Design Rationale

Refactoring Techniques

Extract Method and Extract Variable

The createContract method inside the OpenBidAction class was too long. Since this method is an important one, in order to improve readability the sections of code which deal with retrieving bid offer data from other classes of the system were grouped together in the creatOffer method which is now called to get offer data.

The **Extract Method** refactoring technique is used to split the showSubDetails method in CloseBidAction class. The section of code in showSubDetails concerned with the details about teaching arrangements in a close bid, is moved to the showLessonDetails method. This is done so we are able to isolate specific UI texts, labels and messages which in turn improve the readability of our UI code logic.

The **Extract Method** refactoring technique is used in MakeOpenBidOffer class to isolate UI specific code from the constructor to the showUI method. This is done so that all UI responsibility is placed on showUI and the constructor can now only deal with creating the bid offer instance. Furthermore the sections of code to create the bid message and for posting the message are fragmented into their own methods: createMessage and postMessage methods. This improves overall readability as well as frees the built in actionPerformed method to only handle button clicks without having to worry about the logic that needs to be executed. Initially, the postMessage method also handled calls to the web api but the GUIAction interface already has concrete implementation to do just this. So we removed the code to call the api in postMessage method and replaced it with a call to updateWebApi method in GUIAction. This is done to remove repeated code from the system as having repeated code raises the risk of having to fix numerous codes after a bug is found.

The very first implementation of the viewContractAction class had a single method, showFinalizedContracts() which was 121 lines long and it was to show the signed contracts of the student. The showFinalizedContracts() was a bloated method due to its length which is bad design since there were sections of the method that can exist independently[1]. Moreover this made readability difficult. The **Extract Method** refactoring technique is used to split the large showFinalizedContracts() method into five independent methods each concerned with a specific task[1]. For a given node:

1. getStudentAndTutorNames() method would provide the details of the two parties
2. getcontractDetails() method provides the agreed contract details between the two parties
3. showFinalizedContracts() method simply displays the data received from the previous two methods in the UI.
4. contractNotification() method checks if a contract is about to expire soon or not
5. and expiryNotification() method adds an expiry alert on each contract.

The first three methods are where we applied the Extract refactoring method to separate the data retrieval methods - getStudentAndTutorNames() and getcontractDetails(), from showFinalizedContracts() . Therefore we were able to isolate independent sections of the previously bloated showFinalizedContracts() method into three separate methods which in turn

also improved readability[1]. The fourth and fifth points prove the advantage of using the Extract method because with the refactored design we could easily extend the class to include two new methods for contract expiry notifications[1].
Furthermore since showFinalizedContracts() displays the data it received from getStudentAndTutorNames(), getcontractDetails() and expiryNotification() methods, the **Extract Variable** refactoring technique is used to place the complex return from the methods into readable variables[1]. This further improved the readability of the code.

Disadvantages
The use of the Extract Method caused our class to have 5 methods in total which puts too much responsibility on the viewContractAction class. The use of the Extract Variable refactoring technique means that the viewContractAction class has lots of local variables in it which increases the overall length of the class and makes it cluttered[1].

References:
[1] Week 9 lecture slides 13, 19 and 21
[2]
[3]
[4]
[5]

(Draft 2)