Design Document

for the ninth Object Oriented homework

**Attention**

The sequence of all classes are arranged by alphabetical order. All classes’ attribute are public except the class MapException.

Content

[Design Document 1](#_Toc450954652)

[一、 ChangeIndex 3](#_Toc450954654)

[二、 Index 3](#_Toc450954655)

[三、 Light\_ctl 4](#_Toc450954656)

[四、 Map 4](#_Toc450954657)

[五、 MapException 7](#_Toc450954658)

[六、 Passenger\_Monitor 8](#_Toc450954659)

[七、 Passenger 8](#_Toc450954660)

[八、 PassengerQuene 10](#_Toc450954661)

[九、 Schedule 10](#_Toc450954662)

[十、 Taxi\_main 11](#_Toc450954663)

[十一、 Taxi 11](#_Toc450954664)

[十二、 Traffic\_light 13](#_Toc450954665)

[十三、 Types 14](#_Toc450954666)

1. ChangeIndex
2. Overview

Record the change of indexes.

1. Process Specifications

public boolean repOK() {

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the true if the rep variant holds for this. otherwise return false.

\*/

(The repOK method in all class have the same specification so only write once here)

public ChangeIndex(Index index, int change)

/\*

\* Requires: Two Index variables.

\* Modifies: Nothing.

\* Effects: Construct a ChangeIndex.

\*/

public Index getIndex()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the index.

\*/

public int getChange()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the change.

\*/

1. Indicated Object

private Index index

private int change

1. Abstract Function

AF(c) = (index, change path), where index = c. index, change path = c. change.

1. Invariance

c. index != null && 0<=change<=3

1. Index
2. Overview

Record the index.

1. Process Specifications

public int getX()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the x.

\*/

public int getY() {

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the y.

\*/

public Index(int x, int y) {

/\*

\* Requires: Two integer.

\* Modifies: Nothing.

\* Effects: Construct a index.

\*/

1. Indicated Object

private final int x

private final int y

1. Abstract Function

AF(c) = (x,y), where x = c. x, y = c. y.

1. Invariance

c. x ∈R && c. y ∈R

1. Light\_ctl
2. Overview

Control all the traffic lights on the simulative road.

1. Process Specifications

public Light\_ctl(Traffic\_light[][] light) {

/\*

\* Requires: two-dimensional array of Traffic\_light.

\* Modifies: Nothing.

\* Effects: Initialize the light..

\*/

1. Indicated Object

private Traffic\_light[][] light

1. Abstract Function

AF(c) = (light), where light = c. light.

1. Invariance

c. light != null

1. Map
2. Overview

Simulate the roads and traffic lights.

1. Process Specifications

public Map()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Initialize the flows, map\_p, changeIndex, map

\*/

public static boolean isConnect(Index a, Index b)

/\*

\* Requires: Two indexes which is border upon.

\* Modifies: Nothing.

\* Effects: check this two indexes whether border upon.

\*/

public static Vector<Passenger> findPasg(int x, int y)

/\*

\* Requires: Two integers which is an index.

\* Modifies: Nothing.

\* Effects: Find all the passengers near the index which passed in then return a Vector contains them.

\*/

public static Vector<Integer> shortestPath(int x1, int y1, int x2, int y2)

/\*

\* Requires: Four integers which are two indexes.

\* Modifies: Nothing.

\* Effects: Find the shortest path of this two indexes.

\*/

public static int shortestPath2(int x1, int y1, int x2, int y2)

/\*

\* Requires: Four integers which are two indexes.

\* Modifies: Nothing.

\* Effects: Find the first step of the shortest and least car flow path of this two indexes and return.

\*/

public static void addReq(int x, int y, Passenger p)

/\*

\* Requires: Two integers which is an index and a passenger.

\* Modifies: Nothing.

\* Effects: Map the passenger into the map\_p.

\*/

public static void deleteReq(int x, int y, Passenger p)

/\*

\* Requires: Two integers which is an index and a passenger.

\* Modifies: Nothing.

\* Effects: delete the passenger in the map\_p.

\*/

private boolean init\_map()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Initialize the map by Map.txt.

\*/

private void init\_lights()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Initialize the light.

\*/

private int countConnect(int i, int j)

/\*

\* Requires: Two integers.

\* Modifies: Nothing.

\* Effects: Get the number of connected path.

\*/

public synchronized static boolean deletePath(Index co, int num)

/\*

\* Requires: An index in map which needs to be changed to the num.

\* Modifies: Nothing.

\* Effects: Delete a path in the map..

\*/

public synchronized static void recoverPath(int i)

/\*

\* Requires: A number which is a index of changeIndex.

\* Modifies: Nothing.

\* Effects: Recover a path in map.

\*/

public static Vector<ChangeIndex> getChanged()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the changeIndex.

\*/

public static void addFlow(int x, int y, int direction)

/\*

\* Requires: An Index and a direction..

\* Modifies: flows.

\* Effects: Add the flow in corresponding edge.

\*/

public static void minusFlow(int x, int y, int direction)

/\*

\* Requires: An Index and a direction..

\* Modifies: flows.

\* Effects: Minus the flow in corresponding edge.

\*/

public static int getFlow(int x, int y, int direction)

/\*

\* Requires: An Index and a direction..

\* Modifies: Nothing.

\* Effects: Return the flow in corresponding edge.

\*/

public static boolean haslight(int x, int y)

/\*

\* Requires: Two integers.

\* Modifies: Nothing.

\* Effects: Return the light[x][y].isHas().

\*/

public static boolean canPass(int x, int y, int di)

/\*

\* Requires: Three integers.

\* Modifies: Nothing.

\* Effects: Return the whether can pass.

\*/

1. Indicated Object

private static final int [][] map

private static final int [][] connect

private static final Traffic\_light[][] light

private static Vector<Passenger>[][] map\_p

private static Vector<ChangeIndex> changeIndex

private static AtomicIntegerArray flows

1. Abstract Function

AF(c) = (map, connect, light, map\_p, changeIndex, flows), where map = c. map, connect = c. connect, light = c. light, map\_p = c. map\_p, changeIndex = c. changeIndex，flows = c. flows.

1. Invariance

c. map != null && c. connect != null && c. light != null && c. map\_p != null && c. changeIndex != null && c. flows != null

1. MapException
2. Overview

An user-defined exception.

1. Process Specifications

public MapException(String msg)

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Nothing

\*/

1. Indicated Object

private static final long serialVersionUID

1. Abstract Function

nothing

1. Invariance

nothing

1. Passenger\_Monitor
2. Overview

Simulate the passenger.

1. Process Specifications

private void addPSG(Index loc, Index des)

/\*

\* Requires: Two Index variables which indicate the passenger location and destination.

\* Modifies: Nothing.

\* Effects: Construct a passenger and then add the passenger request into the passengers.

\*/

public Passenger\_Monitor(Taxi[] taxis)

/\*

\* Requires: An array of Taxi

\* Modifies: this.taxis

\* Effects: set the taxis

\*/

1. Indicated Object

private Taxi [] taxis

1. Abstract Function

AF(c) = (taxis), where taxis = c. taxis.

1. Invariance

c. taxis!= null

1. Passenger
2. Overview

A fake Passenger ☺.

1. Process Specifications

public Passenger(Index location, Index destination)

/\*

\* Requires: Two Indexes.

\* Modifies: Nothing.

\* Effects: Initialize the passenger.

\*/

public boolean addTaix(Taxi taxi)

/\*

\* Requires: A taxi.

\* Modifies: Nothing.

\* Effects: Add the taxi into taxis.

\*/

public Taxi selectTaxi()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Arrange a taxi to serve this passenger.

\*/

public Index getLocation()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the location.

\*/

public Index getDestination()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the destination.

\*/

public String toString()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the passenger's string.

\*/

1. Indicated Object

private Index location

private Index destination

private Vector<Taxi> taxis

1. Abstract Function

AF(c) = (location, destination, taxis), where taxis = c. taxis, destination = c. destination, location = c. location.

1. Invariance

c. taxis!= null && c. location!= null && c. destination!= null

1. PassengerQuene
2. Overview

A container of all the fake passengers.

1. Process Specifications

public static void pushPassenger(Passenger p)

/\*

\* Requires: A passenger.

\* Modifies: Nothing.

\* Effects: if passengers' size less than 400 then add the passenger into passengers.

\*/

public static Passenger pullPassenger()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Push a passenger and return.

\*/

public static int getsize()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the passengers' size now..

\*/

1. Indicated Object

private static Vector<Passenger> passengers

private static int size

1. Abstract Function

AF(c) = (passengers, size), where passengers = c. passengers, size = c. size.

1. Invariance

c. size >= 0

1. Schedule
2. Overview

Schedule the passenger.

1. Process Specifications
2. Indicated Object

private static int i = 0

1. Abstract Function

Nothing.

1. Invariance

c.i >= 0

1. Taxi\_main
2. Overview

Initialize all the classes and make this program running.

1. Process Specifications

public static void main(String[] args)

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Initialize all the classes and make this program running.

\*/

1. Indicated Object
2. Abstract Function
3. Invariance
4. Taxi
5. Overview

Simulate the taxi.

1. Process Specifications

public Taxi(int id)

/\*

\* Requires: Taxi id.

\* Modifies: Nothing.

\* Effects: Initialize a taxi.

\*/

public void setPassenger(Passenger passenger)

/\*

\* Requires: A passenger.

\* Modifies: this.passenger and credit.

\* Effects: Allocate a passenger to this taxi.

\*/

private void runTaxi(int di)

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: run the taxi.

\*/

public int getID()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the taxi'ID.

\*/

public int getCredit()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the taxi' credit.

\*/

public int getState()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the taxi' state.

\*/

public int getNow\_x()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the taxi' x now.

\*/

public int getNow\_y() {

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the taxi' y now.

\*/

public int getTime() {

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the time.

\*/

1. Indicated Object

private int now\_x

private int now\_y

private int state

private int ID

private int credit

private Passenger passenger

private int Direction

private int exDirection

private int time

private int rest\_count

1. Abstract Function

AF(c) = (now\_x, now\_y, state, ID, credit, passenger, Direction, exDirection, time, rest\_count), where now\_x = c. now\_x, now\_y = c. now\_y , state = c. state, ID = c. ID, credit = c. credit, passenger = c. passenger, Direction = c. Direction, exDirection = c. exDirection, time = c. time, rest\_count = c. rest\_count.

1. Invariance

0<=now\_x<80 && 0<=now\_y<80 && 4<= state<=7 && 0<= ID<100 && credit >= 0 && -1 <= Direction<=3 && -1<= exDirection<=3 && time >= 0 && 0<=rest\_count<=200

1. Traffic\_light
2. Overview

Simulate the traffic light.

1. Process Specifications

public Traffic\_light(boolean has)

/\*

\* Requires: A boolean.

\* Modifies: Nothing.

\* Effects: Initialize a traffic light.

\*/

public boolean isHas()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the has.

\*/

public int getL\_r()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the l\_r.

\*/

public int getU\_d()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Return the u\_d.

\*/

public void change()

/\*

\* Requires: Nothing.

\* Modifies: Nothing.

\* Effects: Change the light status..

\*/

1. Indicated Object

private boolean has

private int l\_r

private int u\_d;

1. Abstract Function

AF(c) = (has, l\_r, u\_d), where has = c. has, l\_r = c. l\_r, u\_d = c. u\_d.

1. Invariance

|c.l\_r| == 1 && |c.u\_d| == 1

1. Types
2. Overview

Define all the base types in this project.

1. Process Specifications
2. Indicated Object

public static final int UP = 0

public static final int DOWN = 1

public static final int LEFT = 2

public static final int RIGHT = 3

public static final int size = 80

public static final int WAIT = 4

public static final int GETPSG = 5

public static final int SERVING = 6

public static final int REST = 7

public static final long BASE\_TIME = 100

public static final long CALL\_TIME = 3000

public static final int WAIT\_TIME = 200

public static final int REST\_TIME = 10

1. Abstract Function
2. Invariance