La Rochelle Université

La Rochelle Université
Licence Informatique
UE Génie logiciel 2

Design pattern Decorator

Sujet d'Armelle Prigent

Version enseignant

1 Objectif de la séance

L'objectif de cet exercice est de vous faire découvrir par vous même le design pattern *Decorator* que vous implémenterez lors du TP2. La partie théorique sera présentée dans un second temps.

2 Vente de dessert

2.1 Spécification

Le cahier des charges est le suivant :

— le système développé doit permettre d'afficher dans la console le nom complet du dessert choisi et son prix ;

— les clients ont le choix entre deux desserts : des crêpes ou des gaufres ;

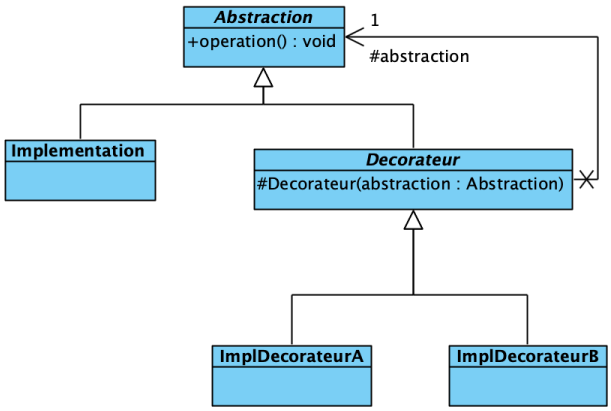
— sur chaque dessert, les clients peuvent ajouter un nombre quelconque d'ingrédients afin de faire leurs propres assortiments ;

— il existe 2 ingrédients (le chocolat (du Nutella® of course sinon ça ne vaut pas le coup) et la chantilly) mais il faut garder à l'esprit que l'ajout de nouveaux ingrédients doit être simplifié ;

— la tarification est simple : une crêpe nature coûte 1.50€ et une gaufre 1.80€. L'ajout de chocolat est facturé 0.50€ et de chantilly 0.40€.

2.2 Travail demandé

En vous inspirant de la structure du design pattern Decorator, proposez un diagramme de classe permettant de répondre au cahier des charges en intégrant ce design pattern.



Vous trouverez en figure 1 le diagramme de classe de cet exercice ainsi que son code Java. Donnez le diagramme de séquences du main.

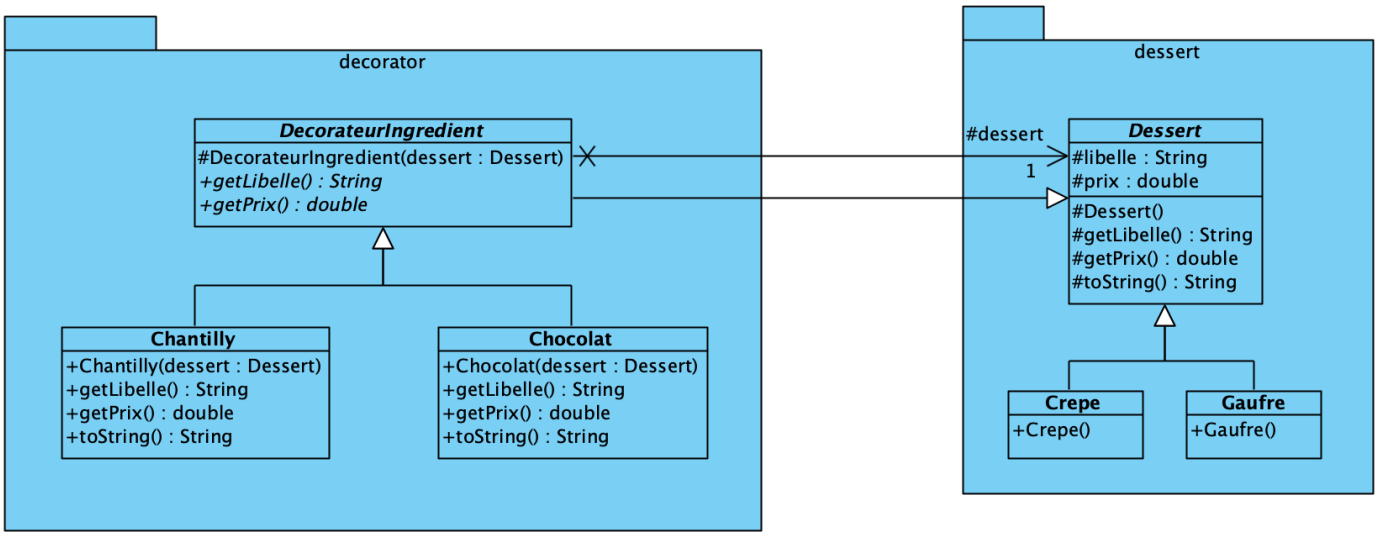


FIGURE 1 – Solution du design pattern Decorator sur l'exercice Dessert

```
1  public abstract class Dessert {
2     protected String libelle;
3     protected double prix;
4
5     protected Dessert() {
6        this.libelle = "";
7        this.prix = 0.0;
8     }
9
10    @Override
11    public String toString() {
12       return this.libelle + ", prix : " + prix + "€";
13    }
14
15    public String getLibelle() {
16       return this.libelle;
17    }
18
19    public double getPrix() {
20       return this.prix;
21    }
22 }
23
24 public class Crepe extends Dessert {
25    public Crepe() {
26       this.libelle = "crepe";
27       this.prix = 1.5;
28    }
29 }
30
31 public class Gaufre extends Dessert {
32    public Gaufre() {
33       this.libelle = "gaufre";
```

```
        this.prix = 1.8;
    }
}

public abstract class DecorateurIngredient extends Dessert {
    protected Dessert dessert;

    protected DecorateurIngredient(Dessert dessert) {
        this.dessert = dessert;
    }

    public abstract String getLibelle();
    public abstract double getPrix();
}

public class Chocolat extends DecorateurIngredient {
    public Chocolat(Dessert dessert) {
        super(dessert);
    }

    @Override
    public String getLibelle() {
        return this.dessert.getLibelle() + " chocolat";
    }

    @Override
    public double getPrix() {
        return this.dessert.getPrix() + 0.5;
    }

    @Override
    public String toString() {
        return this.dessert.getLibelle() + " chocolat, prix : " + (this.dessert.getPrix() + 0.5 )+ "€";
    }
}

public class Chantilly extends DecorateurIngredient {
    public Chantilly(Dessert dessert) {
        super(dessert);
    }

    @Override
    public String getLibelle() {
        return this.dessert.getLibelle() + " chantilly";
    }

    @Override
    public double getPrix() {
        return this.dessert.getPrix() + 0.4;
    }

    @Override
    public String toString() {
    return this.dessert.getLibelle() + " chantilly, prix : " + (this.dessert.getPrix() + 0.4) + "€";
    }
}
```

```java
public class Main {
   public static void main(String[] args) {
      Logger log = Logger.getLogger("log");

      // Creation d'une gaufre au chocolat
      Dessert gaufre = new Gaufre();
      Dessert gaufreChocolat = new Chocolat(gaufre);
      log.log(Level.INFO, gaufreChocolat.toString());

      // Creation d'une crepe chocolat + chantilly
      Dessert crepe = new Crepe();
      Dessert crepeChocolat = new Chocolat(crepe);
      Dessert crepeChocolatChantilly = new Chantilly(crepeChocolat);
      log.log(Level.INFO, crepeChocolatChantilly.toString());
   }
}
```

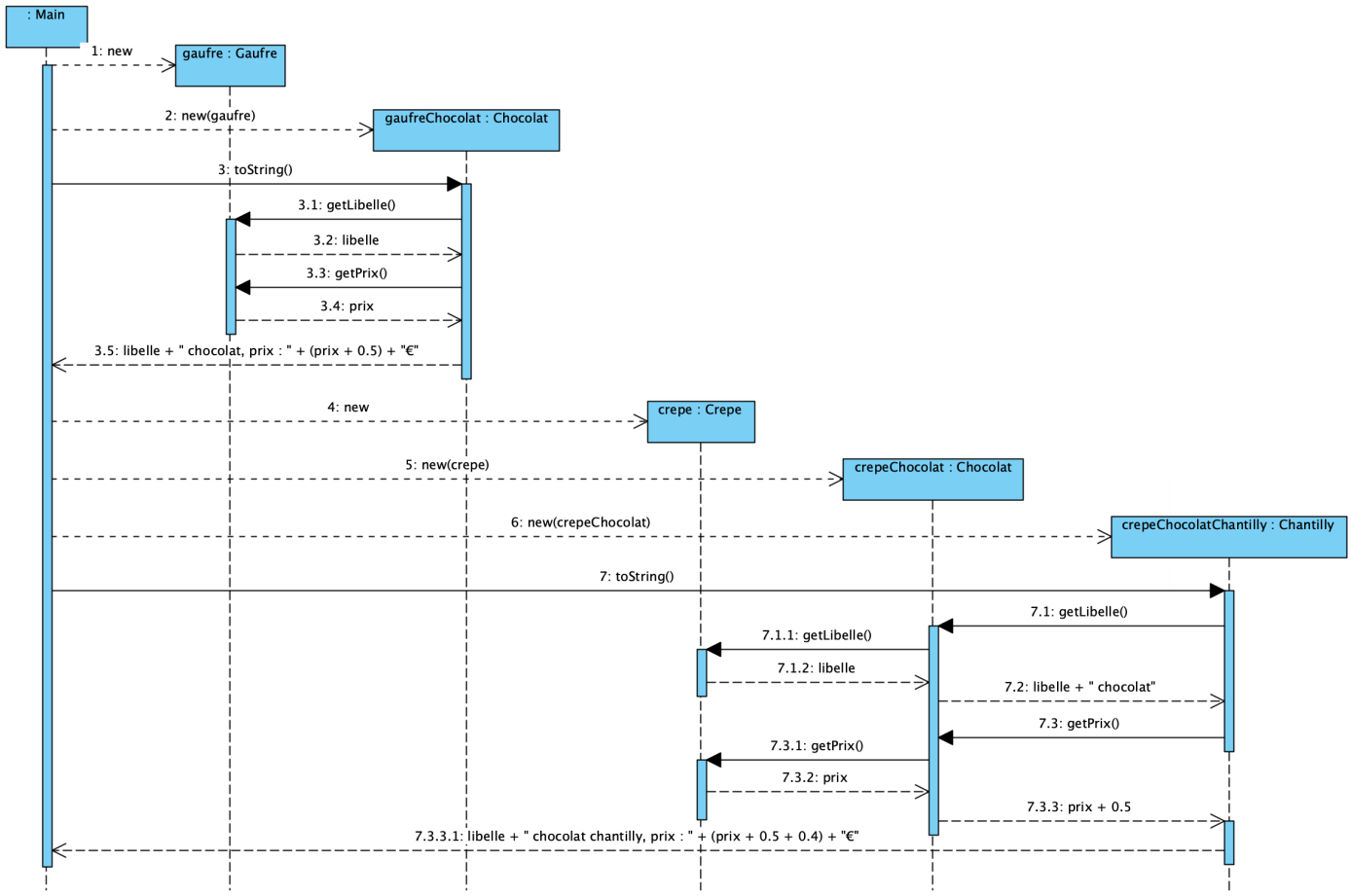


FIGURE 2 – Solution du diagramme de séquences