JSF Doc

Elevator – ALS (Project 3)

**Classes:**

1. **ALS**

public ALS(Scheduler sched) {

/\*\*

\* @REQUIRES:src (Point), dst (Point), req\_time (double);

\* @MODIFIES:this;

\* @EFFECTS:(\result = \this) &&(\this.isEmpty());

\*/

}

public void pickUp(Queue queue, Elevator elevator) throws CannotRemoveException {

/\*\*

\* @REQUIRES:queue, elevator;

\* @MODIFIES:queue, this.sched.isPickUp;

\* @EFFECTS:

\* queue.size > 0 ==>

\* queue(1).type == "FR" ==>

\* (elevator.floor == queue(1).floor && elevator.dir == queue(1).dir && elevator.time <= queue(1).time) ==> queue.remove(1), \this.sched.isPickUp == true;

\* queue(1).type == "ER" ==>

\* (elevator.floor == queue(1).floor && elevator.time <= queue(1).time) ==> queue.remove(1), \this.sched.isPickUp == true;

\*/

}

public void runElevator() throws Exception {

/\*\*

\* @REQUIRES:None;

\* @MODIFIES:this;

\* @EFFECTS:

\* queue.size > 0 ==>

\* elevator.floor < curReq.floor ==> moveUp;

\* elevator.floor > curReq.floor ==> moveDown;

\* elevator.floor == curReq.floor ==> doorMove;

\*/

}

1. **Elevator**

public void print(Request req) {

/\*\*

\* @REQUIRES: req (Request);

\* @MODIFIES: System.out;

\* @EFFECTS: \req.toString();

\*/

}

public boolean moveUp() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this;

\* @EFFECTS: (\result == true) ==> (\this.curFloor == \old.curFloor+1) && (\this.elevTime == \old.elevTime + \this.moveTime);

\*/

}

public boolean moveDown() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this;

\* @EFFECTS: (\result == true) ==> (\this.curFloor == \old.curFloor-1) && (\this.elevTime == \old.elevTime + \this.moveTime);

\*/

}

public boolean doorMove() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this;

\* @EFFECTS: (\result == true) ==> (\this.direction == "STILL") && (\this.elevTime == \old.elevTime + \this.doorTime);

\*/

}

1. **Floor**

None

1. **Main**

None

1. **Queue**

public boolean addQueue (Request req) throws InvalidRequestException, SameRequestException {

/\*\*

\* @REQUIRES: req (Request);

\* @MODIFIES: this.queue;

\* @EFFECTS:

\* (\result==false)==>(\old(\this).contains(req)==true)

\* (\this.size == \old(\this).size+1) && (\this.contains(req)==true)&&(\result==true);

\*/

}

public boolean removeQueue (int i) throws CannotRemoveException {

/\*\*

\* @REQUIRES: i (int);

\* @MODIFIES: this.queue;

\* @EFFECTS:

\* (this.queue.size == \old(\this.queue).size-1);

\* (\forall int e; e != i; this.queue.contains(e) == \old(this.queue).contains(e));

\* exceptional\_behaviour(CannotRemoveException("Remove index is out of array bounds!"));

\*/

}

1. **Request**

None

1. **Scheduler**

public boolean addReq(Request req) throws InvalidRequestException, SameRequestException {

/\*\*

\* @REQUIRES: req (Request);

\* @MODIFIES: this.queue;

\* @EFFECTS: queue.addQueue(req);

\*/

}

1. **InputHandler**

public void input() throws Exception {

/\*\*

\* @REQUIRES: System.in;

\* @MODIFIES: input;

\* @EFFECTS:

\* input == System.in;

\* input.contains(" ") == false && input.contains("\t") == false;

\*/

}

public void validateInput(String input) {

/\*\*

\* @REQUIRES: input (String);

\* @MODIFIES: m, input;

\* @EFFECTS:

\* input.contains("(") == false && input.contains(")") == false;

\* (m.matches(input) == false ==> exceptional\_behavior(Exception("Invalid Format!")

\* (m.matches(input) == true ==> input.contains(FR) == true ==> parseFloor(input);

\* (m.matches(input) == true ==> input.contains(ER) == true ==> parseElevator(input);

\*/

}

public void parseFloor(String input) throws Exception {

/\*\*

\* @REQUIRES: input (String);

\* @MODIFIES: req\_floor, direction, req\_time;

\* @EFFECTS:

\* req\_floor == temp[1] && direction == temp[2] && req\_time == temp[3];

\* (\old(\this.sched).addReq(req\_floor, direction, req\_time) == false)==>exceptional\_behavior(Exception("Cannot add query!"))

\*/

}

public void parseElevator(String input) throws Exception {

/\*\*

\* @REQUIRES: input (String);

\* @MODIFIES: req\_floor, req\_time;

\* @EFFECTS:

\* req\_floor == temp[1] && req\_time == temp[3];

\* (\old(\this.sched).addReq(req\_floor, direction, req\_time) == false)==>exceptional\_behavior(Exception("Cannot add query!"))

\*/

}