Rapport Projet JAVA: Gestion de tri sélectif

Laura MARLIER, Antonin GRIFFON,
Romain PIGNOL, Elya LARCHER

23 avril 2023

Table des matières

1 Le sujet

Le sujet de était de créer une application de tri sélectif mettant en oeuvre plusieurs acteurs : en particulier des consommateurs (les ménages) et des centres de tri. Il était question de représenter la gestion des déchets, de leur dépôt dans des poubelles par des ménages, à leur collecte, puis leur transport, et enfin les étapes de tri.
Nous rappelons le principe du tri sélectif :

- poubelle verte : contient les produits en verre,
- poubelle jaune : contient les produits en carton, métal et plastique,
- poubelle bleue : contient le papier,
- poubelle classique : contient tous les autres déchets ne pouvant pas aller dans les poubelles citées juste avant

Ce principe de tri sélectif est encouragé par la possibilité pour les ménages de gagner des points de fidélité auprès de commerces partenaires des centres de tri s'ils appliquent correctement la méthode de tri et placent leurs déchets dans les poubelles appropriées.

2 Le projet

2.1 Le diagramme UML

Nous avons représenté le sujet par le diagramme UML suivant :

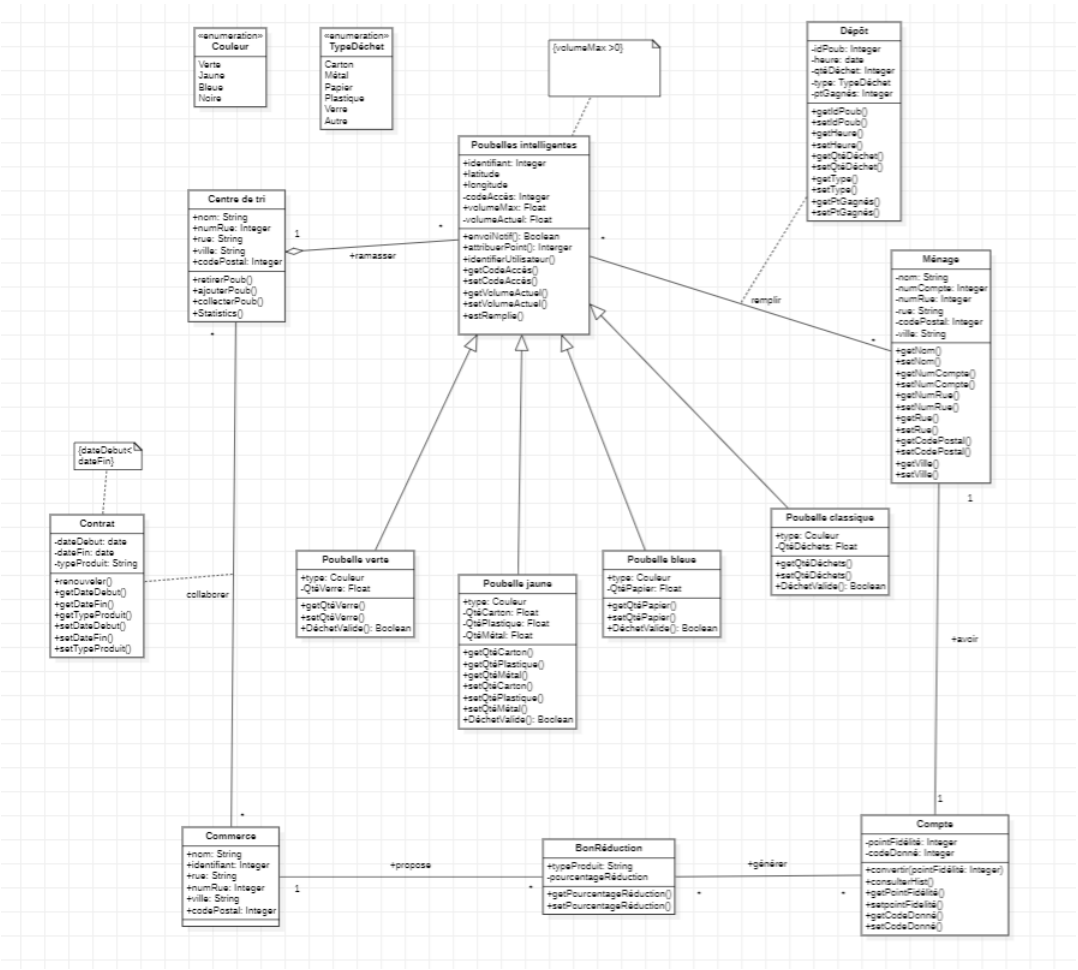

FIGURE 1 – Diagramme UML fait sur StarUML

2.2 La partie terminal

2.2.1 La classe Dechet

La classe Dechet est décrite par 2 attributs, son volume de type float, et un type de type TypeDechet, que nous détaillerons juste après.

```
public class Dechet {
    public TypeDechet type;
    public float volume;

     public Dechet(TypeDechet type, float volume){
        this.type = type;
        this.volume = volume
    }
}
```

2.2.2 L'énumération TypeDechet

L'énumération TypeDechet liste les 6 différents déchets possibles dans cette application. Elle permet à un déchet d'avoir un type spécifique.

```
public enum TypeDechet {
    Carton,
    Metal,
    Papier,
    Plastique,
    Verre,
    Autre,
}
```

2.2.3 La classe PoubelleIntelligente

La classe PoubelleIntelligente a 3 attributs, un identifiant entier, un volume max de type float et une liste de déchets. Elle a comme méthodes le fait d'ajouter des déchets dans une poubelle, de vider la poubelle et de lister les différents déchets contenus dans la poubelle. Elle a 2 "getter", un pour l'identifiant, et un pour le volume actuel de déchets.

```
public class PoubelleIntelligente {
    public Integer identifiant;
    public float volumeMax;
    public List<Dechet> listDechets;

```

```
6       public boolean ajouteDechet(Dechet dechet) {
7           if (volumeMax < getVolume() + dechet.volume){
8               System.out.println("La poubelle est trop
                    remplie");
9               return false;
10          }
11          else {
12              System.out.println("Dechet ajoute : " +
                    dechet.volume + " du type " +
                    dechet.type);
13              this.listDechets.add(dechet);
14              return true;
15          }
16      }
17
18      public void viderPoubelle() {
19          this.listDechets.clear();
20      }
21
22      public Integer getIdentifiant() {
23          return identifiant;
24      }
25
26      public float getVolume() {
27          float accumulateur = 0;
28          for (int i = 0; i < this.listDechets.size();
                i++){
29              accumulateur += listDechets.get(i).volume;
30          }
31
32          return accumulateur;
33      }
34
35      public void listerDechets() {
36          System.out.println("Affichage des dechets dans
                la poubelle ");
37          for (int i = 0; i < this.listDechets.size();
                i++) {
38              System.out.println(listDechets.get(i).volume
                    + ", " + listDechets.get(i).type);
39          }
40      }
41  }
```

2.2.4 Les classes PoubelleVerte, PoubelleBleue, PoubelleJaune, PoubelleClassique

Ces 4 classes héritent de la classe PoubelleIntelligente. Elles ont 3 attributs, une couleur de type string, une latitude et une longitude, de type integer, qui permettent de déterminer la position de la poubelle.

```
1  public class PoubelleVerte/Bleue/Jaune/Classique extends
       PoubelleIntelligente {
2      public String couleur;
3      public Integer latitude;
4      public Integer longitude;
5
6      public PoubelleVerte/Bleue/Jaune/Classique(Integer
           identifiant, float volume) {
7          this.identifiant = identifiant;
8          this.couleur =
               "verte"/"bleue/"jaune"/"classique";
9          this.volumeMax = volume;
10         this.listDechets = new ArrayList<>();
11     }
12
13     public void setPlace(Integer latitude1, Integer
           longitude1) {
14         latitude = latitude1;
15         longitude = longitude1;
16     }
17
18     public String getCouleur() {
19         return couleur;
20     }
```

2.2.5 La classe CentreTri

La classe CentreTri a 3 attributs, un nom de type string, une liste de déchets non triés et une liste de liste de déchets, ce qui forme une matrice, avec pour chaque déchet, sa quantité.

```
1  public class CentreTri {
2      public String nom;
3      public List<Dechet> dechetnontrie;
4      public List<List<Dechet>> listdechets;
5
6      public CentreTri(String nom) {
```

```
7           this.nom = nom;
8           listdechets = new ArrayList<>();
9           for (int i = 0; i < 6; i++){
10              listdechets.add(new ArrayList<>());
11          }
12          dechetnontrie = new ArrayList<>();
13      }
14
15
16      public void ajouterDechet(PoubelleIntelligente
            poubelle) {
17          dechetnontrie.addAll(poubelle.listDechets);
18      }
19
20
21      public void afficherDechetsNonTries() {
22          for (Dechet dechet : dechetnontrie) {
23              System.out.print(dechet.type + ", ");
24          }
25          System.out.println("\n");
26      }
27
28
29      public void afficherDechetsTries() {
30          TypeDechet[] type = TypeDechet.values();
31          for (int i = 0; i < type.length; i++){
32              System.out.print("Volume des elements tries
                    du type " + type[i] + " : ");
33              for (int j = 0; j <
                    listdechets.get(i).size(); j++){
34                  System.out.print(listdechets.get(i).get(j).volume
                        + ", ");
35              }
36              System.out.println("\n");
37          }
38      }
39
40      public void trier() {
41          TypeDechet[] type = TypeDechet.values();
42          while (dechetnontrie.size() != 0) {
43              int trie = 0;
44              Dechet dechet = dechetnontrie.get(0);
45
46              for (int i = 0; i < type.length; i++) {
47                  if (dechet.type == type[i]){
```

```
48                      listdechets.get(i).add(dechet);
49                      trie = 1;
50                  }
51              }
52              if (trie == 1){
53                  System.out.println("Element trie : " +
                        dechet.type);
54              }
55              else {
56                  System.out.println("Impossible de trier
                        le dechet : "+ dechet.type);
57              }
58              dechetnontrie.remove(dechet);
59          }
60      }
61  }
```

2.2.6 La classe Ménage

La classe Ménage contient 4 attributs, son nom et son adresse de type string, son nombre de points de fidélité de type integer, et un historique des déchets qu'il a déposé. Un ménage peut déposer des déchets dans une poubelle, acheter à un commerce partenaire des centres avec des points de fidélité obtenus en déposant correctement ses déchets, afficher ses points de fidélité et l'historique de ses dépôts.

```
1   public class Menage {
2
3       public String nom;
4       public String adresse;
5       public int pointsFidelite;
6       public ArrayList<Dechet> historique;
7
8       public Menage(String nom) {
9           this.nom = nom;
10          pointsFidelite = 0;
11          historique = new ArrayList<>();
12      }
13
14      public void deposerDechet(Dechet dechet,
            PoubelleIntelligente poubelle) {
15          if (poubelle.ajouteDechet(dechet)){
16              pointsFidelite += dechet.volume;
17              historique.add(dechet);
```

```
18          }
19      }
20
21      public void acheter(Commerce commerce, int prix)
            throws IOException {
22          int newPrix;
23          if ((prix - pointsFidelite)>0){
24              newPrix = prix - pointsFidelite;
25          }
26          else{
27              newPrix = prix;
28          }
29          pointsFidelite -= (prix - newPrix);
30          if (pointsFidelite < 0){
31              pointsFidelite = 0;
32          }
33      }
34
35      public void afficherPoints() {
36          System.out.println("Vous possedez " +
                pointsFidelite + " points de fidelite");
37      }
38
39      public void afficherHistorique() {
40          System.out.println("Historique des depots");
41          for (Dechet dechet : historique) {
42              System.out.println(dechet.type + " : " +
                    dechet.volume);
43          }
44      }
45
46  }
```

2.2.7 La classe Commerce

La classe Commerce est définie par 2 attributs, son nom et une liste de produits.

```
1   public class Commerce {
2       public String name;
3       public List<String> listeproduits;
4
5       public Commerce(String name) {
6           this.name = name;
```

```
    }
}
```

2.2.8 La classe Contrat

La classe Contrat relie un commerce et un centre de tri. Elle a 2 attributs de type Date, une date de début de contrat et une date de fin de contrat.

```
public class Contrat {
    public Date datedebut;
    public Date datefin;

    public Contrat(Date datedebut, Date datefin,
        Commerce commerce, CentreTri centreTri)
}
```

2.2.9 Le MainTest

Le fichier MainTest crée un ménage, des déchets, une poubelle, un centre de tri et un commerce. Un ménage dépose des déchets dans sa poubelle, jusqu'à ce qu'elle soit pleine, puis les déchets sont affichés, l'historique des dépôts est affiché, et les poubelles sont vidées au centre de tri. Les déchets du centre de tri sont non triés, ils sont affichés puis triés. Le ménage affiche ses points de fidélité et les utilise. Les principales fonctionnalités de l'application sont testées.

2.3 L'interface graphique

L'interface graphique n'affiche qu'une fenêtre ayant en fond l'image du sujet du projet ainsi que 2 boutons sans autres fonctionnalités.

```
public class InterfaceGraphique extends Application {

    @Override
    public void start(Stage primaryStage) {

        Image backgroundImage = new
            Image("H:\\Desktop\\Cours\\3eme annee - ING1
            GM\\Semestre 2 - GMI\\Maths-Info\\Analyse et
            programmation orientee
            objet\\Projet\\poubelle.png");

        HBox panneauBoutons = new HBox();
```

```
9           Button bouton1 = new Button("Bouton 1");
10
11          /* Tentative de faire en sorte que le bouton
                execute une commande de la partie terminale
                de l'application, non fonctionnel
12          bouton1.setOnAction(new
                EventHandler<ActionEvent>() {
13              @Override
14              public void handle(ActionEvent actionEvent) {
15                  CentreTri.afficherDechetsNonTries();
16              }
17          }); */
18
19          Button bouton2 = new Button("Bouton 2");
20          panneauBoutons.getChildren().addAll(bouton1,
                bouton2);
21
22          ImageView backgroundImageView = new
                ImageView(backgroundImage);
23
24          StackPane root = new StackPane();
25          root.getChildren().addAll(backgroundImageView,
                panneauBoutons);
26
27          StackPane.setAlignment(backgroundImageView,
                Pos.CENTER);
28
29          Scene scene = new Scene(root, 800, 600);
30
31          primaryStage.setTitle("Interface graphique : Tri
                selectif");
32          primaryStage.setScene(scene);
33          primaryStage.show();
34      }
35
36      public static void main(String[] args) {
37          launch(args);
38      }
39  }
```

3 Conclusion

Ce projet a été assez compliqué car la programmation orientée objet est une notion totalement nouvelle que nous n'avions jamais manipulé auparavant. De plus, l'interface graphique est une partie que nous avons dû mettre de côté pour se concentrer sur la partie terminale de l'application, pour qu'elle soit le plus fonctionnel possible.