Assignment 3

Due date

- 11.59 PM EST on Oct 18th.

Submit your code as per the provided instructions.

Updates

Assignment Goal

Apply the design principles you have learned so far to develop software for the given problem.

Team Work

- You need to work alone on this assignment.
- You cannot discuss with anyone the design pattern(s) to be used for this assignment.

Programming Language

You are required to use Java.

Compilation Method

- You are required to use ANT to compile the code.

Policy on sharing of code

- EVERY line of code that you submit in this assignment should be written by you. Do NOT show your code to any other group. Our code-comparison software can very easily detect similarities.
- Post to the listserv if you have any questions about the requirements. Do NOT post your code to the listserv asking for help with debugging. However, it is okay to post design/concept questions on programming in Java/C/C++.

Project Description

Problem Statement

Let's call the factors, that affect the security implementation of an airport, **SecurityFactors**. Let's call the operation of tightening/loosening security, **tightenOrLoosenSecurity()**. The following are the states that an airport can be in,

- **LOW_RISK**
- **MODERATE_RISK**
- **HIGH_RISK**

Two metrics are used to determine the state of the airport. The 2 metrics are,

- **Average Traffic Per Day**
  Computed as ⇒ Total number of travellers ÷ Total number of days
- **Average Prohibited Items Per Day**
  Computed as ⇒ Total number of prohibited items ÷ Total number of days

The following are the list of prohibited items,

- **Gun**
- **NailCutter**
- **Blade**
- **Knife**

For each state that the airport can be in, the following are the conditions that need to be satisfied,

- **LOW_RISK**
  - *0 ≤ average traffic per day < 4* **OR**
  - *0 ≤ average prohibited items per day < 1*
- **MODERATE_RISK**
  - *4 ≤ average traffic per day < 8* **OR**
  - *1 ≤ average prohibited items per day < 2*
- **HIGH_RISK**
  - *average traffic per day ≥ 8* **OR**
  - *average prohibited items per day ≥ 2*

If conditions for 2 states get satisfied, then the state with the higher risk is chosen.

Let's assume that there are 10 operations that can be performed by the security agency, based on the current state of the aiport. Each operation is identified by an ID. Thus, we have 10 IDs in the range [1, 10]. For each state of the airport, the security performs a subset of the operations. These are mentioned below,

- **LOW_RISK**
  - OperationID = 1
  - OperationID = 3
  - OperationID = 5
  - OperationID = 7
  - OperationID = 9
- **MODERATE_RISK**
  - OperationID = 2
  - OperationID = 3
  - OperationID = 5
  - OperationID = 8
  - OperationID = 9
- **HIGH_RISK**
  - OperationID = 2
  - OperationID = 4
  - OperationID = 6
  - OperationID = 8
  - OperationID = 10

For every traveller that enters the airport, the program should output the subset of operations that need to be performed in order to maintain peace and harmony.

Input

Input would be given in the form of a file. The input file will contain many lines, each line corresponding to information about 1 traveller. The following information would be given about each traveller,

- *Day of Travel* (key = **Day**)
- *Time of Day (24 hour clock)* (key = **TOD**)
- *Airline* (key = **Airline**)
- *Item* (key = **Item**)

The input format would be ⇒ **Day:<d1>;TimeOfDay:<d1>;Airline:<d1>;Item:<d1>**
An example input is shown below,

```
Day:1;TOD:10:00;Airline:United;Item:ShavingBrush
Day:2;TOD:17:30;Airline:JetBlue;Item:NailCutter
Day:2;TOD:21:00;Airline:Delta;Item:Gun
Day:2;TOD:06:30;Airline:AmericanAirlines;Item:Wine
Day:2;TOD:07:00;Airline:CapeAir;Item:Knife
Day:2;TOD:08:50;Airline:Southwest;Item:Blade
```

Output

An output file should be generated by the program. For each line in the input file, the output file should contain the list of operations (just mention the OperationIDs) to be performed, separated by ‘<space>’.

Output file (for the example input given above) is as shown below,

```
1 3 5 7 9
1 3 5 7 9
2 3 5 8 9
2 3 5 8 9
2 3 5 8 9
2 4 6 8 10
```

Sample Input Files from Students

. Please note that I have not verfied these input files. If you find any error, or have any comments, please send them to the instructor via email.

- From Harshad C Loya. Thanks!
  - input-1.txt
- From Lauren A Pagano. Thanks!
  - input-2.txt
- From Jaydeep Ingle. Thanks!
  - input-3.txt
- From Swapnil. Thanks!
  - input-4.txt
- From Manav Panchal. Thanks!
  - input-5.txt
- From Girish Kurkute. Thanks!
  - input-6.txt
- From Priyanka Desai. Thanks!
  - input-7.txt
- From Dylan Conroy. Thanks!
  - input-8.txt
- From Kishankumar Patel. Thanks!
  - input-9.txt
- From Abhijit Bhandarkar. Thanks!
  - input-10.txt
- From Tushit Jain. Thanks!
  - input-11.txt
- From Vidhi Kamdar. Thanks!
  - input-12.txt
- From Kunal Shira. Thanks!
  - input-13.txt

Code Organization

- Your directory structure should be the following:

```
-firstName_lastName_assign_3
  ---airportSecurityState
  ----- README.txt
    ----- src
         ----- build.xml
         ---airportSecurityState
          ----------driver
          -----------------Driver.java
          ----------util
          -----------------[Whatever interfaces and class you need]
          -----------------MyLogger.java
          ----------airportStates
          -----------------AirportStateI [interface for State pattern]
          -----------------[Whatever interfaces and classes you need to implement the States]
          ----------other packages and classes that you need
```

Submission

- Same as Assignment-1

General Requirements

- Same as Assignment-1, except Javadoc is not required

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed.

Grading Guidelines

TBA

mgovinda at cs dot binghamton dot edu
Back to Programming Design Patterns