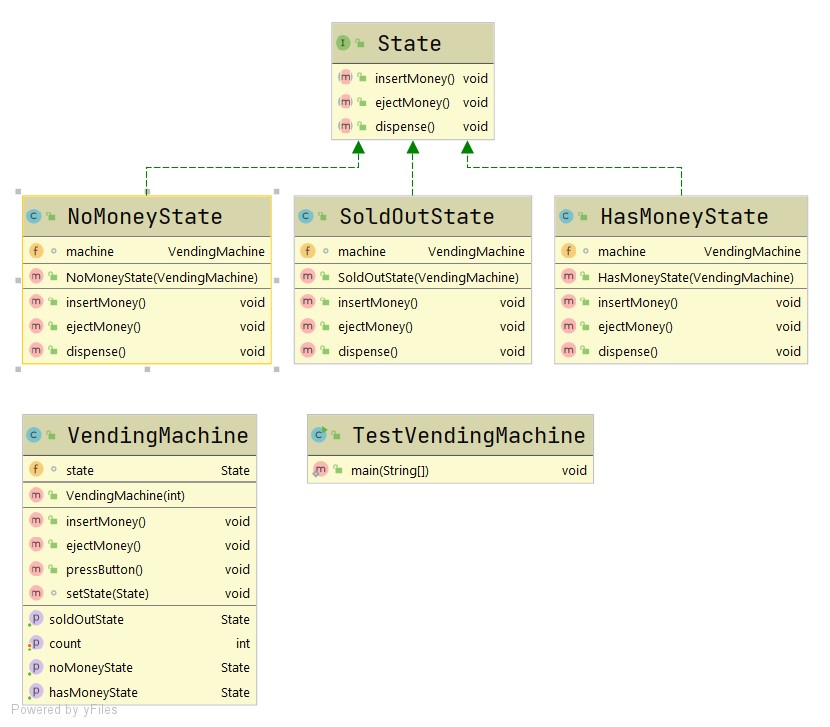
**Mẫu State:**

**Giới thiệu:**

Cho phép một đối tượng thay đổi hành vi khi trạng thái của nó thay đổi. Điều này giúp giảm thiểu sự phức tạp của mã bằng cách tách biệt logic dựa trên trạng thái.

**Sau đây là ví dụ về mẫu State:**



Ví dụ được cung cấp cho mẫu State mô tả cách triển khai một máy bán hàng tự động thay đổi hành vi dựa trên trạng thái hiện tại của nó. Ví dụ này minh họa các nguyên tắc cốt lõi của mẫu State.  
**Cấu trúc:**

1. Interface State:

This interface defines a contract for the vending machine's states.

Nó khai báo ba phương thức:

* InsertMoney(): Mô phỏng việc nhét tiền vào máy bán hàng tự động.
* ejectMoney(): Mô phỏng việc đẩy tiền ra khỏi máy bán hàng tự động.
* phân phối (): Mô phỏng phân phối sản phẩm từ máy bán hàng tự động.

public interface State {

void insertMoney();

void ejectMoney();

void dispense();

}

1. Class HasMoneyState:

Việc triển khai cụ thể giao diện Trạng thái thể hiện trạng thái mà máy bán hàng tự động đã nhét tiền vào.

public class HasMoneyState implements State {

private VendingMachine machine;

public HasMoneyState(VendingMachine machine) {

this.machine = machine;

}

*@Override*

public void insertMoney() {

System.***out***.println("Money already inserted");

}

*@Override*

public void ejectMoney() {

System.***out***.println("Money returned");

machine.setState(machine.getNoMoneyState());

}

*@Override*

public void dispense() {

System.***out***.println("Product dispensed");

machine.setState(machine.getNoMoneyState());

}

}

1. Class NoMoneyState:

Việc triển khai cụ thể giao diện Trạng thái thể hiện trạng thái mà máy bán hàng tự động không có tiền được đưa vào.

public class NoMoneyState implements State {

private VendingMachine machine;

public NoMoneyState(VendingMachine machine) {

this.machine = machine;

}

*@Override*

public void insertMoney() {

System.***out***.println("Money inserted");

machine.setState(machine.getHasMoneyState());

}

*@Override*

public void ejectMoney() {

System.***out***.println("No money to eject");

}

*@Override*

public void dispense() {

System.***out***.println("Payment required");

}

}

1. Class SoldOutState:

Triển khai cụ thể giao diện Trạng thái thể hiện trạng thái máy bán hàng tự động đã bán hết.

public class SoldOutState implements State {

private VendingMachine machine;

public SoldOutState(VendingMachine machine) {

this.machine = machine;

}

*@Override*

public void insertMoney() {

System.***out***.println("Machine sold out");

}

*@Override*

public void ejectMoney() {

System.***out***.println("No money to eject");

}

*@Override*

public void dispense() {

System.***out***.println("Machine sold out");

}

}

1. Class TestVendingMachine:

Một lớp trình điều khiển đơn giản để mô phỏng việc sử dụng máy bán hàng tự động.

Nó tạo ra một đối tượng VendingMachine với một số lượng sản phẩm nhất định.

Nó gọi các phương thức của máy bán hàng tự động (insertMoney và pressButton) để mô phỏng các tương tác của người dùng như nhét tiền và nhấn nút để phân phát sản phẩm.

public class TestVendingMachine {

public static void main(String[] args) {

VendingMachine vendingMachine = new VendingMachine(5);

vendingMachine.insertMoney();

vendingMachine.pressButton();

vendingMachine.insertMoney();

vendingMachine.pressButton();

vendingMachine.insertMoney();

vendingMachine.pressButton();

vendingMachine.insertMoney();

vendingMachine.pressButton();

vendingMachine.insertMoney();

vendingMachine.pressButton();

}

}

1. Class VendingMachine:

Lớp cốt lõi đại diện cho chính máy bán hàng tự động.

Nó duy trì một tham chiếu đến đối tượng trạng thái hiện tại (trạng thái) bằng cách sử dụng mối quan hệ thành phần.

Nó có nhiều đối tượng trạng thái khác nhau (soldOutState, noMoneyState và hasMoneyState) để thể hiện các trạng thái máy bán hàng tự động khác nhau.

Nó cũng duy trì số lượng sản phẩm có sẵn (đếm).

Nó cung cấp các phương pháp.

public class VendingMachine {

private State state;

private State soldOutState;

private State noMoneyState;

private State hasMoneyState;

private int count;

public VendingMachine(int count) {

soldOutState = new SoldOutState(this);

noMoneyState = new NoMoneyState(this);

hasMoneyState = new HasMoneyState(this);

this.count = count;

if (count > 0) {

state = noMoneyState;

} else {

state = soldOutState;

}

}

public void insertMoney() {

state.insertMoney();

}

public void ejectMoney() {

state.ejectMoney();

}

public void pressButton() {

state.dispense();

}

public void setState(State state) {

this.state = state;

}

public State getSoldOutState() {

return soldOutState;

}

public State getNoMoneyState() {

return noMoneyState;

}

public State getHasMoneyState() {

return hasMoneyState;

}

public int getCount() {

return count;

}

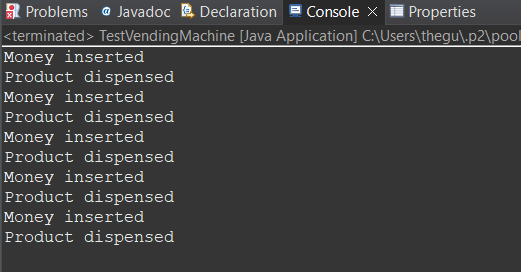
public void setCount(int count) {

this.count = count;

}

}

**Kết quả và thử nghiệm:**



Máy bán hàng tự động:

Trạng thái NoMoneyState: Máy không có tiền, không thể bán hàng.

Trạng thái SoldOutState: Sản phẩm hết hàng, không thể bán hàng.

Trạng thái HasMoneyState: Máy có tiền, có thể bán hàng.

Trong ví dụ này, máy bán hàng tự động sẽ thay đổi trạng thái của nó tùy theo hành động của người dùng như nạp tiền, yêu cầu sản phẩm, hoặc yêu cầu hoàn tiền. Điều này giúp tách biệt logic của các trạng thái khác nhau và giảm sự phức tạp của lớp VendingMachine.

**Lợi ích:**

Nhìn chung, mã được cung cấp triển khai mẫu thiết kế Trạng thái để quản lý hoạt động của máy bán hàng tự động dựa trên trạng thái hiện tại của nó. Máy bán hàng tự động chuyển đổi giữa các trạng thái khác nhau (HasMoneyState, NoMoneyState, SoldOutState) tùy thuộc vào hành động của người dùng (đưa tiền, nhấn nút) và tình trạng sẵn có của sản phẩm.