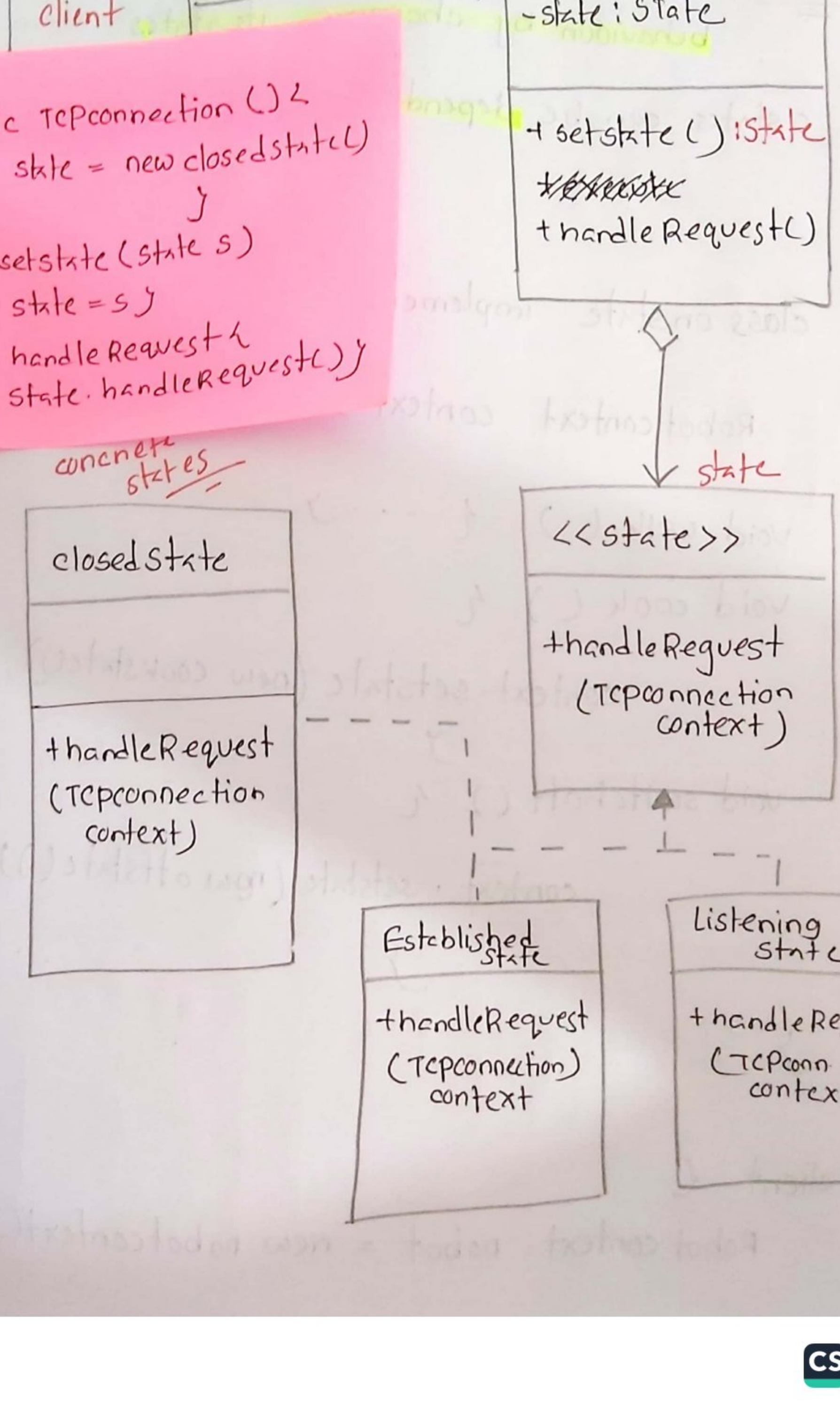
17

state pattern

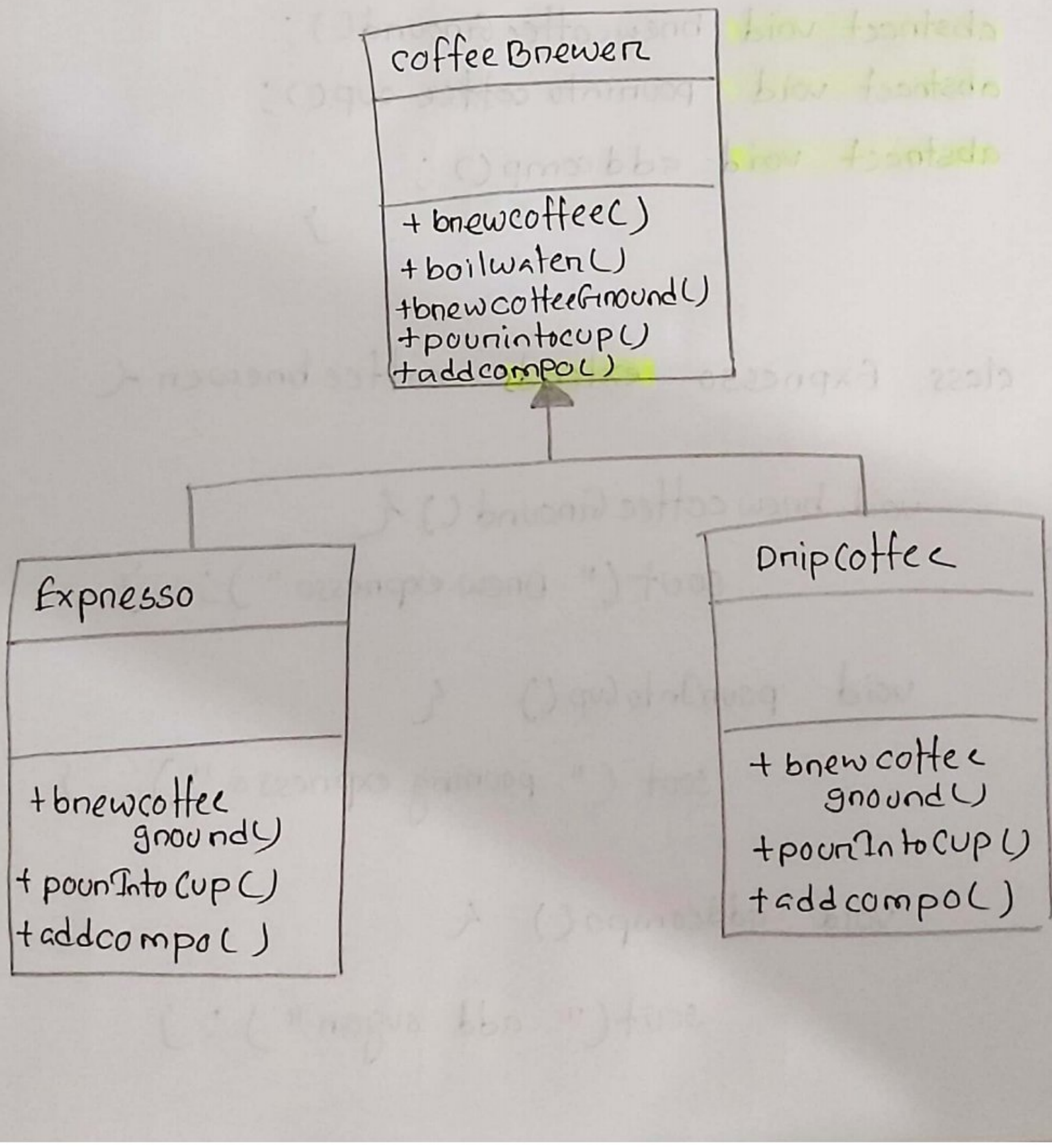
(*) best example follow this

```
Public TCPconnection (){
    state = new closedstate()
            }
void setstate (State s)
  { state = s }
void handleRequest {
    state.handleRequest() }
```



Template Design pattern

Template is a behavonical pattern that provides a way to define a skeleton of an algonithm in a method of base class and allows subclasses to ovennide specific step of ~~the~~ the algonithm without changing its stnuctune



```
abstract class  coffeebnewer {

    final void bnewcoffee () {

        boilwater();
        bnewcoffeeGround();
        pourInto cup();
        addcomp();
                                }

    void boilwater () {                 }

    abstract void bnewcoffeeGround();
    abstract void pourinto coffee cup();
    abstract void addcomp();
                                }

class Expnesso extends coffee bnewer {

    void bnew coffee Ground () {
            sout(" Bnew expnesso ");  }

    void  pourInto Cup ()  {
            sout (" pouring expnesso ");  }

    void  addcompo() {

            sout(" add sugan ");  }
```

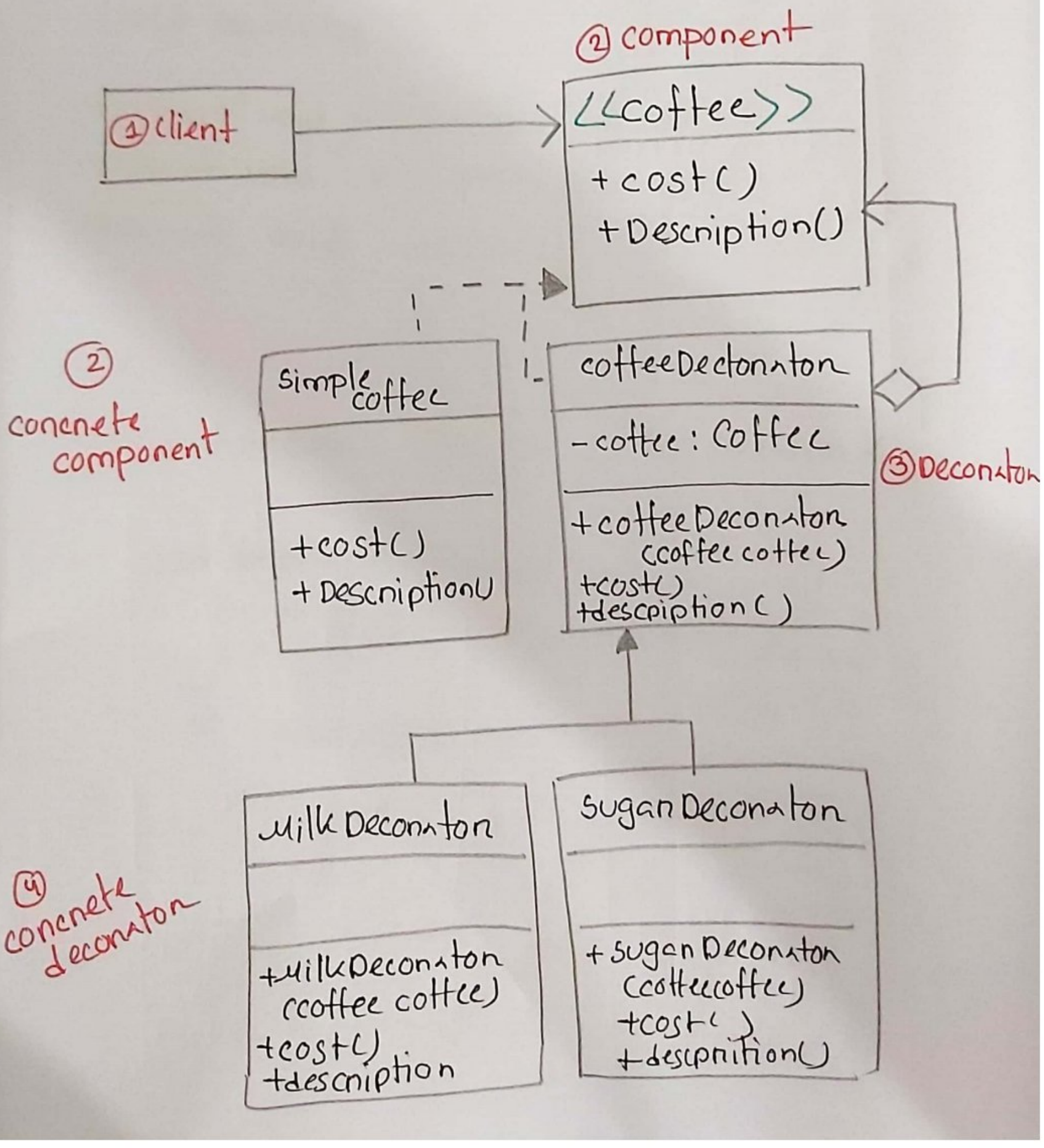
(x) strategy pattern
→ changes the guts / behaviour of an object

(x) Decorator pattern
→ changes the skin / appearance of an object

enchance object without changing its interface

Decorator Design Pattern

is a structural pattern to provide a flexible and dynamic way to extend the functionality of objects without using subclassing. Allows us to add new behaviour of object at Runtime



① interface coffee ( ) {

}

② class simplecoffee implements coffee {

}

③ class coffee Decorator implements coffee {

private Coffee coffee;

coffeeDecorator (coffee coffee)

{ this coffee = coffee }

void cost ()

{ coffee.cost() }

void description ()

{ coffee. description() }

④ class Milk Decorator extends coffee Decorator {

public MilkDecorator (coffee coffee)

{ super (coffee); }

void cost()

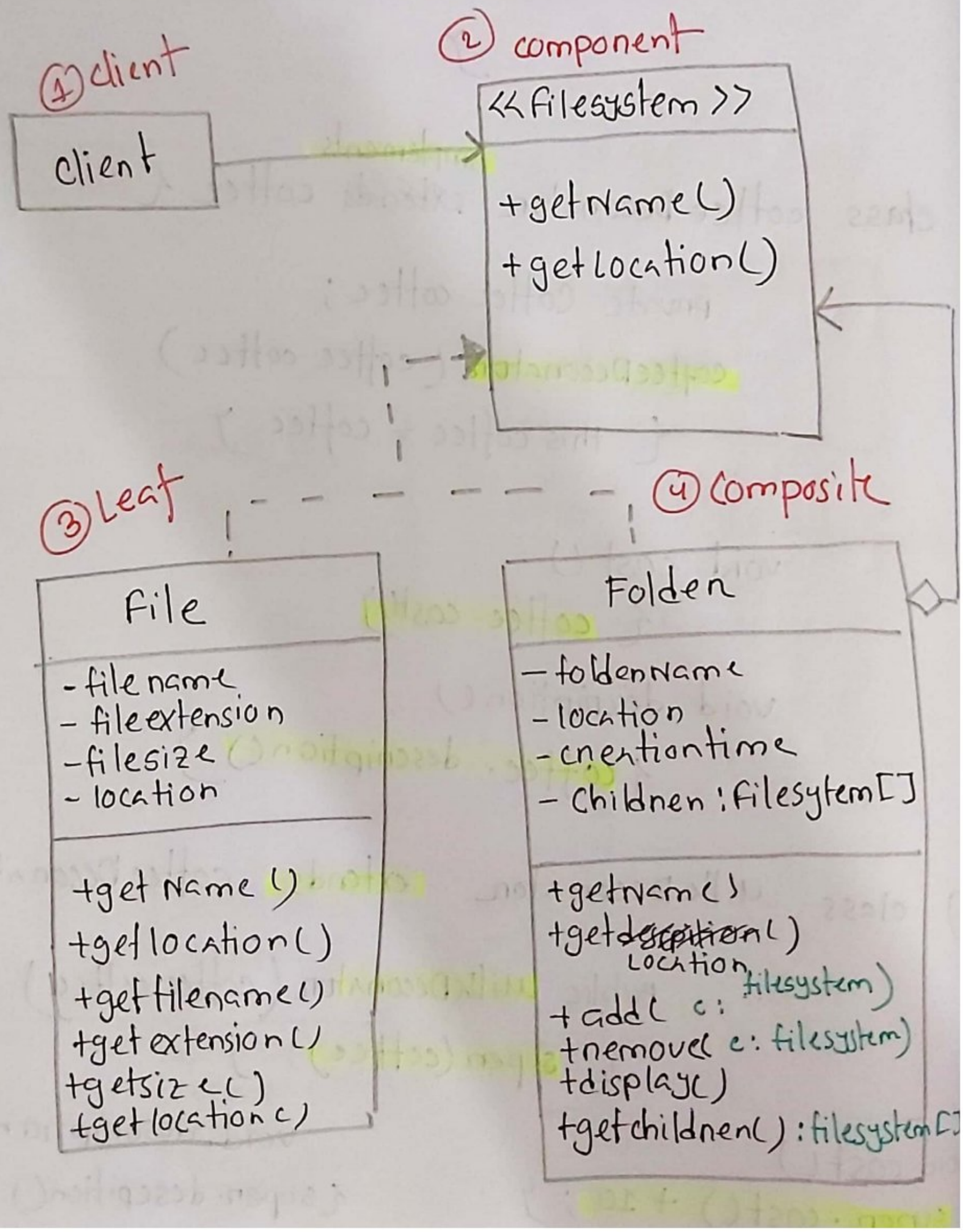
{ super.cost() + 10 ; }

void description ()

{ super. description() + "with Milk" }

Composite Pattenn

is a structural design pattenn that lets you compose objects .. into tnee structures and then work with these structune as if they ... whene individual object



```
interface filesystem() {

}

class file implements filesystem {
        private string filename
        :
file( string filename, . . . . )
{   this.filename = filename
    :
getfile name()
{ }
}

class folden implements filesystem {
        private string folden Name ;
        :
List< filesystem> child = new AnayList<>() ;

Folden (        )
{ :
}

void add ( filesystem c)
{ child .add (c) }

List<filesystem)
void getchildnen(){
    netunn childnen ;
}
```

Patterns similar to composite

① Decoration Pattern

similarities

① both involve concept of composition

② use a tree like structure

③ treat individual objects and composition of objects uniformly

④ allow addition of new feature

dissimilarities

① purpose

composite :-
creating an hierarchical structures of objects

Decoration :-
add responsibility to an Individual object extend behavior

② composite :-
all component less, composite all treated uniformly under common interface

Decoration :- Distinction between component and Decoration

(2) chain of Responsibility

Similanities

(1) (2) (3)

Dissimilan

(1) COR → structure linear chain of handlers

composite → hierarchical tree like structure

(2) COR → each handler decide to handle it or pass

composite → all component handle the request

(x) Strategy pattern

→ changes the guts / behaviour of an object

(x) Decoration pattern

→ changes the skin / appearance of an object

—— o ——