基本需求分析：

1. 地图
   1. 大小80\*80，向边界外不可有路，（测试者需保证为连通图），
   2. 4种类型：

0（右和下均无路）1（右有路下无路）2（下有路右无路）3（下和右均有路）

1. 出租车
   1. 一共100辆，初始化时随机分布于地图
   2. 每200ms改变状态到达下一个点，等待状态下遇到道路分支，会随机选一条道路走（包括掉头），
   3. 4种状态：
      1. 等待 (2)：可接单（包括抢单时间）
      2. 停止 (0)：不运行也不抢单
      3. 服务 (1)：处理请求始发到目的地
      4. 接单：已接到单从当前位置到请求发出地
   4. 状态转换方式
      1. 每等待20s，进入停止状态1s
      2. 抢到单后立刻进入接单状态
      3. 接到单后到达出发地点后停1s，然后进入服务状态
      4. 到达目的地后停1s，然后进入等待状态
   5. 司机信用：初始信用为0。抢单一次+1；接单一次+3
   6. 处理原则：
      1. 接单状态和服务状态都必须走最短路径（有相同则任选一条）
      2. 接到多单的话，按照请求发出时间依次处理
2. 请求
   1. 可在任意一点发出请求，请求发送给发出后3s内在发出点4\*4范围内等待状态的所有车均会抢单。3s内离开区域依然可以响应配单。
   2. 3s结束时，若无抢单则告知乘客无可用，结束响应；若有车响应，则选择等待服务状态的车中信用最高的，信用相同则选取距请求发出点最近的，前两种条件相同则随机选取
   3. 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。
   4. 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。
3. 对程序其他要求
   1. 提供一个接口用于查看出租车任意时刻的状态，
   2. 输出请求发出时，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。

程序结构设计：

1. 设计思想

我认为可以将整个系统抽象一下，使用程序的一共有三方：用户（request），出租车，服务器。

服务器并不是线程，访问信息的过程不涉及信息安全。出租车和用户均设计为线程，可以通过服务器进行交互。

服务器体量较大存储总的地图数据，作为静态对象进行访问，可以根据用户所在的位置返回相应的地图信息，根据车辆的位置返回行驶路。

用户向服务器发送请求，从服务器读取自己附近的出租车信息，并选择最新的出租车，然后指引出租车来接，并且将出租车指引到目的地。

出租车平时可以自己运行，在用户发来请求时，根据指示前往接客，可以设计成状态机来实现不同状态的相互顺序转换。

指令进入到窗口期结束前，用户和出租车通过服务器进行交互。到用户端选定自己的出租车后，出租车和用户端直接进行交互，减少服务器负载。个人觉得这样的思路比较符合现实情况

1. 类功能分析
   1. 控制器类：
      1. 用于出租车和指令从Map中获取数据
      2. 读取符合指令要求的出租车
      3. 启动指令线程
      4. 向gui发送出租车数据
   2. 输入：
      1. 读取指令
      2. 判断是否有效以及同质
      3. 将指令发给控制器
   3. 地图类：
      1. 读取和解析地图
      2. 通过地图寻找最短路径，以及最短距离
   4. 出租车类：
      1. 出租车运行
      2. 获取行驶路线
      3. 返回状态的接口
   5. 点：
      1. 存储点位置信息（包括上下左右的链接情况）
      2. 寻找相邻点
      3. 返回随机相邻点
   6. 指令：
      1. 发送指令给出租车
      2. 向gui发送指令信息
      3. 选择出租车
      4. 向出租车发送运行路线
2. 类图（使用ModelSpoon生成）

