Rapport de Projet Rendu n°2 – Gestion intelligente du tri sélectif

Date de rendu : 24 mars 2024

Classe : GM Groupe 2

Formation : 1ère année Cycle Ingénieur – CY Tech

Matière : Programmation orientée objet en Java

Rendu : 2 (implémentation Java en mode texte)

Membres du groupe :
Mariam Kardallas, Yasmine Moutaouafiq, Shaïma Ouadah, Lucas Gournay, Florian Vo

Table des matières

Table des matières

1. Introduction

Ce rapport correspond au rendu 2 du projet de programmation orientée objet en Java. L'objectif était d'implémenter en mode texte un système intelligent de tri sélectif.

Nous avons dû créer toutes les classes nécessaires pour simuler un centre de tri, les ménages, les commerces partenaires et le système de points fidélité. Certaines fonctionnalités comme les conditionneurs et recycleurs n'étaient pas à développer dans ce rendu.

Chaque classe codée devait être accompagnée de tests unitaires, avec une classe `MainTest` qui lance tous les tests et vérifie que tout fonctionne correctement. Aucune interface utilisateur n'était demandée, seulement des jeux de données pour valider les comportements.

Le projet est structuré en plusieurs packages : un pour les classes principales, un pour les tests, et un pour le lancement global. Nous avons également préparé une connexion à une base de données MySQL via JDBC, afin d'assurer une bonne gestion des données.

Ce travail nous a permis de mettre en pratique nos connaissances en Java, et d'apprendre à organiser un projet complet en équipe.

2. Architecture orientée objet

L'architecture repose sur une modélisation orientée objet forte. Chaque entité du monde réel est représentée par une classe Java : centre de tri, bac, dépôt, ménage, commerce, contrat, etc. Les relations entre les classes sont définies de façon cohérente : un ménage effectue des dépôts dans un bac, le bac est géré par un centre, les points obtenus donnent accès à des bons utilisables dans des commerces partenaires.

Le `CentreDeTri` joue un rôle central : il coordonne les interactions, gère les données et offre des méthodes de consultation et d'analyse. La classe `Bac` vérifie les déchets déposés, les stocke temporairement, et transmet les informations aux autres entités. Les déchets sont modélisés via la classe `Depot`, qui contient toutes les informations : poids, type, date, ménage déposant, etc.

Les ménages sont représentés par la classe `Menage`, cumulant des points en fonction de leurs dépôts. Les `BonReduction` sont générés à partir de ces points et peuvent être utilisés dans des `Commerce`, eux-mêmes liés au centre par un `ContratPartenariat`.

Des énumérations (`Couleur`, `Type`, `ResCat`) viennent renforcer la rigueur du code en standardisant les types de données utilisés.

3. Fonctionnalités principales

Le système développé propose un ensemble de fonctionnalités destinées à simuler un environnement de tri sélectif complet, interactif et intelligent. Ces fonctionnalités concernent aussi bien les usagers que les gestionnaires du centre de tri.

Tout d'abord, les utilisateurs (ménages) peuvent effectuer des dépôts de déchets dans les bacs. Chaque dépôt est analysé automatiquement par le bac concerné, qui vérifie la correspondance entre le type de déchet et le type autorisé. Si le tri est conforme, des points de fidélité sont attribués au ménage. Ces points sont calculés en fonction du type de déchet et du poids déposé.

Les ménages peuvent consulter leur solde de points et les convertir en bons de réduction. Ces bons sont utilisables uniquement dans les commerces partenaires liés à un contrat actif avec le centre de tri. La génération d'un bon réduit automatiquement le nombre de points du ménage.

Le système offre également une visualisation des statistiques de tri : par type de déchet, par couleur de bac, par rue, ou encore sur une période donnée. Ces statistiques peuvent être utilisées pour analyser le comportement des usagers, identifier les zones les plus actives ou repérer des erreurs de tri fréquentes.

Un autre aspect essentiel est la gestion des bacs. Le centre peut être alerté lorsque certains bacs sont pleins grâce à un système de notification. Des fonctionnalités permettent ensuite de vider ou déplacer ces bacs. Cette flexibilité permet de simuler une gestion opérationnelle efficace.

Enfin, le système prend en charge la gestion des contrats de partenariat entre le centre de tri et les commerces. Chaque contrat a une durée de validité et peut être renouvelé. Le centre peut vérifier l'état des contrats et déterminer si un bon est encore valable chez un commerçant donné.

4. Choix techniques

— Langage : Java 17
— API : Java standard (`java.util`, `java.time`, etc.)
— Données : stockage en mémoire avec structures HashMap et ArrayList
— Base de données : MySQL, connectée via JDBC
— Identifiants uniques : `UUID` pour bacs, dépôts, bons
— Outils : VSCode, IntelliJ, GitHub

5. Organisation du travail

Le projet a été réalisé en groupe de cinq. Chacun a codé des classes de son côté selon la répartition décidée ensemble. Cela nous a permis de gagner du temps et de travailler en parallèle.

Une fois les classes terminées, Florian s'est chargé de faire le lien entre elles. Il a vérifié que tout s'assemblait bien et que les méthodes fonctionnaient correctement ensemble.

Lucas a réfléchi à une base de données SQL. Elle a été mise en place pour stocker les données de manière durable.

Yasmine et Shaïma ont rédigé le rapport.

6. Difficultés rencontrées

Au début du projet, on a eu du mal à organiser toutes les classes. Par exemple, on avait prévu une classe `Poubelle`, mais on a fini par tout faire directement avec la classe `Bac` pour simplifier. Il a donc fallu modifier plusieurs parties du code.

Un autre problème qu'on a rencontré, c'était la communication entre les classes. Parfois, les objets ne se transmettaient pas bien les infos, surtout entre `Depot`, `Menage` et `CentreDeTri`. On a dû revoir certains liens pour que tout fonctionne.

On a aussi hésité sur la façon d'identifier les objets. Finalement, on a utilisé des `UUID` pour qu'il n'y ait jamais de doublon. C'était un peu plus

compliqué au début, mais c'est plus sûr.

Enfin, on avait tous d'autres projets à gérer en parallèle. Il a donc fallu bien s'organiser, se répartir les tâches et se relancer quand il fallait avancer. Malgré tout ça, on a réussi à finir dans les temps.

7. Tests et validation

Pour s'assurer que le système fonctionne, on a créé des tests pour chaque classe importante. Ces tests vérifient par exemple si les points sont bien donnés, si les bons de réduction sont valides, ou si les bacs sont bien détectés comme pleins.

Une classe `MainTest` permet de lancer tous les tests d'un coup et de voir s'ils passent ou non. C'est pratique pour valider l'ensemble du projet.

On a aussi fait des tests manuels, comme simuler plusieurs dépôts ou générer un bon de réduction. Ces tests nous ont permis de corriger quelques erreurs de logique au fur et à mesure.

8. Chronologie du projet

— 10/03/2024 : Début du projet, UML initial
— 15/03/2024 : Ajustement de l'UML (classe `Poubelle` discutée)
— 17/03/2024 : Début de l'implémentation Java
— 22/03/2024 : Intégration des fonctionnalités principales
— 23/03/2024 : Finalisation du dépôt Git + README
— 24/03/2024 : Rendu 2

9. Annexe :

```
79          */
80
81          ct.inscrire("VO", "VOFlorian2005", new Adresse(11, "Avenue du Général Saussier", 10000, "Troyes"));
82          ct.inscrire("VO", "voflorian", new Adresse(9, "Rue Jean Moulin", 10000, "Troyes"));
83          ct.inscrire("GOURNAY", "LucasG", new Adresse(4, "Rue Lebon", 95000, "Cergy"));
84
85          System.out.println("\n\n\n----------------------------------------------------------------------------\n\n\n");
86          System.out.println(Menage.getMapMenage());
87
88          Menage m1 = bacArray.get(0).identifierMenage("VO", "voflorian2005");
```

Problems @ Javadoc Declaration Search Console × Terminal

<terminated> Main (6) [Java Application] /opt/eclipse-java/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.10.v20240120-1143/jre/bin/j

```
Identifiant déjà existant


----------------------------------------------------------------------------


{VO=Menage {
        Nom compte : VO
        Mot de passe : VOFlorian2005
        Points fidélité : 0
        Adresse ménage : Adresse {
                Numero : 11
                Nom rue : Avenue du Général Saussier
                Code postal : 10000
                Ville : Troyes
        }

}
, GOURNAY=Menage {
        Nom compte : GOURNAY
        Mot de passe : LucasG
        Points fidélité : 0
        Adresse ménage : Adresse {
                Numero : 4
                Nom rue : Rue Lebon
                Code postal : 95000
                Ville : Cergy
        }

}
}
Mot de passe incorrect
```

Tentative repoussée de créer deux ménages avec le même identifiant, affichage de ménages, connexion manquée si mot de passe incorrect

```
150
151         System.out.println("\n\n\n----------------------------------------------------------------------------------\n\n\n");
152         System.out.println(Menage.getMapMenage());
153
154         m2.echangerPts(m2.getPoints(), cm2, "Viande");
155         m3.echangerPts(m3.getPoints(), cm1, "Ordinateur");
156
157         ArrayList<BonReduction> bonArray = new ArrayList<BonReduction>(m2.getMapBons().values());
158
159         m2.utiliserBon(bonArray.get(0));
160
161         System.out.println("\n\n\n----------------------------------------------------------------------------------\n\n\n");
162         System.out.println(m2.getMapBons());
163         System.out.println(m3.getMapBons());
164
165         System.out.println("\n\n\n----------------------------------------------------------------------------------\n\n\n");
```

Problems @ Javadoc Declaration Search Console × Terminal

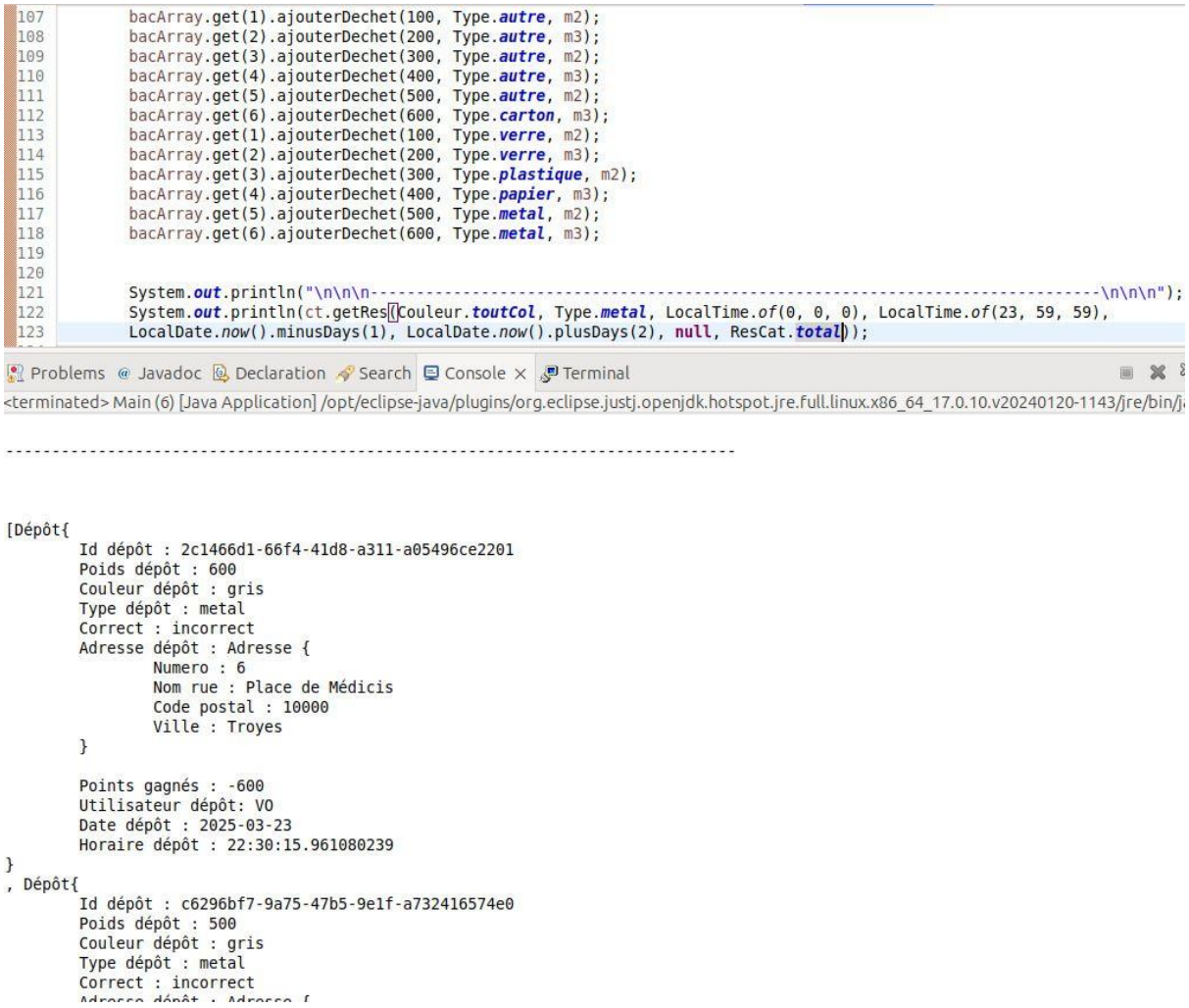
<terminated> Main (6) [Java Application] /opt/eclipse-java/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.10.v20240120-1143/jre/bin/j

```
--------------------------------------------------------------------------------


{de8cee80-b1a7-42e7-a244-522dae4ede3d=BonReduction {
        Id bon : de8cee80-b1a7-42e7-a244-522dae4ede3d
        Valeur : 6.89
        Bon utilisé : true
        Commerce bon : 0c309311-f99e-45c0-b205-929065fe4f87
        Ménage bon : GOURNAY
        Date expiration : 2025-04-23
}
}
{936564d8-e641-4c3e-985f-e661ae131653=BonReduction {
        Id bon : 936564d8-e641-4c3e-985f-e661ae131653
        Valeur : 175.0
        Bon utilisé : false
        Commerce bon : 73845bb4-1362-4457-9eb5-f77fcc0d120b
        Ménage bon : VO
        Date expiration : 2025-04-23
}
}


--------------------------------------------------------------------------------
```

Ce test vérifie que deux bons de réduction sont correctement enregistrés : l'un, d'une valeur de 6,89 €, a été utilisé par le ménage GOURNAY, tandis que l'autre, d'une valeur de 175 €, associé au ménage VO, est encore disponible.



```
107         bacArray.get(1).ajouterDechet(100, Type.autre, m2);
108         bacArray.get(2).ajouterDechet(200, Type.autre, m3);
109         bacArray.get(3).ajouterDechet(300, Type.autre, m2);
110         bacArray.get(4).ajouterDechet(400, Type.autre, m3);
111         bacArray.get(5).ajouterDechet(500, Type.autre, m2);
112         bacArray.get(6).ajouterDechet(600, Type.carton, m3);
113         bacArray.get(1).ajouterDechet(100, Type.verre, m2);
114         bacArray.get(2).ajouterDechet(200, Type.verre, m3);
115         bacArray.get(3).ajouterDechet(300, Type.plastique, m2);
116         bacArray.get(4).ajouterDechet(400, Type.papier, m3);
117         bacArray.get(5).ajouterDechet(500, Type.metal, m2);
118         bacArray.get(6).ajouterDechet(600, Type.metal, m3);
119
120
121         System.out.println("\n\n\n----------\n\n\n");
122         System.out.println(ct.getRes(Couleur.toutCol, Type.metal, LocalTime.of(0, 0, 0), LocalTime.of(23, 59, 59),
123         LocalDate.now().minusDays(1), LocalDate.now().plusDays(2), null, ResCat.total));
```

Problems | Javadoc | Declaration | Search | Console | Terminal

<terminated> Main (6) [Java Application] /opt/eclipse-java/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.10.v20240120-1143/jre/bin/j

```
----------


[Dépôt{
        Id dépôt : 2c1466d1-66f4-41d8-a311-a05496ce2201
        Poids dépôt : 600
        Couleur dépôt : gris
        Type dépôt : metal
        Correct : incorrect
        Adresse dépôt : Adresse {
                Numero : 6
                Nom rue : Place de Médicis
                Code postal : 10000
                Ville : Troyes
        }

        Points gagnés : -600
        Utilisateur dépôt: VO
        Date dépôt : 2025-03-23
        Horaire dépôt : 22:30:15.961080239
}
, Dépôt{
        Id dépôt : c6296bf7-9a75-47b5-9e1f-a732416574e0
        Poids dépôt : 500
        Couleur dépôt : gris
        Type dépôt : metal
        Correct : incorrect
        Adresse dépôt : Adresse {
```

Ce test, test la recherche de dépôt il affiche deux dépôts de déchets de type métal : le premier, effectué par l'utilisateur VO, est incorrect, pèse 600 g, et a entraîné une perte de 600 points ; le second, de 500 g, est également incorrect, mais l'utilisateur n'est pas précisé. Les deux dépôts ont été enregistrés avec leur adresse complète et leur horaire.