Requirements and Approach for Unit 3

Unit 3

Your first step is to re-implement our current system, with the same functionality, but with code that's better designed to handle multiple models.

Technical Requirement – Set of Models(Automobile) should be saved using LinkedHashMap. Set of OptionSet in each Model and respective Options can be saved in an ArrayList.

In addition, both Automobile and OptionSet will need some methods for keeping track of which options a user has chosen. To try and keep straight which methods are for defining options, and which are for choosing options, I've put Choice in the name of the new methods related to tracking user choices.

Here's a UML class diagram for additional things in Automobile:

```
Automobile
– make: String
– model: String
– optionSets: LinkedHashMap
- choice: Option
+ getMake(): String
+ setMake(make: String): void
+ getModel(): String
+ setModel(model: String): void
+ getOptionChoice(setName: String): String
+ getOptionChoicePrice(setName: String): int
+ setOptionChoice(setName: String, optionName: String): void
+ getTotalPrice(): int
```

The method setOptionChoice() is for choosing a particular option in an option set. E.g.,

```
setOptionChoice("transmission", "standard");
```

would choose the standard transmission option. After the above choice is set, getOptionChoice("transmission") would return "standard" and getOptionChoicePrice("transmission") would return -815.

To make it easy to define the Automobile methods for tracking option choices, add the following methods to OptionSet:

```
OptionSet
...
...

+ getOptionChoice(): Option

+ setOptionChoice(optionName: String): void
```

setOptionChoice(), given the name of an option, would save that choice inside the option set. getOptionChoice() would return the option chosen, if any, otherwise it should return null.

To test this new Automobile class, write a driver program to read your text input file, populate an instance of Automobile class and print the OptionSet's and their respective options.

Your application should work just as it did before, but will be much closer to being able to handle multiple models. (This means that you can still select an optionset with their respective options and calculate the car price, given user's selection.)

Working with LinkedHashMap –
Note that to access elements in LinkedHashMap you will need to associate an Iterator and then use it to find the elements in LinkedHashMap.

Proposed action items for Unit 3

1. Action 1
    a. Refactor the code to convert Option Array to ArrayList.
    b. Refactor the code to convert OptionSet Array to ArrayList.
    c. Learn how ArrayList works.
    d. Review API and example provided.
    e. Finish this action and test to make sure things are working - Running the test driver from Lab2.
2. Action 2:
    a. Refactor the code to convert the "static Automobile instance" in proxyAutomotive to a "static LinkedHashMap Automobile instance".
    b. Add methods to: add and remove for Automobiles.
    c. Do not change the API - createAuto, buildAuto.
    d. Test this with BuildAuto class instance.
3. Action 3
    a. Go to OptionSet class and add a private variable called Choice of type Option
    b. Write protected Getter/Setters.
    c. Go to Automobile and add the following methods:
        i. + getOptionChoice(setName: String): String
        ii. + getOptionChoicePrice(setName: String): int
        iii. + setOptionChoice(setName: String, optionName: String): void
        iv. + getTotalPrice(): int

Submitting your work

Total points (50)

Pl. review your work against this checklist before submission:

1. Program Specifications / Correctness (35)
    a. No errors, program always works correctly and meets the specification(s).
    b. The code could be reused as a whole or each routine could be reused.
    c. Collection - LinkedHashMap is implemented for group of Automobile class. OptionSet and Option implement ArrayList
    d. Automobile class has way to track user selection (through creation of Option choice variable and related functionality).
2. Readability (5)
    a. No errors, code is clean, understandable, and well-organized.
    b. Code has been packaged and authored based on Java Coding Standards.
    c. Text input file is readable and easy to follow.
3. Documentation (5)
    a. The documentation is well written and clearly explains what the code is accomplishing and how.
    b. Class Diagram is provided.
4. Code Efficiency (5)
    a. No errors, code uses the best approach in every case. The code is extremely efficient without sacrificing readability and understanding.