Factory Pattern là một mẫu thiết kế phần mềm (design pattern) trong lập trình hướng đối tượng. Nó cho phép chúng ta tạo ra các đối tượng một cách linh hoạt, giúp giảm bớt sự phụ thuộc vào việc tạo đối tượng và giúp cho việc bảo trì mã nguồn dễ dàng hơn.

Trong mô hình Factory Pattern, chúng ta sử dụng một lớp Factory để tạo ra các đối tượng. Lớp Factory này đóng vai trò là một cánh cửa (gateway) để tạo đối tượng, giúp giảm sự phụ thuộc giữa các đối tượng và mã nguồn, và làm cho việc tạo đối tượng trở nên dễ dàng hơn.

Ví dụ về mô hình Factory Pattern trong bệnh viện:

Giả sử chúng ta đang xây dựng một hệ thống quản lý bệnh nhân cho một bệnh viện. Trong hệ thống này, chúng ta có các loại bệnh nhân khác nhau như bệnh nhân nội trú, bệnh nhân ngoại trú, bệnh nhân điều trị tại nhà, và chúng ta muốn tạo các đối tượng bệnh nhân tương ứng.

Bước 1: Định nghĩa các Lớp Bệnh nhân

Chúng ta tạo ra các lớp InPatient, OutPatient và HomePatient kế thừa Lớp Absstract Patient để thể hiện các loại bệnh nhân khác nhau.

public abstract class Patient {  
 protected String name;  
 protected String id;  
  
 public Patient(String name, String id) {  
 this.name = name;  
 this.id = id;  
 }  
  
 public abstract void getInfo();  
}

public class InPatient extends Patient {  
 public InPatient(String name, String id) {  
 super(name, id);  
 }  
  
 @Override  
 public void getInfo() {  
 System.*out*.println("This is an InPatient: " + name + ", ID: " + id);  
 }  
}

public class OutPatient extends Patient {  
 public OutPatient(String name, String id) {  
 super(name, id);  
 }  
  
 @Override  
 public void getInfo() {  
 System.*out*.println("This is an OutPatient: " + name + ", ID: " + id);  
 }  
}

public class HomePatient extends Patient {  
 public HomePatient(String name, String id) {  
 super(name, id);  
 }  
  
 @Override  
 public void getInfo() {  
 System.*out*.println("This is a HomePatient: " + name + ", ID: " + id);  
 }  
}

Bước 2: Định nghĩa lớp Factory

Chúng ta tạo ra một lớp Factory để tạo ra các đối tượng bệnh nhân khác nhau.

public class PatientFactory {  
 public Patient createPatient(String type, String name, String id) {  
 if(type.equalsIgnoreCase("InPatient")) {  
 return new InPatient(name, id);  
 }  
 else if(type.equalsIgnoreCase("OutPatient")) {  
 return new OutPatient(name, id);  
 }  
 else if(type.equalsIgnoreCase("HomePatient")) {  
 return new HomePatient(name, id);  
 }  
 else {  
 return null;  
 }  
 }  
}

Bước 3: Thực hiện tạo các Lớp Bệnh nhân thông qua Lớp Factory

public class Main {  
 public static void main(String[] args) {  
 PatientFactory patientFactory = new PatientFactory();  
  
 Patient inPatient = patientFactory.createPatient("InPatient", "Patient 1", "212323");  
 inPatient.getInfo();  
  
 Patient outPatient = patientFactory.createPatient("outPatient", "Patient 2", "524323");  
 outPatient.getInfo();  
  
 Patient homePatient = patientFactory.createPatient("homePatient", "Patient 3", "4111212");  
 homePatient.getInfo();  
  
 }  
}

**Kết quả:**

