John Quinn
CSCI E-97
Assignment 1
Review
February 6th, 2017

## Any changes made to proposed design:

I followed the design document, adhering to it as closely as possible. One divergence from it may not even be considered divergence – I added several helper and/or getter functions in most of the source files.

As helpers and/or setters their lesser importance is such that in a design document, even one as prescriptive as this one, likely would exclude them. I mention them here solely for clarity and disclosure. Each is commented above it within each source file nonetheless, and since they only serve to increase readability of the source files they continue to support the requirements.

I have, in addition, added two classes, ImportException and QueryEngineException. Class Import, of course, throws the former while QueryEngine throws the latter. The two are strictly required, and, of course, custom Exceptions are *per se* new classes in Java. Keeping such custom exception classes as such allows for better readability and extensibility as well and was strongly recommended by all teaching staff in Java and the Hadoop Distributed File System.

## Design Document's help with the implementation:

The design document was most helpful with the implementation. The document was replete with detail to a point, e.g., that return values, especially for the methods of the KnowledgeGraph class, made the intended code quite clear.

Two more quasi-divergences exist. The first relates to where whitespace is trimmed. I have not done so with the Importer class, as was *suggested* under "Comment your code" towards the end of the online section hand-out of two days ago.

The second relates to simple terminology. In my comments within the source files I have referred to the KnowledgeGraph as an ostensible database while the 1st page of Assignment 1 appropriately labels it a lattice structure. Given the ostensible qualifier I do not see the former as inapposite.

## Potential betterments to the design document (& the design):

I have fairly strictly adhered to the design document, but one item was discussed in lecture and in section. The design document, together with this other guidance, could be read to intimate a more plausible use of the KnowledgeGraph, querying it from a user interface ("UI").

Directly related to such a UI is, of course, implementing it. This could be done as easily with additional arguments (i.e., Subject, Predicate, and Object) delivered to main() as part of the argument vector via Command Line Input ("CLI"). A more robust implementation would be a friendlier UI, but that would require an event listener and other code. As this Assignment 1 was intended to test facility with somewhat basic Java, its absence is appropriate.
In that same vein, adding concurrency would be a betterment to this design. One common theme in this assignment is parsimony with respect to memory. Because a KnowledgeGraph could include many Nodes and many Predicates, space optimization is a high priority. Such objective could be bettered with, among other things, caching with storing to disc or dynamic memory allocation. Again, as this assignment is not meant as a Java *learning* exercise, the exclusion of the former is appropriate. The exclusion of the latter is also appropriate, explained simply by the back that this is not a systems programming course as well.

The documentation itself could be bettered by shortening it. Brevity itself is a betterment. Furthermore, more clarity might result One might shorten the documentation, e.g., by providing only one document, a design document with numbered footnotes and endnotes. Indeed, the Fowler text in several spots recommends divergence from what some perceive to be the strictures of the Unified Modeling Language ("UML"), and my proposed approach, one document, follows that recommendation.