Projet 3A DD :

Gestion d'un système de tri sélectif

Implémentation JAVA

BECK Naïma – BENAMAR Yassine

BIETTE Antonin – LEPOUL Axelle

PLAN DU RAPPORT

I. INTRODUCTION

II. CODE JAVA ET EXPLICATIONS

III. RETOUR SUR LE PROJET ET CONCLUSION

I. INTRODUCTION

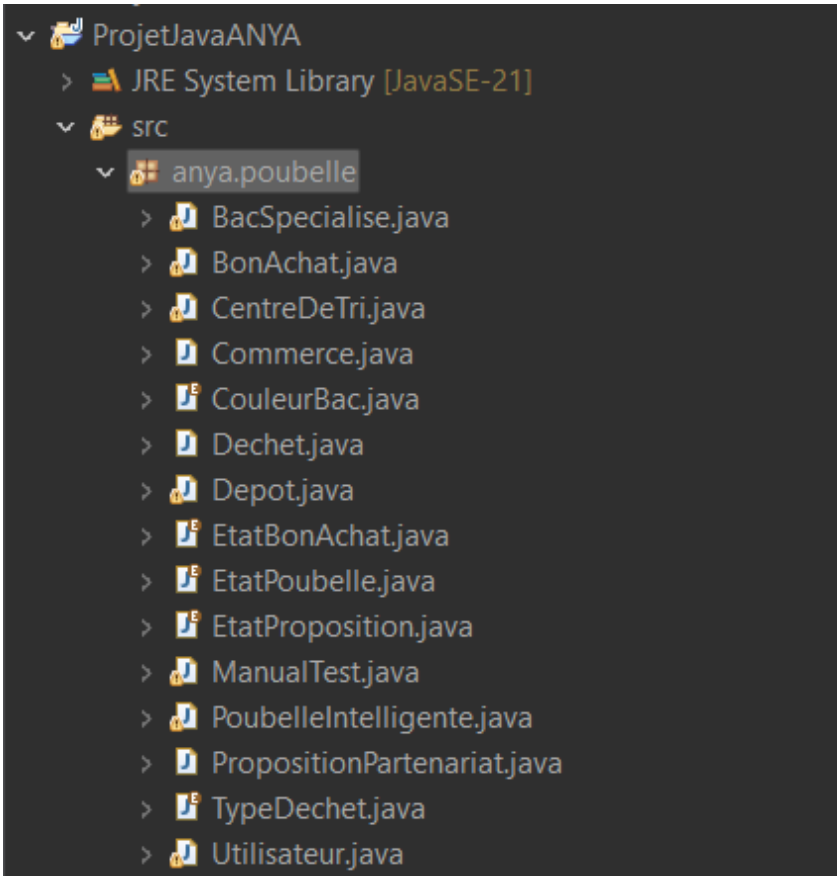
Lors de notre précédent travail nous avions créé un diagramme UML détaillant la structure d'un réseau d'acteurs autour de la gestion du recyclage à l'échelle d'une ville. Cette schématisation du réseau nous avait permis de faire émerger les relations entres les différents acteurs, notamment entre le centre de tri, les commerces, et les usagers. Cependant cette première étape servait avant tout en tant qu'outil pour la seconde étape de notre projet présentée dans ce rapport.

L'objectif de ce nouveau devoir fut de produire un code JAVA fonctionnelle (via l'utilisation de l'application eclipse IDE) permettant la bonne réalisation des comportements et fonctionnalités de chacune des classes décrites antérieurement dans l'UML. Pour ne pas disperses notre travail, nous n'avons implémenté que les classes se rattachant à la collecte des déchets, à la gestion des poubelles, et au système de bon d'achat. De fait, le recyclage et le reconditionnement des déchets n'ont pas été développées.

Dans la suite de ce rapport nous nous focaliserons principalement sur la structure de la classe « ManualTest » qui fait office de classe test et retrace le circuit reliant chacune des autres classes. Nous expliquerons le fonctionnement général des autres classes sans toutes les détaillées afin de ne pas saturer le rapport d'informations non-essentielles.

II. CODE JAVA ET EXPLICATIONS

Afin d'avoir une meilleure compréhension du code produit, voici la liste des 15 classes développées :



Comme expliqué précédemment, commençons par nous intéresser au contenu de la classe « ManualTest ». Chaque bloc de code qui la compose sera observé individuellement afin de faciliter la lecture du programme.

```java
package anya.poubelle;

import java.time.LocalDate;
import java.util.ArrayList;

public class ManualTest {
    public static void main(String[] args) {
        System.out.println("Début du test manuel.");

        // Création du centre de tri
        CentreDeTri centreDeTri = new CentreDeTri("Centre de tri de Saint-Germain-En-Laye");
        System.out.println("Centre de tri créé : " + centreDeTri.getNom());

        // Création de la poubelle intelligente
        PoubelleIntelligente poubelle = new PoubelleIntelligente(48.90058, 2.07033, centreDeTri);
        System.out.println("Poubelle intelligente créée à la position : (" + poubelle.toStringCoordonnees() + ")");
        centreDeTri.setPointsPourCouleur(CouleurBac.VERTE, 10.0);
```

La première étape du test consiste à appeler la classe « CentreDeTri », il est primordial de commencer par celle-ci car les autres classes interagissent avec elle. Vient ensuite la création d'une poubelle (instance de la classe « PoubelleIntelligente ») caractérisée par sa localisation et le centre de tri auquel elle est rattachée. La dernière ligne de code visible sur l'image ci-dessus instancie le nombre de points de fidélité au kilo (ici 10pts) pour les déchets placés dans une bac de couleur verte. Lors de la mise en pratique du code il est évidement nécessaire d'effectuer cette opération pour les 4 types de bacs.

```java
    // Création d'un commerce partenaire potentiel
    Commerce supermarche = new Commerce("Carrefour");
    System.out.println("Commerce créé : " + supermarche.getNom());
    try {
        centreDeTri.ajouterPartenairePotentiel(supermarche);
        System.out.println("Commerce ajouté comme partenaire potentiel.");
    } catch (Exception e) {
        System.out.println("Erreur lors de l'ajout du commerce comme partenaire potentiel : " + e.getMessage());
    }
```

La seconde étape est l'instantialisation d'un supermarché (issu de la classe « Commerce »). Puisque le code est développées avec comme référence le centre de tri, un commerce n'a d'utilité que si une proposition de contrat peut se faire avec lui. Cette action est ici représentée par la méthode « ajouterPartenairePotentiel » qui ajoute le commerce nouvellement crée à une ArrayList de Commerce avec lesquels le centre de tri n'a pas encore enclenché de contrat. A noter que la procédure de création de contrat n'est ici pas encore débutée.

```java
// Création d'un utilisateur
Utilisateur user = new Utilisateur("Yassine", centreDeTri);
System.out.println("Utilisateur créé : " + user.getNom());

// Création d'un déchet
Dechet dechet = new Dechet(0.5, TypeDechet.verre);
System.out.println("Déchet créé : " + dechet.getType() + " avec un poids de " + dechet.getPoid() + " kg");

// Connexion de l'utilisateur et déverrouillage de la poubelle
System.out.println("Avant le jet, le nombre de points de fidélités de l'utilisateur est " + user.getPointsFidelite() + " points.");

try {
    user.seConnecterEtDeverouiller(poubelle);
    System.out.println("Utilisateur connecté et poubelle déverrouillée.");
} catch (Exception e) {
    System.out.println("Erreur lors de la connexion ou du déverrouillage de la poubelle : " + e.getMessage());
}

// Placement du déchet dans la poubelle
boolean dechetPlacé = user.placerDechetDansPoubelle(dechet, poubelle, CouleurBac.VERTE);
if (dechetPlacé) {
    System.out.println("Jet de déchet a réussi.");
} else {
    System.out.println("Jet de déchet a échoué.");
}

System.out.println("Après le jet, le nombre de points de fidélités de l'utilisateur est " + user.getPointsFidelite() + " points.");
```

Dernier acteur encore inexistant dans le code, la classe « Utilisateur » (caractérisé par un nom et un centre de tri de référence) est instanciée sous la forme de "user". Ceux-ci ont la capacité de jeter des déchets qui sont caractérisés par un poids et un type (il en existe uniquement 7+1(autre) contenus dans l'énumération « TypeDechet »). Lorsqu'un "user" veut jeter ses déchets il doit au préalable s'identifier sur une poubelle (cela à pour effet de changer l'état de la poubelle passant de « verouillee » à « deverouillee »). Une précision faite dans la méthode correspondante est que l'état de la poubelle ne doit pas être « pleine » auquel cas l'utilisateur ne pourra pas s'identifier. S'il parvient à se connecter, il peut alors jeter ses déchets. Ce jet de déchets est alors enregistré comme une instance de la classe « Depot » (caractérisé par un id, un nombre de points rapporté, un utilisateur, un bac, une date, une quantité et un booléen représentant le succès du dépôt). L'historique des dépôts est alors mis à jour dans la classe « Utilisateur » ainsi que le nombre de points de fidélité de ce dernier (tout comme son score qui correspond au nombre total de points de fidélité accumulé sans prendre en compte les dépenses potentiels faites par l'utilisateur). A noter que les points de fidélités obtenues peuvent être négatifs si l'utilisateur s'est trompé de bac OU si le système de vérification du type de déchet s'est trompé et à considérer à tort que le déchet n'était pas dans le bon bac (ce risque d'erreur de la poubelle est définit selon une gaussienne de moyenne arbitrairement fixé à 0.85).

```java
// Proposer un partenariat
try {
    centreDeTri.proposerPartenariat(supermarche, 10, 5);
    System.out.println("Proposition de partenariat envoyée.");
} catch (Exception e) {
    System.out.println("Erreur lors de la proposition de partenariat : " + e.getMessage());
}

// Gestion des propositions de partenariat
try {
    ArrayList<PropositionPartenariat> propositions = supermarche.getPropositionsEnAttente(0, 10);
    System.out.println("Propositions reçues : " + propositions.size());
    if (!propositions.isEmpty()) {
        System.out.println("Acceptation de la première proposition.");
        supermarche.accepter(propositions.get(0), LocalDate.now().plusDays(5));
        if (propositions.size() > 1) {
            System.out.println("Refus de la deuxième proposition.");
            propositions.get(1).refuser();
        }
    }
} catch (Exception e) {
    System.out.println("Erreur lors de la gestion des propositions : " + e.getMessage());
}
```

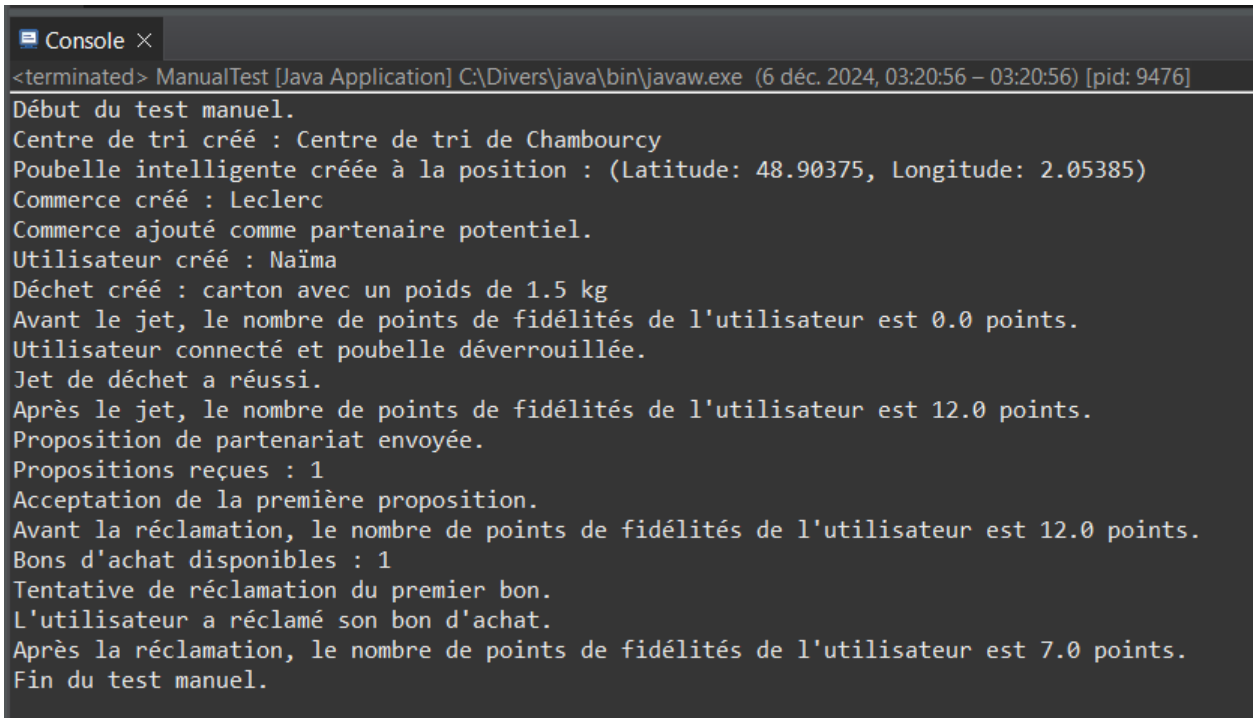
Dans cette nouvelle partie de notre classe test sont traités les échanges entre un commerce et un centre de tri. L'initiative revient au centre de tri qui est le seul à pouvoir proposer un partenariat (en précisant le commerce avec lequel il passe un contrat, le montant des bons d'achats qui seront crée et le nombre de points de fidélité nécessaire pour en acheter), la première version du contrat est alors envoyé dans une liste d'attente au sein de la classe « Commerce ». C'est ensuite à ce dernier d'accepter ou non la proposition. S'il décide d'accepter alors il renvoie le contrat au centre de tri en y ajoutant la date d'expiration. S'il refuse alors le commerce prévient le centre de tri de son refus. Dans les deux cas, la proposition de contrat est supprimer de la liste d'attente. En fonction de la réponse du commerce, le centre de tri ajoute le commerce à ses partenaires ou à ses refus. Les lignes de codes ci-dessus illustre chacun des deux cas.

```java
        // Consultation des bons d'achat disponibles
        System.out.println("Avant la réclamation, le nombre de points de fidélités de l'utilisateur est " + user.getPointsFidelite() + " points.");
        try {
            ArrayList<BonAchat> bons = user.consulterBonsAchatDisponibles(0, 10);
            System.out.println("Bons d'achat disponibles : " + bons.size());
            if (!bons.isEmpty()) {
                System.out.println("Tentative de réclamation du premier bon.");
                if (user.reclamerBonAchat(bons.get(0))) {
                    System.out.println("L'utilisateur a réclamé son bon d'achat.");
                } else {
                    System.out.println("Le bon d'achat n'a pas pu être réclamé.");
                }
            }
        } catch (Exception e) {
            System.out.println("Erreur lors de la consultation ou de la réclamation des bons d'achat : " + e.getMessage());
        }

        System.out.println("Après la réclamation, le nombre de points de fidélités de l'utilisateur est " + user.getPointsFidelite() + " points.");


        System.out.println("Fin du test manuel.");
    }
}
```

Pour finir, un utilisateur doit être en mesure de dépenser ses points de fidélités en bons d'achats. Pour cela il consulte tous les bons d'achats en possession de son centre de tri de référence. Si celui-ci en possède il peut alors tenter d'en réclamer un (identifié à partir de son indice dans la liste), à condition que l'utilisateur possède suffisamment de point de fidélité.

Lorsqu'on exécute la classe « ManualTest » voici deux exemples de résultat obtenu :

```
Console ×
<terminated> ManualTest [Java Application] C:\Divers\java\bin\javaw.exe  (6 déc. 2024, 03:09:36 – 03:09:36) [pid: 13880]
Début du test manuel.
Centre de tri créé : Centre de tri de Saint-Germain-En-Laye
Poubelle intelligente créée à la position : (Latitude: 48.90058, Longitude: 2.07033)
Commerce créé : Carrefour
Commerce ajouté comme partenaire potentiel.
Utilisateur créé : Yassine
Déchet créé : verre avec un poids de 0.5 kg
Avant le jet, le nombre de points de fidélités de l'utilisateur est 0.0 points.
Utilisateur connecté et poubelle déverrouillée.
Jet de déchet a réussi.
Après le jet, le nombre de points de fidélités de l'utilisateur est -2.5 points.
Proposition de partenariat envoyée.
Propositions reçues : 1
Acceptation de la première proposition.
Avant la réclamation, le nombre de points de fidélités de l'utilisateur est -2.5 points.
Bons d'achat disponibles : 1
Tentative de réclamation du premier bon.
Le bon d'achat n'a pas pu être réclamé.
Après la réclamation, le nombre de points de fidélités de l'utilisateur est -2.5 points.
Fin du test manuel.
```



Dans le premier résultat on observe que l'utilisateur « Yassine » a perdu des points (bien que non indiqué, il avait au préalable jeté dans la bonne poubelle) cela est dû à une mauvaise interprétation du type de déchet par la poubelle. De fait lorsque « Yassine » veut obtenir un bon d'achat en échange de ses points de fidélités la réclamation échoue car il n'a pas suffisamment de points (-2.5<5 qui est la valeur d'un bon d'achat tel que défini dans le test). Dans le deuxième cas, l'utilisateur « Naïma » a jeter son carton dans la bonne poubelle qui cette fois-ci à bien interprété le type de déchet. « Naïma a donc reçu ses points et pu réclamer son bon d'achats (entrainant par la même occasion la baisse de ses points).

III. RETOUR SUR LE PROJET ET CONCLUSION

Avant de parvenir au résultat tel qu'il est présenté dans ce rapport, s'est déroulé tout un travail de discussion et de choix sur les méthodes à implémenter en priorité. Certaines idées ont ainsi pu être modifiées où abandonnées pour garantir la bonne fonctionnalité du code.

D'autre part, il fut essentiel de produire en amont un visuel de ce à quoi pourrait ressembler notre application. Cela à permit entre autre de mieux comprendre les échanges logiques entre le centre de tri et les commerces.

Enfin, ce travail bien que d'ores et déjà satisfaisant, peut encore être poursuivi et amélioré. Pour cela nous poursuivrons la mise en pratique de notre code Java en développant des interfaces graphiques en IHM permettant de visualiser ce à quoi ressemblerait notre application et son fonctionnement aussi bien du point de vue du centre de tri que celui d'un commerce ou d'un utilisateur.