48024
Programming 2
Assignment 1

Topics:

OO Design, Standard Patterns, Lists

Learning Outcomes:

This assessment task addresses the following subject learning objectives (SLOs): 1, 2 and 3

Due date:

12:00 AM Monday 3 October

Weight:

35%

Individual Work

All work is individual. You may discuss ideas, approaches and problems, but you should write every line of code yourself except for code copied from the lecture notes, lecture code or lab code. You **MUST NOT** let another student see your solution code, and you MUST NOT look at another student's solution code. More information about Academic Misconduct can be found at:

http://www.gsu.uts.edu.au/rules/student/section-16.html

Working Language

You can choose either Java or Python to complete assignment 1. The higher mark between your Java solution and Python solution will be counted into your final grade. However, you are only credited with one of your solutions, either Java or Python, not both of them or the mixture.

The specification is illustrated based on Java. You can simply translate the Java syntax to Python for your Python solution. Detailed explanations about Python criteria will be posted on the FAQ page on ED.

Specification

The Travel Agency is hiring a software engineer to develop a new management system, which consists of two main components, an administrative component, and a trip booking component.

The administrative section will allow, through text-based menus, the login and logout of agents, the addition and removal of destinations in the system, the display of all destinations, and the addition and removal of flights in the system.

The trip booking component will store a list of the destination information and a list of flight information. The trip booking component will allow the creation and display of the trip booking information. A trip consists of multiple destinations, with multiple flights that connect these destinations together based on the country they are in.

Each destination record will include the destination's name and country.

Each trip has a minimum of 2 destinations. This means that you should always expect a Trip to have n destinations and n-1 flights.

Each flight record will include the flight's airline name, flight number, takeoff place, landing place and cost. The agency can only book one existing flight between two countries. The takeoff and landing for a Flight are represented as a String of the country, and the combination of these two fields is unique. You do not have to worry about the dates or times of the flight.

The flights in a trip are based solely on the destinations. For example, if a trip consists of 3 destinations:

- Eiffel Tower | France
- Opera House | Australia
- Machu Picchu | Peru

Then your code should search the agency's flight list for 2 connecting flights. For example:

- American Airlines | 232 | France -> Australia
- QANTAS | 189 | Australia -> Peru

An aside

While reading the first part of the specification, you will notice there is a lot going on.

- How many functions did you identify?
- How many classes did you identify?
- What are the fields in each class?
- How many goals did you identify?
- How many patterns did you think of that might be applicable?

This assignment will be challenging, and you will probably want to manage your time well.

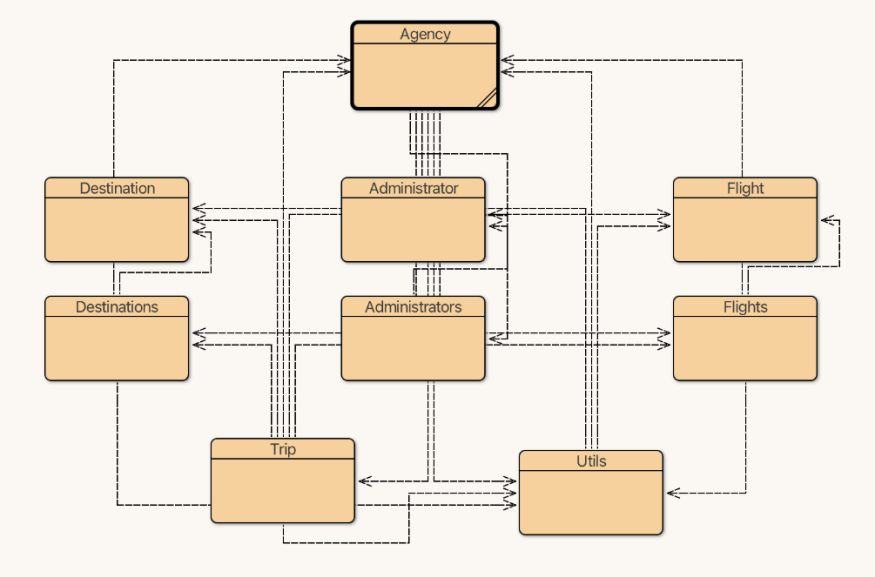
- How long do you think it will take you to code the functions?
- How long do you think it will take you to code each goal?
- A good rule of thumb is to think of an estimate, and then multiply that number by 3 or 4!
- To manage your time well, you may need to figure out which parts of the assignment you can start early.
- Which parts can you start now?

- Which parts can you start in week 6?

If you complete parts in the same week that you learn the topics (while they are fresh in your mind), they will take less time to complete.

Requirements

- Your design will consist of exactly the following classes with the listed fields, declared as indicated. You are not allowed to add or remove classes or fields; however, you may add constructors, functions and procedures to complete your design (in fact, you will have to!). You should pay careful attention to the tests on ED, as these will help guide you with some (but not all) of these methods.
- To help visualize the design, a partial class diagram has been provided



- Classes – your design will consist of these 9 classes:
    1. Agency
    2. Administrator
    3. Administrators
    4. Destination
    5. Destinations
    6. Flight
    7. Flights
    8. Trip
    9. Utils (this is the class to facilitate format, do not change)
- Fields – All the fields have been clarified in each class and they should not be modified. The fields also have some additional requirements and structures:

  Lists all have the abstract type of List<>, but must be instantiated with a concrete type that implements the List<> behavior (you can choose either – you may also want to think about why you might do things this way).

- Constructors – the constructors of the class have the following requirements:

1. All constructors initialize the fields of their class.
2. The Agency, Administrators, constructors take no parameters.
3. The Administrator constructor takes three parameters, the name, email and password, and initializes the fields corresponding to the four fields identically named.
4. The Destination constructor takes two parameters, the name and country, and initializes the fields corresponding to the three fields identically named.
5. The Flight constructor takes five parameters, the airline, flightNo, takeoff place, landing place and cost, and initializes the fields corresponding to the six fields identically named.
6. The Trip, Flights and Destinations constructors take agency as the parameter to initialize the fields identically named.
7. Administrators has the following data for Login:

David Dyer	david46@uts.com	123
Angela Huo	angela123@uts.com	mypw

8. Destinations has the following data.

Name	Country
Eiffel Tower	France
Opera House	Australia
Uluru	Australia
Machu Picchu	Peru
Great Pyramids	Egypt
Niagara Falls	Canada

9. Destination display format: {destination name} in {country}

   For example: Eiffel Tower in France

10. Flight display format: {airline} Flight {flight number} from {takeoff country} to {landing country} for the price of ${cost of the flight}

    For example: American Airlines Flight 677 from Argentina to Spain for the price of $260.44

11. Trip display format: Prints all the destinations and flights related to the trip, line by line. This should be in chronological order starting from the first destination, alternating between destination and flight, and ending with the final destination. Afterwards, print the total cost of the trip. This should use the Trip header and footer found in the Utils class.

    For example:

    Eiffel Tower in France

    United Airlines Flight 144 from France to Peru for $120.99

    Machu Picchu in Peru

    JetStar Flight 99 from Peru to Australia for $559.20

    Opera House in Australia

12. Utils Class – the Class defines the input methods and the printing format for the headers and footers of flight, destination and trip display. **All the methods in Utils are static and can be called without an object reference using Utils.methodName()**

    addFlightsForDestination(Destination destination, Agency agency): This is a utility method to automatically generate flights into the agency for a newly added Destination. This method should be called any time you add a new Destination to the agency in any way.

13. The main method of the program will be in the Agency class.

Non-existing and Existing check

Ensure your code handles non-existing and existing checks from the user's input. In some cases, you will need to utilize a read loop pattern to continuously ask the user for input until they input something valid; in other cases, you may rely on the menu pattern to instantly return to the upper menu it was called from. Use the "Check" button on Ed to verify your output. For examples:

1. Add a flight: **The combination of takeoff and landing must be unique.** There should not be 2 Flight objects in the system that have the same takeoff and landing country, regardless of whether their airline name or flight number is different. Your code must check that a flight does not already exist with that combination. If the input is wrong, ask the user to re-enter the values.

2. Add a connecting flight: For any 2 consecutive destinations, your code must find a flight from the agency's flight list that connects the destination countries and add the flight to the Trip's flight list. Your code must do this for all destinations in the Trip's destinations list. If the flight list is already populated, clear that list before repopulating it. If there are less than 2 destinations in the destinations list, inform the user and return to the Trip main menu.

3. Add a destination: **The combination of country and name must be unique.** There should not be 2 Destination objects in the system that have the same name and country. Your code must check that a flight does not already exist with that combination. If the input is wrong, ask the user to re-enter the values.

4. Remove a flight or destination: The flight/destination must exist inside the flights/destinations list. If the input is wrong, ask the user to re-enter the values.

5. Display: If there are no matching flights, inform the user.

Expected Workload

The time to do the assignment to a credit/distinction level has been estimated at 35 hours for a student of average ability who has completed all the tutorial and lab exercises. It is always better to start earlier than later, as you don't know which small error will consume all your time.

Online Support

A FAQs (Frequently Asked Questions) page will be posted as a slide in the Assignment page on Ed. This is not a static page filled with text, it is a question thread where you can post questions for the teaching staff to answer**. Your question should strictly relate to the specification, questions regarding code problems or testcase issues should go on the regular discussion board.** Any questions that do not adhere to this rule will be deleted. If you have a question, check the FAQ first, it may already have been answered there. **Anything posted to the FAQ is considered to be part of the assignment specification.**

As for "normal" assignment questions and assignment help, the preferred way to ask is through participating in the lectures, lab activities, UPASS and consultation sessions.

The Subject Coordinator may be contacted by email if you have matters of a personal nature to discuss, e.g., illness, personal issues or other matters of importance. All emails sent to the subject coordinators, tutors or lecturers must have a clear subject line that states the subject number followed by the subject of the email. e.g.:

[Subject 48024, Appointment Request], and must be sent from your UTS email address.

Submission

There are 2 parts to your submission. You must submit your source code through Ed by clicking the 'Mark' button AND you must submit your filled in cover sheet to Canvas.

ED Submission

Your solution is marked for correctness by comparing the output of your system to the output of the benchmark system in ED. You can submit a solution to ED many times by press "MARK"; I urge you to do this, so you receive credit for your work. Any code hasn't been "MARK" by ED won't be credited.

ED will test the features of your program in a certain order, but it cannot test the more advanced goals until the basic goals are working. To receive marks, you must pass ED's test cases in the order in which ED tests them.

Your code is marked by software, so you can get a good mark by fooling or spoofing the software. If you spoof a task worth N marks, you receive a penalty of 2*N marks. Design rules and spoofy check will be manually performed after due. Detailed specification about design rules and spoofy check refers to the document "Spoofy Check".

Canvas Submission

You need to submit the cover sheet of assignment 1 to Canvas portal and submit your source code to ED. **Missing cover sheet submission will result in "0" mark for the assignment.**

Return

Your provisional mark and feedback is generated immediately each time you submit to ED. However, it takes time for the analysis of spoofing, plagiarism, collusion and general cheating, which will start two weeks following the due date. If you are suspected of Academic Misconduct, I will forward your case to the Misconduct Committee and will notify you by manual feedback in ED.

There is no scheduled late submission period. An extension of up to 3 days will be granted automatically with a late penalty. An extension CANNOT be given after the due date. For any extension beyond one week, you will need to submit a Special Consideration following the Special Consideration process for reasons including unexpected health, family or work problems. More information about how to apply for special consideration can be found at: http://www.sau.uts.edu.au/assessment/consideration.html.

Marking Scheme

The marks for the assignment are divided into the following functionality components (note that individual tests may test several functionality components, and a functionality component may be tested by several tests):

Agency		
Main Menu	5	**10**
Login	5	
Destinations		
Menu	5	**25**
View Destinations	5	
View Destinations Filtered	5	
Add Destination	5	
Remove Destination	5	
Flights		
Menu	5	**25**

View Flights	5	
View Flights Filtered	5	
Add Flight	5	
Remove Flight	5	
Trip		
Menu	5	**40**
Add/Remove Destinations	15	
Add Connecting Flights	20	

This adds to a mark out of 100, at makes up 35% of your final assessment mark.

View Trip is integrated into the ‘Add/Remove Destinations’ and ‘Add Connecting Flights’.