用于列车售票的可线性化并发数据结构

September 7, 2016

给定Ticket类：

```
class Ticket {
    long tid;
    String passenger;
    int route;
    int coach;
    int seat;
    int departure;
    int arrival;
}
```

其中，tid是车票编号，passenger是乘客名字，route是列车车次，coach是车厢号，seat是座位号，departure是出发站编号，arrival是到达站编号。

给定TicketingSystem接口：

```
public interface TicketingSystem {
    Ticket buyTicket(String passenger,int route,
        int departure,int arrival);
    int inquiry(int route,int departure,int arrival);
    boolean refundTicket(Ticket ticket);
}
```

其中,

- buyTicket是购票方法，即乘客passenger购买route车次从departure站到arrival站的车票1张。若购票成功，返回有效的Ticket对象；若失败（即无余票），返回无效的Ticket对象（即return null）。
- refundTicket是退票方法，对有效的Ticket对象返回true，对无效的Ticket对象返回false。
- inquriy是查询余票方法，即查询route车次从departure站到arrival站的余票数。

每位学生使用Java语言设计并完成一个用于列车售票的可线性化并发数据结构：TicketingDS类，该类实现TicketingSystem接口，同时提供

```
TicketingDS(routenum,coachnum,seatnum,stationnum);
```

构造函数。其中，routenum是车次总数（缺省为5个），coachnum是列车的车厢数目（缺省为8个），seatnum是每节车厢的座位数（缺省为100个），stationnum 是每个车次经停站的数量（缺省为10个，含始发站和终点站）。

为简单起见，假设每个车次的coachnum、seatnum和stationnum都相同。车票涉及的各项参数均从1开始计数，例如车厢从1到8编号，车站从1到10编号等。

每位学生需编写多线程测试程序，在main方法中用下述语句创建TicketingDS类的一个实例。

```
TicketingDS tds = new
TicketingDS(routenum, coachnum, seatnum, stationnum);
```

系统中同时存在若干个线程（缺省为16个），每个线程是一个票务代理，按照60%查询余票，30%购票和10%退票的比率反复调用TicketingDS类的三种方法若干次（缺省为总共10000次）。按照线程数为4，8，16，32，64个的情况分别给出每种方法调用的平均执行时间，同时计算系统的总吞吐率（单位时间内完成的方法调用总数）。

正确性要求

- 每张车票都有一个唯一的编号tid，不能重复。
- 每一个tid的车票只能出售一次。退票后，原车票的tid作废。
- 每个区段有余票时，系统必须满足该区段的购票请求。
- 车票不能超卖，系统不能卖无座车票。

作业评分标准

作业评分分为两部分，基本分（50%）和性能分(50%)。

1. 首先保证并发数据结构功能正确。如果发现实现有错误，只能按照完成情况给基本分。如果能够给出验证并发数据结构正确性的方法并实现验证程序（须提交说明文档：阐述验证程序设计思路），可以额外加分。
2. 对于所有正确实现的并发数据结构，用统一的多线程基准程序在同一测试环境下测试系统的延迟和吞吐率，并按照并发数据结构的性能测试结果进行加权排序，从高到低依次给出性能分。

作业清单

大作业提交package的名字为ticketingsystem，至少包含3个文件：

1. TicketingSystem.java是规范文件，不能更改。
2. TicketingDS.java是并发数据结构的实现。
3. Test.java实现多线程测试。

TicketingSystem.java:

```
package ticketingsystem;

class Ticket{
        long tid;
        String passenger;
        int route;
        int coach;
        int seat;
        int departure;
        int arrival;
}
```

```java
public interface TicketingSystem {
    Ticket buyTicket(String passenger,int route,
        int departure,int arrival);
    int inquiry(int route,int departure,int arrival);
    boolean refundTicket(Ticket ticket);
}
```

TicketingDS.java:

```java
package ticketingsystem;

public class TicketingDS implements TicketingSystem {
    //ToDo
}
```

Test.java:

```java
package ticketingsystem;

public static void main(String[] args) {

    TicketingDS tds = new
    TicketingDS(routenum, coachnum, seatnum, stationnum);

    //ToDo
}
```

作业按照project打包提交，提交前需编译测试通过，同时提交测试和性能评价报告：阐述并发数据结构和多线程测试程序的设计思路，分析系统的正确性和性能，解释所实现的每个方法是否可线性化、是否deadlock-free、starvation-free、lock-free或wait-free。