用于列车售票的可线性化并发数据结构

2019 年 9 月 25 日

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，对错误或无效的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, threadnum);
```

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），threadnum 是并发购票的线程数（缺省为16个）。

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。

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

系统中同时存在threadnum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。
- 查询余票的约束放松到静态一致性。查询结果允许不精确，但是在某个车次查票过程中没有其他购票和退票的“ 静止”状态下，该车次查询余票的结果必须准确。

作业评分标准

作业评分分为三部分，基本分（50%），性能分(50%)和奖励分。

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。

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。

3. 如果采用或实现通用的验证工具对并发数据结构进行验证（须提交验证代码，并阐述验证思路、过程和结果），可以在基本分和性能分的基础上，额外加奖励分，奖励分单独计入总成绩。

作业清单

大作业提交package的名字为ticketingsystem，所有Java程序放在ticketingsystem目录中，trace.sh文件放在ticketingsystem目录的上层目录中。如果程序有多重目录，那么将主Java程序放在ticketingsystem目录中。至少包含5个文件(见附件ticketingsystem.tgz)：

1. TicketingSystem.java是规范文件，不能更改。

2. Trace.java是trace生成程序，用于正确性验证，不能更改。

3. trace.sh是trace生成脚本，用于正确性验证，不能更改。

4. TicketingDS.java是并发数据结构的实现。

5. Test.java实现多线程性能测试。

TicketingSystem.java:

```
package ticketingsystem;

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


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:

```
package ticketingsystem;

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

Test.java:

```
package ticketingsystem;

public static void main(String[] args) {

final TicketingDS tds = new
TicketingDS(routenum, coachnum, seatnum, stationnum, threadnum);
    //ToDo
}
```

trace.sh:

```
#!/bin/sh
javac -encoding UTF-8 -cp . ticketingsystem/Trace.java
```

作业按照ticketingsystem目录打包提交，程序编码为UTF-8格式（要求程序必须在Linux系统上能够正常编译和运行）。若提交多种不同的实现，每种实现须按上述目录结构，以ticketingsystem, ticketingsystem1, ticketingsystem2, ... 的目录命名方式单独打包，评分时以ticketingsystem目录的代码为准。提交前需编译测试通过，要求能正确执行trace.sh脚本文件生成trace（不得更改trace.sh所在目录位置）。同时提交性能评测报告：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。

程序需本人独立完成，不得抄袭他人代码。如果发现抄袭，成绩为零分。