OO 第十一次作业指导书

一、 作业背景和目标

本次作业在第十次作业基础上，进一步对出租车服务和监控的功能进行扩充，并进一步训练类规格、过程规格的规范书写和对LSP（Liskov Substitution Principle）原则的理解与应用。

二、 核心概念定义

1. 城市地图

1) 使用网格区域来模拟城市地图。所有的道路要么是水平方向，要么是垂直方向，如果两个点之间有道路，则这两个点之间存在一条连接。

2) 城市地图通过ASCII编码的文本文件输入。文件内容为80行字符串，每行有80个数字字符（字符之间允许出现空格或制表符），每个数字字符为 0 到 3 之间的整数，表示一个 80×80 的邻接矩阵 $A_{80\times80}$。输入文件中除数字0,1,2,3，空格，制表符和回车换行外，出现任何其他字符都可判定为无效输入。如果输入少于或多于80行，或者存在某行输入少于或多于80个0~3之间的数字，则可判定为无效输入。

   输入第 i 行的第 j 个数字 $A_{i,j}$ 记录的是地图中 (i,j) 坐标位置的点到与右方坐标 $(i,j+1)$ 的点和下方坐标 $(i+1,j)$ 的点的连接情况，若

   a) $A_{i,} = \mathbf{0}$ 表示 (i,j) 与 $(i,j+1)$ 和 $(i+1,j)$ 均无连接。

   b) $A_{i,} = \mathbf{1}$ 表示 (i,j) 与 $(\boldsymbol{i,j+1})$ 有连接，但与 $(i+1,j)$ 无连接。

   c) $A_{i,} = \mathbf{2}$ 表示 (i,j) 与 $(i,j+1)$ 无连接，但与 $(\boldsymbol{i+1,j})$ 有连接。

   d) $A_{i,} = \mathbf{3}$ 表示 (i,j) 与 $(\boldsymbol{i,j+1})$ 和 $(\boldsymbol{i+1,j})$ 均有连接。

   提示： (i,j) 与 $(i-1,j)$ 和 $(i,j-1)$ 的连接情况由 $A_{i-1,j}$ 和 $A_{i,j-1}$ 定义。所有的连接均为双向。只有水平和垂直连接，没有对角线连接。

3) 文件需要确保地图上的所有的点都连通，即整个图是连通图，且不能存在点与图外的点有连接，例如 $A_{80,1}$ 不能是 2 或 3，$A_{80,80}$ 只能是 0。

4) 支持在系统运行过程中动态关闭或打开一些地图上已有的连接边（道路临时关闭/打开功能），为测试方便，一个时间窗口内改变的连接边不超过5。任意时刻需要由测试者保地图的连通性，并且打开关闭操作只能在原地图中联通的边上执行。

5) 对于地图上打开的连接边，定义该变道路的流量为单位时间窗内从该边经过的出租车数（不论出租车通过时处于那种状态）。为简单起见，时间窗设置为200ms。

6) 交叉路口（包括十字交叉和丁字交叉）有红绿灯（为简化问题，不设置黄灯，只有红、绿两种灯）控制。针对全部交叉路口，要求不少于30%的路口有红绿灯。对于未添加红绿灯控制的路口，出租车行驶规则保持不变。红绿灯控制的路口信息通过一个独立文件输入，文件内容为80行字符串，每行有80个字符，每个字符为0或1（字符之间允许出现空格或制表符）。0表示无红绿灯控制，1表示有红绿灯控制。

7) 所有路口的红绿灯变化间隔在初始化时随机确定，为200ms~500ms（注意第十次作业是50ms~200ms，间隔短导致测试难以观察，故本次作业进行了调整）之间的一个数值。每个路口有两组灯，南北方向为一组，东西方向为另一组。同组灯的颜色始终相同，不同组之间颜色始终相异。初始化时随机决定南北或东西方向的灯为绿色，然后每经过变化间隔时间，两个方向的灯同时变换颜色。

2. 出租车

1) ［新增］出租车分为两类，其中一类为前几次作业所述的普通出租车；本次作业新增一类特殊的出租车，称之为可追踪出租车。可追踪出租车具有以下特性：

   a) 能够追踪出租车从程序启动运行以来的乘客服务情况，包括该车所抢到的乘客请求（请求产生时刻、请求发出位置、目的地位置），出租车在抢到单时的所处位置，出租车去接乘客以及运送乘客去目的地途中所行驶的路径。要求按照每次乘客服务进行信息组织和管理。

   b) 扩展普通出租车的路径选择方法，使得可追踪出租车能够行走关闭的道路（普通出租车则不可以）。注：关闭的道路的车流为0，且保持不变。

2) 出租车行驶一条格子边的时间为200ms。

3) 每辆出租车的状态有四种：停止运行、服务（在运行且车内有乘客）、等待服务（在运行但没有抢到单）、**准备服务**（在运行，车内无乘客但已抢到单，去接乘客的过程状态）。

4) 出租车的状态有以下转换规则：

a) 出租车处于等待服务状态持续20s后，停止运行1s，然后再次运行立即进入等待服务状态。

b) 出租车在抢到用户请求后，立即进入准备服务状态，并立刻去往乘客所在地址提供服务；

c) 处于准备服务状态的出租车一旦到达用户等待位置，停止运行1s，然后再次运行立即进入服务状态。

d) 出租车完成当前服务（到达用户的目的地）后，停止运行 1s，然后再次运行立即进入等待服务状态。

e) 任何情况下，只要是停止运行，都处于停止运行状态。

5) 在停止运行、准备服务和服务状态下不能响应新的乘客请求。

6) 出租车的行走方式有以下两种：

a) 在等待服务状态时，出租车如果遇到道路分支，选择当时流量最小的一条边行走，如果有多条流量最小的边，可随机选择一条分支边行走。

b) 在准备服务和服务状态时，要求出租车按照最短路径行走，如果有多条最短路径可走，则选择流量最小的路径行走，如果仍有多条流量最小的路径，可任意选择一条行走。一旦做出决定，在行驶过程中不再根据流量对路径进行调整。

7) 出租车有信用积累，初始所有车信用为 0，每抢单一次会使其信用度加 1，每成功服务顾客一次会使其信用度加 3。

8) 针对有红绿灯控制的路口，要求出租车必须遵守如下图所示和相应文字解释的行驶规则（不存在现实中的违规可能性）。图中假设此时南北方向为绿灯，东西方向为红灯，绿色表示可行驶路线，红色表示禁止行驶路线。其中由东西到南北的左转弯路线（如图中的5）与南北直行路线（如图中的4）出现交叉，属于可接受情况（因为我们没有专门设置左转红绿灯控制）。

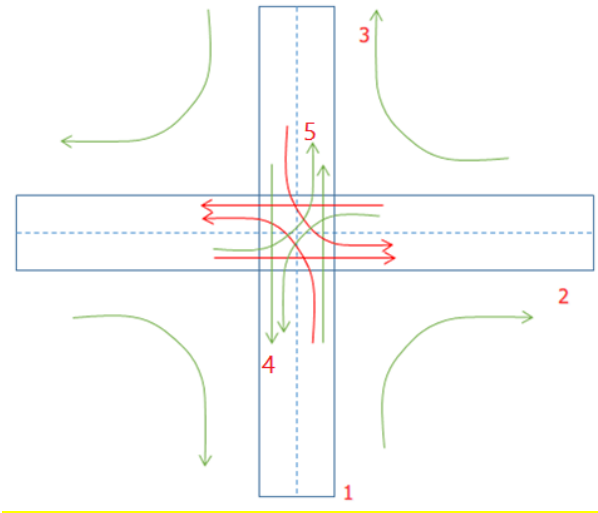

图1：红绿灯控制路口的出租车行驶规则示意图

a) 当出租车车头指向的灯为绿灯时，可直行；

b) 当出租车车头指向的灯为红灯时，可左转；

c) 不论出租车车头指向的灯为何颜色，均可右转时（如图中的四条右转弯线），同时不论红绿灯为何颜色，都可以掉头；

d) 处于路口等待的出租车，不存在故意延误时间不走的情形。

提示：出租车在搜索最短路径时依然采用之前的策略，无需考虑红绿灯变化带来的未来可能影响。出租车等红灯所消耗时间不纳入之前所定义的出租车状态转换中相关时间和一条边的行驶时间。举例来说，之前出租车行驶一条边为 200ms，此时正好在这条边的一个端点处（假设正好是红绿灯控制的交叉路口）等红灯消耗了相应变换间隔时间（如 80ms），则出租车实际行驶这条边所花时间为 280ms。

3. 乘客请求及响应

1) 乘客在任意一点 C 向系统发出呼叫请求后，系统只把呼叫请求发送给在以 C 为中心的 4×4 区域（上下左右均延伸 2 个格子，但不超出 80×80 的大范围）里行驶的出租车。呼叫请求包括点C的坐标和目的地坐标信息，如果目的地坐标无效，则系统拒绝响应该请求。如果请求中点C的坐标与目的地一样，则视为无效请求，不予响应。

2) 系统一旦收到乘客请求，会设置一个时间长度为3s的抢单时间窗口，在窗口内系统向符合条件的出租车广播发送乘客请求，在抢单时间窗口关闭时，系统在抢单的出租车中，按照第5）条规则来选择相应出租车响应乘客请求。

3) 如果 3s 内无出租车对系统应答则视为无车响应，系统告知乘客无可用出租车，系统对该乘客请求的处理结束。

4) 处于等待服务状态的出租车只要在乘客请求的抢单时间窗口内，一旦进入以请求发出地为中心的 4 × 4 网格区域就会收到请求。出租车只要收到请求就会抢单。

5) 如果有车响应，系统为乘客从当前时间窗口中抢单的出租车中自动进行选择，在抢单时间窗口关闭时刻选择处于等待服务中信用度最高的出租车；如果有多辆信用度相同的出租车，则选择当前距离用户请求出发地最近的；如果仍有多辆满足条件的出租车，则从中随机选择一辆。

6) 出租车一旦抢单，系统就会记录，即使出租车在抢单时间窗口关闭时已经离开了以请求发出地为中心的 4 × 4 区域也视为有效。

7) 一辆出租车在同一时刻可以抢不止一单，但每辆车一次服务只能够响应一个乘客请求。

8) 抢单后到抢到单之间这个时间段，出租车仍然处于等待服务状态。

9) 同一时刻在同一地点发出的去同一目的地的请求视为相同请求，相同请求只处理一个和一次。

10) 对于乘客请求的响应，两种类型出租车没有区别。

三、 设计要求

1. 对设计者的要求

1) 要求对所有方法按照所提供的JSF规范指南来书写过程规格和类规格。务必确保规格和代码实现的一致性。

2) [新增]要求修复之前被报告的bug，并在文档中加以说明。

3) [新增]完善相应的类规格设计和设计文档

4) [新增]要求在设计文档中就新增的可追踪出租车类论证其如何满足LSP原则

5) 要求使用多线程和线程安全设计。提供线程安全的乘客请求队列，供测试使用。注意请求队列容量不得小于300个。

6) 程序可通过控制台来获得乘客请求，乘客请求格式为[CR,src,dst]，其中CR为标识符，src和dst均为(i,j)形式的坐标位置,表示乘客请求的发出地和目的

地。任何超出地图范围的坐标都视为无效请求而被直接忽略，不影响对其他有效请求的处理。乘客请求的产生时间自动从系统获得。系统的基本时间单位100ms。

7) 要求程序把对乘客请求的处理过程输出到文件中，作为测试判断的依据。针对每个乘客请求，需要记录的数据包括：请求发出时，处于以请求src为中心的4×4区域中的所有出租车状态、信用信息；在抢单时间窗内所有抢单的出租车；系统选择响应相应请求的出租车；出租车响应相应请求过程中的实际行驶路径。

8) 要求使用继承机制来设计实现新的出租车类型。

9) 要求新类型的出租车提供双向迭代器来访问其乘客服务历史。

10) 要求出租车调度、管理等用户类不能够直接在代码中使用新出租车类型。要求设计者提供一个空的方法（如init_taxi方法），并在readme中介绍。该方法的功能是返回初始化创建的100个出租车对象。设计者应严格定义该方法的规格，测试者保证按照规格来实现这个方法，可按自己设想的顺序来构造70个普通出租车和30个可追踪出租车。出租车对象的创建必须调用设计者提供的相应构造方法。

11) 要求提供测试接口来查询出指定出租车的状态（这里的测试接口不是 JAVA 里面的接口），至少能够在任意时刻给出指定出租车的状态信息，信息包括但不限于当前的时刻，出租车当前坐标，以及在任意时刻返回处于指定状态的出租车对象（如服务状态）。

12) 要求以可交互的方式在程序运行中提供道路临时关闭或打开功能（在readme中具体说明交互方式），从而便于测试。

13) 要求使用一个线程类来模拟交叉路口的红绿灯控制。

14) 要求每个类都实现repOK方法，且必须与不变式逻辑一致。

15) 本次作业提供一个GUI程序包，该程序包提供相应接口来展示地图和出租车的位置和运动，通过调用GUI程序包可在测试时直接在GUI界面上进行判断，便于进行测试判断。但是在发现bug时要求提供输出的文件内容作为证据。（GUI的使用说明另行提供）

16) 要求提供测试者能够完成测试要求的相关接口。

17) 要求写清楚为测试者提供的测试接口功能说明。

2. 对测试者的要求

1) 对照JSF规范指南检查过程规格和类规格书写的规范性，每发现一个问题报告一个incomplete类型bug，但一个方法规格最多只能报告一个问题。在报告问题时务必准确提供不符合规范的规格内容。

2) 检查规格描述与代码实现之间的一致性（包括repOK方法是否与不变式逻辑一致）。发现不一致，报告一个incompelete类型bug，但一个方法规格和类规格最多只能报告一个问题。在报告问题时务必准确提供不一致的规格内容和相应代码片段。

3) 检查设计者提供的测试接口是否满足了实验的功能要求和测试要求，如果有遗漏，则记为功能性缺失，但同一个原则不重复扣分。

4) 编写测试线程，向请求队列发送请求来模拟乘客呼叫出租车，并通过访问相关出租车对象的状态自动判断程序处理是否正确。

5) 要求针对每个类都进行不变式满足情况的检查测试：每个方法被至少调用一次，在满足规格要求的前提下，测试不受限制。以方法为单位，只要方法执行后导致后置条件不满足或者所在对象的不变式不满足，记为一个wrong类型的bug。

6) 提交bug时，需要提供测试使用的地图文件、红绿灯文件、相应的测试代码，以及记录乘客请求-响应过程的文件。其中乘客请求-响应过程的文件至少包括这样一组信息：

   a) 乘客请求内容：时刻、请求位置、目的地位置

   b) 4*4请求区域内的车辆信息：车辆编号、车辆位置、车辆状态、车辆信用

   c) 响应请求的车辆运行信息：车辆ID、分派任务时的车辆位置坐标、时刻、达到乘客位置的时刻、位置坐标、到达目的地坐标、时刻、关键途经点（运行方向改变的位置点）的坐标和时刻。

7) 建议以类为单位编写测试代码，按照不变式和方法规格后置条件来组合设计测试用例。

   测试时注意检查类实现是否满足类规格，并通过可执行的测试用例（针对类的测试用例）来报告相应的bug。

3. 建议和说明

1) 由于100个车辆在6400网格内不停的变化，通过人工测试和检查是一个非常艰

苦和耗时的工作，因此建议并鼓励通过写程序的方式进行测试。

2) 作为具有说服力的测试参考，应将各种状态结果输出到文件中，作为运行正确性评判的最重要依据。

四、 关于GUI程序包的使用说明

本次作业会提供一个GUI包，为大家调试多线程提供和检查结果方便。GUI表现出来的问题不作为程序设计的问题进行评定。

1) GUI提供了一个TaxiGUI类，有关地图，出租车以及请求可以通过可视化方式进行展示

2) 在使用之前需要对可视化对象进行地图初始化，使用boolean LoadMap(int[][] map, int size)将地图载入可视化对象，要求输入为一个2维数组，大小为size（本次作业应该是80），地图坐标左上角为坐标原点(0,0),该地图是否有效由调用者负责。

3) 在使用之前需要将出租车进行初始化，每一个出租车需要通过调用void SetTaxiStatus(int index, Point point, int status)来改变出租车的状态，index取值范围[0,99]，Point为出租车坐标值，status取值为0,1,2，其中0-停止运行；1-服务（在运行且车内有乘客）；2-等待服务（在运行但车内无乘客），3-准备服务。

4) 在出租车运行过程中，每一次出租车移动一步后仍然需要调用void SetTaxiStatus(int index, Point point, int status)来改变出租车的状态。

5) 系统收到出租车请求，也需要将该请求输入可视化对象，通过调用void RequestTaxi(Point src, Point dst)来输入，src，dst分别为乘客提出出租车请求的起始坐标和目的坐标。

6) 在断开/联通邻近的两条道路时，调用void SetRoadStatus(Point p1, Point p2, int status)来进行设置，其中status=0:断开，1：联通。由开发者保证该函数调用的逻辑正确性（如两个点应为邻接点，不能是同一个点等等）。

7) 针对路口红绿灯，调用void SetLightStatus(Point p, int Status)来进行设置，其中Status取值：0 没有红绿灯 1 东西方向为绿灯（南北向红灯） 2 东西方向为红灯（南北向绿灯）。

8) 对每一个出租车，调用void SetTaxiType(int index, int type)来进行类型

设置，index为出租车编号，type为出租车类型，0是普通出租车，1为可追踪出租车(在GUI上该类型出租车显示为粉色)。

9) GUI不是测试内容，任何GUI层次的问题都不作为测试内容项且无需报告bug。