Programming Assignment 5 - Boggle

Testing Report

Steven Cheng & Kyutae Sim

October 2020

1 Summary

Overall, we found the peer review process as an interesting and effective take on testing code in general. Since each team had their own perspectives on what to test and what to expect from certain less well-defined behaviors, we were able to get a more objective look at not only which parts of our code works, but also which parts needed more consistent behavior with both the handout and our assumptions.

When we tested other projects, our biggest obstacle was the fact that we made some basic underlying assumptions that others did not. Although we tried to adapt some of our test code to be more inclusive of all assumptions, some of our tests still failed to work (most notably, we could not get our addWord to work even though we tried various combinations of cases, various locations in the code to load the dictionary, etc.) Since we could not look at others' code, we were sadly unable to resolve these issues and could not provide feedback to those teams for these tests.

Additionally, one of the teams we were supposed to review had an issue with their .jar. Although the UI worked well, we could not import their GameManager or GameDictionary classes for use in our testing harness. Because of this, we were only able to give surface level feedback based on how their UI performed, though we tried to give pointers as to what to test based on what the other 3 teams generally missed.

When reading the peer reviews we received, it became clear to us that while our feedback on the failed tests was sufficiently thorough, we could have been much more specific when initially testing our methodology. We were reluctant to detail our methodology since there would be simply too much to mention - our testing files compose over 1500 lines of JUnit testing, but most of the tests are repetitive and some wouldn't even work. We opted to pick a more high level explanation of the general things we tested, and only went specific when describing the failed tests. In hindsight, it would have been useful to describe the whole methodology, regardless of how long the full description would have been.

2 Evaluation

The peer reviews we received from best to worst (in our opinion) were from Team 3, Team 20, Team 9 and Team 15. Overall, all of these reviews were really solid and provided useful feedback. Additionally, each team provided some insight that none of the others did, which was very helpful for our revised solution's design choices.

We found Team 3's to be the most helpful review, mostly because they simply had the most tests. Both their black box GUI tests and white box interface tests were very thorough, and their descriptions of what exactly our code failed on was very detailed. Additionally, they provided some pointers as to what fixes we should make to our GUI to fit certain requirements.

Team 20's review was also quite solid. They had a wide variety of tests just like Team 3, but the insights they provided were not as detailed as Team 3's. For some of the feedback, we had to infer the context of the failure, though this was usually pretty obvious ("hO" failing on "HOUSE" pretty clearly means they added HOUSE and tested the prefix hO).

Team 9's review had a lot of tests as well. The results of the tests were not very detailed, but this was largely because they had already extensively described the tests they had used. The main reason why we ranked Team 9's lower is because we passed all but 2 of their tests. However, in both Team 20 and team 3's review, there were many more issues they had caught that Team 9 didn't, suggesting their tests were not as rigorous as they could've been.

Team 15's was ranked last because they seemed to have less tests than Team 9, and also had the issue where certain tests passed because they likely weren't rigorous enough. Additionally, our code passed on some of the test cases where they specifically mentioned that our code should have failed, again suggesting that their tests did not adequately check this (or that our code is very inconsistent, which could've also easily been the case).

Despite these critiques, however, all the reviews provided very helpful criticisms, and also mentioned the parts we had excelled at (most commonly, the overall look and feel of the UI). Each team also did a great job of detailing their specific tests as well. Additionally, they all pointed some major bugs (which can be seen in the revised solution).

3 Revised Solution

Though our solution generally passed, there were a few major issues we needed to address. Some of these came in part from our testing, but the majority of these issues were noted by all the peer reviews we received:

- **Bad GameDictionary iterator.** This was by far the biggest issue - we had incorrectly tested our GameDictionary iterator and failed to catch major issues with its functionality.
- **UI Fixes and Additions.** Many teams noted that our GUI did not provide functionality to play with multiple players or different board sizes. Additionally, there were some suggestions to fix various bugs and create more specific error messages.
- **Case Sensitivity.** In our original code, we designed it under the assumption that everything should be case sensitive. After going through the peer review testing process and viewing different programs, we realized it would be much more logical to make our code case insensitive.
- **Invalid parameters.** Many teams also noted that we do not handle most kinds of invalid input, and instead throw various errors caused by attempting to use these inputs.
- **Pointer issues.** This came from our testing harness, since we realized that we failed to cover the pointer issues created through getBoard.

In our revised solution, we sought to fix all of these. Firstly, we fixed the bugs in the GameDictionary iterator. We also added a menu for the newGame button that would allow the user to pick player numbers, board size, word/cube files, and more, as well as added some checks to provide more specific messages. We also added checks to all the parameters, made our code case insensitive wherever applicable, and modified the getBoard method to create a copy of the board rather than pass the direct reference.