# Assignment 4

## Due date

- 11.59 PM EST on November 8th.

Submit your code as per the provided instructions.

### git

- https://classroom.github.com/a/mkzIqP5_

### 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 increasing/decreasing security **increaseOrDecreaseSecurity()**. 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** *(avgTrafficPerDay)*
  Computed as ⇒ Total number of travellers ÷ Total number of days

- **Average Prohibited Items Per Day** *(avgProhibitedItemsPerDay)*
  Computed as ⇒ Total number of prohibited items ÷ Total number of days

The following are the list of prohibited items,

- **Grains**
- **NailCutters**
- **Plants**
- **EndangeredAnimals**

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

- **LOW_RISK**
  - $0 \leq avgTrafficPerDay < 4$ **OR**
  - $0 \leq avgProhibitedItemsPerDay < 2$
- **MODERATE_RISK**
  - $4 \leq avgTrafficPerDay < 8$ **OR**
  - $2 \leq avgProhibitedItemsPerDay < 4$
- **HIGH_RISK**
  - $avgTrafficPerDay \geq 8$ **OR**
  - $avgProhibitedItemsPerDay \geq 4$

If either conditions (*avgTrafficPerDay* or *avgProhibitedItemsPerDay*) gets satisfied for more than one state, then the higher risk state is chosen.
*avgTrafficPerDay* and *avgProhibitedItemsPerDay* need to have corresponding setters and getters in the context. These setter and getter methods SHOULD be used from the states to update/retrieve values.

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 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**)
- *Item* (key = **Item**)

The input format would be ⇒ **Day:<day>;Item:<item>**
An example input is shown below,

```
Day:1;Item:NailCutters
Day:1;Item:Grains
Day:2;Item:Wine
Day:2;Item:EndangeredAnimals
Day:2;Item:Plants
```

*Note: The input file will be well formatted and will not contain any spaces between elements in each line.*

## Output

An output file should be generated by the program. *The name of the output file will be provided via command line*. 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
2 3 5 8 9
1 3 5 7 9
1 3 5 7 9
2 3 5 8 9
```

### More requirements

- Decide on what you want to print for Debug values 1-4.
- From the command line, accept the Debug value as an integer. Check for exceptions when you convert the command line argument from String to integer. Also, validate that the Debug value is in the range 0-4.
- An example of the command line arguments we will use for grading is the following: input.txt output.txt 0

## Code Organization

- Your directory structure should be the following:

```
-firstName_lastName_assign_4
  ---airportSecurityState
  ----- README.txt
    ----- src
         ----- build.xml
         ---airportSecurityState
          ----------driver
          -----------------Driver.java
          ----------util
          -----------------[Whatever interfaces and class you need]
          -----------------MyLogger.java
          ----------airportStates
          -----------------AirportContextI [interface for Context]
          -----------------AirportStateI [interface for State]
          -----------------[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

Grading Guidelines.

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