Decorator Pattern là một trong những mẫu thiết kế phần mềm cơ bản, nó cho phép bổ sung các chức năng mới cho một đối tượng hiện có mà không cần phải thay đổi cấu trúc của đối tượng đó. Đây là một mẫu thiết kế linh hoạt và mạnh mẽ cho phép mở rộng các tính năng của một đối tượng theo cách dễ dàng và linh hoạt.

Ý tưởng chính của Decorator Pattern là sử dụng một lớp Decorator để bọc lấy một đối tượng hiện có, sau đó bổ sung các chức năng mới hoặc thay đổi hành vi của đối tượng gốc bên trong. Lớp Decorator sẽ có cùng interface với đối tượng gốc, do đó nó có thể thay thế đối tượng gốc và sử dụng những tính năng mới được bổ sung mà không làm ảnh hưởng đến chức năng ban đầu của đối tượng gốc.

Ưu điểm:

* Cho phép thêm hoặc loại bỏ các tính năng của một đối tượng mà không cần thay đổi cấu trúc của đối tượng đó.
* Giúp tách biệt phần mở rộng chức năng của đối tượng và phần chức năng cố định.
* Giúp giảm sự phức tạp của hệ thống bằng cách chia nhỏ chức năng thành các đối tượng nhỏ hơn và đơn giản hơn.

Nhược điểm:

* Tăng độ phức tạp của hệ thống vì cần tạo nhiều lớp Decorator khác nhau.
* Có thể dẫn đến việc nhiều lớp Decorator sẽ phải được kết hợp với nhau, dẫn đến việc quản lý và bảo trì đối tượng khó khăn hơn.

Ví dụ:

Trong một bệnh viện, chúng ta có thể sử dụng Decorator Pattern để bổ sung các tính năng cho các đối tượng như bệnh nhân, y tá hoặc bác sĩ. Ví dụ, chúng ta có thể sử dụng Decorator Pattern để bổ sung chức năng thêm thông tin sức khỏe cho một bệnh nhân, hoặc bổ sung các chức năng của một bác sĩ như chẩn đoán bệnh, chữa bệnh và kê đơn thuốc.

Chúng ta có thể tạo ra một lớp Decorator mới có tên là PatientDecorator để bổ sung các chức năng mới cho đối tượng bệnh nhân

Việc bổ sung chức năng mới cho đối tượng bệnh nhân có thể được thực hiện bằng cách tạo ra các lớp Decorator khác nhau, mỗi lớp có thể bổ sung một tính năng cụ thể.

Ví dụ, chúng ta có thể tạo ra lớp SicknessDecorator để bổ sung thông tin về các bệnh tật của bệnh nhân, hoặc tạo ra lớp InsuranceDecorator để bổ sung thông tin về bảo hiểm y tế của bệnh nhân.

Mỗi lớp Decorator sẽ có cùng interface với đối tượng bệnh nhân, do đó chúng ta có thể sử dụng các lớp Decorator này để thay thế đối tượng bệnh nhân ban đầu mà không làm ảnh hưởng đến các chức năng ban đầu của bệnh nhân.

Nếu chúng ta muốn bổ sung nhiều tính năng khác nhau cho đối tượng bệnh nhân, chúng ta có thể kết hợp các lớp Decorator lại với nhau để tạo ra một đối tượng bệnh nhân hoàn chỉnh.

Ví dụ cụ thể về cách sử dụng Decorator Pattern trong mô hình bệnh viện:

// Interface đại diện cho đối tượng bệnh nhân  
public interface Patient {  
 String getName();  
 String getGender();  
 int getAge();  
}

// Lớp bệnh nhân cơ bản  
public class BasicPatient implements Patient {  
 private String name;  
 private String gender;  
 private int age;  
  
 public BasicPatient(String name, String gender, int age) {  
 this.name = name;  
 this.gender = gender;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getGender() {  
 return gender;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}

// Lớp Decorator bổ sung tính năng thông tin bảo hiểm y tế  
public class InsuranceDecorator implements Patient {  
 private Patient patient;  
 private String insuranceNumber;  
  
 public InsuranceDecorator(Patient patient, String insuranceNumber) {  
 this.patient = patient;  
 this.insuranceNumber = insuranceNumber;  
 }  
  
 public String getName() {  
 return patient.getName();  
 }  
  
 public String getGender() {  
 return patient.getGender();  
 }  
  
 public int getAge() {  
 return patient.getAge();  
 }  
  
 public String getInsuranceNumber() {  
 return insuranceNumber;  
 }  
}

// Lớp Decorator bổ sung tính năng thông tin bệnh tật  
public class SicknessDecorator implements Patient {  
 private Patient patient;  
 private String sickness;  
  
 public SicknessDecorator(Patient patient, String sickness) {  
 this.patient = patient;  
 this.sickness = sickness;  
 }  
  
 public String getName() {  
 return patient.getName();  
 }  
  
 public String getGender() {  
 return patient.getGender();  
 }  
  
 public int getAge() {  
 return patient.getAge();  
 }  
  
 public String getSickness() {  
 return sickness;  
 }  
}

Chúng ta đã định nghĩa ba lớp khác nhau: BasicPatient, SicknessDecorator và InsuranceDecorator.

Lớp BasicPatient đại diện cho đối tượng bệnh nhân cơ bản, chứa các thông tin cơ bản như tên, giới tính và tuổi.

Lớp SicknessDecorator và InsuranceDecorator đại diện cho các lớp Decorator, mỗi lớp Decorator sẽ bổ sung một tính năng cụ thể cho đối tượng bệnh nhân. Lớp SicknessDecorator bổ sung thông tin về bệnh tật của bệnh nhân, trong khi lớp InsuranceDecorator bổ sung thông tin về bảo hiểm y tế của bệnh nhân.

Sau đây ta sẽ triển khai các Decorator đã khởi tạo:

public class Main {  
 public static void main(String[] args) {  
 Patient patient = new BasicPatient("John", "Male", 30);  
  
 patient = new SicknessDecorator(patient, "Flu");  
  
 patient = new InsuranceDecorator(patient, "123456789");  
  
 System.*out*.println(patient.getName());  
 System.*out*.println(patient.getGender());  
 System.*out*.println(patient.getAge());  
 System.*out*.println(((SicknessDecorator) patient).getSickness());  
 System.*out*.println(((InsuranceDecorator) patient).getInsuranceNumber());  
 }  
}