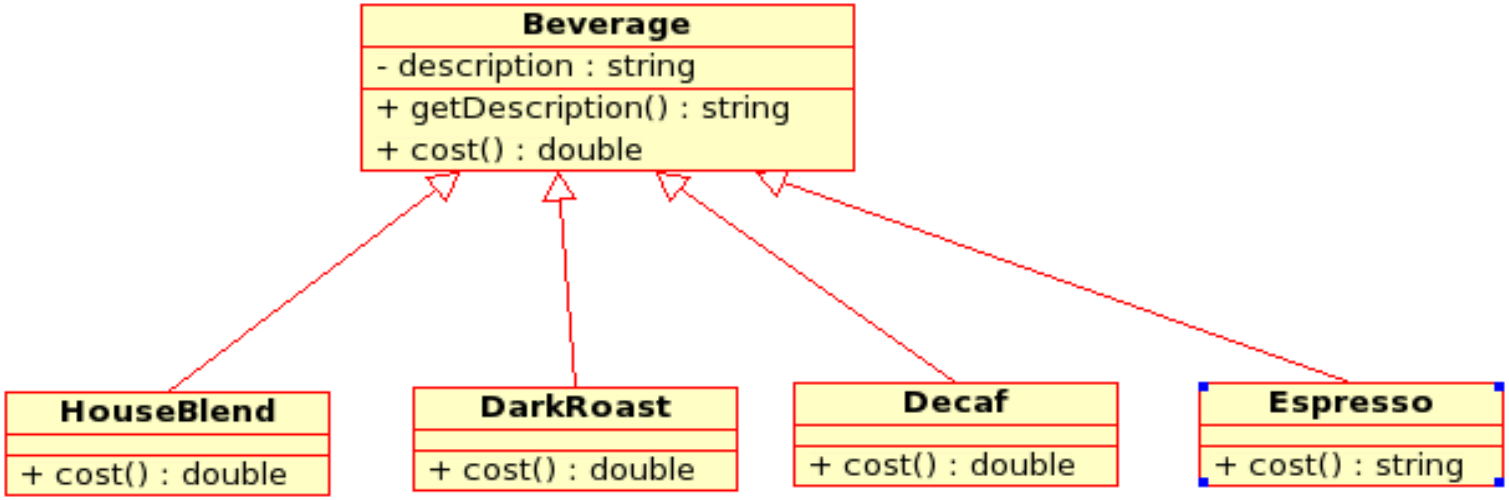
Het decorator pattern

Objectgeoriënteerde analyse en ontwerp

katholieke hogeschool
associatie KU Leuven
vives

Welcome to Starbuzz Coffee



Starbuzz Coffee verkoopt koffie.
Er is een beperkt aantal soorten koffie.
Die kunnen zonder probleem geprogrammeerd worden met 1 klasse per soort.

Maar er zijn ook toevoegingen…

- Er is koffie met melk, koffie met suiker, koffie met melk en suiker, koffie met spam, koffie met melk en spam, koffie met spam en spam, ….
  En al die toevoegingen hebben een eigen prijs!

-> Enorm veel combinaties mogelijk! Hoe programmeer je dat?

- Eerste oplossing: subklassen (specialisatie/ inheritance). 1 klasse per combinatie van toevoegingen.
- Geen goed idee: veel te veel klassen, veel te veel codeduplicatie.

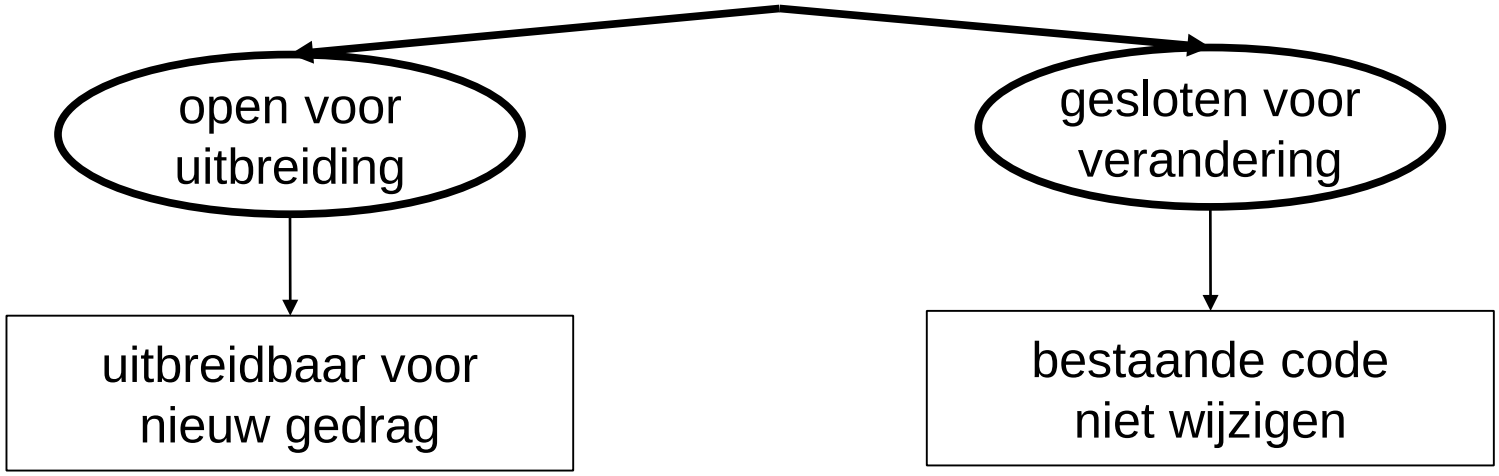
Toevoegingen (2)

- Tweede oplossing: aan de drankklassen attributen (booleans) toevoegen per soort toevoeging.
- De operatie cost() wordt ingewikkelder maar lijkt ok.
- Nadeel van deze aanpak: de drankklassen zijn afhankelijk van de toevoegingen

-> Dat is in strijd met het open/closed-principe.

Het principe open-closed.

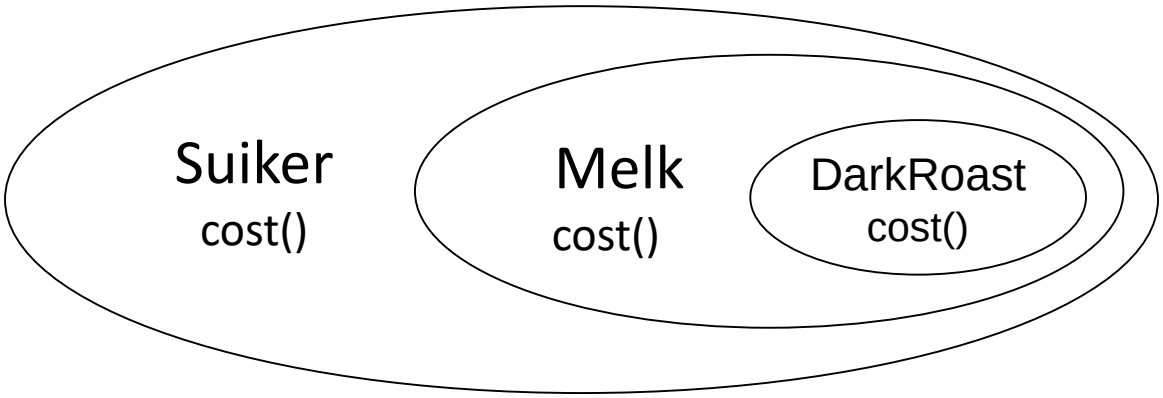
- Code moet zijn:



Het decoratorprincipe

- Neem een DarkRoast-koffie-object.
- Verpak dat in een Melk-object.
- Verpak dat geheel in een Suiker-object.
- Roep nu de operatie cost() aan bij de buitenste klasse. Die klasse doet beroep op de andere klassen op die te berekenen.

Het decoratorprincipe

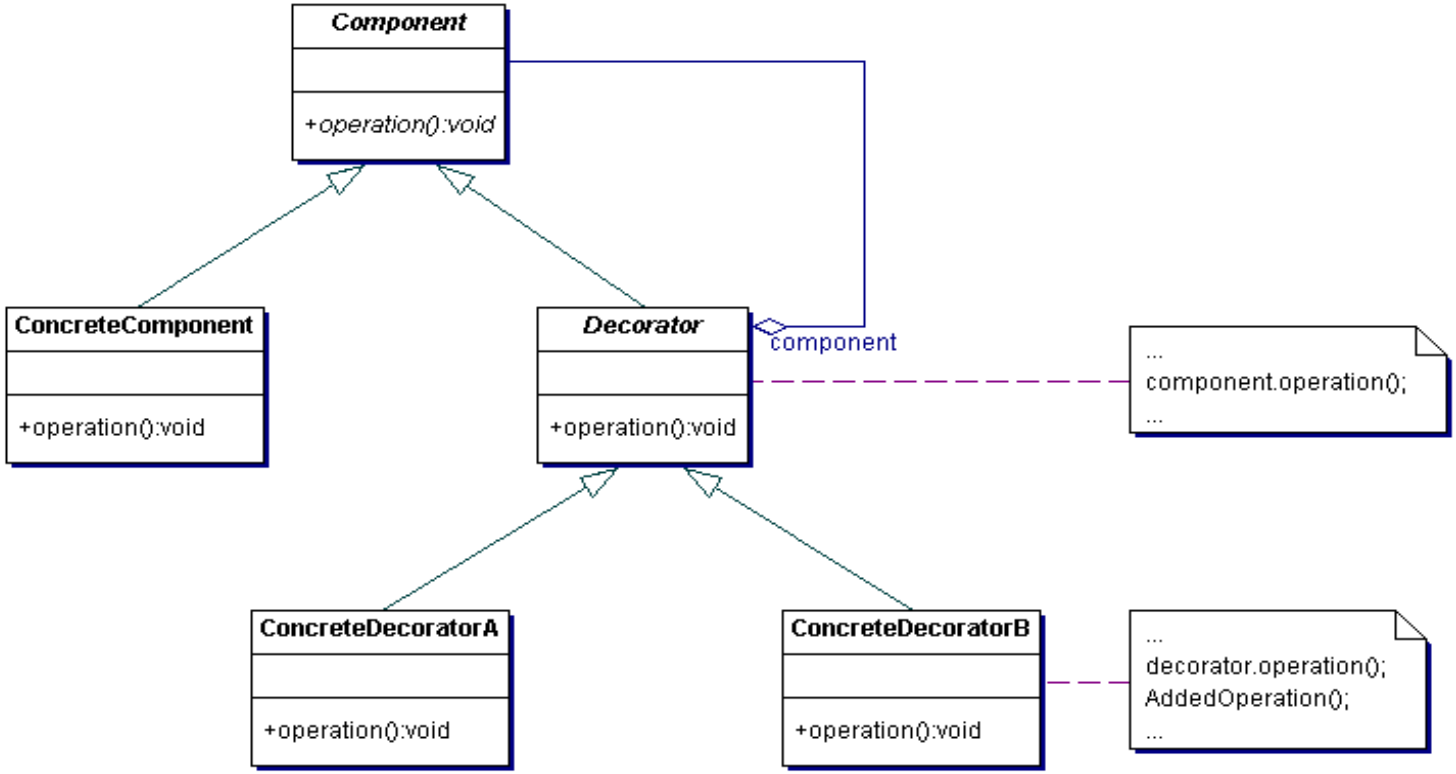


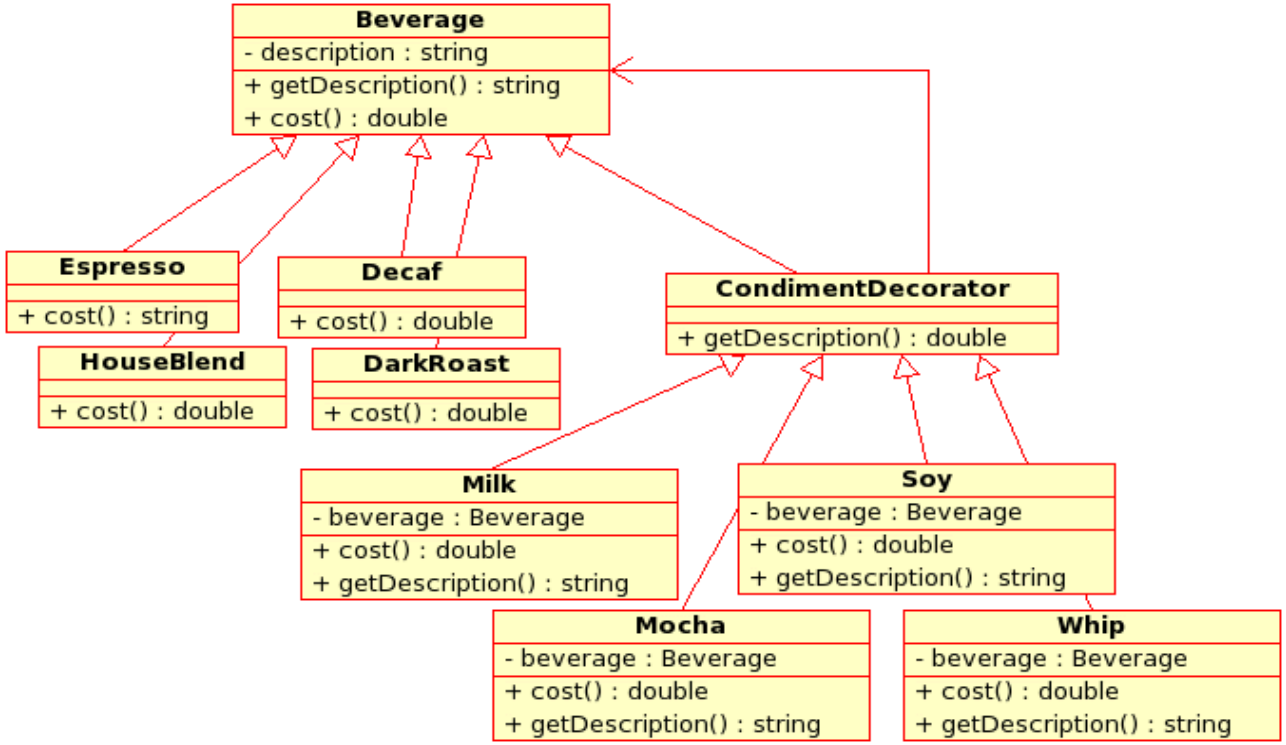
cost()

- Wij roepen cost() aan bij Suiker.
- Suiker roept cost() aan bij Melk.
- Melk roept cost() aan bij DarkRoast.
- DarkRoast geeft zijn kost: $ 2.99
- Melk voegt er 20 cent aan toe: $ 3.19
- Suiker voegt er nog eens 10 cent aan toe: $ 3.29

Decorator

- Decorator-objecten hebben hetzelfde supertype als de objecten die ze decoreren.
- Je kan één of meer decorators gebruiken om een object mee te omvatten.
- cf. punt 1: je kan dus een gedecoreerd object gebruiken als was het het originele object.
- De decorator voegt extra gedrag toe voor of na het gedrag van het object dat het omvat.
- We kunnen decorators dynamisch, i.e. “at runtime”, toevoegen.





Beverage

```
public abstract class Beverage {
        String description = “Onbekende drank”;
        public String getDescription() {
                return description;
                }
         public abstract double cost();
}
```

CondimentDecorator

```
public abstract class CondimentDecorator extends Beverage {
    public abstract String getDescription();
}
```

moet opnieuw geïmplementeerd worden!

De concrete koffiesoorten

```
public class DarkRoast extends Beverage {
        public DarkRoast() {
                description = “DarkRoast”;
        }
        public double cost() {
                return 1.99;
        }
}
```

De extra's

```
public class Suiker extends CondimentDecorator {
        Beverage beverage;
        public Suiker(Beverage beverage) {
                this.beverage = beverage;
        }
        public String getDescription() {
                return beverage.getDescription() + ", met suiker";
        }
        public double cost() {
                return .20 + beverage.cost();
        }
}
```

de 'kern' die zal uitgebreid worden

iets toevoegen aan de kern

Gebruik

```
Beverage beverage = new DarkRoast();
Beverage drMetMelk = new Melk(beverage);
Beverage drMetMelkEnSuiker = new Suiker(drMetMelk);
```

ofwel

- Beverage beverage = new Suiker(new Melk(new DarkRoast)))

Nadelen

- Soms kan een overdadig gebruik van het Decorator-patroon leiden tot een grote verzameling kleine klassen. Dit maakt de code minder doorzichtig.
- Let er ook op dat niemand client-code schrijft die een specifiek type verwacht, want dat type kan met een decorator getooid zijn en dus een licht gewijzigd gedrag hebben (=> is in feite fout van client).
- Constructie ziet er soms ingewikkeld uit. (Evtl Builder pattern gebruiken om dat te verbergen.)

Wat hebben we geleerd?

- Decorator. Patroon om dynamisch extra verantwoordelijkheden aan een object toe te voegen.
- Klassen moet open zijn voor uitbreiding, maar gesloten voor wijzigingen in hun code.