Structural - Decorator Design Pattern

1. Şöyle bir durum düşünelim. Bir rapor formatı hazırladınız ve sundunuz. Ardından her bir satır başına çizgi istendi, başka bir gün hem çizgili hem imza yeri olan bir rapor istendi, veya bu tür istekler devam etti. Dolayısıyla basit rapor formatımızı dekore etmek mantığı için kullanılan bir pattern'dir. Yani bir nesneye yeni özellikler eklemek için kullanılır

Faydaları Nedir?

1. loosely-coupled uygulamalar yapmayı sağlar.
2. Runtime zamanında(dinamik olarak) bir nesneye yeni özellikler eklenmesini sağlar.
3. Özellikleri kalıtım yolu dışında composition ve delegation ile de alınabilmesini sağlar.
4. open-closed prensibinin uygulandığı tasarım desenidir.

```
interface Report{
   String getText();//raporun metni
}
*****************************************
class BasicReport implements Report{
   //Düz rapor hiç bir özelliği yok
   private String text;
   public BasicReport(String newText){
      text = newText;
   }
   @Override
   public String getText() {
      return text;
   }
}
*****************************************
class LittleReport extends ReportDecorator{
   //küçük rapor
   public LittleReport(Report report) {
      super(report);
   }

   @Override
   public String getText() {
      return super.getText() + "Kucuk rapor oluşturuldu";
   }
}
*****************************************
class LinedReport extends ReportDecorator{
   //çizgili rapor
   public LinedReport(Report report) {
      super(report);
   }

   @Override
   public String getText() {
      String text = super.getText();
      return ReportUtil.getLinedReport(text);
   }
}

//Report'ları oluşturmak için yardımcı bir class
class ReportUtil{
   public static String getLinedReport(String text){
      //logic yazmak istemediğim mock bir string yazdım
      //gelen rapor text'inin her bir satırına çizgi eklemek aslında amacımız
      String description = "Her bir satıra çizgi ekle" + text;
      return description;
   }
}
```

```
//Decorator
abstract class ReportDecorator implements Report{
    private Report report;
    public ReportDecorator(Report report) {
        this.report = report;
    }
    @Override
    public String getText() {
        return report.getText();
    }
}

//Main
public static void main(String[] args) {
    BasicReport basicReport = new BasicReport("asdadsasdads");
    String text = basicReport.getText();
    System.out.println(text);
    LinedReport report = new LinedReport(basicReport);
    String linedReportText= report.getText();
    System.out.println(linedReportText);
}
```