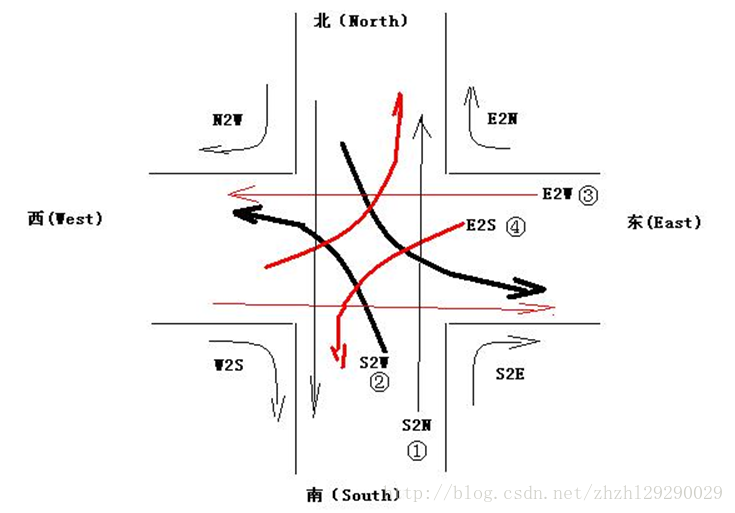
# [黑马程序员--JAVA交通灯管理系统项目](http://blog.csdn.net/zhzhl29290029/article/details/12753865)

**一、项目需求**

模拟实现十字路口的交通灯管理系统逻辑，具体要求如下：  
1、异步随机生成按照各个路线行驶的车辆。  
例如：  
   由南向而来去往北向的车辆----直行车辆  
   由西向而来去往南向的车辆----右转车辆  
   由东向而来去往南向的车辆----左转车辆  
       。。。。。。。。。。  
2、信号灯忽略黄灯，只考虑红灯和绿灯。  
3、应考虑左转车辆控制信号灯，右转车辆不受信号灯控制。  
4、具体信号灯控制逻辑与现实生活中普通交通灯控制逻辑相同，不考虑特殊情况下的控制逻辑。  
注：南北向车辆与东西向车辆交替放行，同方向等待车辆应先放行直行车辆而后放行左转车辆。  
5、每辆车通过路口时间为1秒（提示：可通过线程sleep的方式模拟）。  
6、随机生成车辆时间间隔以及红绿灯交换时间间隔自定，可以设置。  
7、不要求实现GUI,只考虑系统逻辑实现，可通过Log方式展现程序运行结果。

**二、需求分析**



总共有12条路线，为了统一编程模型，可以假设每条路线都有一个红绿灯对其进行控制，右转弯的4条路线的控制灯可以假设称为常绿状态，另外，其他的8条线路是两两成对的，可以归为4组，所以，程序只需考虑图中标注了数字号的4条路线的控制灯的切换顺序，这4条路线相反方向的路线的控制灯跟随这4条路线切换，不必额外考虑。

**面向对象的分析与设计：**  
a)        每条路线上都会出现多辆车，路线上要随机增加新的车，在灯绿期间还要每秒钟减少一辆车。  
1设计一个Road类来表示路线，每个Road对象代表一条路线，总共有12条路线，即系统中总共要产生12个Road实例对象。

2每条路线上随机增加新的车辆，增加到一个集合中保存。

3每条路线每隔一秒都会检查控制本路线的灯是否为绿，是则将本路线保存车的集合中的第一辆车移除，即表示车穿过了路口。  
b)        每条路线每隔一秒都会检查控制本路线的灯是否为绿，一个灯由绿变红时，应该将下一个方向的灯变绿。  
1设计一个Lamp类来表示一个交通灯，每个交通灯都维护一个状态：亮（绿）或不亮（红），每个交通灯要有变亮和变黑的方法，并且能返回自己的亮黑状态。  
2总共有12条路线，所以，系统中总共要产生12个交通灯。右拐弯的路线本来不受灯的控制，但是为了让程序采用统一的处理方式，故假设出有四个右拐弯的灯，只是这些灯为常亮状态，即永远不变黑。  
3除了右拐弯方向的其他8条路线的灯，它们是两两成对的，可以归为4组，所以，在编程处理时，只要从这4组中各取出一个灯，对这4个灯依次轮询变亮，与这4个灯方向对应的灯则随之一同变化，因此Lamp类中要有一个变量来记住自己相反方向的灯，在一个Lamp对象的变亮和变黑方法中，将对应方向的灯也变亮和变黑。每个灯变黑时，都伴随者下一个灯的变亮，Lamp类中还用一个变量来记住自己的下一个灯。  
4无论在程序的什么地方去获得某个方向的灯时，每次获得的都是同一个实例对象，所以Lamp类改用枚举来做显然具有很大的方便性，永远都只有代表12个方向的灯的实例对象。  
5设计一个LampController类，它定时让当前的绿灯变红。  
（我们初步设想一下有哪些对象：红绿灯，红绿灯的控制系统，汽车，路线。汽车看到自己所在路线对应的灯绿了就穿过路口吗？不是，还需要看其前面是否有车，看前面是否有车，该问哪个对象呢？该问路，路中存储着车辆的集合，显然路上就应该有增加车辆和减少车辆的方法了。再看题目，我们这里并不要体现车辆移动的过程，只是捕捉出车辆穿过路口的过程，也就是捕捉路上减少一辆车的过程，所以，这个车并不需要单独设计成为一个对象，用一个字符串表示就可以了。  
面向对象设计把握一个重要的经验：谁拥有数据，谁就对外提供操作这些数据的方法。再牢牢掌握几个典型的案例就可以了：人在黑板上画圆，列车司机紧急刹车，售货员统计收获小票的金额，你把门关上了等。  
学员的两个面向对象的面试题，用面向对象的方式设计如下情景。  
“两块石头磨成一把石刀，石刀可以砍树，砍成木材，木材做成椅子”，  
“球从一根绳子的一段移动到了另一端”，）

**三、代码实现**

**一）Road类的编写：**

分析：  
每个Road对象都有一个name成员变量来代表方向，有一个vehicles成员变量来代表方向上的车辆集合。  
在Road对象的构造方法中启动一个线程每隔一个随机的时间向vehicles集合中增加一辆车（用一个¡°路线名\_id¡±形式的字符串进行表示）。  
在Road对象的构造方法中启动一个定时器，每隔一秒检查该方向上的灯是否为绿，是则打印车辆集合和将集合中的第一辆车移除掉。（在讲Road对象的定时器代码时，因为开始阶段还没有设计Lamp类，所以，检查该方向上的灯是否为绿的代码暂时先采用短路方式）  
 代码实现：

**[java]** [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12753865) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12753865)

1. package com.sohu.interview.test;
2. import java.util.List;
3. import java.util.ArrayList;
4. import java.util.Random;
5. import java.util.concurrent.ExecutorService;
6. import java.util.concurrent.Executors;
7. import java.util.concurrent.ScheduledExecutorService;
8. import java.util.concurrent.TimeUnit;
9. /\*
10. \* @author zhzhl29290029
11. \*/
12. public class Road {
13. //定义一个集合，用于存储车辆和移除车辆
14. private List<String> vechicles = new ArrayList<String>();
15. //依据路名产生车名
16. private String name = null;
17. public Road(String name){
18. this.name = name;
20. //模拟车辆不断随机上路的过程
21. //使用线程池，通过产生单个线程的方法，创建一个线程池
22. ExecutorService pool = Executors.newSingleThreadScheduledExecutor();
23. //调用execute方法，可向线程池提交一个任务，让池中的线程执行任务
24. pool.execute(new Runnable(){
25. //执行的任务，随机产生车，并加入集合
26. public void run() {
27. for(int i=1;i<1000;i++){
28. try {
29. //每个10秒随机产生一辆车
30. Thread.sleep((new Random().nextInt(10)+1)\*1000);
31. } catch (InterruptedException e) {
32. e.printStackTrace();
33. }
34. vechicles.add(Road.this.name + "\_" + i);
35. }
36. }
37. });
39. //每隔一秒检查对应的灯是否为绿，是则放行
40. //产生一个单线程，创建定时器
41. ScheduledExecutorService timer = Executors.newScheduledThreadPool(1);
42. timer.scheduleAtFixedRate(
43. //执行具体的任务
44. new Runnable(){
45. public void run(){
46. //判断路上是否有车，有则进行相应的操作
47. if(vechicles.size()>0){
48. //每隔1秒让车通行，通行前要先判断灯是否亮，亮了才能通行，即从集合中移除
49. boolean lighted = Lamp.valueOf(Road.this.name).isLighted();
50. if(lighted)
51. System.out.println(vechicles.remove(0) + " is traversing !");
52. }
53. }
54. },
55. 1,
56. 1,
57. TimeUnit.SECONDS
58. );
59. }
61. }

**二）Lamp类的编写：**

分析：  
系统中有12个方向上的灯，在程序的其他地方要根据灯的名称就可以获得对应的灯的实例对象，综合这些因素，将Lamp类用java5中的枚举形式定义更为简单。  
每个Lamp对象中的亮黑状态用lighted变量表示，选用S2N、S2W、E2W、E2N这四个方向上的Lamp对象依次轮询变亮，Lamp对象中还要有一个oppositeLampName变量来表示它们相反方向的灯，再用一个nextLampName变量来表示此灯变亮后的下一个变亮的灯。这三个变量用构造方法的形式进行赋值，因为枚举元素必须在定义之后引用，所以无法再构造方法中彼此相互引用，所以，相反方向和下一个方向的灯用字符串形式表示。  
增加让Lamp变亮和变黑的方法：light和blackOut，对于S2N、S2W、E2W、E2N这四个方向上的Lamp对象，这两个方法内部要让相反方向的灯随之变亮和变黑，blackOut方法还要让下一个灯变亮。  
除了S2N、S2W、E2W、E2N这四个方向上的Lamp对象之外，其他方向上的Lamp对象的nextLampName和oppositeLampName属性设置为null即可，并且S2N、S2W、E2W、E2N这四个方向上的Lamp对象的nextLampName和oppositeLampName属性必须设置为null，以便防止light和blackOut进入死循环。  
代码实现：

**[java]** [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12753865) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12753865)

1. package com.sohu.interview.test;
2. /\*
3. \* @author zhzhl29290029
4. \*/
5. public enum Lamp {
6. S2N("N2S","S2W",false),S2W("N2E","E2W",false),E2W("W2E","E2S",false),E2S("W2N","S2N",false),
7. N2S(null,null,false),N2E(null,null,false),W2E(null,null,false),W2N(null,null,false),
8. S2E(null,null,true),E2N(null,null,true),N2W(null,null,true),W2S(null,null,true);
10. //定义当前灯对应方向的灯，同亮
11. private String oppsiteLampName;
12. //定义当前灯的下一个灯，当前灯亮，下个灯不亮
13. private String nextLampName;
14. //标记当前灯是否为绿灯（亮）
15. private boolean lighted;
16. //灯的判断是否亮的方法
17. public boolean isLighted(){
18. return lighted;
19. }
20. //构造方法，初始化对象
21. private Lamp(String oppsiteLampName,String nextLampName,boolean lighted){
22. this.oppsiteLampName = oppsiteLampName;
23. this.nextLampName = nextLampName;
24. this.lighted = lighted;
25. }
26. //当前灯变绿时，对应的灯也绿了
27. public void light(){
28. this.lighted = true;
29. //当前灯变绿了，让对应的灯也变绿
30. if(oppsiteLampName!=null)
31. Lamp.valueOf(oppsiteLampName).light();
32. //调试用
33. System.out.println(name() + " lamp is green,下面总共应该有6个方向能看到汽车穿过！");
34. }
36. //当前灯变红了，对应灯也要变红，并启动下个灯变绿
37. public Lamp blackOut(){
38. this.lighted = false;
39. if(oppsiteLampName!=null)
40. Lamp.valueOf(oppsiteLampName).blackOut();
41. Lamp nextLamp = null;
42. if(nextLampName!=null){
43. nextLamp = Lamp.valueOf(nextLampName);
44. //调试用
45. System.out.println("绿灯从" + name() +"切换为" + nextLampName);
46. nextLamp.light();
47. }
48. return nextLamp;
49. }
50. }

**三）LampContriller类的编写：**

分析：交通灯控制器  
整个系统中只能有一套交通灯控制系统，所以，LampController类最好是设计成单例。  
LampController构造方法中要设定第一个为绿的灯。  
LampController对象的start方法中将当前灯变绿，然后启动一个定时器，每隔10秒将当前灯变红和将下一个灯变绿。

**[java]** [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12753865) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12753865)

1. package com.sohu.interview.test;
3. import java.util.concurrent.Executors;
4. import java.util.concurrent.ScheduledExecutorService;
5. import java.util.concurrent.TimeUnit;
7. /\*
8. \* @author zhzhl29290029
9. \*/
11. public class LampController {
12. //定义当前灯
13. private Lamp currentLamp;
15. //构造函数，指定当前灯并让其为绿
16. public LampController(){
17. currentLamp = Lamp.S2N;
18. currentLamp.light();
20. //每隔10秒将当前绿灯变为红灯，并让下一个方向的灯变绿
21. //创建定时器
22. ScheduledExecutorService timer = Executors.newScheduledThreadPool(1);
23. //通过定时器的方法，创建线程
24. timer.scheduleAtFixedRate(
25. new Runnable() {
26. public void run() {
27. //让当前灯变红，并返回下一个灯
28. System.out.println("coming...");
29. currentLamp = currentLamp.blackOut();
30. }
31. },
32. 10,
33. 10,
34. TimeUnit.SECONDS);
35. }
36. }

**四）MainClass类的编写**

用for循环创建出代表12条路线的对象。  
接着再获得LampController对象并调用其start方法。  
（String [] directions = {"S2N","N2S","S2W","N2E","E2W","W2E",  
"E2S","W2N","S2W","N2W","E2N","W2S"};刚开始把上面的倒数第4个元素写错了（应该写成S2E），所以总是只看到5个方向上有车通过的情况。）

**[java]** [view plain](http://blog.csdn.net/zhzhl29290029/article/details/12753865) [copy](http://blog.csdn.net/zhzhl29290029/article/details/12753865)

1. package com.sohu.interview.test;
2. public class TraffTest {
4. public static void main(String[] args) {
5. //将12个方向上的路线存入一个数组
6. String[] directions = new String[]{
7. "S2N","S2W","E2W","E2S",
8. "N2S","N2E","W2E","W2N",
9. "S2E","E2N","N2W","W2S"
10. };
11. //通过循环创建12个对象，即12条路线
12. for(int i=0;i<directions.length;i++){
13. new Road(directions[i]);
14. }
15. //产生整个交通灯系统
16. new LampController();
17. }
18. }