

Projet 3A DD :

Gestion d'un système de Tri sélectif



BECK Naïma – BENAMAR Yassine
BIETTE Antonin – LEPOUL Axelle

I. INTRODUCTION

II. DIAGRAMME UML ET ANALYSE

III. RETOUR SUR LE PROJET ET CONCLUSION



I. INTRODUCTION

Dans un contexte de crise environnementale majeure, le XXI^{ème} siècle est marqué par une volonté de limiter l'impact néfaste de l'espèce humaine sur son écosystème mais aussi sur lui-même. Entre autres, la question de la gestion des ressources est au centre de l'attention dans un monde où nombre d'entre elles sont de plus en plus limitées. Ainsi, des technologies nouvelles et innovantes sont mises en place pour contrer cet épuisement des ressources, la première d'entre elles étant la poubelle de tri sélectif.

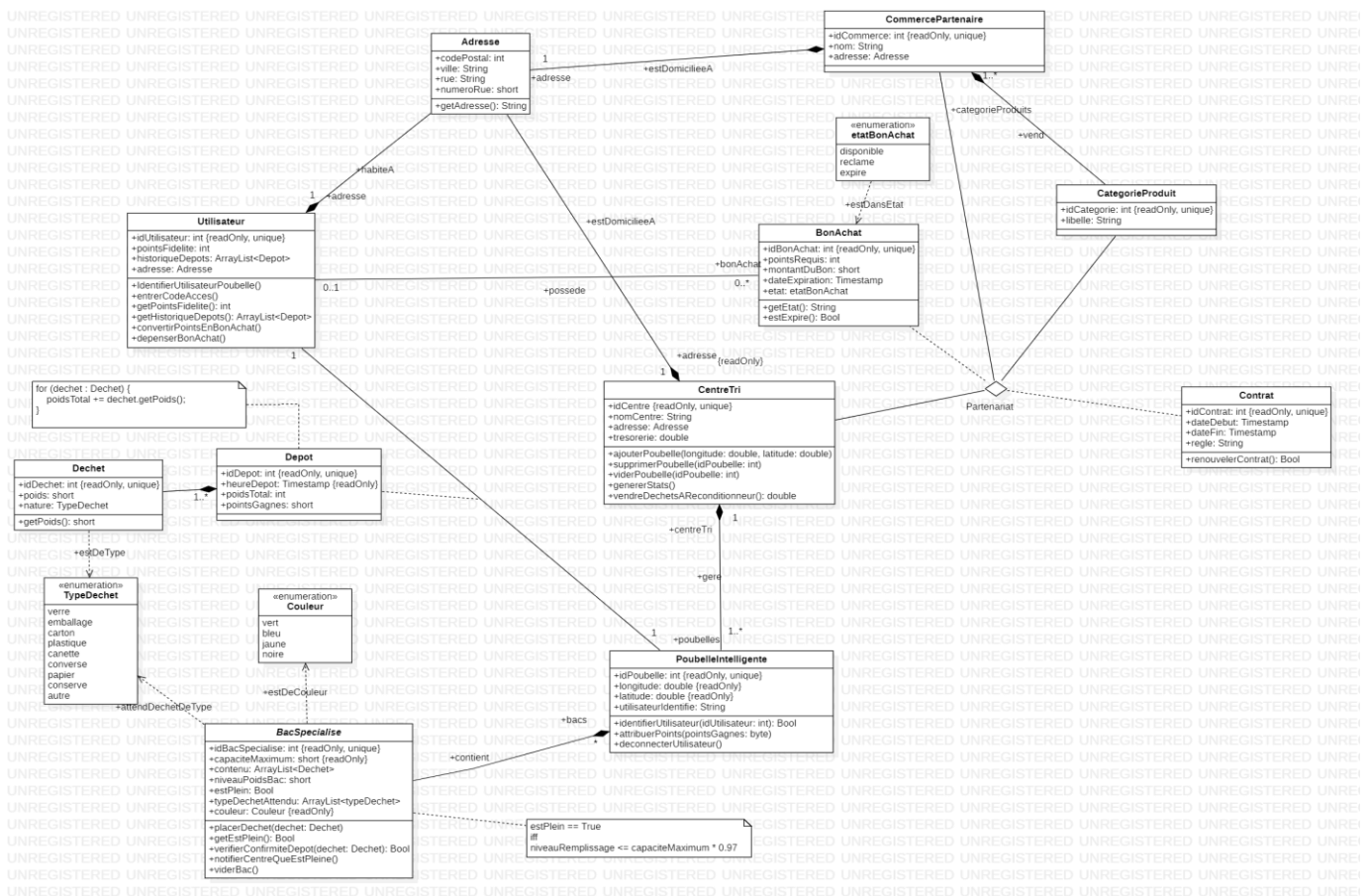
L'invention de la poubelle telle que nous la connaissons aujourd'hui est relativement ancienne puisqu'elle fut créée en 1884 par le juriste et diplomate français Eugene Poubelle. Cependant la réglementation et la démocratisation du tri n'arrive qu'un siècle plus tard en 1974 en réponse économique aux chocs pétroliers ayant eu lieu l'année précédente. Encore de nos jours la pratique du tri bien que connues de tous n'est systématique que chez 51% des français selon un sondage de Ipsos en 2019. Le sujet est donc loin d'être clos et la marge de progression encore vaste.

Lorsque nous nous intéressons au fonctionnement du réseau de tri sélectif en France il apparaît que trier ses déchets ne fait sens que si cette action est perpétuée aux différentes échelles de la gestion des déchets. Celle-ci est donc l'œuvre aussi bien d'individus au sein de leur domicile mais également de tout un réseau d'acteur comme les centres de tri ou les reconditionneurs.

Dans ce projet, c'est cette gestion de la fin de vie des déchets du quotidien que nous avons étudié. Notre travail a consisté en la représentation d'un système de tri à l'échelle d'une ville. C'est sous la forme d'un diagramme UML (Unified Modeling Language), construit grâce à l'application StarUML, que nous avons représenté les différents acteurs ainsi que les relations qui les unis.

II. DIAGRAMME UML ET ANALYSE:

Voici ci-dessous le diagramme réalisé au cours de ce projet :



Celui-ci comprend un total de 13 classes : Adresse, Utilisateur (représentant le compte dédié à un ménage), PoubelleIntelligente, BacSpecialise (représentant les trous et compartiment au sein d'une poubelle intelligente), Couleur (d'un bac de tri), Dechet, Type Dechet, Depot(représentant l'ensemble des déchets jeté par un utilisateur lors d'un déplacement vers une poubelle intelligente), CentreTri, CommercePartenaire, etatBonAchat, BonAchat, et Contrat.

Lors de la conception de ce diagramme UML, de nombreuses questions ont émergé au sein du groupe et bon nombre d'éléments ont été débattus avant d'être incorporés. Aussi, voici une liste des décisions prises ayant conduit au diagramme tel qu'il est présenté :

Classe « BacSpecialise » :

➤ Nous avons fait le choix de créer un attribut dérivé « estPlein : Bool » dépendant du niveau de remplissage (en termes de Poids) qui se calcule comme

0.97*la capacité maximum de la poubelle. Cet attribut permet de savoir quand le bac est presque plein et ainsi empêcher un utilisateur de continuer de vider ses déchets au risque de le surcharger.

➤ Ce sont les bacs de tri (et non la poubelle intelligente) qui notifie le centre de tri lorsqu'ils sont pleins. La différence est que le centre de tri devra venir vider les bacs chaque fois que l'un d'entre eux est plein (et non attendre qu'ils le soient tous car cela aurait pour conséquence de bloquer certains bacs qui se rempliraient plus rapidement que les autres).

➤ Le nombre de couleurs que peut prendre un bac de tri étant fini et puisque celles-ci n'apportent pas de description supplémentaire (ni attribut ni comportement) nous avons fait le choix de les représenter par une énumération.

Classe « Dechet » :

➤ Un raisonnement similaire à celui des couleurs des bacs a été appliqué pour le type des déchets jetés. Cela est possible car nous avons pris le parti que les déchets étaient triés individuellement et donc que leur poids n'avait pas besoin d'être dépendant du type du déchet (il n'y a pas de distinction entre 10kg de plastique et 10kg de verres puisque ceci sont jeter dans des bacs différents).

Classe « PoubelleIntelligente » :

➤ Dans la classe « Poubelle Intelligente » l'utilisateur a la possibilité de s'identifier (voir classe Utilisateur) mais la déconnexion est automatique pour éviter les problèmes d'utilisateurs partant sans se déconnecter. La poubelle est en mesure d'attribuer directement des points aux utilisateurs à partir de leur identifiant.

➤ C'est la poubelle qui a la charge de l'attribution des points de fidélité à travers l'opération « attribuerPoints(pointsGagnes):byte ».

Classe « Adresse » :

➤ Cette classe a pour utilité de ne pas répéter les attributs composant une adresse (code postal, ville, quartier, rue et numéro de rue) pour les différentes classes en nécessitant à savoir les utilisateurs, les centres de tri et les commerces partenaires.

Classe « Utilisateur » :

- Chaque ménage est volontairement représenté par un compte utilisateur car le ménage n'existe dans ce système qu'à travers son compte. De plus un compte est unique par ménage quel que soit le nombre de personnes y vivant (d'où l'absence de nom ou autre attribut qualificatif d'un individus unique)
- L'historique des dépôts d'un utilisateurs est stocké sous la forme d'une liste de Dépôts (voir classe « Depot »)
- Le nombre de point de fidélité d'un utilisateur est gardé en attribut, l'utilisateur à la possibilité de convertir un certain nombre de ses points en bon d'achats s'il en possède suffisamment (ceux-ci sont définis dans une classe à part entière, voir classe « BonAchat »).
- Un utilisateur doit entrer le code d'accès pour accéder à la poubelle via l'opération « entrerCodeAcces() », celui-ci est identique pour tous les utilisateurs et sert juste à vérifier que celui-ci possède l'application pas l'identifier.
- L'utilisateur à la possibilité de s'identifier mais n'en est pas contraint.

Classe « Depot » :

- La classe « Depot » est une classe d'association résultant de l'association entre « Utilisateur » et « PoubelleIntelligente », ainsi chaque fois qu'un utilisateur vide ses déchets dans une poubelle un objet de la classe dépôt est créé.
- Celui-ci stocke l'heure du dépôt le poids total du dépôt et le nombre de point de fidélité gagné.
- Ce dernier est un attribut dérivé qui se calcule par : $\text{poidsTotal}/1000$ auquel on soustrait $10 \times \text{poids}$ pour toutes les valeur False renvoyées par `verifierConformiteDechet(dechet :Dechet)`. Cette formule revient à attribuer 1 point de fidélité par kilogramme de déchet correctement triés et à en retirer de manière 10 fois plus sévères pour chaque déchet mal trié.
- Le poids d'un dépôt est calculé comme la somme du poids des déchets jetés au cours du dépôt.

Classe « CentreTri » :

- Nous avons fait le choix de suivre le budget du centre de Tri, ce suivi est représenté par l'attribut « tresorerie ». L'idée est de stocker l'argent que possède le centre de tri afin de rendre possible la création de statistique comptable

notamment le coup d'entretien des poubelles intelligente et la rentabilité des contrats avec les différents commerces.

- Ces statistiques sont obtenues à partir de l'opération « `genererStats()` » qui sera codée ultérieurement en Java.

- L'opération « `vendreDechetsAReconditionneur()` » marque la fin du processus de tri. Nous avons fait le choix de ne pas représenter de classe « reconditionneur » car la remise en circulation des produits recyclés n'est pas à traiter dans le projet.

Classe « `CommercePartenaire` » et « `CategorieProduit` » :

- Ne sont représenté dans cette classe que les commerces ayant passé un ou plusieurs contrats avec des centre de tri.

- Chaque commerce possède différentes catégories de produits. Les commerces et leurs catégories de produits sont reliés par une agrégation.

Classe « `BonAchat` » :

- La classe « `BonAchat` » est une classe d'association entre un commerce partenaire, une catégorie de produit de celui-ci et un centre de tri.

- Ses attributs sont : le nombre de point requis pour acheter un bon d'achat spécifique, la valeur de celui-ci en €, sa date d'expiration et son état.

- Ce dernier est défini sous la forme d'une énumération : {disponible, reclame, expire}. Ainsi, lorsqu'un bon d'achat est encore en possession du commerce son état est dit « disponible », lorsqu'il est en possession d'un utilisateur il est « reclame » et si celui-ci n'a pas été utilisé avant expiration il est logiquement « expire »

Classe « `Contrat` » :

- La classe « `Contrat` » est aussi une classe d'association autour de la même association n-aire que « `BonAchat` », il est défini par des règles, une date de début et une date de fin

- Un contrat a la possibilité d'être renouvelé lorsqu'il arrive à son terme.

III. RETOUR SUR LE PROJET ET CONCLUSION :

Lors de la réalisation de ce travail de groupe nous avons pu expérimenter la nature plurielle de la construction d'un diagramme UML. En effet, après les premières tentatives individuelles pour faire émerger une structure globale de ce à quoi ressemblerait notre diagramme nous avons observé que chacun avait une analyse et une interprétation du sujet et donc de sa retranscription en UML. Ainsi, la discussion a pu faire émerger ou disparaître des idées selon les problèmes auxquelles elles répondaient ou engendraient. Cette perfectibilité continue s'est prolongée même après la création d'un diagramme commun et validé par les membres du groupe car certains détails pouvaient sans cesse être améliorés.

En somme, ce travail servira de base pour la suite du projet, notre objectif sera alors d'implémenter dans un premier temps du code Java afin de tester le fonctionnement de nos classes et du système de tri en général, puis dans un second temps de développer des interfaces graphiques en IHM pour l'exécution de notre code.