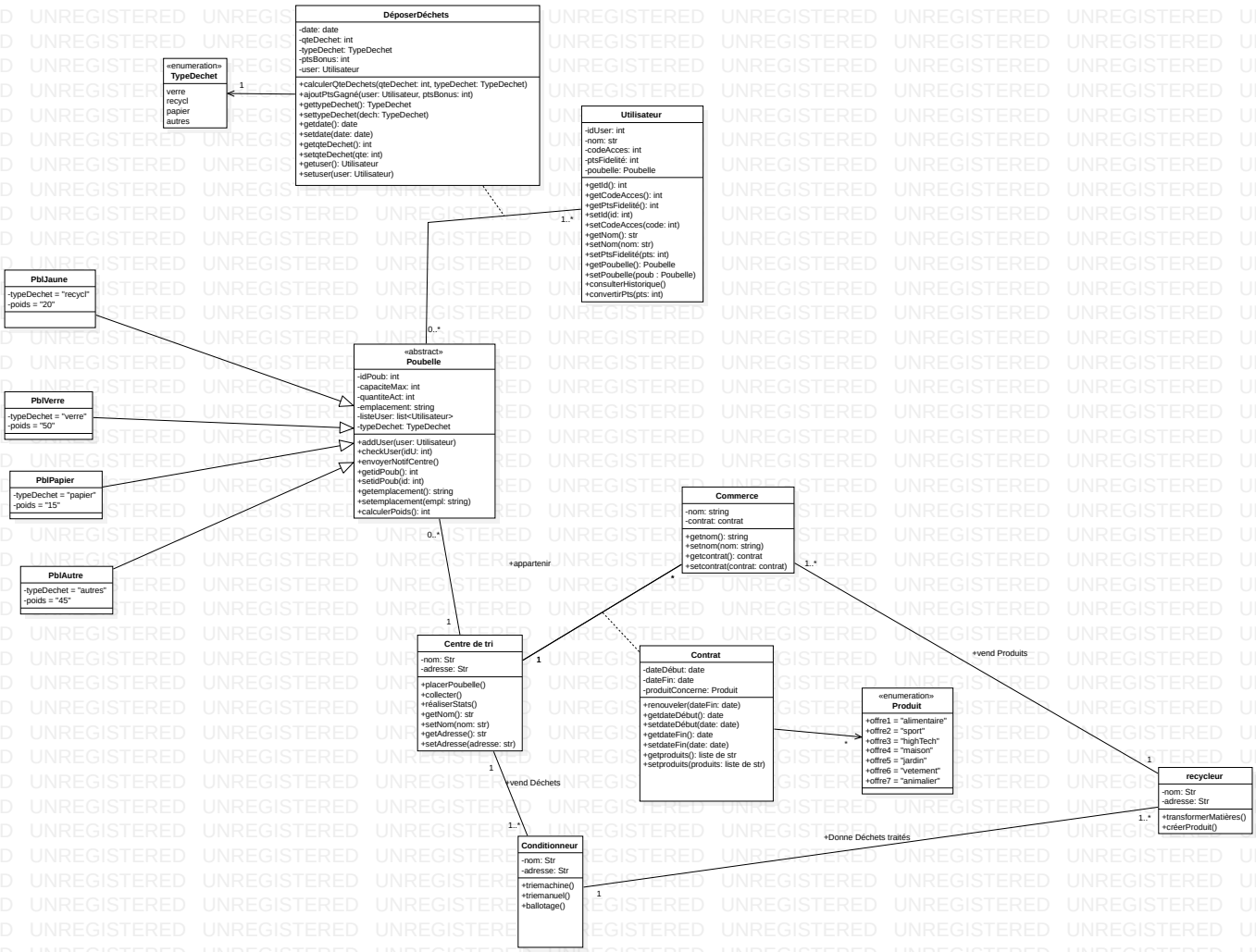
Rapport groupe numéro 6 GMI

Diagramme UML adapté :

Model.Main



Les différentes classes :

-Utilisateur
getId() int permet de récupérer l'id de l'utilisateur
getCodeAcces():int permet de récupérer le Code d'accès de l'utilisateur
getPtsFidelité():int permet de récupérer le nombre de point de l'utilisateur
setId(id:int) :permet de définir l'id d'un utilisateur
setCodeAcces(code : int) :permet de définir le Code d'accès de l'utilisateur
getNom() : str permet de récupérer le nom de l'utilisateur
setNom(nom : str) :perdmet de définir le nom de l'utilisateur
setPtsFidelité(in pts:int) permet de définir les points de fidéité de l'utilisateur
getPoubelle(): Poubelle permet de donner la poubelle atribuée à l'utilisateur
setPoubelle(in poub :Poubelle) permet de définir une poubelle pour l'utilisateur
consulterHistorique() permet de consulter l'historique des dépôts de déchets
convertirPts(in pts:int) :permet de convertir les points pour les transformer en bon de réduction

-Poubelle
addUser(in user:Utilisateur) permet d'ajouter un utilisateur à cette poubelle
checkUser(in idU:int) permet d'avoir l'id de l'utilisateur
envoyerNotifCentre() permet d'envoyer une notification au centre de tri (notamment quand une poubelle est pleine)
getidPoub(): int permet d'avoir l'id de la poubelle
setidPoub(in id:int) permet de définir l'id de la poubelle
getemplacement(): string permet d'avoir l'emplacement de la poubelle
setemplacement(in empl:string) permet de définir l'emplacement de la poubelle
calculerPoids(): int permet de calculer le poids dans la poubelle (qui dépendra du type de déchet)

-Centre de tri
placerPoubelle() permet de placer une poubelle
collecter() permet de collecter une poubelle
réaliserStats() permet de réaliser des stats (avec en paramètre le type de déchet et le nombre de déchets)
getNom(): str permet d'avoir le nom du centre
setNom(in nom:str) permet de définir le nom du centre
getAdresse(): str permet d'avoir l'adresse du centre
setAdresse(in adresse:str) permet de définir l'adresse du centre

-Commerce
getnom(): string permet d'avoir le nom du commerce
setnom(in nom:string) permet de définir le nom du commerce
getcontrat(): contrat permet d'avoir accès au contrat du commerce avec le contre de tri

setcontrat(in contrat:contrat) permet de définir un contrat

-DéposerDéchets
calculerQteDechets(in qteDechet:int, in typeDechet:TypeDechet) permet de calculer la quantité de déchet en fonction du type de déchets
ajoutPtsGagné(in user:Utilisateur, in ptsBonus:int) permet d'ajouter un nombre de point « ptsBonus » à un « Utilisateur »
gettypeDechet(): TypeDechet permet de savoir quel type de déchet a été jeté
settypeDechet(in dech:TypeDechet) permet de définir le type de déchet jeté
getdate(): date permet de savoir la date du dépôt de déchets
setdate(in date:date) permet de définir la date du dépôt de déchets
getqteDechet(): int retourne la quantité de déchets déposée
setqteDechet(in qte:int) permet de définir la quantité de déchets déposés
getuser(): Utilisateur retoune l'utilisateur qui a déposé les déchets
setuser(in user:Utilisateur) permet de définir l'utilisateur qui a déposé les déchets

-Contrat
renouveler(in dateFin:date) permet de renouveler le contrat en changeant la date de fin
getdateDébut(): date retourne la date de début du contrat
setdateDébut(in date:date) permet de définir la date de début de contrat
getdateFin(): date retourne la date de fin de contrat
setdateFin(in date:date) permet de définir la date de début de contrat
getproduits(): liste de str permet d'avoir accès à la liste des produits sous contrat
setproduits(in produits:liste de str) permet de définir une liste de produits

-Conditionneur
triemachine() permet de trier les déchets à la machine
triemanuel() permet d trier les déchets à la main
ballotage() permet de ballotter des déchets

-recycleur
transformerMatières() permet de transformer les déchets venant du conditionneur
créerProduit() permet de créer un produit à partir des déchets transformés

Le dossier contient :

Un README

Un jar exécutable permettant de lancer directement notre projet

Une archive jar possédant toutes les sources de notre projet

Les fichiers de données de notre projet sont directement dans notre main.java donc, **pour faire les tests** démontrant la fonctionnalité de nos méthodes, il faut aller dans le main.java, et suivre les instructions.
Par exemple :

```
// Enlever les /* */ Pour essayer les méthodes de DeposerDechet

/*
DeposerDechet dechetUser2 = new DeposerDechet();
dechetUser2.setDate(new Date());
dechetUser2.setQteDechet(10);
dechetUser2.setTypeDechets(TypeDechets.VERRE);
dechetUser2.setUtilisateur(user2);
dechetUser2.setPtsgagne(0);
dechetUser2.calculerQteDechets(dechetUser2.getQteDechet(),
dechetUser2.getTypeDechets());
dechetUser2.ajoutPtsGagné(user2,dechetUser2.getPtsgagne());
int i=user2.getPtsFidelite();
System.out.println("Le nombre de point de fidelité après ajout est :" + i);
*/
```

Il suffit d’enlever le /* au début et le */ à la fin du bloc qui suis l’explication, puis exécuter le main afin de pouvoir vérifier la fonctionnalité des méthodes.

```
DeposerDechet dechetUser2 = new DeposerDechet();
dechetUser2.setDate(new Date());
dechetUser2.setQteDechet(10);
dechetUser2.setTypeDechets(TypeDechets.VERRE);
dechetUser2.setUtilisateur(user2);
dechetUser2.setPtsgagne(0);
dechetUser2.calculerQteDechets(dechetUser2.getQteDechet(),
dechetUser2.getTypeDechets());
dechetUser2.ajoutPtsGagné(user2,dechetUser2.getPtsgagne());
int i=user2.getPtsFidelite();
System.out.println("Le nombre de point de fidelité après ajout est :" + i);
```

Le principe est le même pour toutes les vérificaitons qui suivent dans le main.

Pour l’organisation :
Saif : classes Commerce, Poubelle, TypeDechets et main
Sofiane : classes utilisateur et centre de tri
Hugo : classes déposerdéchet, contrat, produit et main
Logan : correction Uml et rapport
Titouan : Gestion + README