EECS 3311
FALL 2019

PROJECT REPORT

NAME: RAJANBIR SINGH

CSE LOGIN: rajanbir

REPORT:

For basic overall design of the classes in my project, I have chosen a stable and conservative approach for the design of classes.

I chose to make use of the composite design pattern for the classes in “Language of Expressions” table of project description. This was done to make sure that OPEN-CLOSED principle was obeyed in the design.

Furthermore, the classes in the outer-structural components of the analyzer (for eg, model classes in the sub-cluster CLASS_STRUCTURE, like ATTR_OBJ, CLASS_OBJ, COMMAND_OBJ, QUERY_OBJ, have their names components with the export status {NONE} so that client cannot refer to those insider structural components without an accessor method (like get_name for CLASS_OBJ). This is the part of the model cluster that’s kept hidden from the client. Since it’s hidden, this is also the part that may be changed at the supplier’s comfort, which means the design pattern implemented makes it possible to add more features to the analyzer, without changing much design of other classes and/or without making much changes in the model cluster.

Apart from the hidden data structures, the commands portion (which basically forms part of the ABSTRACT USER INTERFACE) is the one that’s kept revealed (not hidden) to the client. Thus, making the commands section stable. No matter what happens on the supplier’s side of code (any changes, implementation improvements, any design pattern changes, etc) may not be shown to the client so that the client’s side of code remains stable and concrete.

All this was done to follow the INFORMATION HIDING principle closely, keeping in mind that any future additions or design changes to the project do not result in any expectation changes for the client.

The model and etf_model_access features in the key classes of the model cluster were designed in such as way as to make sure of the SINGLETON ACCESS of the model and it’s features was not violated.

This also makes use of the UNIFORM ACCESS PRINCIPLE, as access to the model remains singular and consistent across all classes. And only etf_model_access provides access to the model, this is done to make sure that not all classes have the privilege to tamper with the model structure and it’s components.

This design pattern choice further increases the security and stability of the classes and their individual features.

The composite pattern improves the design by enabling a hierarchal structure, and encourages general functionality across the structure.

For eg, in the context of this project's user tool coded, the assignment instruction operations like addition, subtraction, modulus, etc, use double inheritance (in theory). They inherit form Expression class, and the Composite class and theres components that can be specified individually (without operators, like Boolean constant, Integer constant) and then there's components that have to be specified as compositions of expressions (with operators, like addition := Integer constant + Integer constant). The design choice of these classes makes the gist of composite design pattern.

This also makes use of the principle of non-code-repetition. Having more classes with less code in each of them, forming cell-like structures, each of them being part of the bigger structure and contributing individually to support operations like pretty printing, analysis, type checking etc. All this leads to code being evenly distributed among classes (more understandable and easy to make changes in the future, less confusing to read and debug, etc.) and less code repetition, as compared to less number of giant classes containing majority of the code, defeating the purpose of Object Oriented Programming (meaning huge chunks of code and repetition amongst classes, similar looking code, less genericity, etc).

I did not implement the visitor design pattern with the composite, as I felt that it would over-clutter things in the model structure. Although it was a strong contender in the beginning while making choice of which design patterns to use, I felt like I could do without it, and chose to move forward with the composite design pattern for most part of the project (Context free grammar of expression language).

Note: Please find the BON diagrams attached in the "docs" folder of online submission. I couldn't include them here in the hard copy as they were getting printed very blurred and unreadable.