La Rochelle Université

La Rochelle Université
Licence Informatique
UE Génie logiciel 2

TD 1 - Diagrammes de classes et de séquences

Version enseignant

1 Interprétation d'un diagramme de classes

Interprétez le diagramme de classes de la figure 1 par un ensemble de phrases en langage naturel. Par exemple : « Les factures sont numérotées et datées. ». N'oubliez pas d'interpréter les éléments liés aux associations entre classes.

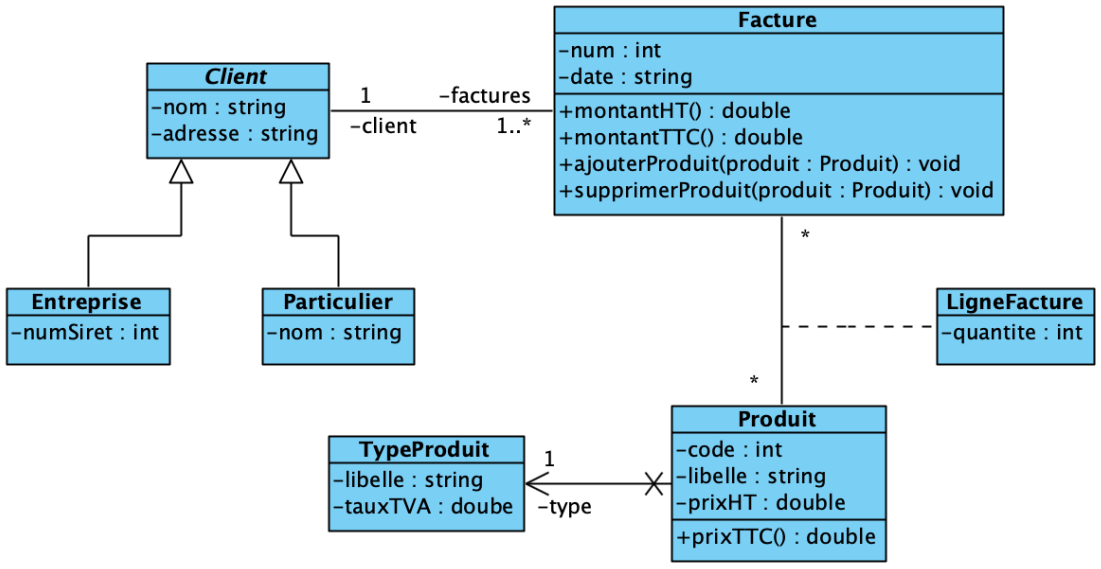

FIGURE 1 – Diagramme de classes de l'exercice 1

- les factures sont numérotées et datées ;
- chaque facture fait apparaître un certain nombre de produits. Une quantité est associée à chacun des produits d'une facture ;
- un produit est désigné par un code, un libellé permettant de l'identifier de manière plus claire et un prix HT ;
- le montant TTC d'un produit et d'une facture est calculé ;
- la TVA applicable à chaque produit dépend de son type. Pour chaque type de produit, on connaît le libellé et le taux de TVA associé ;
- les factures ne concernent qu'un seul client ;
- les clients peuvent être des entreprises ou des particuliers ;
- on peut ajouter et supprimer des produits d'une facture.

2 Un petit café

Donnez le diagramme de séquences du code suivant suite à l'appel du *main*.

```
interface ICoffee {
   public void serveCoffee(CoffeeContext context);
}

public class Coffee implements ICoffee {
   private final String flavor;

   public Coffee(String newFlavor) {
      this.flavor = newFlavor;
      System.out.println("Coffee is created! - " + flavor);
   }

   public String getFlavor() {
      return this.flavor;
   }

   public void serveCoffee(CoffeeContext context) {
       System.out.println("Serving " + flavor + " to table " + context.getTable());
   }
}

public class CoffeeContext {
   private final int tableNumber;

   public CoffeeContext(int tableNumber) {
      this.tableNumber = tableNumber;
   }

   public int getTable() {
      return this.tableNumber;
   }
}

public class CoffeeFactory {
   private HashMap<String, Coffee> flavors = new HashMap<String, Coffee>();

   public Coffee getCoffeeFlavor(String flavorName) {
      Coffee flavor = flavors.get(flavorName);
      if (flavor == null) {
      flavor = new Coffee(flavorName);
         flavors.put(flavorName, flavor);
      }
      return flavor;
   }

   public int getTotalCoffeeFlavorsMade() {
      return flavors.size();
   }
}

public class Waitress {
   //coffee array
   private static Coffee[] coffees = new Coffee[20];
```

```
54      //table array
55      private static CoffeeContext[] tables = new CoffeeContext[20];
56      private static int ordersCount = 0;
57      private static CoffeeFactory coffeeFactory;
58
59      public static void takeOrder(String fl, int table)
60      {
61         coffees[ordersCount] = coffeeFactory.getCoffeeFlavor(fl);
62         tables[ordersCount] = new CoffeeContext(table);
63         ordersCount++;
64      }
65
66      public static void main(String[] args)
67      {
68         coffeeFactory = new CoffeeFactory();
69         takeOrder("Cappuccino", 2);
70
71         for (int i = 0; i < ordersCount; ++i) {
72            coffees[i].serveCoffee(tables[i]);
73         }
74
75         System.out.println("\nTotal Coffee objects made: " +  coffeeFactory.getTotalCoffeeFlavorsMade());
76      }
77   }
78
```

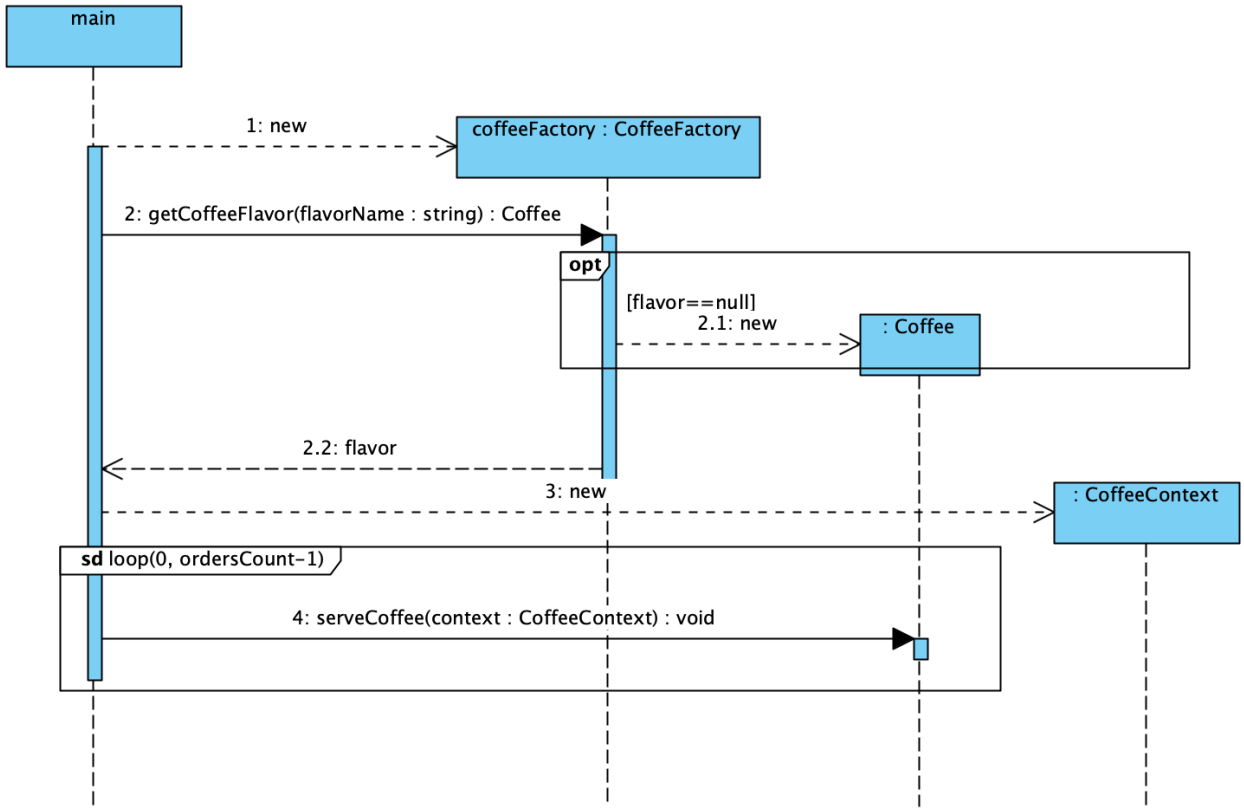

FIGURE 2 – Solution du diagramme de séquences de l'exercice 2

3 Magasin

Un magasin doit gérer un stock de produits de différentes catégories : des articles de jardinage et des articles de bricolage. Ces articles ont des propriétés communes à savoir :

— le nom ;
— l'identifiant ;
— le prix ;
— la quantité disponible en magasin.

Selon la catégorie, des informations supplémentaires sont enregistrées :

— article de jardinage : une saison d'utilisation ;
— articles de bricolage : un rayon (quincaillerie, plomberie...) et une information indiquant s'il s'agit d'un appareil électrique.

Les informations sur le stock sont enregistrées dans un fichier texte catalogue.txt dont voici un exemple :

```
    Jardinage;Rat345;50;Rateau;12;Ete
Jardinage;Rat456;150;Rateau;5;Ete
Bricolage;Clou12;2;Boite clous;50;Quiquaillerie;Nonelectrique
```

On fait l'hypothèse que le fichier catalogue.txt existe (vous le créerez manuellement) et l'on souhaite mettre en place le système capable de charger en mémoire le catalogue par lecture du fichier en utilisant le design pattern *Factory*.

Ici, la fabrique va créer les instances des produits à partir de chacune des lignes du fichier. Pour vous guider dans la conception, la classe *Client* vous est donnée ci-dessous.

Proposez un diagramme de classe mettant en oeuvre la fabrique en vous inspirant de l'exemple vu en cours.

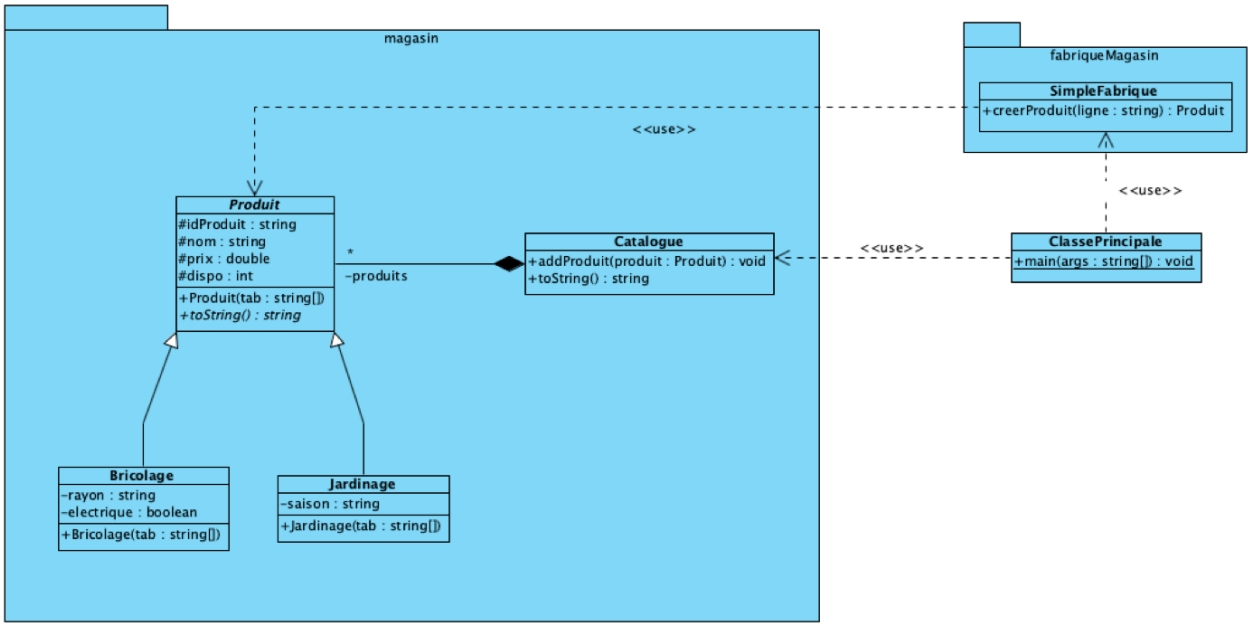

FIGURE 3 – Solution du diagramme de classes de l'exercice 3

Annexe

```
import java.io.BufferedReader;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ClassePrincipale {
   public static void main(String[] args) {
      SimpleFabrique simpleFabrique = new SimpleFabrique();
      Catalogue catalogue = new Catalogue();
      Path path = Paths.get("catalogue.txt");
      String ligne;
      Logger log = Logger.getLogger("log");

      try(BufferedReader lecteurAvecBuffer = Files.newBufferedReader(path, StandardCharsets.UTF_8))
      {
            while ((ligne = lecteurAvecBuffer.readLine()) != null)
                  catalogue.addProd(simpleFabrique.creerProduit(ligne));
      }
      catch (Exception exc)
      {
            log.log(Level.SEVERE, "Erreur à la lecture du fichier de stock {0}: " , exc.getMessage());
      }
      log.log(Level.INFO, "Le catalogue contient : \n {0}", catalogue);
   }
}
```