Proxy Pattern là một mô hình thiết kế được sử dụng để tạo ra một đối tượng đại diện cho một đối tượng khác. Đối tượng proxy có thể kiểm soát truy cập đến đối tượng thật và thực hiện các hành động bổ sung trước hoặc sau khi truy cập đến đối tượng thật. Proxy Pattern được sử dụng để bảo vệ đối tượng thật khỏi những truy cập trái phép, cải thiện hiệu suất và giảm thiểu sử dụng tài nguyên của hệ thống.

Ví dụ về mô hình bệnh viện:

Trong mô hình bệnh viện, chúng ta có lớp Doctor và lớp Prescription. Lớp Doctor có thể kê đơn cho bệnh nhân. Trong trường hợp này, lớp Prescription được sử dụng để lưu trữ thông tin kê đơn. Tuy nhiên, không phải ai cũng có thể truy cập được đến lớp Prescription, mà chỉ có bác sĩ mới được phép.

Để đảm bảo rằng chỉ có bác sĩ mới có quyền truy cập đến lớp Prescription, chúng ta có thể sử dụng mô hình Proxy. Lớp PrescriptionProxy sẽ được sử dụng để kiểm soát truy cập đến lớp Prescription và chỉ cho phép bác sĩ truy cập vào nó.

Ví dụ:

public class Doctor implements Prescription {  
 private String name;  
  
 public Doctor(String name) {  
 this.name = name;  
 }  
  
 public static boolean checkDoctorAuthorization() {  
 return true;  
 }  
  
 @Override  
 public void writePrescription(Patient patient, Medicine medicine) throws AuthorizationException {  
 Prescription prescription = new PrescriptionProxy();  
 prescription.writePrescription(patient, medicine);  
 }  
}

public interface Prescription {  
 void writePrescription(Patient patient, Medicine medicine) throws AuthorizationException;  
}

public class PrescriptionImpl implements Prescription {  
 private List<Prescription> prescriptionList = new ArrayList<>();  
  
 public PrescriptionImpl(Patient patient, Medicine medicine) {  
 }  
  
 public PrescriptionImpl() {  
 }  
  
 @Override  
 public void writePrescription(Patient patient, Medicine medicine) {  
 prescriptionList.add(new PrescriptionImpl(patient, medicine));  
 }  
}

public class PrescriptionProxy implements Prescription {  
 private Prescription prescription;  
  
 @Override  
 public void writePrescription(Patient patient, Medicine medicine) throws AuthorizationException {  
 if (Doctor.*checkDoctorAuthorization*()) {  
 if (prescription == null) {  
 prescription = new PrescriptionImpl();  
 }  
 prescription.writePrescription(patient, medicine);  
 } else {  
 throw new AuthorizationException("You are not authorized to write prescription.");  
 }  
 }  
}

Trong đó, lớp Doctor được sử dụng để kê đơn cho bệnh nhân. Tuy nhiên, thay vì gọi trực tiếp phương thức writePrescription của lớp PrescriptionImpl, nó sẽ sử dụng lớp PrescriptionProxy để kiểm soát truy cập đến lớp Prescription.

Lớp PrescriptionProxy kiểm tra xem bác sĩ có được phép truy cập vào lớp Prescription hay không thông qua phương thức checkDoctorAuthorization của lớp Doctor. Nếu có quyền truy cập, nó sẽ tạo một đối tượng PrescriptionImpl để thực hiện việc kê đơn. Ngược lại, nếu bác sĩ không được phép truy cập, lớp PrescriptionProxy sẽ ném ra ngoại lệ AuthorizationException.

Ưu điểm của Proxy Pattern:

* Kiểm soát truy cập: Proxy Pattern giúp kiểm soát truy cập đến đối tượng thật và chỉ cho phép những đối tượng được ủy quyền có thể truy cập đến nó.
* Cải thiện hiệu suất: Proxy Pattern giúp cải thiện hiệu suất của hệ thống bằng cách giảm thiểu việc tải các tài nguyên không cần thiết và chỉ tải chúng khi cần thiết.
* Bảo vệ đối tượng thật: Proxy Pattern giúp bảo vệ đối tượng thật khỏi những truy cập trái phép.

Nhược điểm của Proxy Pattern:

* Tăng thêm độ phức tạp cho hệ thống: Việc sử dụng Proxy Pattern có thể làm tăng độ phức tạp cho hệ thống, đặc biệt là khi có nhiều lớp Proxy phải được tạo ra để kiểm soát truy cập đến đối tượng thật.