**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---------------o0o---------------**

****

**BÁO CÁO THỰC HÀNH MÔN HỌC ĐẶC TẢ PHẦN MỀM**

**CÀI ĐẶT CÁC MẪU THIẾT KẾ**

Giảng viên hướng dẫn: Trần Văn Dũng

Sinh viên thực hiện: Hà Duy Anh

Lớp: Công nghệ thông tin 3 – K60

Mã sinh viên: 191201864

Hà Nội, tháng 9 năm 2022

**Mục lục**

**[1. Mẫu thiết kế Bride](#_Toc119133937)** [2](#_Toc119133937)

**[1.1 Mô tả bài toán](#_Toc119133938)** [2](#_Toc119133938)

**[1.2 Cài đặt](#_Toc119133939)** [2](#_Toc119133939)

**[1.3 Chạy chương trình](#_Toc119133940)** [4](#_Toc119133940)

**[2. Mẫu thiết kế Builder](#_Toc119133941)** [4](#_Toc119133941)

**[2.1 Mô tả bài toán](#_Toc119133942)** [4](#_Toc119133942)

**[2.2 Cài đặt](#_Toc119133943)** [5](#_Toc119133943)

**[2.3 Chạy chương trình](#_Toc119133944)** [8](#_Toc119133944)

**[3. Mẫu thiết kế Singleton](#_Toc119133945)** [9](#_Toc119133945)

**[3.1 Mô tả bài toán](#_Toc119133946)** [9](#_Toc119133946)

**[3.2 Cài đặt](#_Toc119133947)** [9](#_Toc119133947)

**[3.3 Chạy chương trình](#_Toc119133948)** [10](#_Toc119133948)

**[4. Mẫu thiết kế Adapter](#_Toc119133949)** [11](#_Toc119133949)

**Các mẫu thiết kết**

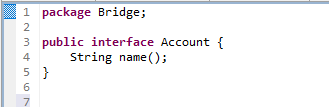
**1. Mẫu thiết kế Bride**

### **1.1 Mô tả bài toán**

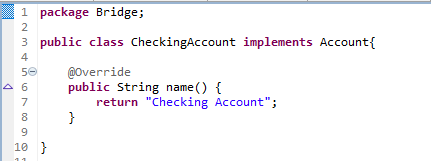
Một hệ thống ngân hàng cung cấp các loại tài khoản khác nhau cho khách hàng, chẳng hạn: Checking account và Saving account. Hệ thống có nhu cầu thêm nhiều ngân hàng và mỗi ngân hàng sẽ có các loại tài khoản khác nhau. Sử dụng mẫu thiết kế Bride sẽ giúp việc thêm các ngân hàng và các loại tài khoản một cách dễ dàng mà không làm thay đổi cấu trúc của nhau.

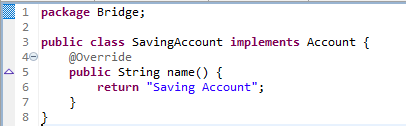
### **1.2 Cài đặt**

* Interface Account:

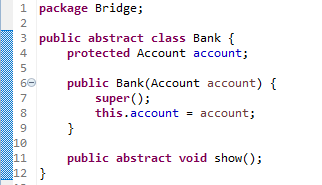


* Class CheckingAccount và SavingAccount được implements từ Interface Account:

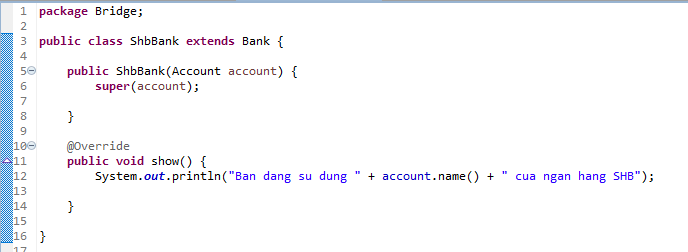


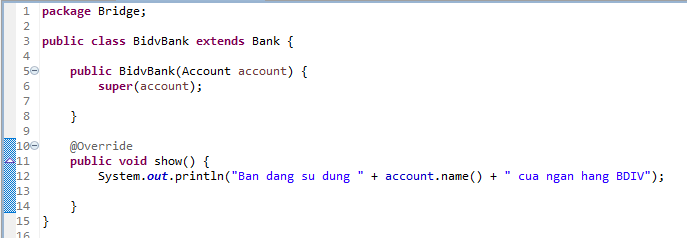


* Abstract class Bank:



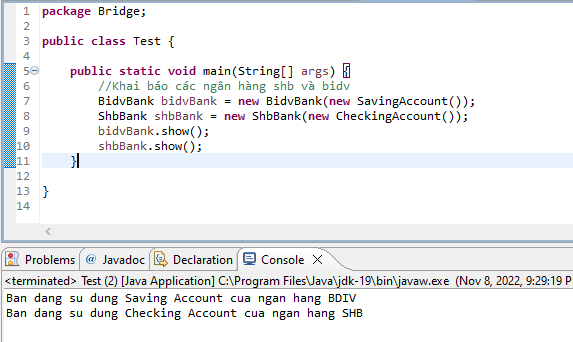
* Class ShbBank và BidvBank được extends từ Abstract class Bank:





### **1.3 Chạy chương trình**

* Kết quả chạy chương trình test:



**2. Mẫu thiết kế Builder**

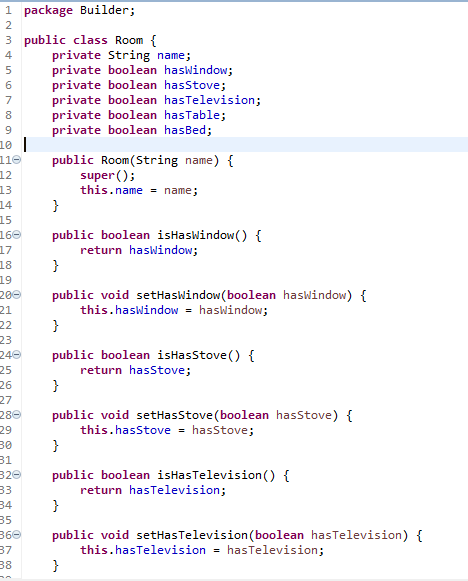
### **2.1 Mô tả bài toán**

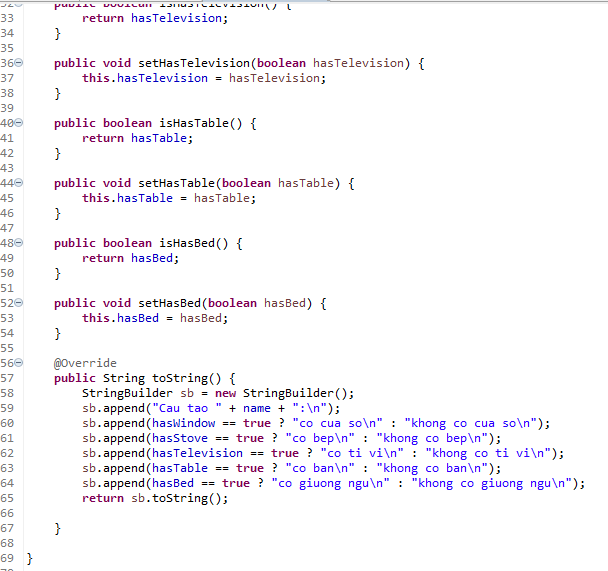
Một hệ thống gồm các đối tượng Room, chúng có chứa rất nhiều thuộc tính.

Mỗi đối tượng Room khác nhau có các thuộc tính với giá trị khác nhau nên việc khởi tạo một đối tượng Room rất phức tạp và khó khăn. Sử dụng mẫu thiết kế Builder sẽ giúp việc khởi tạo các đối tượng Room một cách dễ dàng hơn.

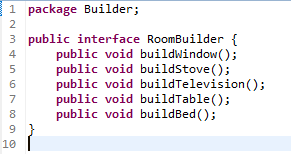
### **2.2 Cài đặt**

* Class Room:

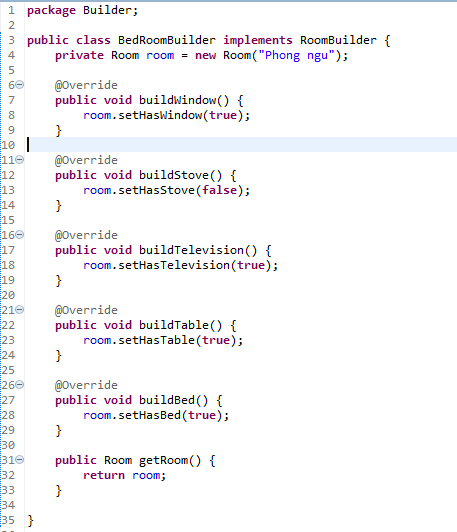


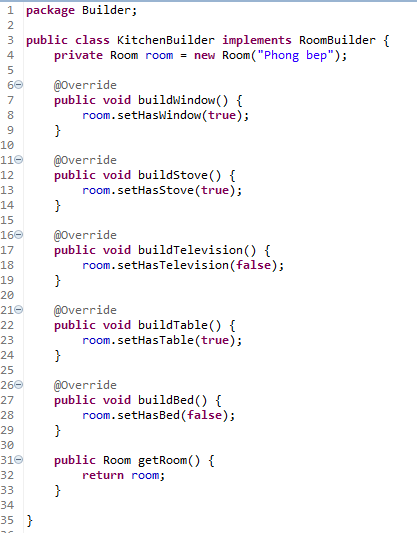


* Interface RoomBuilder:

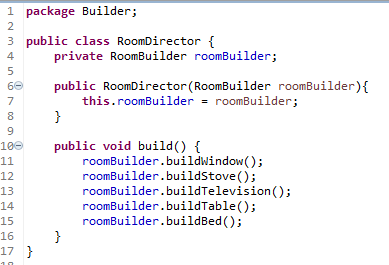


* Class BedRoomBuilder và KitchenBuilder được implements từ Interface RoomBuilder:



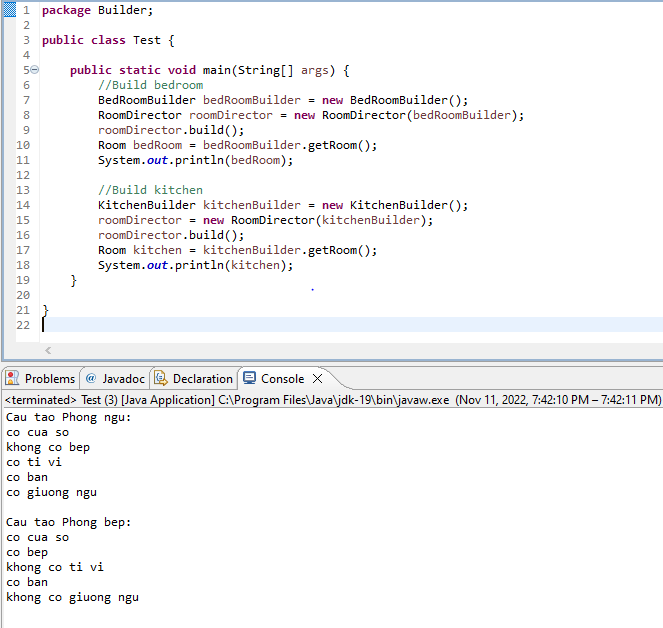


* Class RoomDirector:



### **2.3 Chạy chương trình**

* Kết quả chạy chương trình test:



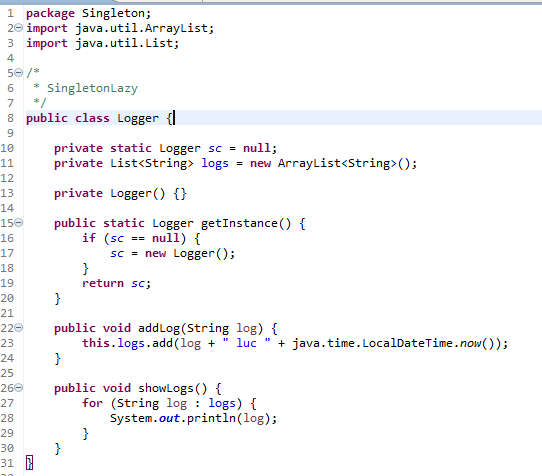
**3. Mẫu thiết kế Singleton**

### **3.1 Mô tả bài toán**

Một hệ thống cần khởi tạo một đối tượng Logger để lưu lại nhật ký hoạt động của hệ thống. Hiện tại, cứ mỗi lần hệ thống sử dụng Logger thì phải tạo một đối tượng tương ứng, điều đó rất tốn kém. Sử dụng mẫu thiết kế Singleton sẽ giúp chúng ta chỉ tạo duy nhất một đối tượng Logger trong suốt quá trình sử dụng, cụ thể sẽ sử dụng Singleton Lazy.

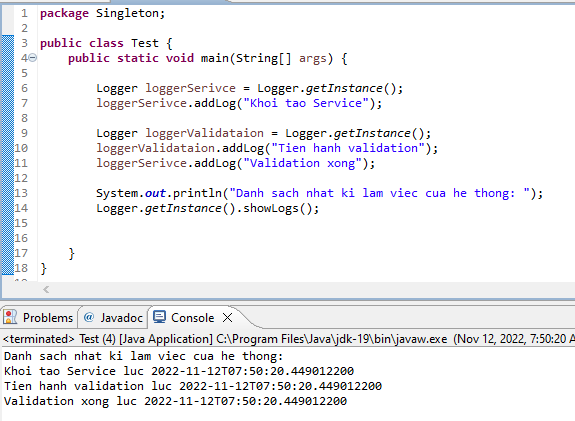
### **3.2 Cài đặt**

* Class Logger:



### **3.3 Chạy chương trình**

* Kết quả chạy chương trình test:



**4. Mẫu thiết kế Vistor**

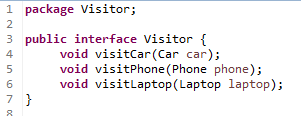
### **4.1 Mô tả bài toán**

Một hệ thống gồm các sản phẩm như: Ô tô, điện thoại, laptop… Hệ thống có nhu cầu tính toán các loại thuế cho các sản phẩm: Thuế thường, thuế ngày lễ…

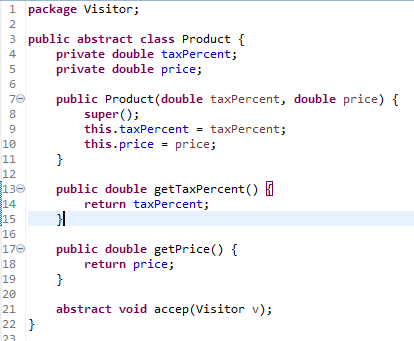
Việc mở rộng hệ thống như bổ sung thêm thao tác tính toán của các loại thuế diễn ra thường xuyên nên hệ thống cần phải xử lí một cách nhanh chóng mà không cần phải cài đặt lại quá nhiều. Sử dụng mẫu thiết kế Visitor sẽ giúp hệ thống dễ dàng trong việc bổ sung các tính toán của các loại thuế khác nhau mà không làm ảnh hưởng đến cài đặt của các sản phẩm.

### **4.2 Cài đặt**

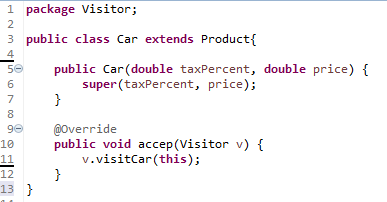
* Interface Visitor:

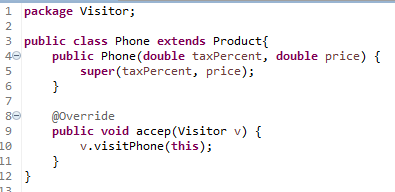


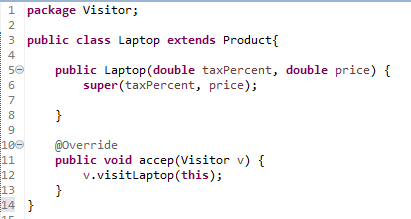
* Abstract Class Product:



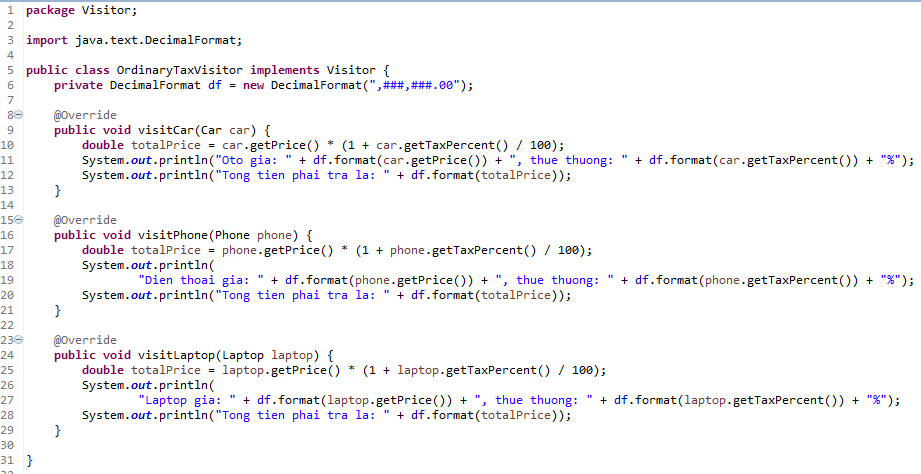
* Class Car, Phone, Laptop được extends từ Abstract Class Product:

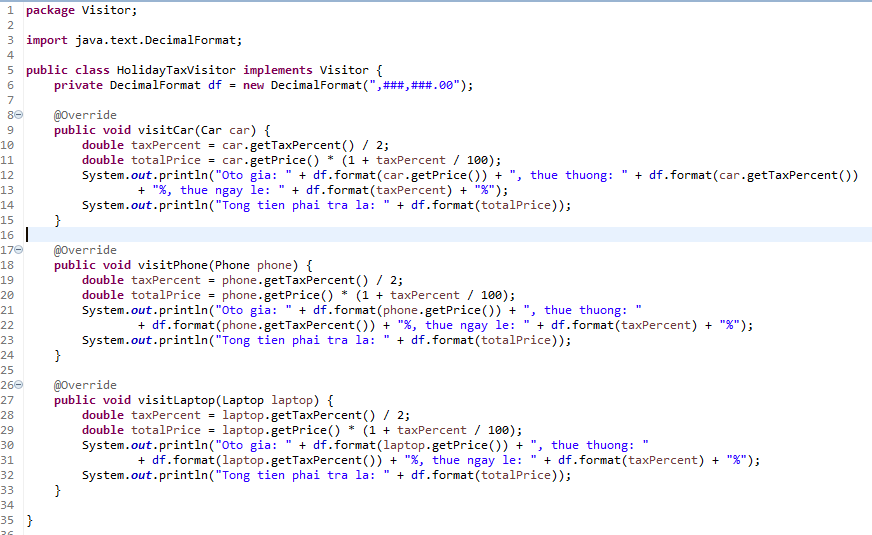






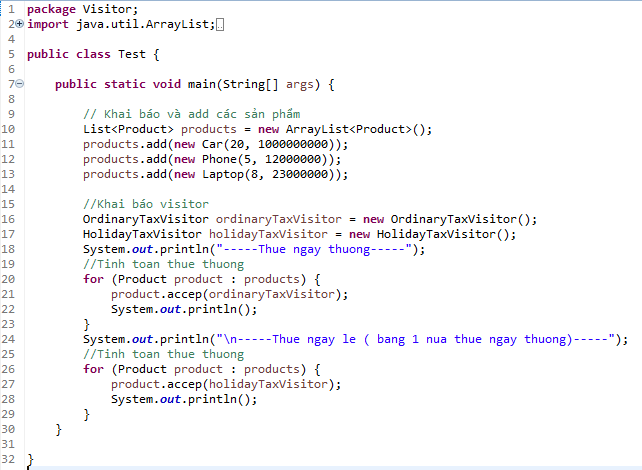
* Class Car, Phone, Laptop được implements từ Interface Visitor:

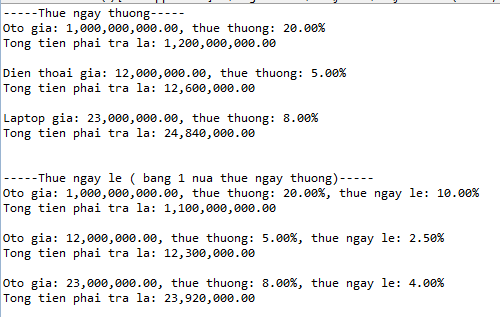




### **4.3 Chạy chương trình**

* Kết quả chạy chương trình test:





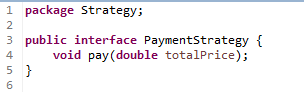
**5. Mẫu thiết kế Strategy**

### **5.1 Mô tả bài toán**

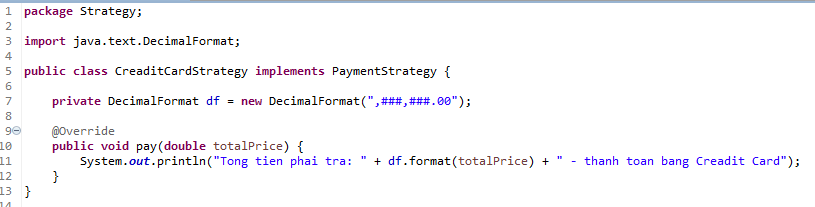
Một hệ thống bán hàng muốn tích hợp nhiều cổng thanh toán trong hệ thống của mình để người sử dụng dễ dàng chọn loại thanh toán mà mình mong muốn. Sử dụng mẫu thiết kế Strategy sẽ giúp hệ thống dễ dàng định nghĩa, đóng gói và dễ dàng thay đổi, sử dụng các cổng thanh toán khác nhau một cách độc lập mà không làm ảnh hưởng nhiều đến hệ thống.

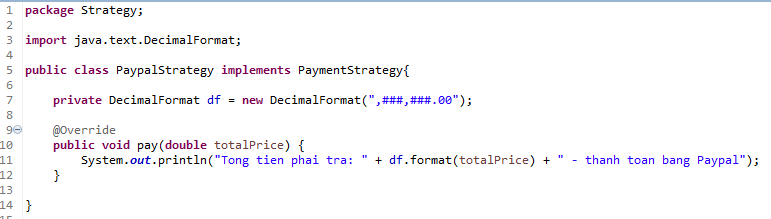
### **5.2 Cài đặt**

* Interface PaymentStrategy:

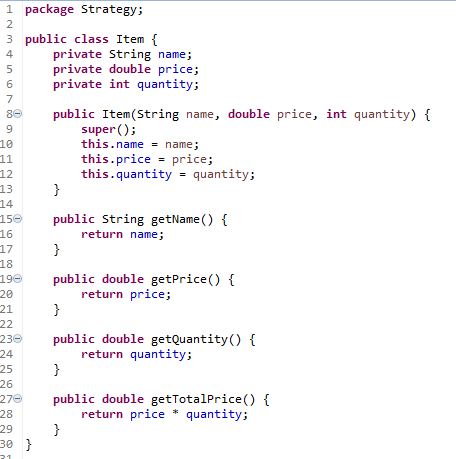


* Class CreaditCardStrategy, PaypalStrategy được implements từ Interface PaymentStrategy:





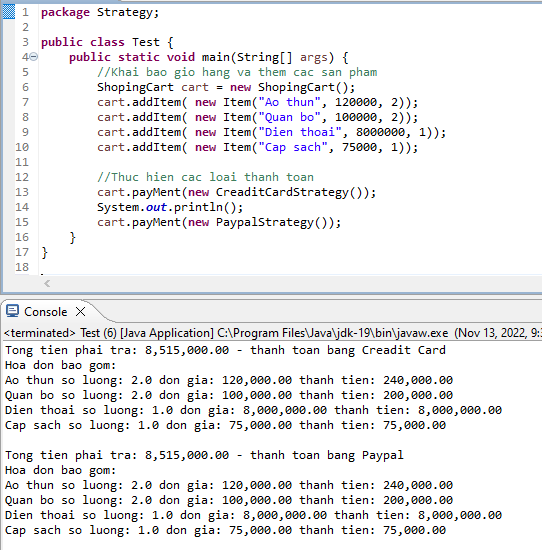
* Class Item, ShopingCart được implements từ Interface PaymentStrategy:





### **5.3 Chạy chương trình**

* Kết quả chạy chương trình test:



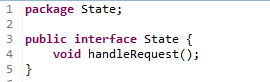
**6. Mẫu thiết kế State**

### **6.1 Mô tả bài toán**

Một hệ thống cần quản lý các trạng thái của một Document. Một Document có thể bao gồm các trạng thái: tạo mới (New), nộp (Submitted) và phê duyệt (Approved)… Sử dụng mẫu thiết kế State sẽ giúp chúng ta thêm, chuyển trạng thái của một Document một cách dễ dàng mà không làm thay đổi cài đặt của Document.

### **6.2 Cài đặt**

* Interface State:



* Class NewState, ApprovedState, SubmittedState được implements từ Interface State: