Command Pattern là một mẫu thiết kế phần mềm thuộc nhóm hành vi (Behavioral Design Pattern) được sử dụng để thực hiện các hành động hoặc yêu cầu của người dùng một cách linh hoạt và dễ dàng mở rộng. Pattern này được sử dụng để giảm sự phụ thuộc giữa các đối tượng và chức năng trong chương trình, đồng thời cung cấp một phương thức tổ chức chức năng và hành động của chương trình.

Trong Command Pattern, một yêu cầu được đóng gói trong một đối tượng riêng biệt gọi là "Command". Đối tượng này chứa tất cả thông tin cần thiết để thực hiện yêu cầu, bao gồm cả phương thức để thực hiện hành động đó. Sau đó, đối tượng "Command" này được truyền cho đối tượng "Invoker" (người kích hoạt) để thực hiện yêu cầu. Invoker có thể biết về các đối tượng Command mà nó nhận được, nhưng không cần biết chi tiết về cách thực hiện hành động. Các đối tượng Command cung cấp một cách tách biệt để định nghĩa các hành động khác nhau mà có thể được thực hiện trong chương trình.

Ưu điểm:

* Command Pattern giảm sự phụ thuộc giữa các đối tượng trong chương trình bằng cách đưa các yêu cầu vào trong đối tượng riêng biệt.
* Nó cho phép mở rộng các hành động của chương trình một cách dễ dàng, do đó làm cho mã chương trình dễ bảo trì và dễ tái sử dụng hơn.
* Nó giúp tách biệt chức năng và hành động của chương trình, từ đó tạo ra một phương thức tổ chức mạnh mẽ để quản lý các hành động của chương trình.

Nhược điểm:

* Việc sử dụng Command Pattern có thể làm tăng độ phức tạp của mã chương trình và tăng số lượng đối tượng trong chương trình.
* Đôi khi, Command Pattern không được sử dụng hiệu quả nếu các yêu cầu của chương trình quá đơn giản.

Ví dụ về mô hình bệnh viện:

Trong một hệ thống quản lý bệnh viện, Command Pattern có thể được sử dụng để thực hiện các yêu cầu khác nhau, chẳng hạn như tạo hồ sơ bệnh nhân

Việc tạo hồ sơ bệnh nhân là một yêu cầu phổ biến trong hệ thống quản lý bệnh viện. Trong Command Pattern, yêu cầu này được đóng gói trong một đối tượng riêng biệt gọi là "CreatePatientCommand". Đối tượng này chứa tất cả thông tin cần thiết để tạo hồ sơ bệnh nhân, bao gồm tên, tuổi, giới tính, địa chỉ, lý do khám bệnh và các thông tin y tế khác. Nó cũng cung cấp phương thức để thực hiện yêu cầu này.

Sau đó, đối tượng "CreatePatientCommand" này được truyền cho đối tượng "Invoker" (người kích hoạt) để thực hiện yêu cầu. Đối tượng Invoker có thể là một đối tượng quản lý bệnh viện hoặc một đối tượng điều hành chung của chương trình. Khi đối tượng Invoker nhận được yêu cầu, nó sẽ gọi phương thức để thực hiện yêu cầu đó.

Đối tượng "CreatePatientCommand" cũng có thể được sử dụng để thực hiện các chức năng khác trong hệ thống quản lý bệnh viện, chẳng hạn như chỉnh sửa thông tin bệnh nhân, xóa bệnh nhân khỏi hệ thống hoặc phát hành đơn thuốc.

Ví dụ :

public class Patient {  
 private String name;  
 private int age;  
 private String gender;  
 private String address;  
 private String reasonForVisit;  
 private String medicalHistory;  
  
 public Patient(String name, int age, String gender, String address, String reasonForVisit, String medicalHistory) {  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 this.address = address;  
 this.reasonForVisit = reasonForVisit;  
 this.medicalHistory = medicalHistory;  
 }  
}

public class Hospital {  
 private static List<Patient> *patients* = new ArrayList<Patient>();  
  
 public static void addPatient(Patient patient) {  
 *patients*.add(patient);  
 }  
  
 public static void createPatient(String name, int age, String gender, String address, String reasonForVisit, String medicalHistory) {  
 Command createPatientCommand = new CreatePatientCommand(name, age, gender, address, reasonForVisit, medicalHistory);  
 createPatientCommand.execute();  
 }  
}

public interface Command {  
 void execute();  
}

public class CreatePatientCommand implements Command {  
 private String name;  
 private int age;  
 private String gender;  
 private String address;  
 private String reasonForVisit;  
 private String medicalHistory;  
  
 public CreatePatientCommand(String name, int age, String gender, String address, String reasonForVisit, String medicalHistory) {  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 this.address = address;  
 this.reasonForVisit = reasonForVisit;  
 this.medicalHistory = medicalHistory;  
 }  
  
 @Override  
 public void execute() {  
 Patient patient = new Patient(name, age, gender, address, reasonForVisit, medicalHistory);  
 Hospital.*addPatient*(patient);  
 }  
}

Trong ví dụ trên, đối tượng "CreatePatientCommand" chứa tất cả thông tin cần thiết để tạo hồ sơ bệnh nhân và phương thức "execute()" để thực hiện yêu cầu. Đối tượng "Hospital" được sử dụng để thực hiện yêu cầu thông qua phương thức "createPatient()". Khi phương thức này được gọi, nó tạo một đối tượng "CreatePatientCommand" và gọi phương thức "execute()" để thực hiện yêu cầu.