Adapter Pattern là một mẫu thiết kế trong lập trình hướng đối tượng (OOP), nó cho phép một đối tượng hoạt động với các đối tượng khác mà không cần biết chi tiết cài đặt của chúng.

Ví dụ: Giả sử ta có một lớp "Client" muốn gọi một phương thức của một đối tượng "Service". Tuy nhiên, đối tượng "Service" này đã được triển khai bằng một interface khác với interface mà "Client" đã biết. Để giải quyết vấn đề này, ta sẽ sử dụng Adapter Pattern để chuyển đổi interface cũ thành interface mới để "Client" có thể gọi được phương thức đó.

Ví dụ về cách áp dụng Adapter Pattern trong mô hình bệnh viện: Ta có thể sử dụng Adapter Pattern để chuyển đổi các hệ thống quản lý thông tin bệnh nhân khác nhau thành một interface chung để các bác sĩ, y tá và nhân viên y tế có thể truy xuất thông tin bệnh nhân một cách đồng nhất và thuận tiện hơn.

Giả sử ta có một ứng dụng quản lý bệnh viện với các lớp như sau:

public interface ElectronicHealthRecord {  
 String getPatientName();  
 int getPatientAge();  
}

public class HospitalEHR implements ElectronicHealthRecord {  
 @Override  
 public String getPatientName() {  
 return "John";  
 }  
  
 @Override  
 public int getPatientAge() {  
 return 35;  
 }  
}

public class Patient {  
 private String name;  
 private int age;  
  
 public Patient(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
}

Ở đây, ElectronicHealthRecord là một interface mô tả thông tin về bệnh nhân. Lớp HospitalEHR triển khai interface này để cung cấp thông tin về bệnh nhân. Lớp Patient là một đối tượng đại diện cho một bệnh nhân trong hệ thống.

Tuy nhiên, nếu ta muốn sử dụng đối tượng Patient để truy xuất thông tin của bệnh nhân từ đối tượng HospitalEHR, ta cần sử dụng Adapter Pattern để chuyển đổi interface HospitalEHR thành interface mới, phù hợp với lớp Patient.

Vì vậy, ta có thể tạo một Adapter như sau:

public class PatientAdapter implements ElectronicHealthRecord {  
 private Patient patient;  
  
 public PatientAdapter(Patient patient) {  
 this.patient = patient;  
 }  
  
 public String getPatientName() {  
 return patient.getName();  
 }  
  
 public int getPatientAge() {  
 return patient.getAge();  
 }  
}

Lớp PatientAdapter triển khai interface ElectronicHealthRecord và sử dụng đối tượng Patient để cung cấp thông tin về bệnh nhân. Bây giờ, ta có thể sử dụng đối tượng Patient để truy xuất thông tin từ đối tượng HospitalEHR như sau:

public class Main {  
 public static void main(String[] args) {  
 Patient patient = new Patient("John", 35);  
 ElectronicHealthRecord ehr = new PatientAdapter(patient);  
 System.*out*.println("Patient name: " + ehr.getPatientName());  
 System.*out*.println("Patient age: " + ehr.getPatientAge());  
 }  
}

Bằng cách sử dụng Adapter Pattern, ta có thể chuyển đổi một interface không phù hợp cho một lớp thành một interface mới phù hợp với lớp đó, giúp các đối tượng khác có thể tương tác với lớp đó một cách dễ dàng hơn.