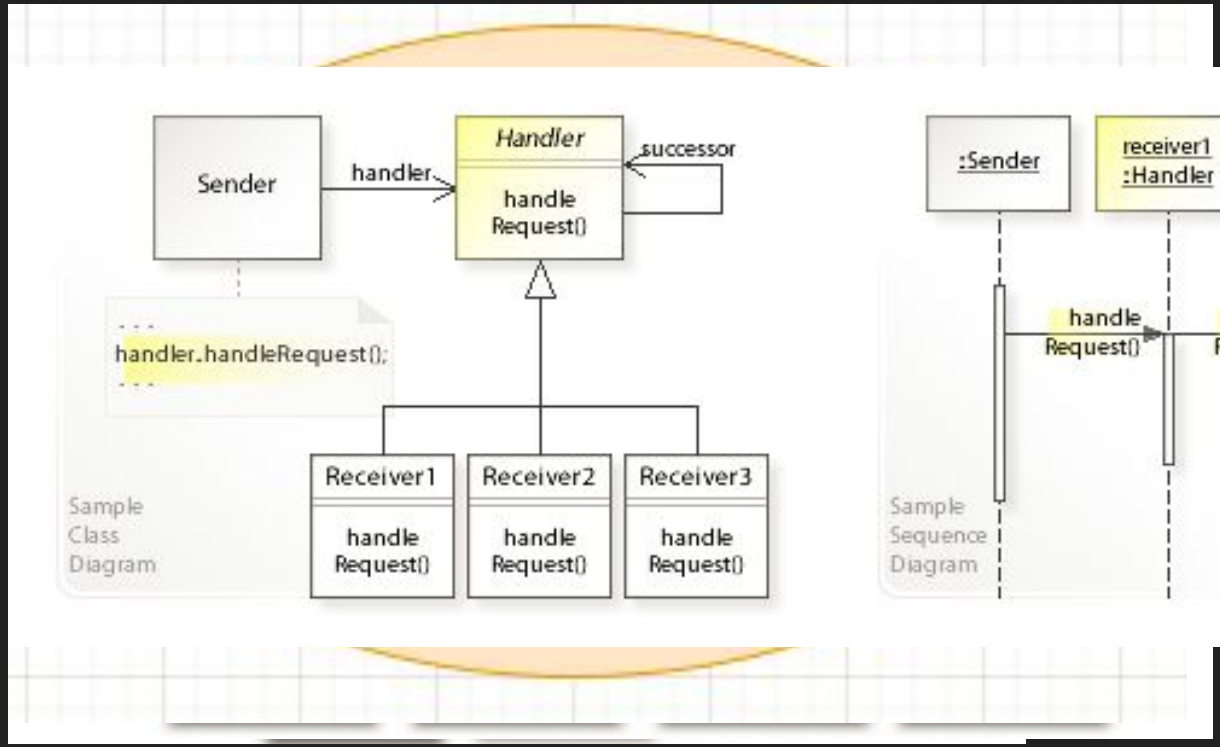
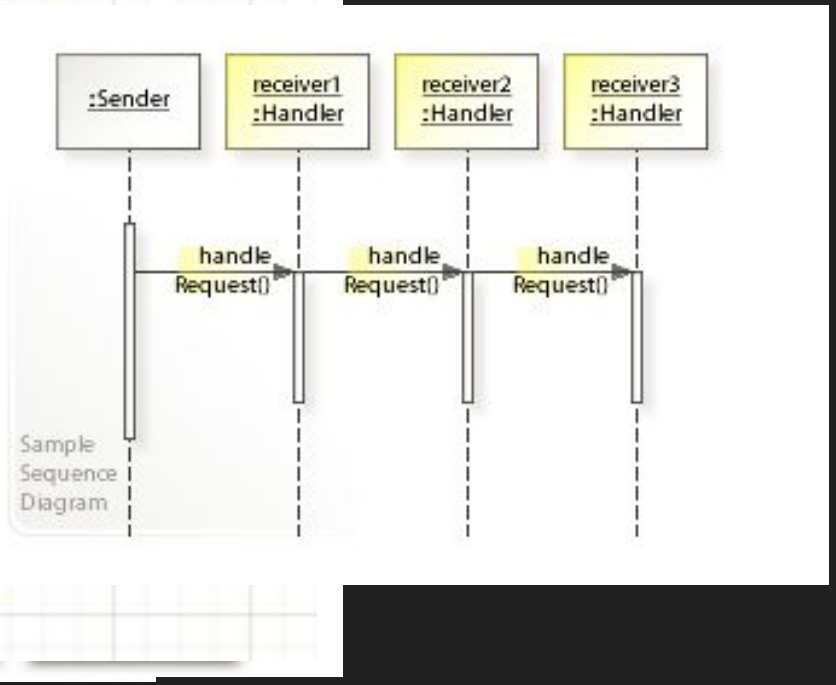
Decorator Pattern

Group 6
Tri & Arseniy

About the Decorator Pattern



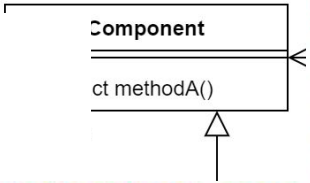


Decorator Patte

- Type
- St
- Dy

```
public abstract class Food
{
    //Default constructor
    public Food()
    {
        _description = "Unspecified food.";
        _cost = 0.00;
    }

    public abstract string _description { get; set; }
    public abstract double _cost { get; set; }
}
```

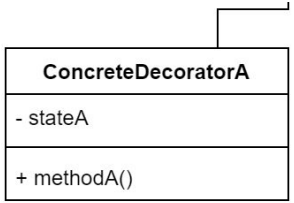
```
public class Pizza : Food
{
    //Constructor
    public Pizza()
    {
        _description = "Pizza";
        cost = 65.00;
    }
```

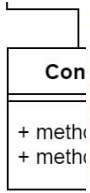


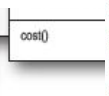

```
public abstract class ToppingDecorator : Food
{
    public ToppingDecorator()
    {

    }
```

```
//We want a pizza with marinara sauce and mozzarella cheese
Pi PS C:\Users\jeg\Documents\GitHub\I6SWD> dotnet run
Ma We want a Pizza, Marinara, Mozzarella
Mo Which costs: 75,-
Co
Console.WriteLine("Which costs: " + mozzarella1._cost + ",-\n");
```

```
get;set;}
_description {get;set;}
_cost {get;set;}

la : ToppingDecorator

(Food food)

tiveFood = food;
```








```
    //Property implementations
    public override string _description {
        get{
            return _additiveFood._description + ", Mozzarella";
        }
        set{
        }
    }
    public override double _cost {
        get{
            return _additiveFood._cost + 5.00;
        }
        set{
        }
    }
}
```

Decorator Pattern in practice

- Drawbacks?
  - Added decorators cause added maintenance
  - Rigid structure
  - Overuse of open close principle
  - Client needs to be aware of decorators present

- Uses
  - Places where you want to add functionality on run time.
  - Avoid subclassing nightmare
  - Practical case would be file read/write