Mettons maintenant en évidence les relations et associations entre les classes.

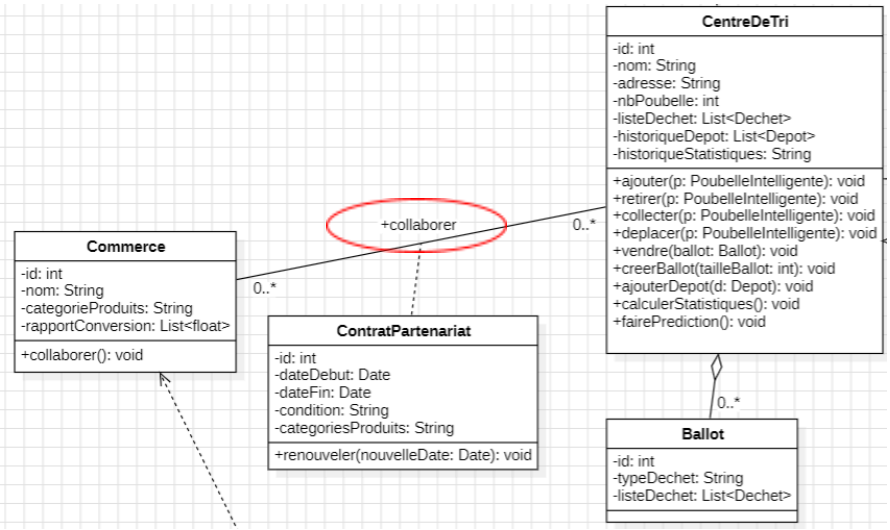
Le centre de tri peut recevoir des notifications de ses poubelles lorsqu'elles sont pleines. C'est pourquoi nous avons placé une association notifier entre ces deux classes :



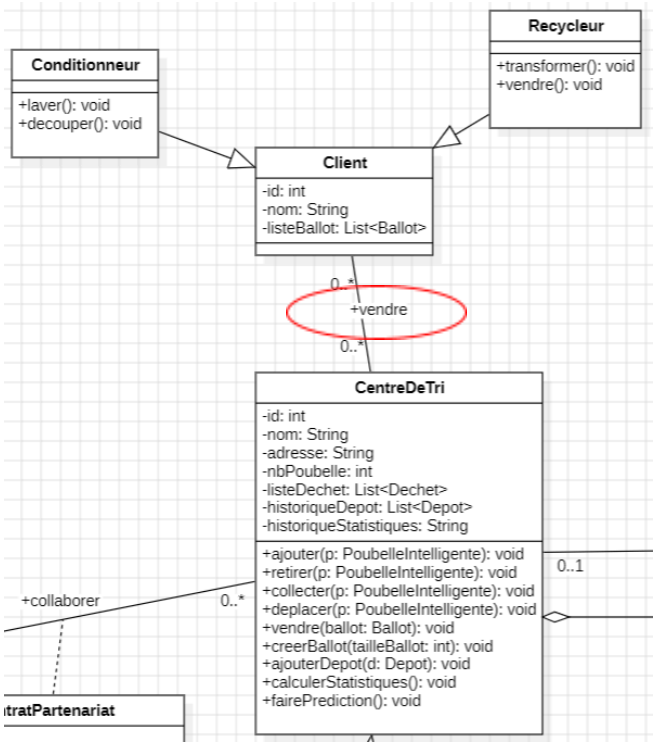
Ensuite, lorsqu'un utilisateur veut jeter ses déchets, il interagit avec une poubelle intelligente via une association que l'on a appelé utiliser.



On représente aussi les différents partenariats à l'aide d'une association collaborer qui relie commerce et centre de tri.



Enfin, on modélise la capacité du centre de tri à vendre ses déchets par une association vendre qui relie le centre de tri et ses clients.



Scénario typique du Gestion de Tri sélectif

Legende: Classe/Objet ; Attribut ; Méthode

1.Le Ménage initie le cycle de tri en utilisant sa Corbeille.

- Action : Un Dechet est ajouté à la corbeille via la méthode remplirCorbeille(dech: Dechet).
- Mécanisme interne : Cette méthode appelle remplirListe(listeCible: int, dech: Dechet) pour ajouter le Dechet à la liste appropriée (plastique, verre, carton, métal) selon le choix de l'utilisateur/Menage. *Note : L'utilisateur peut se tromper de liste.*

2. Lorsque la corbeille est vidée dans une PoubelleIntelligente :

- Création d'un Depot :
  - identifiant de la PoubelleIntelligente
  - heureDepot du dépôt
  - quantiteDechets et typeDechets de déchets déposés (déduits du compartiment de la corbeille)
  - pointsGagnes potentiellement attribués.
- Mise à jour de la Corbeille : La méthode actualiserCorbeille() est appelée pour réinitialiser la liste de déchets correspondant à celle qui vient d'être vidée, la rendant prête pour de nouveaux déchets.
- Mise à jour de la PoubelleIntelligente : La méthode actualiserNbDechet() est exécutée pour incrémenter le nombre total de déchets et son type reçus par la PoubelleIntelligente.

3. La PoubelleIntelligente effectue une vérification de conformité :

- Contrôle de la nature des déchets : La méthode natureConforme(dech: Dechet) est utilisée pour vérifier si le déchet déposé correspond à la catégorie de la poubelle.
- Attribution de points (ou pénalité) :
  - Si natureConforme() est vrai (tri correct), la méthode attribuerPoints() récompense le Menage en créditant des pointsFidel sur son CompteUtilisateur mais aussi PointsGagnes sur le Depot.
  - Si natureConforme() est faux (erreur de tri), une pénalité peut être appliquée via la même méthode attribuerPoints().

4. Suivi du niveau de remplissage de la PoubelleIntelligente :

- Calcul du poids : La méthode calcPoids() est exécutée régulièrement pour déterminer le poids total des déchets dans la PoubelleIntelligente.
- Notification de remplissage maximal : Si le poids atteint la capacité maximale, la méthode notifierPlein() est appelée.
- Signal au CentreDeTri : notifierPlein() envoie une notification au CentreDeTri pour planifier la vidange de la PoubelleIntelligente.

5. Processus de Conversion des Points :

- L'utilisateur se connecte à son CompteUtilisateur (seConnecter()).
- Il consulte son solde de pointsFidel (afficherPoints()).
- Il initie la conversion des points et ajoute un nouveau bon d'achat à la listeBonAchats du Menage via convertirPoints(mag: Commerce). Pas besoin de mettre le Menage en paramètre car il n'y a qu'un seul ménage par CompteUtilisateur.
- Le bon d'achat est égal au pointsFidel x rapportConversion.

Alternative envisageable

Au lieu d'ajouter un simple entier à la listeBonAchats, on ajouterait un **objet** BonAchat. La méthode dans Menage ne serait plus ajouterBon(bon: int) mais plutôt ajouterBon(bon: BonAchat). La listeBonAchats deviendrait une liste d'objets de type BonAchat et non plus une simple liste de valeurs numériques ou autre type simple.