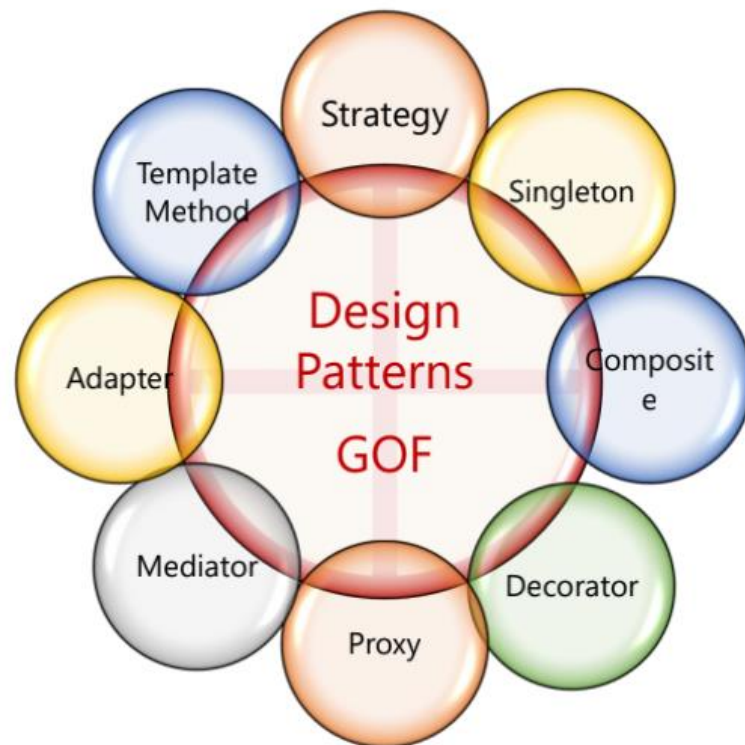


## Decorator Pattern



Réalisé par : REGOUG Zakia

GLSID3

Année universitaire : 2023-2024

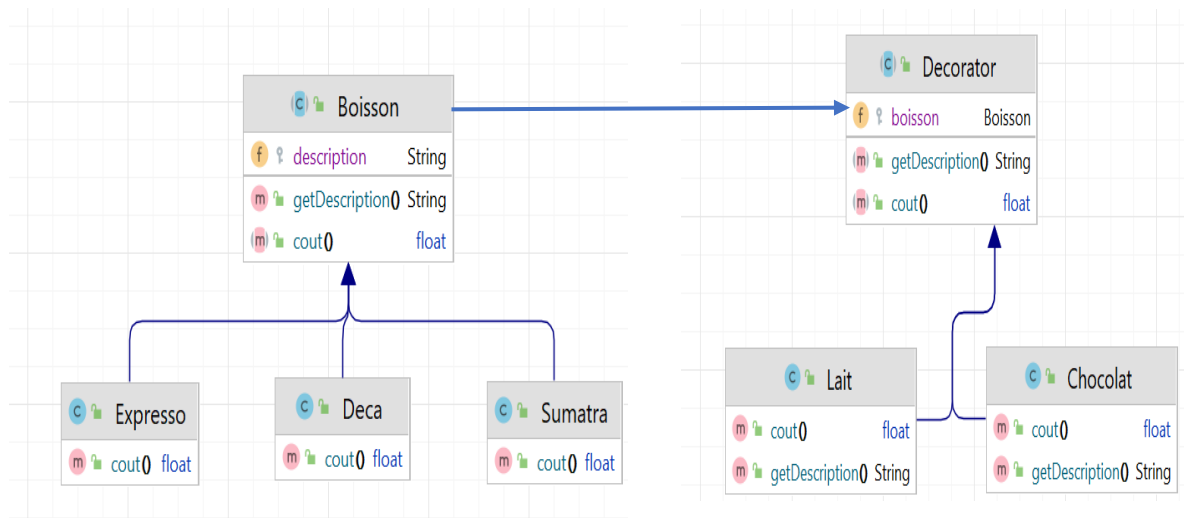
## Catégorie :

Structure

## Objectif :

Le pattern Décorateur attache dynamiquement des responsabilités supplémentaires à un objet. Il fournit une alternative souple à la dérivation pour étendre les fonctionnalités.

## Diagramme de Classe :



## Boisson abstract class :

```
package org.example.boissons;

public abstract class Boisson {
    protected String description;

    public String getDescription() {
        return description;
    }

    public abstract float cout();
}
```

## Deca :

```
public class Deca extends Boisson{

    public Deca() {
        description = "Deca";
    }

    @Override
    public float cout() {
        return 9;
    }
}
```

## Espresso :

```
public class Espresso extends Boisson{

    public Espresso() {
        description = "Espresso";
    }

    @Override
    public float cout() {
        return 11;
    }
}
```

## Summatra :

```
public class Sumatra extends Boisson{
    public Sumatra() {
        description = "Sumatra";
    }

    @Override
    public float cout() {
        return 12;
    }
}
```

## Decorator class :

```
public abstract class Decorator extends Boisson {
    protected Boisson boisson;

    public Decorator(Boisson boisson) {
        this.boisson = boisson;
    }

    public abstract float cout();
    public abstract String getDescription();
}
```

## Chocolat :

```
public class Chocolat extends Decorator{

    public Chocolat(Boisson boisson) {
        super(boisson);
    }

    @Override
    public float cout() {
        return boisson.cout()+3;
    }

    @Override
    public String getDescription() {
        return boisson.getDescription()+" Au chocolat";
    }
}
```

## Lait :

```

public class Lait extends Decorator{

    public Lait(Boisson boisson) {
        super(boisson);
    }

    @Override
    public float cout() {
        return boisson.cout()+5;
    }

    @Override
    public String getDescription() {
        return boisson.getDescription()+" Au Lait";
    }
}

```

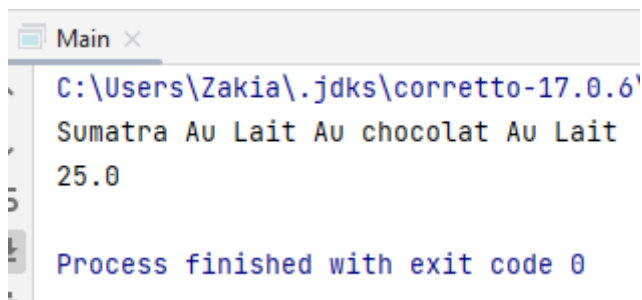
## Main

```

public class Main {
    public static void main(String[] args) {
        Boisson b= new Sumatra();
        b = new Lait(new Chocolat(new Lait(b)));
        System.out.println(b.getDescription());
        System.out.println(b.cout());
    }
}

```

## Affichage :



The screenshot shows a console window titled 'Main' with the following output:

```

C:\Users\Zakia\.jdk\corretto-17.0.6\
Sumatra Au Lait Au chocolat Au Lait
25.0

Process finished with exit code 0

```