Programming Test: Bus Stop

Goal

The goal of this exercise is to create a console application to solve the problem described in the next section and to do it in a way that shows understanding of programming concepts and structures. Console application must take one argument — the path to input file and must generate "output.txt" as result. Source code must be available on any public git repository.

Rules and Requirements

Following rules are in effect during the exercise:

- You can use whatever reference material you wish (book, web, …).
- You may use standard libraries, but no other advanced frameworks.
- The implementation of classes that already has analogue in standard library is allowed only in case when your implementation is better than the standard one for this specific problem (number of operations, memory, code readability, etc.). In such case the comment shall be presented in the code describing why does your implementation is better.
- A valid configuration for some common build system (like Maven or MSBuild) that produces the executable file (jar or exe) must be included.
- The code must be well formed (reasonable and meaningful names, OOP principels, etc).
- If you are stuck or you have questions, just ask them.
- You may adjust the input file for testing other cases and include them into the solution with JUnit or NUnit.

Problem

Background

Bloggersville is served by two bus companies: *Posh Bus Company* and *Grotty Bus Company*. Both companies operate a service from the airport to the central bus stop.
The two companies have decided to produce a joint daily bus timetable. However, bus travelers find it difficult to use the timetable because of following reasons:

1. It is difficult to search faster buses in the timetable.
2. Some of the buses run faster than others. For a frequent bus traveler it is better to miss an earlier bus in order to catch a faster bus which departs later but reaches its destination sooner.
3. The entries in the timetable are not necessarily in order of departure time.

Description

Given the information in the joint timetable, write a program to produce two modified timetables, one for *Posh Bus Company* and one for *Grotty Bus Company*, each satisfying the following requirements:

1. All entries in each timetable are in order of departure time.
2. Any service longer than an hour shall not be included.
3. The provided timetable is valid for any day, i.e. tomorrow it will be the same.
4. There can be entries with departure time greater than arrival time. This means that the bus will arrive tomorrow. E.g.: 23:03 00:01 defines that this service takes 58 minutes and the arrival will be tomorrow.
5. Only efficient services shall be added to the timetable. A service is considered efficient if there are no other services that are more efficient compared to it. A service is considered more efficient compared to the other one:
   - If it starts at the same time and reaches earlier, or

- If it starts later and reaches at the same time, or
- If it starts later and reaches earlier.

6. If both companies offer a service having the same departure and arrival times then always choose *Posh Bus Company* over *Grotty Bus Company*, since *Grotty Bus Company* busses are not as comfortable as those of *Posh Bus Company*.

Input format

The input file has the following format:

```
<service>
<service> …
<service><EOF>
```

Each <service> record is on a separate line and will consist of:

- The character string 'Grotty' or 'Posh' to indicate which company is running the service.
- A space
- The departure time in 24 hours format, represented by 5 characters as 'HH:MM'
- A space
- The arrival time in 24 hours format, represented by 5 characters as 'HH:MM'

Example of a <service>:

```
Posh 10:15 11:10
```

Output format

The output timetables shall be in the same format as the input timetable, with the *Posh Bus Company* timetable first followed by a blank line and the *Grotty Bus Company* timetable:

```
<posh_service>
<posh_service> …
<posh_service>
<blank_line>
<grotty_service>
<grotty_service> …
<grotty_service><EOF>
```

Example input and output

Given the following data:

```
Posh 10:15 11:10
Posh 10:10 11:00
Grotty 10:10 11:00
Grotty 16:30 18:45
Posh 12:05 12:30
Grotty 12:30 13:25
Grotty 12:45 13:25
Posh 17:25 18:01
```

Your program shall produce:

```
Posh 10:10 11:00
Posh 10:15 11:10
Posh 12:05 12:30
Posh 17:25 18:01

Grotty 12:45 13:25
```