Proficious

November 12, 2018

Code Review 1

Team Members:

- Alexander Greff (greffal1) alex.greff@mail.utoronto.ca
- Saad Syed Ali (alisaad2) saadsyed.ali@mail.utoronto.ca
- Mohammed Osumah (osumahmo) mohammed.osumah@mail.utoronto.ca
- Gyeongwon Choi (choigyeo) gyeongwon.choi@mail.utoronto.ca

Table of Contents

Strategy

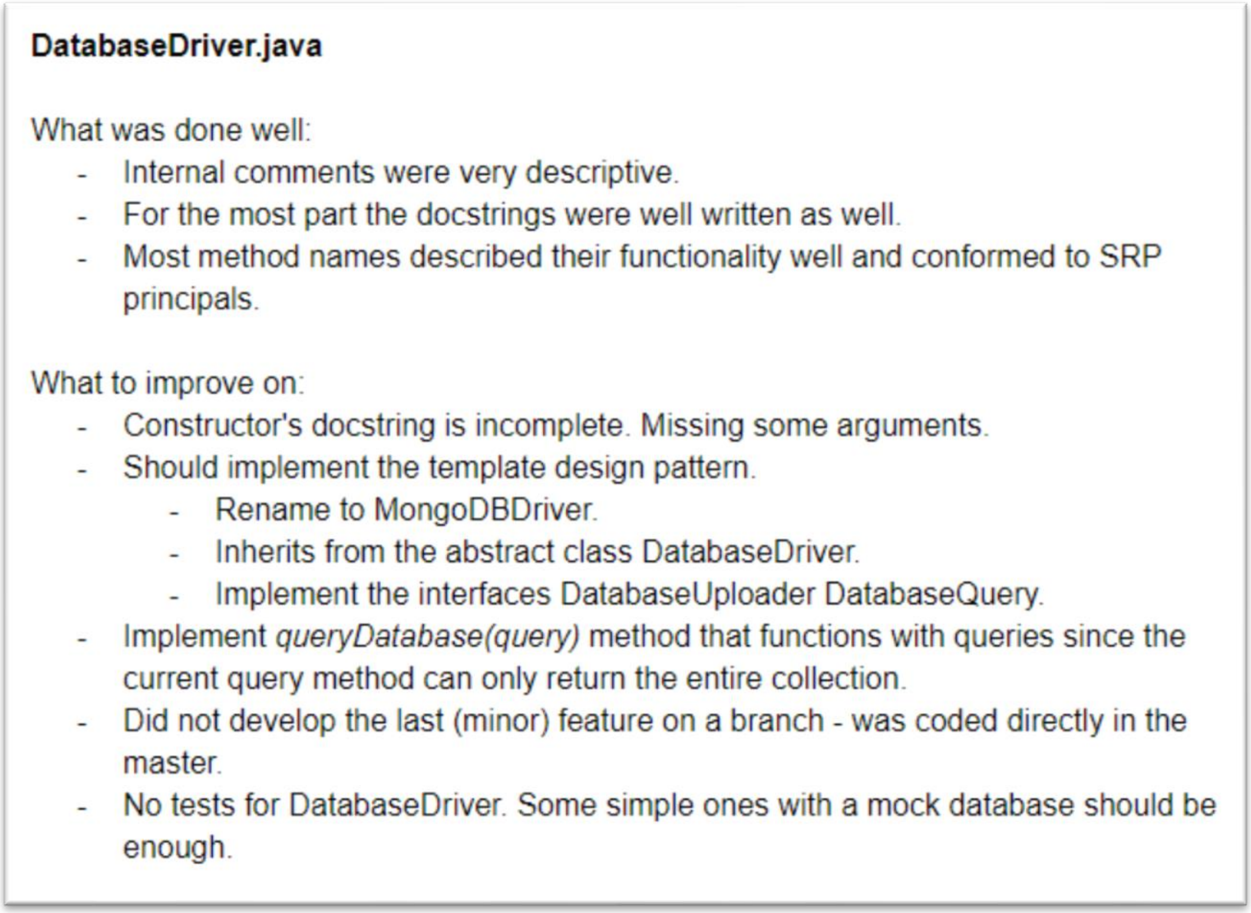
- Look for how well SOLID design principals and the various design patterns are implemented. Is there anything that could be changed that would make it have a better design?
- Check to see that the code conforms to SRP principals.
- Look for how well documented the code is. Is the internal code commented well? Are there docstrings? Are the variables and methods named appropriately?
- Check to see if the code matched how they were presented in the tasks (verification).
- Check to see if the code helps meet the expectations of the clients (validation)
- How well make are the corresponding unit/integration tests for the code? Do they cover all edge-cases adequately?

Review Distribution

Group Member	System	Package
Saad	Project Utilities Template System	com.team5.utilities com.team5.template
Won	GUI Interface	com.team5.gui
Alex	Database Connection System	com.team5.database
Mo	Report System	com.team5.report

Review Summary

- Alex:



DatabaseDriver.java

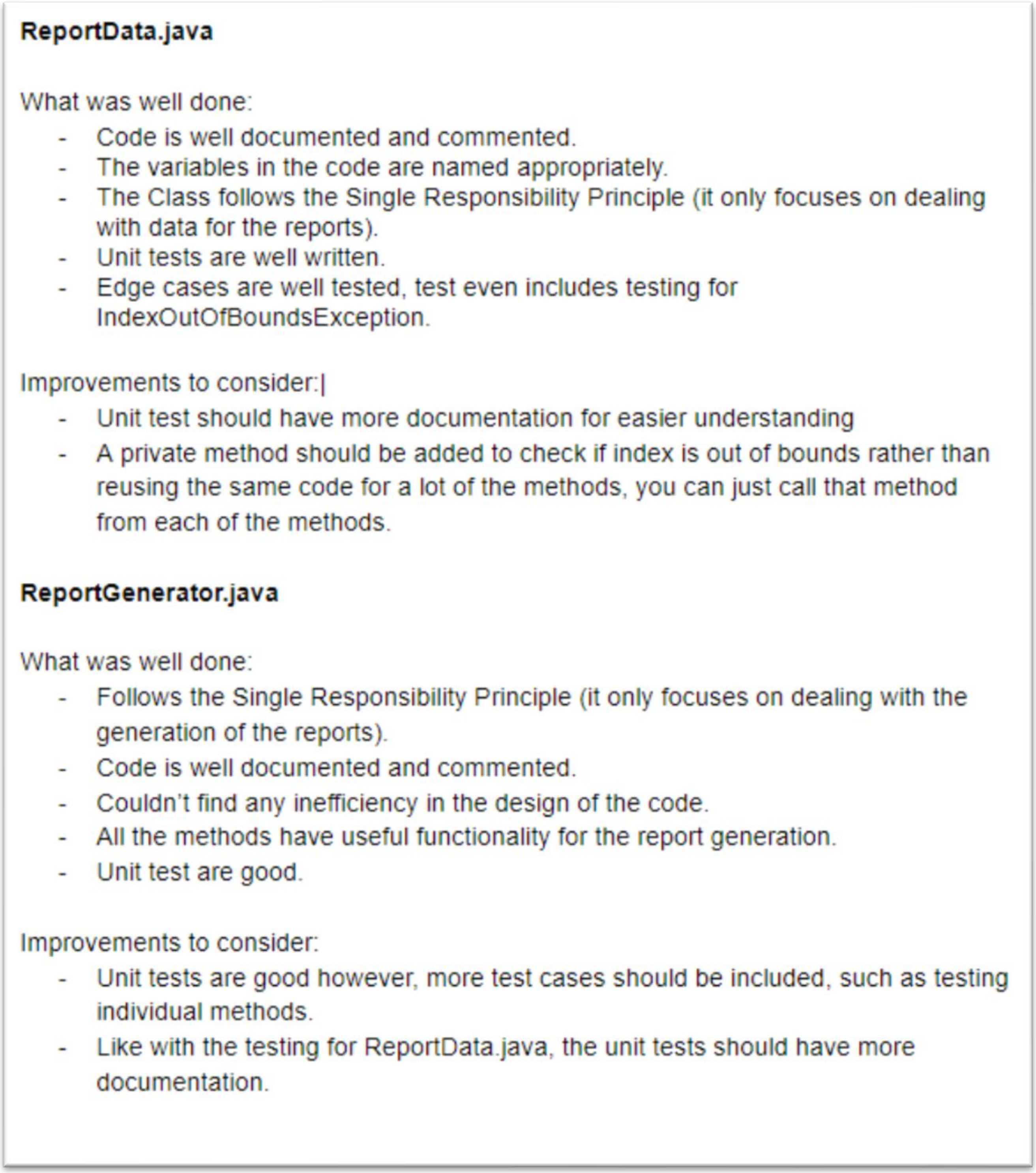
What was done well:

- Internal comments were very descriptive.
- For the most part the docstrings were well written as well.
- Most method names described their functionality well and conformed to SRP principals.

What to improve on:

- Constructor's docstring is incomplete. Missing some arguments.
- Should implement the template design pattern.
  - Rename to MongoDBDriver.
  - Inherits from the abstract class DatabaseDriver.
  - Implement the interfaces DatabaseUploader DatabaseQuery.
- Implement *queryDatabase(query)* method that functions with queries since the current query method can only return the entire collection.
- Did not develop the last (minor) feature on a branch - was coded directly in the master.
- No tests for DatabaseDriver. Some simple ones with a mock database should be enough.

- Mo:



ReportData.java

What was well done:

- Code is well documented and commented.
- The variables in the code are named appropriately.
- The Class follows the Single Responsibility Principle (it only focuses on dealing with data for the reports).
- Unit tests are well written.
- Edge cases are well tested, test even includes testing for IndexOutOfBoundsException.

Improvements to consider:|

- Unit test should have more documentation for easier understanding
- A private method should be added to check if index is out of bounds rather than reusing the same code for a lot of the methods, you can just call that method from each of the methods.

ReportGenerator.java

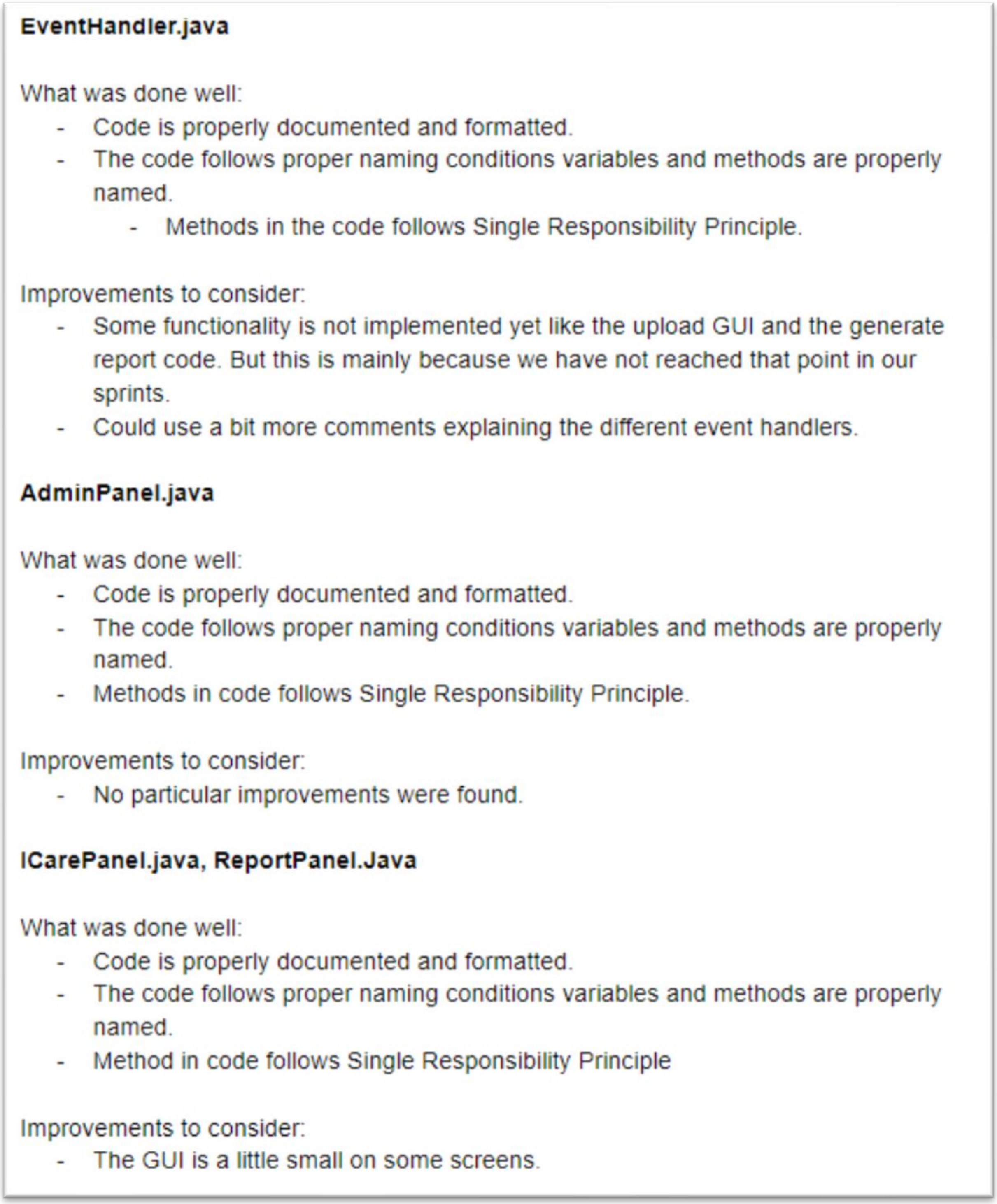
What was well done:

- Follows the Single Responsibility Principle (it only focuses on dealing with the generation of the reports).
- Code is well documented and commented.
- Couldn't find any inefficiency in the design of the code.
- All the methods have useful functionality for the report generation.
- Unit test are good.

Improvements to consider:

- Unit tests are good however, more test cases should be included, such as testing individual methods.
- Like with the testing for ReportData.java, the unit tests should have more documentation.

- Won:



EventHandler.java

What was done well:

- Code is properly documented and formatted.
- The code follows proper naming conditions variables and methods are properly named.
    - Methods in the code follows Single Responsibility Principle.

Improvements to consider:

- Some functionality is not implemented yet like the upload GUI and the generate report code. But this is mainly because we have not reached that point in our sprints.
- Could use a bit more comments explaining the different event handlers.

AdminPanel.java

What was done well:

- Code is properly documented and formatted.
- The code follows proper naming conditions variables and methods are properly named.
- Methods in code follows Single Responsibility Principle.

Improvements to consider:

- No particular improvements were found.

ICarePanel.java, ReportPanel.Java

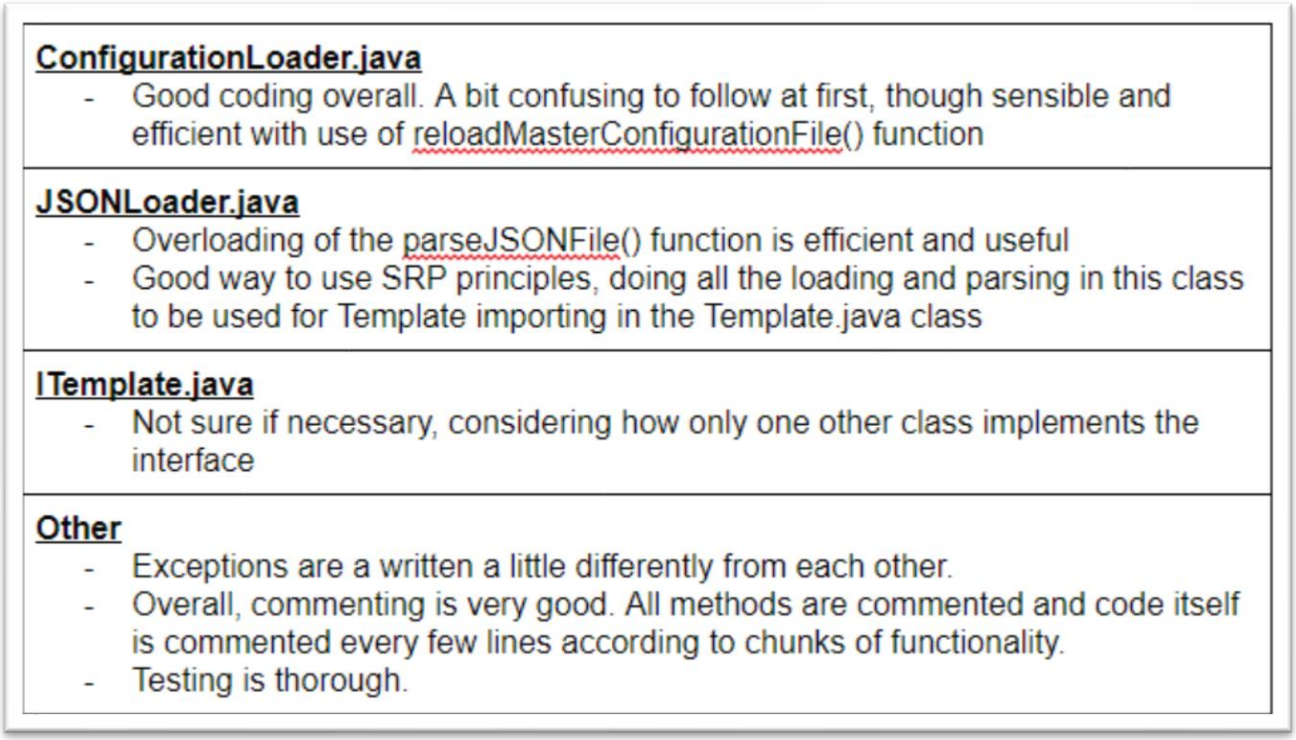
What was done well:

- Code is properly documented and formatted.
- The code follows proper naming conditions variables and methods are properly named.
- Method in code follows Single Responsibility Principle

Improvements to consider:

- The GUI is a little small on some screens.

- Saad:



ConfigurationLoader.java

- Good coding overall. A bit confusing to follow at first, though sensible and efficient with use of reloadMasterConfigurationFile() function

JSONLoader.java

- Overloading of the parseJSONFile() function is efficient and useful
- Good way to use SRP principles, doing all the loading and parsing in this class to be used for Template importing in the Template.java class

ITemplate.java

- Not sure if necessary, considering how only one other class implements the interface

Other

- Exceptions are a written a little differently from each other.
- Overall, commenting is very good. All methods are commented and code itself is commented every few lines according to chunks of functionality.
- Testing is thorough.