La Vité

Vilen Milner
Michael Rossinski
Qi Hang Yang
Tianyi Zeng
Arnob Talukder

Table of Contents

Code Review Summaries

Vilen Milner

I inspected the DataConverter.java class on the backend written by Tony

- Documentation and Naming Conventions: There is no documentation. This can be a challenge if we have to extend the class or add new functionality. There is no spelling or grammar mistake in the code though. The class follows standard Java naming conventions (camelcase, mixedcase). There is also no duplicate code, nor any code that seems extra or redundant.
- Code Structure and SOLID: The code is well structured and not at all cluttered. All applicable SOLID principles are also followed here. For example, it has a single role it performs well, and can be extended for future features. The other principles perhaps do not apply as much to this class specifically
- Understandability and Proper Functionality: The code is very understandable to me despite not having been the one to write it, there is nothing confusing or unnecessary. The code also works without any problems.
- Understandability and Coverage: Testing is not well done, there does exist a test file, however it is located inside the main project package, not the testing package. Furthermore, it is written with a main method instead of as JUnit tests. Even further to the more, there is only one test in there which is not enough. While the limited test cases are understandable, the coverage is lacking.

Qi Hang Yang

Code review of the DbSelectHelper.java file.

- Documentation: There are minimal documentation, which is acceptable for the function of the helper. The code is self explanatory for the most part. The parts that require some explanation are present. One improvement that could be done is to add descriptions to classes that are imported. Since the forms are not self-explanatory, a developer that does have access to those classes will not know the functionality of the classes.
- Naming Convention:
  The naming is consistent within the file. There are camel cased variable names and function names. It is also stylistically consistent between different functions, having the brackets and spacing the same.
- SOLID:
  Follows SOLID design here since the purpose of the class is to act as a helper for other classes, most of the principles are not touched upon/does not apply. The major principle within SOLID that the code uses is the Open/closed principle, which the code follows.
- Code Structure:
  Code is well structured. There are no unnecessary lines of code or any variables that are declared and not used. The code does not have any glaring problems.
- Understandability:
  Very easy to understand for most of the function within the helper file. The most complicated function buildConditionFieldsString() is not hard to understand. It would be better if there are documentation, which can help with the process of understanding the purpose of the query builder.

Tianyi Zeng

Inspection of the RequestHelper.js file in WebContent/helpers/

- Documentation: Lack of documentation in the request helper class. This was a problem upon inspection as I did not help create this piece of code, so it was required for me to trace many of the methods manually in order to fully understand the function of the class. Also, the writer of the code did not document his name, making it hard to go to the author for future potential problems.
- Naming Convention: Many of the methods were named straightforwardly, and self explanatory. Did not required additional assistance when figuring out what a method should do, however as mentioned above, the lack of documentation made it hard to trace through the code itself.
- SOLID and Code Structure: The code is clean and neatly organized. Easy to go through and trace. It satisfies all the SOLID principles, it has single responsibility and can be added on for future use.
- Understandability and Coverage: There is no test suite for RequestHelper.js. This may be because it is connected to the front end, so it is difficult to create mock objects or generate unit test cases for front end functionality.

Michael Rossinski

In-depth code inspection of the FieldBuilder.js file.

- Documentation: Internal documentation is scarce within this file. Only 3 out of the 2 methods within the FieldBuilder.js file are actually commented. Those that are commented however, only have about 1-2 lines of comments only briefly explaining a general use of the method. findForm() and generateField() are both completely uncommented. The other methods also require more in-depth documentation in order to understand what their functionality is.
- Naming Convention: All methods follow the same naming convention, that is camel notation. Some variables names follow camel notation, such as ‘submitButton’, but others do not, such as ‘textbox’. Therefore, some of the variables don’t follow the same naming convention and should be edited. Aside from this small issue, nothing is problematic in this category.
- Code Structure: The findForm() function is completely fine in its structure, only needing 2 lines. The generateField() function also follows a good code design and can’t be shortened. generateFieldString() seems to also not have any problems. generateFieldBoolean(), however, has a lot of lines that can be shortened, since two of the same types of objects are being created. generateFieldSelect() also cannot be shortened.
- Understandability: Understanding the code is laughable, since no component is properly documented and explained. If a worker was to inspect this code and interpret it, they would have strong difficulties. I myself also encountered difficulties understanding the code.

Arnob Talukder

Code review of the QueryBuilder.java file.

- Documentation: There are functions that are named similarly, but the lack of documentation makes it hard to understand the difference between the various functions. For example, addParamFromRequest and addParam are both used to add data to a SQL insert statement, but without documentation, it is hard to know which method to use when.

- Naming Convention: All methods and variable names follow camel notation. All is well in regards to naming.

- Code Duplication: As an example, generateParamsList and generateQuestionMarks methods use identical loops. This could have easily become a common function.

- Violations of SOLID: The various violation of the SOLID principle will surely hamper future development. As of now, no sort of interface/abstract class is used, and thus adding any other sql queries must result in modification the preexisting QueryBuilder class – a clear violation of the Open/closed principle.

- Test Coverage: No tests or documentation for how to test were found. This is most likely due to the class being connected to the front end, making internal testing difficult, but instructions on how to test using the GUI would be much appreciated.