第三单元课程总结

一、规格化设计的优势

1. 在改变一种实现时不会改变任何使用抽象描述的含义。
2. 在不需要检查其他任何抽象的具体实现的情况下，可以阅读与重载这个抽象的实现。
3. 能够实现一个抽象，而不需要更改任何使用该抽象的其他抽象。

二、过程规格

1. 自认为合格的过程规格

a) CityMap 的 turnlight 方法，将地图中的所有路灯的红绿灯反转。

```
public void turnlight() {
        /**
         * @REQUIRES: None
         *
         * @MODIFIES: light
         *
         * @EFFECTS: \all 0 <= i, j < size, light[i][j] == 1 ==>
light[i][j] == 2,
         *           light[i][j] == 2 ==> light[i][j] == 1;
         *
         * @THREAD_REQUIRES:
         *
         * @THREAD_EFFECTS: \locked()
         *
         */
}
```

b) CityMap 类的 isconnect 方法，如果两个点直接相连则返回 true，否则返回 false。

```
public boolean isconnect(Point aPoint, Point bPoint) {
        /**
         * @REQUIRES: aPOint != null, bPoint != null;
         *            aPoint.num <= getnum(Point(79,79)), bPoint.num <=
getnum(Point(79,79));
         * @MODIFIES: None
         *
         * @EFFECTS: 如果 aPoint 与 bPoint 有边直接相连, 返回 true, 否则饭后
false
         * @THREAD_REQUIRES:
         *
         * @THREAD_EFFECTS: \locked()
         */
}
```

c) Taxi 类的 getcredit 方法，返回出租车的信用信息

```
public int getcredit() {
        /**
         * @REQUIRES: None
         *
         * @MODIFIES: None
         *
         * @EFFECTS: \result = credit
         *
         * @THREAD_REQUIRES:
         *
         * @THREAD_EFFECTS: \locked()
         */
}
```

2. 自认写的不好的过程规格

a) CityMap 类的 addflow 方法，增加某两个路口之间的道路流量

```
public void addflow(Point aPoint, Point bPoint) {
        /**
         * @REQUIRES: aPoint != null, bPoint != null;
         *
         *           aPoint.num <= getnum(Point(79,79)), bPoint.num <=
getnum(Point(79,79));
         *
         * @MODIFIES: flow[aPoint.num][bPoint.num]
         *
         * @EFFECTS: flow[aPoint.num][bPoint.num] += 1;
         *
         * @THREAD_REQUIRES:
         *
         * @THREAD_EFFECTS: \locked(flow[aPoint.num][bPoint.num])
         *
         */
}
```

改进方法：Modifies 应该改为：\this

b) CityMap 的 roadclose 方法，关闭某一段道路。

```
public boolean roadclose(Point aPoint, Point bPoint) {
        /**
         * @REQUIRES: sPoint != null, dPoint != null;
         *
         *           sPoint.num <= getnum(Point(79,79)),dPoint.num <=
getnum(Point(79,79));
         *
         * @MODIFIES: map
         *
         * @EFFECTS: if close successfully, return true, otherwise
return false
         *
         * @THREAD_REQUIRES:
         *
         * @THREAD_EFFECTS: \locked()
         *
         */
}
```

改进方法：此函数需要将 CityMap 的 changeflag 改为 true，用于通知出租车道路已更改，所以需要在 EFFECTS 中注明。EFFECTS 应该改为：

```
\result == true ==> \all i, 0 <= i < 100, changeflag[i] == true &&
isconnect(aPoint, bPoint) == false;
\result == false ==> isconnect(aPoint, bPoint) == true;
```

c) InputHandler 类的 getreq 方法，从输入字符串中读取请求信息

```
public static Request getreq(String string, String regex, int size) {
        /**
         * @REQUIRES: regex != null, string != null;
         *
         * @MODIFIES: string
         *
         * @EFFECTS: 从 string 中获得乘客请求并返回，如果输入不合法，返回 null
         */
}
```

改进方法：此处应当采用抛出异常的方法来处理不合法输入。

d) InputHandler 的 getroadreq 方法，获取道路关、打开请求

```
public RoadRequest getroadreq(String strline, String subregex) {
        /**
         * @REQUIRES: subregex != null, strline != null;
         *
         * @MODIFIES: None
         *
         * @EFFECTS: 从 string 中获得道路请求并返回，如果输入不合法，返回 null
         *
         */
}
```

改进方法：此处应当采用抛出异常的方法来处理不合法输入。

e) Inputhandler 的 getmap 方法，从文件中读取地图信息

```
public int[][] getmap(String filepath) {
        /**
         * @REQUIRES: filepath != null && file exist;
         *
         * @MODIFIES: None
         *
         * @EFFECTS: 从文件中获得地图信息并返回，如果输入不合法，返回 null
         *
         */
}
```

改进方法：此处应当采用抛出异常的方法来处理不合法输入。

三、数据规格

1. 自认写的合格的数据规格

a) Taxi 类

```
public class Taxi {
        /**
         * @OVERVIEW: 存储出租车的状态信息，  提供运动方法
         *
         * @RepInvariant: credit > =0 && status >=0 && number >= 0 &&
map.repOK() == true &&
         *                position != null && lastpos != null &&
changeflag != null
         *                && lock != null ==> \result = true;
         */
        protected int credit;
        protected int status;
        protected int number;
        protected CityMap map;
        protected Point position;
        protected Request seReq;
        protected Point lastpos;
        protected ReadWriteLock lock;
        MyFlag changeflag;
}
```

b) TaxiInfo 类

```
public class TaxiInfo {
        /**
         * @OVERVIEW: taxi infomation for each served request
         *
         * @RepInvariant: request != null && position != null &&
pickPath != null &&
         *                servePath != null ==> \result == true;
         */
        protected Request request;
        protected Point position;
        protected Vector<Point> pickPath;
        protected Vector<Point> servePath;
}
```

c) Reqlist 类

```
public class Reqlist {
        /**
         * @OVERVIEW: Thread safe list of requests
         *
         * @Repinvariant: requests != null ==> \result = true,
otherwise, \result = false;
         *
         */
        private Vector<Request> requests;
        private int size;
}
```

2. 自认写的不好的数据规格

我认为我在数据规格的书写上没有大的问题，但在数据抽象方面没有做好，下面列举一下数据抽象没有做好的例子。另外，本次作业的中“有血有肉”的类数量较少，因此没有找到 5 个例子。

a) CityMap 类

```
public class CityMap {
        /**
         * @OVERVIEW: 存储地图信息、道路流量信息、道路红绿灯、红绿灯持续时间
等信息，提供寻找最短路劲、查询红绿灯、流量增减、道路开闭等功能函数。
         *
         * @RepInvariant:(\all 0 <= i, j < size, map[i][j] == 0 ||
map[i][j] == 1 || map[i][j] == 2 || map[i][j] == 3) &&
         *                    (\all 0 <= i, j < size*size, flow[i][j] >=
0) &&
         *                    (\all 0 <= i, j < size, light[i][j] >= 0)
&& (size > 0) && (flags != null) &&
         *                    (\all 0 <= i < 100, flags[i] != null) &&
(gui != null) && (lock != null) ==>\result == true;
         */
        int[][] map;
        Integer[][] flow;
        Integer[][] light;
        int size;
        int lasttime;
        MyFlag[] flags;
        ReadWriteLock lock;
        TaxiGUI gui;
}
```

改进方法：存储了多余的本不应该存储的属性，如 gui、流量信息等，应当去掉；属性均为 package 可见，封装没有做好。

b) RoadRequest 类

```
/**
 * @OVERVIEW: road open or close road request.
 *
 * @RepInvariant: Request.repOK() == true && act != null ==> \result ==
true
 */
class RoadRequest extends Request {
        String act;
}
```

改进方法：道路请求类和乘客请求类有很大，直接继承乘客请求类不合适。另外，即使按照现有的代码，RepInvariant 也应该改为：

```
Request.repOK() == true && (act == “OPEN” || act == “CLOSE”) ==> \result
== true
```

四、Bug 与过程规格质量的关系

1. 打开道路指令北城区读取为关闭指令。
   原因是输入处理有一点疏忽，忘记了字符串是不可变类型，没有将 replace 后的字符串赋给原来的字符串变量。与程序规格没有关系。

另外，因为最近三次作业都是在第七次作业的基础上修改的，除非每次都重构代码，否则每次作业都是先有代码后写规格，这样的话很难说程序 bug 和规格的质量有关系吧？