Project 5 Testing Report

Atharva Pendse & Zhaosong Zhu | CS 314H

In the Boggle project, we are offered the opportunity to conduct peer testing with four other groups from the same class. Personally, we found the peer testing process really helpful, regarding developing a robust testing harness and think about more aspects. Below is a discussion about the testing process.

- Summary of overall Peer Review process
  Our peer review process started with writing test cases based on the sanity test. We sit together to talk about all “traps” that we fell in when writing out the solution and some possible mistakes that other people tend to make. However, merely talking about ideas were not enough - as we received the jar files from others and played with the Boggle game, we were able to update our testing harness and project itself.

  For the one thing, we developed some new insights when testing other people’s code. For example, when checking with group 16’s jar file on loading dictionary, we realized the way they dealt with words with numbers and dashes are more user-friendly. Whenever such situation happens, their program will give an error prompt about why the word is omitted. This reminds us to consider how to deal with words with dashes, which is a possibility that we didn’t think of before.

  For the other thing, from the peer reviews we received, we were able to fix some bugs too. For example, based on team 18’s review, we didn’t address the case where the invalid player number is used for *addWord()*. Many thanks to their suggestion, we were able to find not only this bug but also deal with the similar situation in the GUI with proper way.

  Besides, the list presented is a list of questions we used to test others:

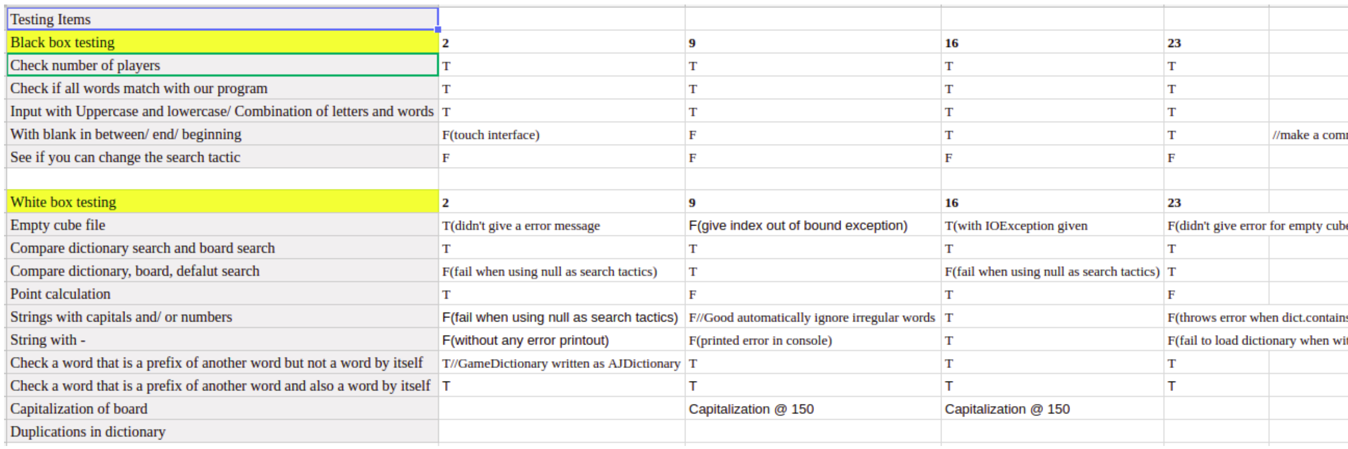
A. Black box testing
    1. Check number of players
    2. Check if all words match with our program
    3. Input with Uppercase and lowercase/ Combination of letters and words
    4. With blank in between/ end/ beginning
    5. See if you can change the search tactic in the UI

B. White box testing - through JUnit
    1. Empty cube file
    2. Compare results of dictionary search and board search
    3. Compare results of dictionary, board, default search
    4. Point calculation of given word
    5. contains(word) word is a string with capitals and/ or numbers
    6. contains(word) word is a string with -
    7. contains(word) Check a word that is a prefix of another word but not a word by itself

8. contains(word) Check a word that is a prefix of another word and also a word by itself
9. Capitalization of board

- Evaluation of Peer Review quality
  This is our ranking of peer reviews, from best to worst: 9-2-23-16. This ranking is based on the number of bugs we discovered for each team and how much we learned from them.

  Based on the items we came up above, our goal is apparently to see if their Boggle game is performing well under both black and white box testing cases. To account for the result, we used a spreadsheet, which records whether a team past a particular item in the test, as shown below:



Testing Items					
Black box testing	2	9	16	23	
Check number of players	T	T	T	T	
Check if all words match with our program	T	T	T	T	
Input with Uppercase and lowercase/ Combination of letters and words	T	T	T	T	
With blank in between/ end/ beginning	F(touch interface)	F	T	T	//make a comm
See if you can change the search tactic	F	F	F	F	
White box testing	2	9	16	23	
Empty cube file	T(didn't give a error message	F(give index out of bound exception)	T(with IOException given	F(didn't give error for empty cube	
Compare dictionary search and board search	T	T	T	T	
Compare dictionary, board, defalut search	F(fail when using null as search tactics)	T	F(fail when using null as search tactics)	T	
Point calculation	T	F	T	F	
Strings with capitals and/ or numbers	F(fail when using null as search tactics)	F//Good automatically ignore irregular words	T	F(throws error when dict.contains	
String with -	F(without any error printout)	F(printed error in console)	T	F(fail to load dictionary when wi	
Check a word that is a prefix of another word but not a word by itself	T//GameDictionary written as AJDictionary	T	T	T	
Check a word that is a prefix of another word and also a word by itself	T	T	T	T	
Capitalization of board		Capitalization @ 150	Capitalization @ 150		
Duplications in dictionary					

  In the spreadsheet, we also wrote down things we noticed as tests are performed on each team's project. For example, when testing search tactics, team 2 fails because the *setSearchTactics* will go wrong when accepting a *null* parameter. We noted this down on the spreadsheet so that we can give back more personalized feedback. This bug we found also helped us find the bug that our *setSearchTactics* method didn't account for null or any other weird String inputs. We were able to realize this when testing other people's codes.

  We believe the testing methodology is decently robust since it accounts for both the UI part and the white box testing part. However, it is definitely not perfect yet. For example, during peer testing, we assumed that the iterator must work correctly if all the search methods are working fine. Although to some degree this assumption is true, it's not a good assumption to ensure a comprehensive test. We then included this into our testing, while not able to address this in all four peer testing feedbacks.
- Description of revised solution
  We made some changes on our solution based on the comments presented below:
  1. "addWord(): does not check for invalid player numbers (i.e., negative and greater than the number of players) "
     This comment points us to a relatively obvious edge case - an invalid integer can be passed in as the number of players. Since this is something we will most likely not encounter when playing the game, we failed to put a check on this case. With this feedback, we then add the following lines to throw an exception when this edge case occurs:

```
if (player >= playerNum || player < 0) {
throw new IllegalArgumentException("Invalid player number");
}
```

2. "contains(): failed for null parameters and capitalization."
   This feedback helps us realize that we forgot to deal with little edge cases on the *contains()* method and *isPrefix()* method. Since the comment points to the *null* parameter, we also put an empty string into consideration. Thus, every time before the search in these two methods are performed, we will check if the parameter is either *null* or empty string. If yes, then we will return no.

   For capitalization and spaces in between, we will then trim the spaces and change everything to lowercase, for the matter of formatting.
3. "We found that the submission updated the last added word when calling get all words. "
   This feedback corrected our wrong assumption that the lastAddedWord needs to be updated whenever a dictionary/ board search is performed. When calling get all words, a search will initially be performed, which is why the last added word was changed. We fixed this error with this feedback. Otherwise, we might not be able to find this out, even with white box testing.