Kenan Abdulky

201504037

EECE 437_Assignment 1 Code Review

Prof. Zaraket

Review of Kamal Haddad's code

The purpose of this paper is to provide a brief code review of Mr. Kamal Hadad's assignment 1 submission. The project was written in java programming language with each class written in a separate .java file.

General review:

In general, the code was written in a nice and neat format. The decision of class hierarchy was done in an elegant and logical form. The code was easy to read and did not include many forms of complexity. The coding style was also consistent in all the .java files. The project includes 28 .java files that each include a class. The maximum level of inheritance was 3, which is suitable for such an assignment. The project also included around 1,400 lines of code including comments.

Ratings:

Readability: 8/10. The code was not complex, and yet it does what it should.

Code style consistency: 7.5/10. The code style was neat and consistent.

Form:

The name of the repository indicates that the project is an EECE 437 assignment. However, it does not show the purpose of the assignment. The repository included no generated files (.class files). It included a folder "src" that includes all the .java files.

In all the classes, the class, variable and methods names were chosen consistently and were able to reflect their functionality and what they actually represent. However, the parameter names passed inside the methods and constructors were not meaningful.

Operation:

The code fully compiles and produces promising results. While testing the code, no bugs were faced. If wrong or inconvenient values were input, then either a compile or a runtime error (exception) would be shown in a way that implements what was asked for in the code submission.

Content and Coupling:

The class hierarchy diagram found in the repository shows clearly how the inheritances were implemented. The main classes were implementing types, values, terms and formulae as requested in the submission guidelines. The repository also had a README file that clearly states how the code was divided and then implemented, along with the design decisions. Moreover, the constructors, evaluate and list methods each had their own functionalities that do work as expected.

The TAPair class was written in a neat way that implements what it should. The class couples primitive data types along with dynamic values. The way it was implemented is that each pair is a pair of TADataType objects. TADataType is the super class of all primitive and dynamic variables. Since all the primitive types such as (TAInt, TADouble…) and dynamic types such as TAArrays are instances of TADataType, then the first() and next() methods return a copy of TADataType object that was at first passed to the TAPair constructor.

Issues:

The classes written did not have a static name property. Instead, the List() method took care of the class names using an "if" condition. Furthermore, no TAGreaterThan, TAGreaterThanOrEquals nor TALessThanOrEquals classes were implemented. Moreover, the code in general lacked comments; the comment to code ratio is very little and does not exceed 5% of the total number of lines.

General concerns:

The submission did put the problem into context. The code hierarchy and style were well-ordered and consistent. The submission took into concern what was asked for in the submission guidelines. Code maintenance and additions would be easy since the code is easily readable and the variable along with the class names represent what they actually were implemented for. Furthermore, the project does not provide additional features (wants). However, it successfully implements most of the (musts) that were asked for.

The submission clearly states the design decisions. In the repository, there is a design decisions text file that explains briefly the design decisions made and the purpose of each decision. A good example would be the TADataType design decision mentioned earlier in the "coupling" section of this paper. This decision has solved the problem of couplings is an easy and a straight way.

Conclusion:

The code is very readable and its style was very consistent. The class hierarchy was simple and logical. Although the code lacked comments and overall documentation, it was easily readable and comprehendible. The variable, method and class names were meaningful and have provided a general idea about what they are actually for. However, most of the parameters passed to the class methods and constructers were meaningless and do not show what they are for. In addition to that, the lack of static name fields was not faithful to the submission guidelines. However, this issue was handled in the List() functionality using an if statement.

For future submissions, the author should name the parameters passed with meaningful names. Moreover, the author should also provide the code with more documentation that would help others understand his code more easily. It would also help him remember the code's purpose in future observations or editing of the code.