FACADE DESIGN PATTERN

One of the design pattern that will be very useful to use for the ClockrBot system is the Facade design pattern. The Facade pattern provides a single interface with the main purpose is to simplify a complex system. The way Facade pattern simplify a complex system is to break the whole system into each small subsystems that have its own responsible interaction sequence functions. We can refer each subsystems as classes, so in regard to that, the ClockrBot system will have three classes name LoginPage, GaPage, and the SupervisorPage. For the LoginPage class, the backend of it the username is declared as a string and the password is declared as an integer. Once the username and password is retrieved from the user, the verifyUsername() and verifyPassword() function perform its role by verifying the inputted data to determine if that input data is for the GaPage or SupervisorPage to direct to its appropriate class. If the username and password data is verified as Ga then the GaPage class will be activated. In the GaPage class, some of its particular function are clockIn(), clockOut(), break(), and logout(). The responsibility of the clockIn() function is to perform the action behind the scenes when the user clicks the clock in button, it should be able to retrieve that specific time of clock in data and display it to the Ga user interface and also sends that data to the supervisor class so that it can be retrieve for the supervisor to obtain that data. The clockOut() and break() functions performs the same actions just like the clockIn() function that was just described. The logout() function performs the action by deactivated the Ga account, but it does not send that data to any other classes. So all those functions is pertain to the GaPage class.

If the username and password data is verified as supervisor then the SupervisorPage class will be activated. In the SupervisorPage class, some of its particular function are disable(), edit(), clockGAOut(), undo(), infoReport(), viewRealTime(), printTime(), and logout(). The responsibility of the disable() function is to perform the action of disabling the GA’s account by restricting to a particular GA from having access to the account. Once the Supervisor clicks on the disable button, from the backend the disable function will perform the action from the SupervisorPage class and sends that data to the GaPage class to take the effect on disabling it. The edit() function perform the action by sending the update changing data to the GaPage class to take the effect on the new changes that the supervisor has made. The clockGAOut() function, the action is done from the SupervisorPage class. Once the Supervisor manually clocks out the Ga the clockGAOut() function data gets sent to the GaPage class to take effect on the changes. The undo() function, data gets sent from the SupervisorPage class to the GaPage class back and forth based on whatever action the supervisor has taken. The infoReport(), viewRealTime(), printTime() functions those function pulls data from the GaPage class to retrieve the Ga’s information to purposely display the data onto the supervisor interface. The logout() function performs the action by deactivated the supervisor account, but it does not send that data to any other classes. With all the backend information from the performance of each sub-systems that was describe from above, by using Facade design pattern the user will only experience with just a single interface of the system without having to know the backend complexity of each single steps about the system performance, the single interface will take care of the rest.

\*\*\*\*\*\*\*\*\*\*\*\*For the Facade Pattern Diagram Cherry said that I can just reference it to Joseph's UML DIAGRAM\*\*\*\*\*\*\*\*\*\*

Composite Design Pattern

Another design pattern that will be very useful to use for the ClockrBot system is called the Composite design pattern. The Composite design pattern is a structural pattern which modifies the structure of an object. It is suitable for the need to work with objects which forms a hierarchy tree. The ClockrBot system will be a hierarchical structure that need a common functionality across its structure such as when the system deals with multiple GA’s accounts that has all the same attributes functions. It can be consider that the supervisorPage class would be the parent managing multiple GA’s accounts, which the GAPage class can be consider as many children. An example of hierarchy tree for this ClockrBot system would be the Supervisor page will be the parent, many GA’s page will be the children, and many features both the Supervisor and GA’s page has will be consider objects as many leaf nodes. The advantage of using the Composite design pattern is that it provide an organize hierarchical structure to the system in an efficient manner. The Component such as the SupervisorPage class has the ability to declare an interface for accessing and managing its child components, which refer to as the GaPage class, and defining the behavior for the primitive objects in the composition would refer to as the features the Supervisor and the GA’s has in its own page.

\*\*\*\*\*\*\*\*\*\*\*\*\*For the Composite Design Pattern Diagram please check it out at "COMPOSITE HIERARCHIAL DIAGRAM"\*\*\*\*\*\*\*\*\*\*\*\*\*\*