SER 216 Team #3

Summer 2018 – Term B 08/07/2018

**Summary Report**

The following are the points needed to be address in this report:

* + Description of the goals of your testing efforts? Due to the lower level experience my teammate and I had with GUIs and/or networking in Java, we tried to focus our attention on the bugs we were able to find with the help of the initially chosen tools. We then tried our best to understand the code as much as necessary in order to write and run some JUnit tests on some of the main classes that integrated methods from other smaller classes.
  + What are the results of your testing? Our testing showed that the code initially given to us was working well.

An initial BlackBox-style testing of the application from a user prospective showed that the game was working properly with no major errors found during the execution of the game. This testing approach did not reveal any of the bug fixes and enhancements revealed with unit testing and defect tracking.  
We noticed a lot of exception handling that needed to be implemented to be able to better test and troubleshoot the code in case an exception was caught; so, we added some very simple console messages to better direct the tester/programmer to the possible problem and to the method that failed.

* + What coverage did you achieve? 17%
  + What did you discover?The more serious errors we found were a couple of possible NullPointerExceptions that needed to be solved. The rest of the bugs were mainly a matter of variable/method declaration, usage, and their static/non-static nature.
  + Summary of bug fixes and enhancements (status of bug fixes, if an error is not fixed then explain why)?

**Bugs:**

* Naked notify: false positive;
* Constructor Invokes Thread.start: since the class was not extended or subclassed, this was also a false positive;
* Possible Null-Pointer: we addressed this bug buy ensuring the ‘listener’ was not null before calling the ‘close’ method;
* Unconditional wait & wait not in loop: false positive;
* Dead store to indicator: we eliminated the extra variable initially declared and never used;
* Field is a mutable collection: since both the PIECE\_MAP and HASH\_MAP collections were declared and then statically assigned the proper values that will not be changed anywhere, this also was a false positive;
* ‘Write to static field’ and ‘Should be a \_static\_ inner class’: these bugs were both addressed by changing the static nature of the variables/methods (made them static);
* Integral division result cast to double or float: solved by making the divisor a double (from 12 to 12.0);
* Unread field - should this field be static: Some instances of this error were solved by making the field static, but a few of them were false positives (the one in ClientBoard.java and ServerBoard.java);

**Enhancements:**

* Creating console messages for the unhandled exceptions of the try-catch blocks within the several files;
* Add a catch to the try-catch block in the Server.java’s main missing the catch part of the block.
  + How would you improve your process in the future? ***First of all, it would be extremely helpful either being part of the development team from the beginning to have a better and more comprehensive understanding of the code, or use a TDD approach. Based on the level of experience acquired with Java, the relatively small amount of experience with GUI coding, and the magnitude of the projects addressed so far, this was a much more involved and diversified software. It would have been easier to come up with proper and more detailed and comprehensive testing if we were part of the development from the beginning to better study the required knowledge of networking and GUIs to fully understand the innerworkings of the game.***