JSF Doc

Elevator – ALS (Project 3)

**Classes:**

1. **Dispatcher**

public void initTaxi(TaxiGUI gui) throws InvalidRequestException, SameRequestException {

/\*\*

\* @REQUIRES: gui;

\* @MODIFIES: this.taxis, this.gui;

\* @EFFECTS:

\* \old(\this.gui) == gui;

\* (\this.taxis.size == \old(\this.taxis).size+100);

\*/

}

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

/\*\*

\* @REQUIRES: req;

\* @MODIFIES: this.queue;

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

\*/

}

public synchronized void getOrder(Request curReq) {

/\*\*

\* @REQUIRES: curReq;

\* @MODIFIES: this.taxis;

\* @EFFECTS:

\* \this.taxis.sort();

\* (\old.taxis.taxi(i).getStatus() == Status.WAITING) ==> (\this.taxis.taxi(i).setStatus() == Status.TAKEORDER);

\* (\old.taxis.taxi(i).getStatus() == Status.TAKEORDER) ==> \this.taxis.taxi(i).start() ==> \this.taxis.remove(i);

\*/

}

1. **Taxi**

public Taxi (int taxinum) {

/\*\*

\* @REQUIRES:taxinum;

\* @MODIFIES:this;

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

\*/

}

public synchronized void addReview(int add) {

/\*\*

\* @REQUIRES: add;

\* @MODIFIES: this.review;

\* @EFFECTS: this.review == \old(this.review) + 1;

\*/

}

public int compareTo(Taxi taxi) {

/\*\*

\* @REQUIRES: taxi;

\* @MODIFIES: taxi;

\* @EFFECTS:

\*/

}

public void moveDown() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: taxiStat, curPos.x;

\* @EFFECTS:

\* \this.curPos.x == \old.curPos.x + 1;

\* \this.taxiStat == Status.SERVING;

\*/

}

public void moveUp() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: taxiStat, curPos.x;

\* @EFFECTS:

\* \this.curPos.x == \old.curPos.x - 1;

\* \this.taxiStat == Status.SERVING;

\*/

}

public void moveRight() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: taxiStat, curPos.y;

\* @EFFECTS:

\* \this.curPos.x == \old.curPos.y + 1;

\* \this.taxiStat == Status.SERVING;

\*/

}

public void moveLeft() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: taxiStat, curPos.y;

\* @EFFECTS:

\* \this.curPos.y == \old.curPos.y - 1;

\* \this.taxiStat == Status.SERVING;

\*/

}

public boolean leftExist() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS:

\* (map[i][j-1] == 1 || map[i][j-1] == 3) ==> \result == true;

\* \result == false;

\*/

}

public boolean rightExist() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS:

\* (map[i][j] == 1 || map[i][j] == 3) ==> \result == true;

\* \result == false;

\*/

}

public boolean upExist() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS:

\* (map[i-1][j] == 2 || map[i-1][j] == 3) ==> \result == true;

\* \result == false;

\*/

}

public boolean downExist() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS:

\* (map[i][j] == 2 || map[i][j] == 3) ==> \result == true;

\* \result == false;

\*/

}

public void getOrder() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS: \this.findRoad(\this.taxi.curPos, \this.curReq.getSrc())

\*/

}

public void moveTaxi() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: None;

\* @EFFECTS: \this.findRoad(\this.taxi.curPos, \this.curReq.getDst())

\*/

}

public boolean addTaxi (int taxinum) {

/\*\*

\* @REQUIRES: taxinum;

\* @MODIFIES: this;

\* @EFFECTS:

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

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

\*/

}

public void sortTaxi() {

/\*\*

\* @REQUIRES: None;

\* @MODIFIES: this.taxis;

\* @EFFECTS: (\all int i, j; 0 <= i < j < taxis.length;  taxis[i] <= taxis[j]);

\*/

}

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**

public Request(Point src, Point dst, double req\_time) {

/\*\*

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

\* @MODIFIES:this;

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

\*/

}

1. **InputHandler**

public void input() throws Exception {

/\*\*

\* @REQUIRES: System.in;

\* @MODIFIES: input;

\* @EFFECTS:

\* input == System.in;

\*/

}

public void validateInput(double time) {

/\*\*

\* @REQUIRES: time (double);

\* @MODIFIES: m, input;

\* @EFFECTS:

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

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

\*/

}

public void parseInput(double time) throws Exception {

/\*\*

\* @REQUIRES: time (double);

\* @MODIFIES: src, dst;

\* @EFFECTS:

\* src.x == temp[1] && src.y == temp[2] && dst.x == temp[3] && dst.y == temp[4];

\* (\old(\this.disp).addReq(src, dst, req\_time) == false)==>exceptional\_behavior(Exception("Cannot add query!"))

\*/

}

1. **OutputHandler**

public synchronized void writeRequest(double time, Point src, Point dest) {

/\*\*

\* @REQUIRES: this.filename != NULL;

\* @MODIFIES: this.filename;

\* @EFFECTS: \this.filename.contains(content);

\*/

}

public synchronized void writeList(int carnum, Point pos, int status, int review) {

/\*\*

\* @REQUIRES: this.filename2 != NULL;

\* @MODIFIES: this.filename2;

\* @EFFECTS: \this.filename2.contains(content);

\*/

}

public synchronized void writeTaken(int carnum, Point pos, double takeorder, double cararrive, Point src, Point dest, double arrivetime) {

/\*\*

\* @REQUIRES: this.filename3 != NULL;

\* @MODIFIES: this.filename3;

\* @EFFECTS: \this.filename3.contains(content);

\*/

}