Project – Unit 3 - Making your kbb (Kelly Blue Book) like app scalable

Requirements (Written by someone in Marketing)

This project needs more work to make sure it's functioning correctly, and especially to ensure that it does the right thing when there are multiple users. Here is a scenario to consider:

How will the system scale if multiple users were to update the same model, same OptionSet or perhaps the same Option value.

Plan of Attack

To ensure that your application can handle concurrency, we have to add new functionality in new classes without modifying our existing code too much.

What I am saying is that proxyAutomotive and BuildAuto (combined with interfaces) is a component of a sort that we want to keep intact. Our Model package containing Automobile, OptionSet and Option classes is representing the data for one Model.

If we want to setup a scalable structure for kbb we added a LinkedHashMap in proxyAutomotive (abstract class) with some simple API's to modify it. (We can always add more methods in interfaces at any time so we are not going to worry about that for now).

We want to add a feature that will improve concurrency which can be done as follows:

1. Design a class called **EditOptions** that can be used to edit Options for a given model in its own thread. You should put this class in its own package called **scale**.
    a. Be sure to use the LinkedHashMap instance of Automobile (that is a static object) in proxyAutomotive class instance.
    b. Consider synchronizing methods in classes that are used in setup of LinkedHashMap instance of Automobile ((that is a static object) in proxyAutomotive class instance). You will have to synchronize all method used for creating, reading, updating and deleting parts of Automobile, OptionSet and Option classes.
2. Now code the new driver class and test it as follows:
    a. Create a driver program for this unit that allows:
        i. Instantiating BuildAuto
        ii. Create two threads using EditOptions that will modify the same model that is a LinkedHashMap instance of Automobile ((that is a static object) in proxyAutomotive class instance).

    b. Test your implementation to ensure that two threads altering same property does not cause data corruption. (Use Goofy.java semantics for this step).
3. Your deliverable for this part:
    a. Updated classes and class diagram
    b. Test program showing the successful implantation of these classes.

Design Considerations

Consideration 1

You have to synchronize methods in Model package. Since methods in OptionSet and Option class are protected you don't really have to synchronize those methods. However Automobile class methods will need to synchronized.

Consideration 2

You have to use instance of Automobile LinkedHashMap in proxyAutomotive (abstract class) in EditOptions class. What is the best way of enabling this interaction?

I am asking you to design this part considering the following:

1. Create an API to expose ProxyAutomotive class through BuildAuto and an Interface. This time however the methods should be accessible internally providing access to the Automobile LinkedHashMap instance. EditOptions class and BuildAuto class will be associated with each other through an API (Interface)
   (This exercise will help you learn how to design Interfaces and value proposition of Java Interfaces. You will need to design this part of the exercise without help from anyone.)
2. EditOptions should use synchronized methods in the Automobile class for operating on Automobile LinkedHashMap instance in proxyAutomotive.

Consideration 3

In this lab I only expect you Change the existing value(s) of an OptionSet and Options. Please do not overwhelm yourself with coding all cases since our goal is to learn multithreading.

Grading your submission

Total 50 mins.

1. Progam Specification/Correctness (35 points)
    a. Class diagram is provided.
    b. No errors, program always works correctly and meets the specification(s).
    c. The code could be reused as a whole or each routine could be reused.
    d. Multhithreading is implemented in EditOptions class or in a separate class.
    e. Object locking is implemented in methods of Automobile class.
    f. Interface is used to enable interaction between EditOption and BuildAuto class.
    g. Demonstrates (code and test) Object Locking usage. (Removing synchronization causes data corruption).

2. Readibility(5)
    a. No errors, code is clean, understandable, and well-organized.
    b. Code has been packaged and authored based on Java Coding Standards.

3. Documentation(5)
    a. The documentation is well written and clearly explains what the code is accomplishing and how.

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.