Decorator

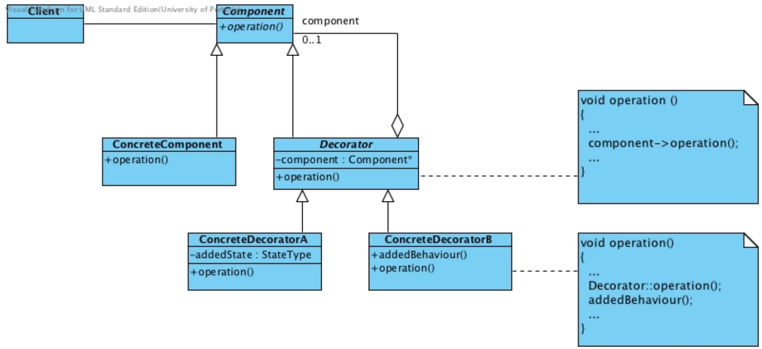
Linda Marshall and Vreda Pieterse

Department of Computer Science
University of Pretoria

29 August 2014

Overview

Name and Classification: Decorator (Object Structural) **Intent:** “Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.” GoF(175)



```
void operation ()
{
  ...
  component->operation();
  ...
}
```

```
void operation()
{
  ...
  Decorator::operation();
  addedBehaviour();
  ...
}
```

- Looks similar to the Composite. A composite may comprise of multiple components, while Decorators comprise of 0 or 1.
- Composite themselves do not have specilisations, while Decorators do.

Component

- interface for objects that can have responsibilities dynamically added to them.

ConcreteComponent

- the object to which the additional responsibilities can be attached

Decorator

- defines a reference to a Component-type object

ConcreteDecorator

- adds the responsibilities to the component

- **Adapter** (139) : Changes the interface to an object while the Decorator only changes responsibilities.
- **Composite** (163) : A Decorator can be seen as a Composite with only one component that has added responsibility.
- **Strategy** (315) : The Strategy pattern changes the inner workings of an object

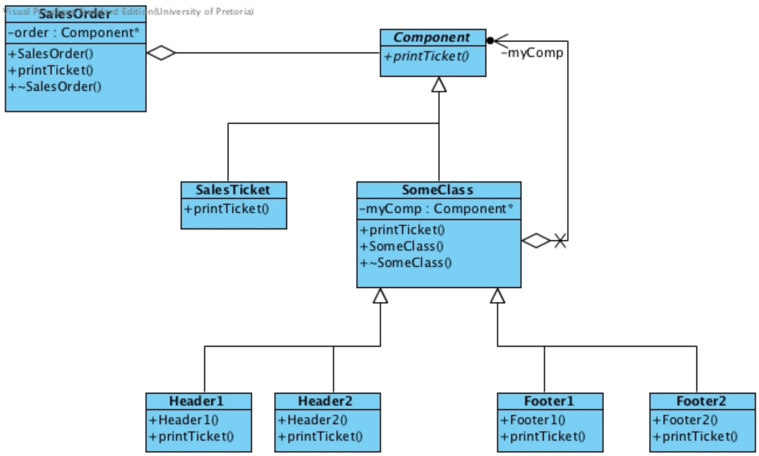
Luigi (the restauranteur from the Ashes to Ashes series) has decided to begin a fast pizza outlet called Dumbo's - he was a Disney fan. Luigi wants his clientele to choose the toppings for the pizza. Each topping has a price associated with it. A Hawaiian pizza for example has a tomato base, mozzarella cheese, ham and pineapple.

```
class Pizza {
  // methods to get a description
  // and the cost
};
class Hawaiian : public Pizza {
  // description and cost methods
  // specifically to Hawaiian pizzas
};
```

What if -

- the cost of ham changes?
- Luigi wants to add salami to the pizza? Or maybe even cheddar cheese!

The previous design will require another class to be defined to inherit from Hawaiian which increases the price of the pizza to include salami and updates the description accordingly. You can see where this is going.........
How would you use the Decorator pattern to solve your problem?


Visual Paradigm Standard Edition(University of Pretoria)
SalesOrder
-order : Component*
+SalesOrder()
+printTicket()
+~SalesOrder()
Component
+printTicket()
-myComp
SalesTicket
+printTicket()
SomeClass
-myComp : Component*
+printTicket()
+SomeClass()
+~SomeClass()
Header1
+Header1()
+printTicket()
Header2
+Header2()
+printTicket()
Footer1
+Footer1()
+printTicket()
Footer2
+Footer2()
+printTicket()