**Explication visibilité de chaque attributs et méthodes**

Pour cette modélisation UML, nous avons choisi de faire un diagramme composé de 7 classes UML :

* CentreDeTri
* Poubelle
* Utilisateur
* Corbeille
* Dechet
* Commerce
* ContratPartenariat

**Visibilité des attributs et méthodes pour chaque classe UML**

**1. CentreDeTri**

Cette classe comporte comme attributs (privés, -) :

* + - nom : String -> identifie les centres de tri
  + - adresse : String -> indique l’emplacement du centre pour organiser la logistique.

Et comme méthode (publique, +) :

* + + placerPoubelle() -> permet d’ajouter ou déplacer des poubelles dans la ville.
  + + collecterDechets() -> organise la collecte des déchets par les camions.
  + + genererStatistiques() -> analyse les données pour optimiser le tri et la collecte.

Le centre de tri est responsable de la gestion des poubelles, du tri et de l'analyse des données pour améliorer le recyclage.

Les attributs sont privés (-) pour garantir l'encapsulation et éviter toute modification directe en dehors de la classe.

Les méthodes sont publiques (+) car elles représentent les actions du centre de tri accessibles par d'autres classes.

**2. Poubelle**

Cette classe comporte comme attributs (privés, -) :

* + – id : int -> chaque poubelle a un identifiant unique.
  + – capaciteMax : int -> définit la capacité maximale pour éviter les débordements.
  + – emplacement : String -> permet de savoir où la poubelle est installée.

Et comme méthode (publique, +) :

* + + identifierUtilisateur() -> vérifie l'accès des utilisateurs avec les identifiants
  + + calculerQuantiteDechets() -> calcul le nombre de déchets jetés
  + + verifierCategorieDechets() -> vérifie que le tri est bien respecté
  + + attribuerPoints() -> attribue des points de fidélité en fonction du tri
  + + notifierCentreTri() -> envoi d’une alerte si la poubelle est pleine ou si le tri n’est pas respecté

La poubelle est assez indépendante pour vérifier si le tri est bien respecté et envoi une alerte dans le cas ou ce n’est pas respecté ou si la poubelle est pleine.

Les attributs sont privés (-) pour empêcher leur modification directe (par exemple : empêcher de changer l’ID d’une poubelle).

Les méthodes sont publiques (+) car elles doivent être appelées par d'autres classes, comme Utilisateur ou CentreDeTri.

**3. Utilisateur**

Cette classe comporte comme attributs (privés, -) :

* + – id : int -> Identifiant unique de chaque utilisateur
  + – nom : String -> Nom de l'utilisateur pour la gestion du compte
  + – pointsFidelite : int -> Stocke les points gagnés grâce au bon tri

Et comme méthode (publique, +) :

* + + deposerDechets() -> Permet à l’utilisateur de jeter ses déchets dans une poubelle connectée
  + + consulterHistorique() -> Affiche les dépôts de déchets et les points gagnés
  + + convertirPoints() -> Permet d’échanger les points en bons d’achat

L’utilisateur est le principal acteur du tri et doit pouvoir suivre ses actions et bénéficier du système de récompenses.

Les attributs sont privés (-) pour protéger les informations personnelles et éviter la modification directe du nombre de points de fidélité.

Les méthodes sont publiques (+) car elles permettent à l’utilisateur d’interagir avec le système.

**4. Corbeille**

Cette classe comporte comme attributs (privés, -) :

* + – listeDechets : List<Dechet> -> Contient tous les déchets collectés avant dépôt dans une poubelle.

Et comme méthode (publique, +) :

* + + ajouterDechets() -> ajoute un déchet à la corbeille.
  + + viderCorbeille() -> vide la corbeille dans une poubelle.

Une corbeille est un contenant temporaire pour stocker les déchets avant qu'ils soient triés et jetés.

L’attribut listeDechets est privé (-) pour éviter que d'autres classes ne modifient directement son contenu.

Les méthodes sont publiques (+) car elles permettent d’ajouter ou de vider la corbeille.

**5. Dechet**

Cette classe comporte comme attributs (privés, -) :

* + – type : String -> définit le type de déchet, tels que plastique, verre, métal….
  + – poids : float -> mesure le poids de chaque déchet.

Et comme méthode (publique, +) :

* + + getType() -> Retourne le type de déchet.
  + + getPoids() -> Retourne le poids du déchet.

Chaque déchet a un type et un poids, nécessaires pour le tri et l'attribution des points.

Les attributs sont privés (-) pour empêcher la modification directe du type ou du poids d’un déchet.

Les méthodes de lecture getType() et getPoids() sont publiques (+) pour que d’autres classes puissent lire ces informations sans pouvoir les modifier directement.

**6. Commerce**

Cette classe comporte comme attributs (privés, -) :

* + – nom : String -> Identifie le commerce.
  + – categoriesProduit : List<String> -> Liste des types de produits qui acceptent les points fidélité.

Et comme méthode (publique, +) :

* + + echnagerPoints() -> Convertit les points fidélité en réductions ou bons d’achat.

Les commerces partenaires permettent aux utilisateurs d’utiliser leurs points pour des réductions sur des produits.

Les attributs sont privés (-) pour éviter des modifications non contrôlées (exemple : éviter que n’importe quelle classe ne modifie les catégories de produits).

La méthode est publique (+) pour permettre aux utilisateurs d’échanger leurs points contre des réductions.

**7. ContratPartenariat**

Cette classe comporte comme attributs (privés, -) :

* + – dateDebut : Date -> Indique la date de début du contrat.
  + – dateFin : Date -> Indique la date de fin du contrat.

Et comme méthode (publique, +) :

* + + definirReglesUtilisation() -> Détermine les conditions d'utilisation des points fidélité.

Chaque commerce a un contrat avec le centre de tri, qui définit la durée et les conditions d’utilisation des points.

Les attributs sont privés (-) pour protéger les informations du contrat.

La méthode est publique (+) pour que le commerce puisse définir les règles d’utilisation.

**Explication cardinalité et choix d’association**

1. **CentreDeTri → gère plusieurs Poubelles**

Le type d’association est une aggregation (\◇). Sa cardinalité est 1 (CentreDeTri) → 0..\* (Poubelle). En effet, un centre de tri possède plusieurs poubelles mais celles-ci peuvent exister indépendamment du centre de tri.

1. **Utilisateur → dépose dans une Poubelle**

Il s’agit ici d’une association simple (—). Sa cardinalité est de 1 (Utilisateur) → 0..\* (Poubelle). En effet, un utilisateur peut déposer ses déchets dans plusieurs poubelles et une poubelle est utilisée par plusieurs utilisateurs.

1. **Poubelle → stocke plusieurs Déchets**

Le type d’association est une composition (◆). Sa cardinalité est de 1 (Poubelle) → 0..\* (Dechet). En effet, une poubelle contient des déchets et ceux-ci n’existent pas sans une poubelle spécifique (si la poubelle disparait, les déchets ne sont plus stockés).

1. **Utilisateur → possède une Corbeille**

Le type d’association est une aggregation (\◇). Sa cardinalité est de 1 (Utilisateur) → 1 (Corbeille). En effet, un utilisateur a une seule corbeille et celle-ci lui appartient mais la corbeille peut exister sans utilisateur.

1. **Corbeille → contient plusieurs Déchets**

Le type d’association est une composition (◆). Sa cardinalité est de 1 (Corbeille) → 0..\* (Dechet). En effet, une corbeille contient plusieurs déchets et ces déchets disparaissent avec la corbeille.

1. **Utilisateur → échange des points avec Commerce**

Il s’agit ici d’une association simple (—). Sa cardinalité est de 1 (Utilisateur) → 0..\* (Commerce). En effet, un utilisateur peut échanger ses points avec plusieurs commerces et un commerce peut avoir plusieurs utilisateurs.

1. **Commerce → définit un ContratPartenariat avec CentreDeTri**

Il s’agit ici d’une association simple (—). Sa cardinalité est de 1 (Commerce) → 1 (ContratPartenariat) → 1 (CentreDeTri). En effet, chaque commerce définit un contrat avec un centre de tri, un contrat lie un seul commerce et un seul centre de tri.