Bridge Pattern là một mẫu thiết kế cấu trúc (structural design pattern) cho phép tách một lớp trừu tượng ra khỏi cài đặt cụ thể của nó và cho phép các lớp con của nó có thể được thay đổi mà không ảnh hưởng đến các lớp khác. Pattern này được sử dụng khi có sự phân cấp lớp trong một hệ thống và khi có nhiều chiến lược (strategy) khác nhau được áp dụng cho mỗi lớp con.

Để minh họa cho pattern này, giả sử chúng ta có một ứng dụng quản lý bệnh viện và chúng ta cần hiển thị thông tin bệnh nhân. Chúng ta có một interface Patient đại diện cho thông tin của một bệnh nhân và các lớp triển khai của interface này đại diện cho các loại bệnh nhân khác nhau, ví dụ như bệnh nhân nội trú (InPatient) hoặc bệnh nhân ngoại trú (OutPatient).

public interface Patient {  
 int getPatientId();  
 String getPatientName();  
}

Chúng ta cũng có một abstract class PatientInformationDisplay để hiển thị thông tin của một bệnh nhân. Tuy nhiên, cách hiển thị thông tin có thể khác nhau tùy thuộc vào loại bệnh nhân. Vì vậy, chúng ta sẽ tách biệt phần hiển thị thông tin của bệnh nhân và loại bệnh nhân để giải quyết vấn đề này.

s

public abstract class PatientInformationDisplay {  
 protected Patient patient;  
  
 public PatientInformationDisplay(Patient patient) {  
 this.patient = patient;  
 }  
  
 public abstract void display();  
}

Sau đó, chúng ta có thể triển khai các lớp con của PatientInformationDisplay để hiển thị thông tin bệnh nhân tương ứng. Ví dụ, chúng ta có lớp PatientInformationDisplayWithId để hiển thị thông tin bệnh nhân kèm theo mã số của bệnh nhân:

public class PatientInformationDisplayWithId extends PatientInformationDisplay{  
 public PatientInformationDisplayWithId(Patient patient) {  
 super(patient);  
 }  
  
 @Override  
 public void display() {  
 System.*out*.println("Patient ID: " + patient.getPatientId());  
 System.*out*.println("Patient name: " + patient.getPatientName());  
 }  
}

Chúng ta có thể sử dụng đoạn mã sau để hiển thị thông tin của một bệnh nhân nội trú có mã số 123 và tên là "John":

public class Main {  
 public static void main(String[] args) {  
 Patient patient = new InPatient(123, "John Doe");  
 PatientInformationDisplayWithId displayWithId = new PatientInformationDisplayWithId(patient);  
 displayWithId.display();  
 }  
}

Như vậy, đối tượng displayWithId có thể hiển thị thông tin của một bệnh nhân nội trú hoặc ngoại trú mà không cần thay đổi lớp PatientInformationDisplay.

Mẫu thiết kế Bridge Pattern giúp chúng ta tách biệt các thành phần cấu trúc của một hệ thống, tạo ra sự linh hoạt và dễ dàng mở rộng hệ thống trong tương lai. Nó cũng cho phép chúng ta thay đổi các lớp con của một lớp trừu tượng mà không làm thay đổi các thành phần khác của hệ thống.