CS5010, Fall 2019

Assignment 2

Tony Mullen

a.mullen@northeastern.edu

- **Assignment 2 (Modular Furniture) is due by 9AM on Tuesday, September 24.**
- **Please push your individual solutions to your designated repo within the CS5010 GitHub organization.**

Resources

You might find the following online resource useful while tackling this assignment:

- https://docs.oracle.com/javase/8/docs/api/
- https://github.com/junit-team/junit4/wiki/getting-started
- https://www.smartdraw.com/uml-diagram/

Submission Requirements

Your repository should contain a separate package for every problem.

The package names should follow this naming convention: `assignmentN.problemM` where you replace `N` with the assignment number, and `M` with the problem number, e.g., all your code for problem 1 for this assignment must be in a package named `assignment1.problem1`. (There is only one problem in Assignment 2.)

Additional expectations for every package:

- One `.java` file per Java class
- One `.java` file per Java test class
- One pdf or image file for each UML Class Diagram that you create
- All methods must have tests
- All non-test classes and non-test methods must have valid Javadoc

Your packages should **not** contain:

- Any `.class` files
- Any `.html` files
- Any IntelliJ specific files

Lastly, a few more rules that you should follow in this assignment ☺ :

- Your classes should have a `public` modifier
- Instance fields should have a `private` modifier
- Use `this` to access instance methods and fields inside a class

ASSIGNMENT 2: Due Tues Sept 24

Problem 1: Modular Furniture

Have you ever come home from a modular furniture store and begun putting together your furniture only to find that you neglected to purchase some important component that is sold separately? This week's assignment will have you developing tools to try to reduce that problem for customers.

You have been tasked by a large modular furniture store to implement the back end for a service to customers to ease the process of deciding which components to buy. Users will use an online form to select their configuration preferences, for example the color, length, and mounting style, and the system will deliver recommended purchase orders including specific items that must be purchased individually. The system takes as input a JSON file describing customer preferences and outputs a JSON file listing possible collections of items that will meet these criteria, along with costs for each item and total costs for each suggested purchase order.

Available products

The company currently has three lines of modular cabinets, the *Yossarian*, the *Luthien*, and the *Atreides*. Each of these lines includes a variety of cabinets. They also have three lines of door and drawer fronts, the *Slothrop,* the *Belacqua,* and the *Gaga* intended to be paired with the cabinets (the cabinet lines do not come with doors or drawer fronts). There are a variety of fixtures and parts that are general purpose and required by certain configurations of cabinets.

The company emphasizes to you that its product lines are growing all the time, and it intends to have new cabinet and door/drawer lines available which will also be similarly modular. For this reason, the application should be designed with this in mind and be as flexible and extensible as possible.

The following describes the configurations available for each cabinet line and the characteristics of each configuration:

Yossarian comes in the following configurations:

Wardrobe size:

- 72″ height, 36″ width, 16″ depth
- Floor mounted (no feet required)
- Requires wall fixture attachment for earthquake safety
- May include up to 7 shelves
- Available colors: brown, black, bone, oxblood

Half size:

- 36″ height, 36″ width, 16″ depth
- Floor mounted (feet optional, available separately) or wall mounted (standard wall mount rail required)
- May include up to 3 shelves, no drawers
- Available colors: brown, black, bone

Quarter size

- 18″ height, 36″ width, 16″ depth
- Floor mounted (feet required, available separately) or wall mounted (standard wall mount rail required)
- May include up to 1 shelf, no drawers
- Available colors: brown, black, bone

Luthien comes in the following configurations:

Half size:

- 36″ height, 36″ width, 18″ depth
- Floor mounted (feet optional, available separately) or wall mounted (standard wall mount rail required)
- May be fitted with up to 3 shelves, or with a one drawer and up to 2 shelves.
- Available colors: black, bone

Quarter size

- 18″ height, 36″ width, 18″ depth
- Floor mounted (feet required, available separately) or wall mounted (standard wall mount rail required)
- May be fitted with a drawer or with a shelf.
- Available colors: black, bone
-

Atreides comes in the following configurations:

Half size:

- 36″ height, 36″ width, 16″ depth
- Floor mounted (feet optional, available separately) or wall mounted (Atreides cabinets use their own special wall-mounting rails, which are distinct from the standard rails, but must be purchased separately.)
- May be fitted with up to 3 shelves.
- Available colors: brown, bone

Quarter size

- 18″ height, 36″ width, 16″ depth
- Floor mounted (feet required, available separately) or wall mounted (Atreides cabinets use their own special wall-mounting rails, which are distinct from the standard rails, but must be purchased separately.)
- May include up to 1 shelf, no drawers
- Available colors: brown, bone

The following describes the configurations available for each door/drawer line and the characteristics of each configuration

Slothrop comes in the following configurations:

Wardrobe size door:

- 72″ height, 36″ width
- Available colors: brown, black
- Includes handle

Half size door:

- 36″ height, 36″ width
- Available colors: brown, black
- Includes handle

Belacqua comes in the following configurations:

Wardrobe size door:

- 72″ height, 36″ width
- Available colors: bone, oxblood
- Includes handle

Half size door:

- 36″ height, 36″ width
- Available colors: bone, oxblood
- Includes handle

Quarter size door:

- 18″ height, 36″ width
- Available colors: bone, oxblood
- Includes handle

(Continued next page)

Gaga comes in the following configurations:

Half size door:

- 36″ height, 36″ width
- Available colors: brown, black, bone, oxblood
- Requires door handle (separate)

Quarter size door:

- 18″ height, 36″ width
- Available colors: brown, black, bone, oxblood
- Requires door handle (separate)

Quarter size drawer front:

- 18″ height, 36″ width
- Available colors: brown, black, bone, oxblood
- Requires drawer handle (separate)

In addition, the following standard components are available. These are easy for customers to forget, so it's important that the system tell them when they must be ordered:

- 18″ drawers (fits quarter-sized cabinets of at least 18″ depth)
- Door handles
- Drawer handles
- Door hinges (required for all doors)
- Standard wall-mount rails
- Wall fixture attachment for earthquake safety
- Cabinet corner feet (sold singly. Typically floor mounted cabinets should have one foot per corner, but when three floor mounted cabinets are mounted together adjacently, a total of six feet is sufficient).

All cabinets can be installed adjacent to other cabinets of the same dimensions.

Actual prices for each item are not currently available to you, and prices change from time to time. Your solution should take into consideration that each item has a price that can be set and accessed.

Customer criteria

Customers will supply their criteria for a purchase via a web form (which you will not implement) and the customer criteria will be provided to your system by means of a JSON file. For the purposes of this assignment you may regard this file as a static file, but your solution should be easily adaptable to the case where the file is being received over a network, for example as the body of a web request.

The structure of this file is up to you (assume that you and the front-end developers are able to plan together). It should represent the existing customer criteria and use an easily extensible format so that other criteria could be added later without breaking existing functionality.

Criteria will include:

- Total desired width of cabinets in feet or inches
- Preferred height of cabinets in feet or inches
- Number of desired shelves
- Number of desired drawers
- Color (if preferred)
- Floor or wall mount preferences
- Budget priority

Output

Your system will deliver a recommended order to the customer that includes all necessary components in appropriate counts. If multiple combinations exist that would meet the customer's needs, then the system should deliver a reasonable selection of them, ordered by price if budget priority is a consideration. If a preferred color is specified, then results are limited to combinations of that color. Otherwise, all colors may be included, but results will still be limited to uniformly-colored combinations.

The output should also be represented as a JSON file. Again, you should consider how this file will represent the necessary data and return the file in such a way that it can be either sent to a static file or returned as part of an HTTP response. You do not have to implement the web API itself. A tutorial on using JSON libraries in Java can be found here:

https://howtodoinjava.com/library/json-simple-read-write-json-examples/

Classes and interfaces

For Modular Furniture, you should begin by thinking about how the data can be organized in classes, subclasses, and interfaces. Think about how to represent characteristics that are common to multiple different products, such as the commonalities across wall-mounted cabinets or across cabinets that take drawers. Think about what will be necessary to extend the options in the future, and keep the solution general enough that it can be applied to other categories of products.

Your tasks:

1. Decide on expressive, extensible representations for user criteria and for output values which can be represented as JSON files. Prepare to discuss ways in which extensions may be required.
2. Create Java classes appropriate for representing user criteria and output values and implement methods for converting JSON representations to objects.
3. Represent available merchandise using classes, subclasses, abstract classes, and/or interfaces, as appropriate. Be prepared to justify your design decisions.
4. Implement the recommendation logic that takes user criteria and outputs a list of possible furniture configurations matching the criteria, with complete lists of items (and counts of items) necessary to purchase.
5. Write tests to cover new functionality.
6. Provide your final UML diagram that includes all relevant methods.