Software Testing and Debugging Final Project Proposal

Zixuan Wang and Gary Wu

Software under test (SUT): https://github.com/Brioal/Game

Basic information:

This is an open-source Gomoku (Five in a row) game written in Java that should let users play through the local area network (LAN). The basic rule would be shown as follows: the game is played on a 19x19 board and there are two players that would take black stones and white stones separately. The black stone side always goes first and two players alternate turns placing a stone of their color on an empty intersection. The winner is the one who first has 5 stones of his/her color connected horizontally, vertically or diagonally.

This project has the following structure

- 5 classes to render the welcome page for the game
- 5 classes to handle the user events (such as mouse clicking and update the panel)
- 3 classes to set up different panels
- 3 classes for web socket that handle the client and server connection
- 2 classes for utils that would set up the main rule (detect winning) and encapsulate the properties of a fundamental stone

The total number of classes in this project is 18 and the total line of code is around 3000. There is a class named the “test” but what it does is just simply create a welcome page without doing any actual test. So, we don’t think this game contains any actual test at all. Thus, this game is suitable for the purpose of our final project.

Tests to conduct:

Blackbox test: Although this game doesn’t have any specific API that we can use, we could still play the game several times for the purpose of blackbox testing since we don’t have any access to the source code when we just play the game. During the initial test, we found several obvious failures shown as follows:

- Sometimes the game only shows one color of the stone.
- The game won’t stop when players win the game. There is an issue with the winning detection algorithm.
- The game doesn’t support multiplayer mode through LAN. It seems like there is a connection issue with the server.

Whitebox testing: Since we have access to the source code after our initial testing, we could dig into it and find out where the faults are. For the missing color failure, we could go to the function with signature *public void addPoint(Point point)* and for the winning detection issue, we need to look into the function with signature *public boolean JudegeWin(Point point))*. For network issues, we need to look into the socket folder. After we know where to trace in the source code, writing many unit test cases would be important. Thus, we could use JUnit 5 to help us with whitebox testing. We aim for **100% branch coverage** for this test.

GUI testing: Since the game has a GUI frame that would handle users interactions, it is important to test the GUI as well. We need to test whether users can put their stones on the board and whether the board is shown correctly. Since our app uses JPanel,Jbutton,JLabel, Assert J Switch would be a good tool to test our GUI. (We are also looking at different options with respect to testing our GUI like FEST, java.awt.Robot.)

Property based testing: The game has many fixed properties that should not be changed throughout the entire gaming process. Thus, having property based testing would ensure the game is set up correctly. For example, the board size of this game must be 19x19 and there must be 2 players in order to start the game.

Mutation testing: mutation testing would be interesting to be implemented and we would leave this test as an optional test. We could test the winning detection algorithm and find out the final mutation coverage level in the end. The tool that we can use is PIT.

Test tools:
JUnit5 for whitebox testing
AssertJ Swing for GUITesting (https://joel-costigliola.github.io/assertj/assertj-swing.html)
Jqwik for property-based testing
PIT for mutation testing
Jacoco for test coverage report

Task breakdown:

1. Blackbox Testing (Zixuan)
2. Whitebox Testing (Gary)
3. GUI Testing (Zixuan)
4. Property Based Testing (Gary)
5. Mutation Testing (Gary)

Timeline:

- Blackbox Testing (4/15)
- Whitebox Testing (4/15)
- GUI Testing (4/19)
- Property Based Testing (4/25)
- Mutation Testing (4/19)
- Presentation (4/27)

Due date: 4/29