**581 Software Maintenance**

**Report for Project 2**

**Team 6**

David Kelly

Denis Anzora

Garhgaj Singh

**Overview**

The quality of Elasticsearch was very good before we started our changes. We used advanced tools such as SonarLint, VSCode and the many maintenance tools in Intellij. We also used plug-ins such as MetricsReloaded and Adapter for Eclipse Code Formatter (Allows using Eclipse's Java code formatter directly from IntelliJ). We were able to make improvements to obtain a better product. In summary, the follow changes were completed (the details are in “The implemented improvements”):

1. Updated 3rd party libraries.
2. Local variables should not shadow class fields.
3. Duplicate strings were turned into constants.
4. Adding final clause where data was not changed.
5. Fixed violations of the Law of Demeter.
6. Fixed linting errors in readme files
7. Some variables were not explicitly initialized.
8. Some resources were not closed and caused memory leaks.
9. Refactored Methods that had identical implementations.
10. Generic wildcard types should not be used in return types.
11. Constructors of an abstract class should be declared public.
12. refactored collapsible “if” statements into one statement.
13. Solving the “hide” utility class public constructor.
14. Converting abstract class to interface.
15. Replacing the use of the system.err by a logger.
16. Added private constructors with no arguments.
17. Removing the unnecessary redundant Boolean.
18. Removing parentheses around the t parameter of single lambda.
19. Refactored methods to reduce the files cognitive complexity.
20. Fixed a code smell regarding Stream.toList()

**Metric Project Details**

* LOC - All lines of code in Elasticsearch is 3,267,580. Due to the large number of LOC, more than one developer will need to maintain this project. There are currently 1,736 contributors.
* Number of files: 21,971
* Matrices for Object Oriented Design - Mood
  + Method Hiding Factor (MHF) - 199.83%
    - Used for encapsulation.
  + Attribute Hiding Factor (AHF) - 79.69%
    - Used for encapsulation
  + Method Inheritance Factor (MIF) - 87.37%
  + Attribute Inheritance Factor (AIF) - 83.19%
  + Coupling Factor (COF) - .18%
  + Polymorphism Factor (POF) - 4.28%

**Metrics**

* Changeability

The Elasticsearch program is written using OOP principles. Inheritance is implemented using base classes to encapsulate functionality that can be reused.

Interfaces are used to improve the levels of abstraction. The interfaces are defined as containers that store the signatures of the methods to be implemented in the code segment. They also enforce the OOP structure.

* Testability

Testing is a big part of Elasticsearch. In fact, there are over 15k of unit and integration tests that take over an hour to run on a fast machine. The structure of the tests in the ES project allows a developer a pattern to follow for testing. There is also a testing readme file TESTING.asciidoc that details exactly how to run all or subsets of tests.

* Stability

The ES program is considered stable due to many unit tests that need to run clean before a release is sent out. Also, the program runs on Java which is considered to be easy to use and is therefore easy to write, compile, debug, and learn more than other programming languages. Since Java is object-oriented, ES is very modular.

* Performance

This is a very important metric for ES. The queries must return as fast as possible to make the program enticing for a user and a company. Cache is used throughout the code to improve performance. Binary operations are used to enhance performance.

* Maintainability

The ES program is complicated but there have been extensive efforts to make maintainability easier. One aspect is that the program has several readme files to help in testing, running, setup and execution of managing the code.

**The implemented improvements**

We focused on a few areas for improvements: Benchmarks, Build-tools and Build-Convention. Each team member pushed changes into their own branches.

* Benchmarks – David Kelly - Branch [daveKelly-Deliverable2](https://github.com/denisanzora/ElasticSearch581/tree/daveKelly-Deliverable2)
  + Local variables should not shadow class fields - See [commit](https://github.com/denisanzora/ElasticSearch581/commit/4ae00000fb138ce03ccceaacb75681d431d60920) and Overriding or shadowing a variable declared in an outer scope can strongly impact the readability, and therefore the maintainability, of a piece of code. Further, it could lead maintainers to introduce bugs because they think they’re using one variable but are really using another.
  + Updated the 3rd party libraries to take advantage of security patches and improved methods. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/1628d9aa18bd6f094dce145836d56d1fd4db23b3)
  + Looked for duplicates – this found strings that were changed into constants that were reused. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)
  + Silent Code Cleanup – This found items such as adding a final clause where the data should not be changed. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)
  + Checked and fixed linting errors in readme files. This helped to make reading the readme files easier. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)
  + Fixed violations of the Law of Demeter. The Law of Demeter or principle of least knowledge is a design guideline for developing software, particularly object-oriented programs. Similar to the Single-responsibility principle. The advantage of following the Law of Demeter is that the resulting software tends to be more maintainable and adaptable. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13) for file [AggConstructionContentionBenchmark.java](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13#diff-be035e647df5f6b4f46ae1f11b8c0036faa0af3922b6bdacf8fbd768d62cbda6), lines 141-145.
  + Some variables were not explicitly initialized. By initializing variables, we can avoid run-time errors such as null or out of range issues. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)
  + Removed unnecessarily qualified static access. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)
  + Resources should be closed:
    - An object BytesStreamOutput was not being closed. The classes that implement the Closeable interface or its superinterface, AutoCloseable, need to be closed after use. Failure to properly close resources will result in a resource leak. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/6b3e496d33432f0c21233c59f8e619a322180a0d)
  + Refactored Methods that had identical implementations. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/3b99f8d34dc59668bc7a742a6da8c98150c5611f)
* Build-tools - Denis Anzora - Branch [Deliverable 2](https://github.com/denisanzora/ElasticSearch581/tree/denisAnzora-Deliverable2)
  + Analyzed using Sonarlint and resolved 3 issues: 1 Critical: Generic wildcard types should not be used in return types. 2 Major: Constructors of an abstract class should be declared public. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/6d6226853a5285183a4ff1d212bd9cef15d084d1).
  + Refactored methods that were identically implemented as another method. Major code smell: Methods should not have identical implementations. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/adb3f2c48e5034e3a2e7e41751c9d8d53d1563ba).
  + Introduced a constant for critical code smell major string literals should not be duplicated and refactored collapsible if statements into one statement, other major issues too. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/51c0314382789338d8f1e0f85b099a6155226fd4).
  + Blocker code smell: Stream was implemented that was not closed, furthermore, the call must be made in a try catch finally block otherwise an exception could keep the call from being made. Failure to properly close resources will result in a resource leak which could bring first application and then perhaps the box the application is on to their knees. Extracted the call and created a helper method to introduce a try-catch for the stream. In addition, created constants for major code smells.
  + Refactored method to reduce complexity from 16 to the allowed of 15. Extracted the code block and introduced a new helper method. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/22ef1631f9164956ded5bc10c188db06465e04e0).
* Build-Convention – Garhgaj Singh - Branch [garhgajDeliverable2](https://github.com/denisanzora/ElasticSearch581/compare/garhgajDeliverable2)
  + Analyzed Using SonarLint to find issues. Working on the Build Conventions folder to resolve issues. Resolved 5 Issues: 2 critical: Solving the hide utility class public constructor and converting abstract class to interface. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/c165a252906940d9435617b7d24ecfa07d42e079).
  + 1 minor: Replacing the use of the system.err by a logger. Also, this file did have some critical changes such as adding private constructors with no arguments because it’s good to throw an exception in just in case of good practice. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/ae06cdb94f1a905056d6d07a6206032f1301aa14).
  + 2 minor: Removing the unnecessary Boolean which is most likely an issue within the style but this error suggests that code isn't clear and needs a change in order to make Boolean literals not redundant.
  + And removing parentheses around the t parameter because parentheses should be removed from a single lambda input parameter when the type is inferred. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/55dc80386615aae1fe5aef4e9c481bc03e360dc2)
  + Refactored methods to reduce the files cognitive complexity where the complexity of the methods should not to be too high and is happening because of the nested if statements. So, in order to fix this, we need to offload complexity by adding && operator which can help joining the if statements.
  + Also in the same file another critical problem was that the string literals could not be duplicated so had to define a constant. Plus, there were unnecessary Boolean literals that needed to be fixed.
  + In the same file, fixed a code smell regarding Stream.toList() where it needed to be replaced by lambda. So, lambda needed to be replaced with method references. Moreover, this file needed to reduce the total number of break and continue statements so went over and restricted the number of breaks.

See the changes in this file through this [commit](https://github.com/denisanzora/ElasticSearch581/commit/3360c636b410e4431bdc1d342d31f2ebb425a3af).

**Documentation Changes**

The documentation was updated to reflect the refactored changes if applicable. We also ran a linting tool in VSCode against the readme files and fixed the issues. See [commit](https://github.com/denisanzora/ElasticSearch581/commit/97825c94e83b7b629f1f67ea9acb707c7a05db13)