Multiplayer Yahtzee

Test and Validation Plans

Team Sink

Arjuna Herbst – Finn Dugan – Louis Cerda – Brandon Poblette

Course: CPSC 224 - Software Development

Instructor: Bryan Fischer

I. Introduction

I.1. Project Overview

The software being tested is a multiplayer version of the game Yahtzee. In this version of Yahtzee, users will be able to add more than one player, each with their own scorecard and set amount of turns. The game will end once every player has taken the maximum amount of turns and filled out their scorecard. A winner will be decided based on each player's individual scorecard, and this winner will be announced to the user.

For this project it is very important that data is correctly read from the GUI by the back-end classes. We'll need testing to make sure that adding players in the GUI successfully creates new players in the back-end. We will also need testing to see if each player's scorecard is correctly updated and only updated when it's their turn. Other major functionality to look out for are to make sure that the correct hand is being shown to the user, and that the correct values from the hand are given to the scoreboard class to calculate the upper and lower scores. The lower and upper scoreboard will also need to have test run on them to ensure the correct calculations are being made.

I.2. Scope

Describe the scope and purpose of this document in a short paragraph. A test plan document outlines the kinds of testing to be used to validate the functionality and to demonstrate the completeness of a project. Summarize how that will work for your particular project here:

The purpose of this document is to help us plan in advance on how we are going to create our tests for this project. By planning in advance, this will help us figure out areas in our program where we will need to put extra effort into. For example, now we know that we must put extra attention to details in creating our player class so that it can create multiple players. We must also make sure that our GUI is functioning properly so that user inputs correctly respond to the user. If the testing fails then that means we have not correctly implemented our project and we will have issues that need to be fixed. Working tests further improve the validity of our project. Without working tests, the game might still run, but it doesn't necessarily mean we reached the goals we set out to complete. Working tests will let us know that the project works correctly. It will also let us know that we have reached the goals we set out to reach.

II. Testing Strategy

Describe the overall approach to testing and provide the overall flow of the testing process.

In our overall approach to testing, we plan on writing test case classes for each individual class we create. Once we've identified each major class, we will write a test for every method in each class. It is crucial that we know every single method works correctly in order to prove the validity of our program.

Our arranged portion of the testing process will involve creating objects of the classes which we created that can call functions we have written and which we want to test. We will then call a method to change the state of the object, which would be the act portion of our testing. Lastly, we will create assert statements to see if the method worked correctly. An example of this would be creating a test for diceRoll(). We would start by creating a dice object. We would then create an integer variable called testVal. Once the object exists, we can call rollDice() on the dice object and store that value in our testVal. Finally, we can assert that if testVal is less than or equal to 6, rollDice() has worked correctly.

Whenever a test fails or an issue arises, we will create an entry on our team github issue board. All team members will comment to make it clear which parts of the code they have written, so when there are issues we can talk to the person who created that part of the code and let them know of that issue and how it could potentially be fixed. From there, we can fix issues together as a team and improve the functionality and validity of our program.

III. Test Plans

III.1. Unit Testing

The primary goal of unit testing is to take the smallest unit of testable software in the application, isolate it from the remainder of the code, and test it for bugs and unexpected behavior.

Unit testing will mostly be written by Brandon. Making unit tests was one of his strong suits and he will make them for every class. Progress for our unit testing would mean that each unit test class is created by at most 3 days after the base class is completed. If all assertions are working, then the program is working as expected. Each person will also review the unit tests for the base class they created. For example, if Arjuna creates the dice class, he will also review the dice class tests. He would also be required to look at them at most 2 days after the test class is created. This way we can ensure that his class is working as intended, and all fixes necessary are done in a timely manner.

III.2. Integration Testing

Since the purpose of integration testing is to find faults in a program's intertwined classes and methods, this stage of our program testing will involve a lot of communication with other team members. This is due to the fact that each member of the team will be working on a separate part of our overall project, and when these different parts of our code interact, we will need tests to make sure that they work together without issue. For example, if Finn wrote the Scorecard class, and Arjuna wrote the part of the GUI which is supposed to display the scorecard to the user, these two parts of the code would not only have to work together, but also work smoothly and efficiently together. To test that these two portions of our project are working together well, the 3 A's of testing will be applied. To arrange, we will run through the GUI as if the program is being run, as well as creating a scorecard object. To act, the test we write will get a certain line of the scorecard object which was created in the arrange step, as well as getting the text of the JLabel which displays the scorecard to the user. To assert, the test will check to see if these two lines of the scorecard object and the JLabel match up and are displaying the correct data.

III.3. System Testing

System testing is a type of black box testing that tests all the components together, seen as a single system to identify faults with respect to the scenarios from the overall requirements specifications. Entire system is tested as per the requirements.

During system testing, several activities could be performed. Note if any of these will be used by your group, and how you expect to do it for your project:

III.3.1. Functional testing:

When going through the functional testing stage of a program, it is crucial to check the program's functional requirements, as without basic functionality working correctly, the whole program is useless. In the case of our multiplayer yahtzee program, functional testing will be .critical. Some examples of functionality that needs to be tested are as follows, multiplayer capability, displaying the score to each individual player, displaying dice being rolled and what they land on, and having a working scorecard. We will carry out this process of testing by writing tests specific to breaking possible functionality such as the above.

III.3.2. Performance testing:

By using stress testing to check the performance of our program, we will be able to figure out a set of maxes and minimums for our code. For instance, we plan on writing a test which fills the game with a set of increasing numbers of players. This test will be timed, and when the time begins to become too large due to creating a large amount of objects, this will be noted as a max number of players we could add to our game.

III.3.3. User Acceptance Testing:

Acceptance testing and installation testing check the system against the project agreement. The purpose is to confirm that the system is ready for operational use. This section of testing is one of the most important stages, as we are building a project which involves other users playing it. The simplest way of carrying out this form of testing would simply be to have someone who doesn't quite understand the point of the game or the code behind to sit down and play a game. We plan on having a few of our non computer science friends to play a round and give us feedback as our testing strategy for this section of the process.