CSCD 212 Project 1

Chris Cain

You have been tasked with coming up with a prototype for new airline ticket purchasing for a third-party site (think Google Flights). Your boss has come to you with a minimal set of instructions about exactly how this needs to be done, leaving a lot of room for intelligent designs. Her only stipulations are as follows:

* The design and flow of the software should be clean and easy to use. Our clients come first, and user experience is a cornerstone of our company.
* The code should be extendable. This is a prototype. Lots can and will change in the near future. We don’t want to spend a lot of time refactoring our original design.
  + To achieve this, your boss thinks it would be wise to use the strategy, decorator, and builder design patterns.

A reservation requires the following information:

* How many tickets are part of this reservation
* The total cost of the reservation (made up of the cost of each ticket, which is made up of multiple factors like seat class and checked bags)
* The aircraft (has a name, number of rows, number of seats per row)
* Number of layovers (and the associated airport codes and hours/minutes of each layover)
* The date and time of the flight
* The source airport code
* The destination airport code
* The airline
* For each ticket:
  + The full name of the flyer
  + Flyer’s date of birth
  + The seat class {First class, comfort, or economy} (each of which may add cost to the overall ticket and reservation)
  + The seat number (e.g. 15B)
  + Boarding class (can use A, B, C, etc. or any other system that makes sense)
  + Optional number of checked bags (add their cost to the overall ticket and reservation costs)

With your code, include a short writeup (~1 page single spaced) explaining how and where each design pattern is used (document exactly where, give class names and line numbers) and the benefit you get from using the design pattern. Finally, include a short description of any drawbacks or additional considerations after applying all 3 patterns to your code.

Hints:

* Use the builder pattern to construct the reservation step-by-step
* Use the decorator pattern to wrap your flight information with the appropriate ticket customization options
* Use the strategy pattern to encapsulate any behaviors/algorithms (the boarding process, a fly() method for a plane)
* Use inheritance hierarchies for airline, airplane, and airport at the least (you may need more depending on your design). Include a few concrete subclasses for each. You can use your creativity for the exact names, types, etc.

Grading (100 lab points, will be multiplied by 3 in Canvas to give it 3x the weight of a normal lab)

* Functionality and design (70 points)
  + How easy to use is your application? (10 points)
  + Functionality – does your reservation system include all necessary components and features? Does the code break elegantly when it should? Are the appropriate exceptions thrown? (15 points)
  + Correct implementations of each design pattern (15 points per pattern, 45 total)
* Unit tests (20 points)
  + Write a test suite of about 20 tests for your application. The exact number is not important. This will be graded on the quality of the tests (Are they likely to reveal bugs? When they fail, how easy is it to determine what went wrong?)
* Documentation (10 points)
  + Document all code with Javadoc comments. Include file headers and method comments. Make sure to specify all parameters, returns, preconditions, and postconditions.

Submission: Zip up your src and test directories and submit to Canvas.