软件构造第5次实验

实验报告

**班级**：软件工程14-3

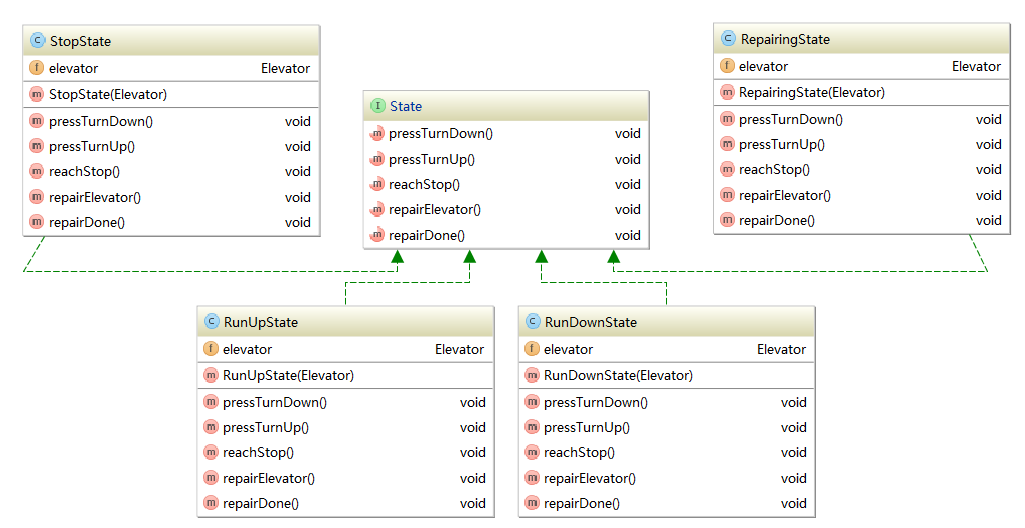
**学号**：201401061038

**姓名**：张正锟

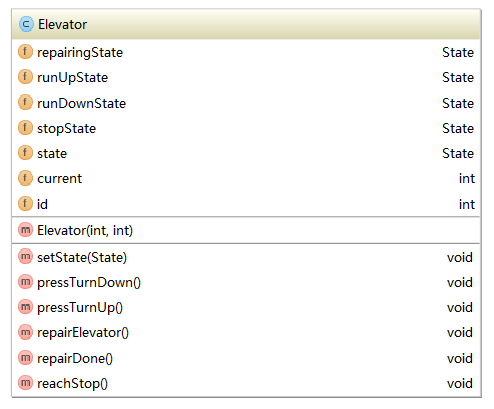
2016年4月25日

# 设计

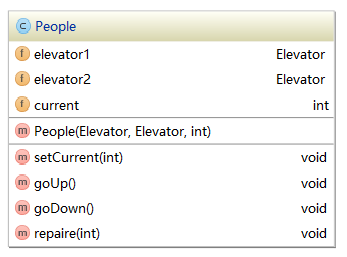
## 电梯状态类继承图



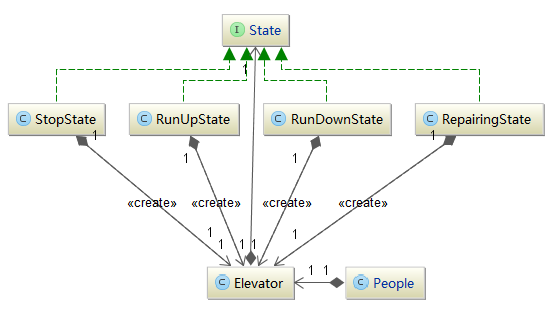
## 电梯类



## 乘客类



## 整体关系类图



# 实现

## 电梯类实现

**public class** Elevator {  
  
 State **repairingState**;  
 State **runUpState**;  
 State **runDownState**;  
 State **stopState**;  
 State **state**;  
  
 **int current** = 0;  
 **int id**;  
  
 **public** Elevator(**int** current, **int** id) {  
 **this**.**current** = current;  
 **this**.**id** = id;  
 **repairingState** = **new** RepairingState(**this**);  
 **stopState** = **new** StopState(**this**);  
 **runUpState** = **new** RunUpState(**this**);  
 **runDownState** = **new** RunDownState(**this**);  
 **state** = **stopState**;  
 }  
  
 **public void** setState(State state) {  
 **this**.**state** = state;  
 }  
  
 **void** pressTurnDown() {  
 **state**.pressTurnDown();  
 }  
 **void** pressTurnUp() {  
 **state**.pressTurnUp();  
 }  
 **void** repairElevator() {  
 **state**.repairElevator();  
 }  
 **void** repairDone() {  
 **state**.repairDone();  
 }  
 **void** reachStop() {  
 **state**.reachStop();  
 }  
}

## 乘客类实现

**public class** People {  
 Elevator **elevator1**, **elevator2**;  
 **int current**;  
  
 **public** People(Elevator elevator1, Elevator elevator2, **int** cur) {  
 **this**.**elevator1** = elevator1;  
 **this**.**elevator2** = elevator2;  
 **this**.**current** = cur;  
 }  
  
 **public void** setCurrent(**int** current) {  
 **this**.**current** = current;  
 }  
  
 **void** goUp() {  
 **if**(**current** == 2) {  
 System.***out***.println(**"您已经在最高层，按钮按不动"**);  
 **return**;  
 }  
 **if**(**elevator1**.**state** == **elevator1**.**stopState** && **elevator1**.**current** == 1) {  
 **elevator1**.pressTurnUp();  
 **current** = 2;  
 } **else if**(**elevator2**.**state** == **elevator2**.**stopState** && **elevator2**.**current** == 1) {  
 **elevator2**.pressTurnUp();  
 **current** = 2;  
 } **else** {  
 System.***out***.println(**"没有可用电梯"**);  
 }  
 }  
  
 **void** goDown() {  
 **if**(**current** == 1) {  
 System.***out***.println(**"您已经在最底层，按钮按不动"**);  
 **return**;  
 }  
 **if**(**elevator1**.**state** == **elevator1**.**stopState** && **elevator1**.**current** == 2) {  
 **elevator1**.pressTurnDown();  
 **current** = 1;  
 } **else if**(**elevator2**.**state** == **elevator2**.**stopState** && **elevator2**.**current** == 2) {  
 **elevator2**.pressTurnDown();  
 **current** = 1;  
 } **else** {  
 System.***out***.println(**"没有可用电梯"**);  
 }  
 }  
  
 **void** repaire(**int** id) {  
 **if**(id == 1) {  
 **elevator1**.repairElevator();  
 **elevator1**.repairDone();  
 }  
 **if**(id == 2) {  
 **elevator2**.repairElevator();  
 **elevator2**.repairDone();  
 }  
 }  
}

## 电梯状态接口实现

**public interface** State {  
 *// 乘客按下上升按钮* **void** pressTurnDown();  
  
 *// 乘客按下下降按钮* **void** pressTurnUp();  
  
 *// 电梯到达目的地，停下来并开门* **void** reachStop();  
  
 *// 检修电梯* **void** repairElevator();  
  
 *// 完成电梯检修* **void** repairDone();  
}

## 电梯维修状态实现

**public class** RepairingState **implements** State {  
  
 Elevator **elevator**;  
  
 **public** RepairingState(Elevator elevator) {  
 **this**.**elevator** = elevator;  
 }  
  
 @Override  
 **public void** pressTurnDown() {  
 System.***out***.printf(**"电梯%d：正在维修中，无法上升\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** pressTurnUp() {  
 System.***out***.printf(**"电梯%d：正在维修中，无法下降\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** reachStop() {  
 System.***out***.printf(**"电梯%d：正在维修中，无法运行\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** repairElevator() {  
 System.***out***.printf(**"电梯%d：正在维修中...\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** repairDone() {  
 **elevator**.**current** = 1;  
 **elevator**.setState(**elevator**.**stopState**);  
 System.***out***.printf(**"电梯%d：电梯维修完毕，可以正常使用\n"**, **elevator**.**id**);  
 }  
}

## 电梯下降状态实现

**public class** RunDownState **implements** State {  
  
 Elevator **elevator**;  
  
 **public** RunDownState(Elevator elevator) {  
 **this**.**elevator** = elevator;  
 }  
  
 @Override  
 **public void** pressTurnDown() {  
 System.***out***.printf(**"电梯%d：正在下降，请耐心等待\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** pressTurnUp() {  
 System.***out***.printf(**"电梯%d：正在下降，无法上升！\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** reachStop() {  
 System.***out***.printf(**"电梯%d：已经到达一楼，门已经打开\n"**, **elevator**.**id**);  
 **elevator**.setState(**elevator**.**stopState**);  
 }  
  
 @Override  
 **public void** repairElevator() {  
 System.***out***.printf(**"电梯%d：正在运行，无法检修\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** repairDone() {  
 System.***out***.printf(**"电梯%d：不是检修状态...\n"**, **elevator**.**id**);  
 }  
}

## 电梯上升状态实现

**public class** RunUpState **implements** State {  
  
 Elevator **elevator**;  
  
 **public** RunUpState(Elevator elevator) {  
 **this**.**elevator** = elevator;  
 }  
  
 @Override  
 **public void** pressTurnDown() {  
 System.***out***.printf(**"电梯%d：正在上升，无法下降！\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** pressTurnUp() {  
 System.***out***.printf(**"电梯%d：正在上升，请耐心等待\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** reachStop() {  
 System.***out***.printf(**"电梯%d：已经到达二楼，门已经打开\n"**, **elevator**.**id**);  
 **elevator**.setState(**elevator**.**stopState**);  
 }  
  
 @Override  
 **public void** repairElevator() {  
 System.out.printf(**"电梯%d：正在运行，无法检修\n"**, elevator.id);  
 }  
  
 @Override  
 **public void** repairDone() {  
 System.out.printf(**"电梯%d：不是检修状态...\n"**, elevator.id);  
 }  
}

## 电梯停止状态实现

**public class** StopState **implements** State {  
  
 Elevator **elevator**;  
  
 **public** StopState(Elevator elevator) {  
 **this**.**elevator** = elevator;  
 }  
  
 @Override  
 **public void** pressTurnDown() {  
 **if**(**elevator**.**current** == 2) {  
 System.***out***.printf(**"电梯%d：正在下降...\n"**, **elevator**.**id**);  
 **elevator**.setState(**elevator**.**runDownState**);  
 **elevator**.reachStop();  
 **elevator**.**current** = 1;  
 } **else if**(**elevator**.**current** == 1) {  
 System.***out***.printf(**"电梯%d：正在一楼，电梯已经开门.\n"**, **elevator**.**id**);  
 }  
 }  
  
 @Override  
 **public void** pressTurnUp() {  
 **if**(**elevator**.**current** == 1) {  
 System.***out***.printf(**"电梯%d：正在上升...\n"**, **elevator**.**id**);  
 **elevator**.setState(**elevator**.**runUpState**);  
 **elevator**.reachStop();  
 **elevator**.**current** = 2;  
 } **else if**(**elevator**.**current** == 2) {  
 System.***out***.printf(**"电梯%d：正在二楼，电梯已经开门.\n"**, **elevator**.**id**);  
 }  
 }  
  
 @Override  
 **public void** reachStop() {  
 System.***out***.printf(**"电梯%d：电梯已经停止.\n"**, **elevator**.**id**);  
 }  
  
 @Override  
 **public void** repairElevator() {  
 **elevator**.setState(**elevator**.**repairingState**);  
 System.***out***.printf(**"电梯%d：电梯正在维修中.\n"**, **elevator**.**id**);  
 System.***out***.println(**"电梯正在维修中"**);  
 }  
  
 @Override  
 **public void** repairDone() {  
 System.***out***.printf(**"电梯%d：抱歉，电梯不是维修状态...\n"**, **elevator**.**id**);  
 }  
}

# 测试

## 测试代码

**public class** ElevatorTest {  
 **public static void** main(String[] args) {  
 Elevator elevator1 = **new** Elevator(1, 1);  
 Elevator elevator2 = **new** Elevator(1, 2);  
 People people1 = **new** People(elevator1, elevator2, 1);  
 People people2 = **new** People(elevator1, elevator2, 1);  
 People people3 = **new** People(elevator1, elevator2, 1);  
 people1.goUp(); people1.goUp();  
 System.***out***.println();  
 people2.goUp(); people2.goDown();  
 System.***out***.println();  
 people3.goUp(); people3.goDown();  
 System.***out***.println();  
  
 people1.repaire(1);  
 people2.repaire(2);  
 }  
}

## 运行结果

电梯1：正在上升...

电梯1：已经到达二楼，门已经打开

您已经在最高层，按钮按不动

电梯2：正在上升...

电梯2：已经到达二楼，门已经打开

电梯1：正在下降...

电梯1：已经到达一楼，门已经打开

电梯1：正在上升...

电梯1：已经到达二楼，门已经打开

电梯1：正在下降...

电梯1：已经到达一楼，门已经打开

电梯1：电梯正在维修中.

电梯正在维修中

电梯1：电梯维修完毕，可以正常使用

电梯2：电梯正在维修中.

电梯正在维修中

电梯2：电梯维修完毕，可以正常使用