IUT Sénart-Fontainebleau, UPEC
CPOO
IUT Info 2A

Florent Madelaine

2022-23

Semaine 5

Les notions

1. usage : synopsis et DCU.
2. structure : DC, DO.
3. comportement : DSS et DC.
4. Abstraction : DC avec classes abstraites et interfaces, API, couplage faible.
5. Patron : design patterns (patrons de conception), reconnaître et mettre en oeuvre quelques réponses élégantes apportées à des problèmes récurrents.

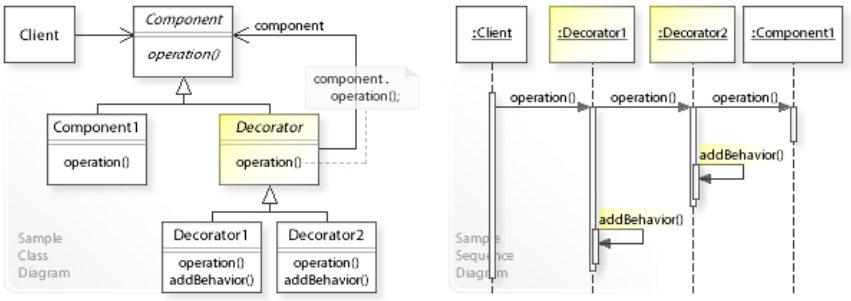
Programme de la semaine

CM : suite du cours sur les patrons de conception.
TD : TD désaltérant sur le patron decorateur
TP : TP noté autour de la bamboche.

Concepts à connaître

- patron de conception (création, structuration, comportement, ...)
- iterateur (iterator)
- observateur (observer)
- adapteur (adapter)
- composite
- singleton
- (abstract) factory

Cette semaine nous étudions en détail un patron : décorateur. Ce n'est pas très compliqué et ça permet de bien comprendre comment la délégation fonctionne.



By Vanderjoe [CC BY-SA 4.0], from Wikimedia Commons

Nous allons voir en détail en CM le patron composite et voir comment s'en servir pour résoudre un petit problème sur les arbres en TP.

Il y a plusieurs difficultés.

D'une part, il convient de comprendre comment on peut coder un arbre enraciné. C'est une structure de donnée récursive.

D'autre part, il faut comprendre le principe de calcul de bas en haut dans l'arbre.

Finalement, il y a la subtilité de l'intérêt du patron composite, pas simple à digérer tant que vous n'avez pas essayé de coder avec.

TD n° 5

Ce TD n'est pas évalué mais je peux regarder votre travail si vous souhaitez un retour individuel. Dans ce cas donnez moi votre réponse en fin de TD.

On souhaite modéliser des boissons à emporter du genre café au lait, double expresso avec de la crème, ou café calva (espresso corretto comme disent les italiens).

Ce cadre de travail correspond bien au patron de conception décorateur.

On a une interface pour les boissons.

```
public interface Boisson {
  public double getCost();
  public String getIngredients();
}
```

On a des exemples concrets de boisson.

```
public class Expresso implements Boisson {
  @Override
  public double getCost() { return 1; }

  @Override
  public String getIngredients() { return "expresso"; }
}
```

On a une classe abstraite pour les boissons « décorées ». Par défaut la boisson décorée (abstraite) ne fait rien de plus et délègue le traitement à la boison qu'on souhaite décorer.

```
public abstract class BoissonEtPlus implements Boisson {
  private final Boisson aDecorer;

  public BoissonEtPlus(Boisson b) {
    this.aDecorer = b;
  }

  @Override
  public double getCost() {
    return aDecorer.getCost();
  }

  @Override
  public String getIngredients() {
    return aDecorer.getIngredients();
  }
}
```

On peut ensuite définir des décorations concrètes.

```
class AvecLait extends BoissonEtPlus {

  public AvecLait(Boisson b) {
    super(b);
  }

  @Override
  public double getCost() {
    return super.getCost() + 0.5;
  }

  @Override
  public String getIngredients() {
    return super.getIngredients() + ", nuage de lait";
  }
}
```

ou encore

```
class Double extends BoissonEtPlus {

  public Double(Boisson b) {
    super(b);
  }

  @Override
  public double getCost() {
    return super.getCost() * 2;
  }

  @Override
  public String getIngredients() {
    return "double dose de (" + super.getIngredients() + ")";
  }
}
```

Exercice 1. Dessiner un diagramme de classe.

Ajouter à ce diagramme des classes pour permettre : de servir du thé, de corriger une boisson en y ajoutant de la goutte.

Dessiner un diagramme objet pour les boissons suivantes : thé au lait, double expresso corrigé, expresso avec deux doses de gouttes, une commande de deux espressi.

Exercice 2 (Comportement)**.** Dessiner un diagramme de séquence pour l'appel printInfo (b1); ci dessous.

```
public class Exemple {

    public static void printInfo(Boisson b) {
        System.out.println("Coût : " + b.getCost() + "; Ingrédients: " + b.getIngredients());
    }

    public static void main(String[] args) {
        Boisson b = new Expresso();
        Boisson b1 = new AvecLait(b);
        printInfo(b1);
    }
}
```