08.03.2018

CENG 112 Data Structures
Homework 1

Deadline: 18.03.2018 23:59

Design and implement an ADT that holds and manipulates the flight table. This table keeps all destinations and direct flights between them.

The flight table should be composed of destinations in its rows and columns, and the ADT should include:

- A **constructor** which takes the destinations as input (as an array of destination names).
- **Modifiers** that allow us to
  - add a flight from a specific destination to another specific one
  - remove a flight from a specific destination to another specific one
  - rename an existing destination
- **Inspectors** that allow us to
  - determine whether flight table contains a specific destination
  - determine whether a destination is covered by any flights
  - determine the number of destinations that have flights
  - determine the number of flights
  - determine all destinations such that a direct flight exists from a specific destination to there
  - determine all destinations such that a direct flight exists to a specific destination from there
  - determine whether there is a direct flight from a specific destination to another one
  - display the current state of flight table in a 2D format

Please implement the ADT for Iztech Airlines Company. The company has a principle that all flights must be bidirectional, i.e. when a flight is added/removed, the flight in the opposite direction will be updated automatically. Additionally, the destination "Izmir" must exist by default and the company has flights from Izmir to every other destination. Please note this company keeps its flight table with a boolean representation of flights as follows: if there is a direct flight between Destination A and Destination B, then the intersection point of them will be "True", otherwise it should be "False".

Please make sure that you implement the ADT as an **interface** in Java**.** The method names in the interface should use destination and connection (instead of flight) words, so that the interface can also be used for implementation of cruise table and bus table. Write a flight table **class that implements this interface**. Provide also a separate **test class** (called FlightTableApp), which only contains the main method to perform the following tasks:

- creates a flight table using an array of destination names
- adds some reasonable flights to the table
- prints the current state of the flight table on the screen
- tests all the above methods

IMPORTANT NOTES

1. We expect you provide a specification of the ADT and its implementer Flight Table, as a well-prepared javadoc. The javadoc should contain an ADT description of the behavior of a connections table and implemented behavior of this ADT as a Flight Table. Do not forget to specify pre-conditions and post-conditions.
2. In your code, throw an appropriate exception whenever it is necessary.
3. In constructor, do not assign destinations to an attribute directly. Instead, make a copy. This will prevent side effects.
4. You should develop your code on **Eclipse IDE** with Version Oxygen. Your Java version should be at least **1.8.**
5. Cheating will NOT be tolerated!
6. Any detected cheating will be graded as 0.

SUBMISSION RULES

1. You should create your java project as **ID1_ID2_HW1** (e.g. 123456_654321_HW1) and export as **ID1_ID2_HW1.zip**.
2. You should upload your zip file as ID1_ID2_HW1.zip.
3. One of the group members is sufficient to upload.
4. Late submissions will not be allowed.

GRADE REDUCTIONS

Please code your programs wisely. Possible grade reductions:

- Missing controls!
- No error handling!
- Unused/dead codes!
- Naming conventions!