Test Plan

CMPT 370 – H2

Amanda Zimolag

Brendan Nykoluk

George Shi

Spencer O’Lain

Xinnan Xie

Zeshan Ahmad

Table of Contents

1. Introduction

This test plan describes the testing approach and framework that will drive the testing of our Hanabi client. Our test plan contains two end-to-end system tests, fourteen integration tests, and 50 unit tests. The tests contained herein are designed to ensure the most important functionality of the program by making sure that a game of Hanabi can be successfully entered and played. The first end to end test is designed to test the main game-play loop, while the second is for other options available to the player. Each test includes the set-up to run the test and the expected result of the test.

2. End to End Tests

Our test plan consists of two end to end system tests which cover all important functionality of our Hanabi client. Each of these tests consist of a connection to a server and playing a game through to its conclusion.

2.1 Basic Interactions Test

The first end to end test checks most of the human interactivity of our client. It consists of eleven integration tests and 38 unit tests. This test is designed to test all of the moves that a human user can perform as part of a game of Hanabi, and as such its success will be mostly visually confirmed by the tester. The test starts with the user creating a server for two people and successfully connecting to it. The deck will be stacked such that each player has one card of each rank in a single suit. Each player will then in turn

- play their “1” (which should be a successful move),
- play their “3” (which should be a failed move burning a fuse token),
- give the other player a hint about their “5”, and
- discard their “2”.

This will test the result of each of the four possible moves from the client and from another player. On the user’s turn for the discard action they will purposely take long enough to violate the turn time limit once to ensure the warning message appears. After the eight moves above have been completed, the user will attempt to play the “4” that was in their hand initially which should burn the final fuse token, ending the game. At this point the end game display should appear shooting off two fireworks to the first level based on the two “1”s that were successfully played and showing the score as two. Finally, the user should be returned to the start menu. The test shall be considered successful if all of the above is completed successfully.

2.2 Alternate Interactions Test

The second end to end test covers some tests that the first test was not able to. It consists of x integration tests and x unit tests. This test involves human interaction to perform most of the actions, and its success will confirmed visually. The test begins with the user joining a game that has been created by another copy of the client. On connection to the game, the user will press the “Take a Break” button initiating the AI taking over move control. On the user’s turn, the AI will make a valid move. After the move is made, the “Take a Break” button will be pressed again, turning off the AI and

returning control to the user. The user will then make one more move to confirm they once again have control. Finally, the user will press the “Leave Game” button. This will return them to the start menu without displaying the fireworks or score. This should also send a message to the server stating that the game should end. If all of the above happens as expected, the test shall be considered successful.

3. Integration Tests

3.1 Create Game Test

The Create a new game integration test will demonstrate the correct creation of a server and joining of that server as specified by section 3.1.2 of the requirements document. A successful create action will consist of three unit tests including

- the Send Message to Server test,
- the Successful Message Received Test, and
- the Check PlayField Start-up Values Test.

This test will be considered successful if all of the unit tests are successful.

3.2 Successful Card Play Test

The play a card successfully integration test will demonstrate the correct updating of the game when the user plays a card as specified by section 3.1.3 of the requirements document. A successful play action will consist of four unit tests including

- the PlayField Successful Play test,
- the played cards UI test,
- the player’s hand UI test, and
- the information token UI test.

This test will be considered successful if all of the unit tests are successful.

3.3 Unsuccessful Card Play Test

The play a card unsuccessfully integration test will demonstrate the correct updating of the game when the user plays a card that does not successfully add to a stack as specified by section 3.1.3 of the requirements document. A successful play action that does not add to a stack will consist of four unit tests including

- the PlayField Unsuccessful Play test,
- the player’s hand UI test,
- the fuse token UI test, and
- the discarded cards UI test.

This test will be considered successful if all of the corresponding unit tests are successful.

3.4 Player Discard Test

One of the ways that a player may act in the game is to discard a card. The action itself can be found along with the other actions in the requirements document, section 3.1.3. To test that our

program will correctly discard a card for the user we will test the whole process of discarding a card with a integration test. The integration test contains five unit tests including

- the PlayField Discard test,
- the Ensure invalid moves are greyed out (UI) test,
- the Player's Hand update test,
- the Discarded Card update test, and
- the Info Tokens update test.

All of the unit tests must pass for the Integration test to be successful.

3.5 Player Give Hint Test

The "give a hint" integration test will be responsible for measuring the functionality of the code modules which will provide the user with the option to give another player a hint. A hint action is described in the requirements document, section 3.1.3. The unit tests that correspond to this integration tests are

- the Info Tokens update test, and
- the Invalid Hints are Grayed out test.

This test will be considered successful if both of the corresponding unit tests succeed.

3.6 Opponent Play Card Test

The Opponent Play Card integration test will demonstrate the correct updating of the game when the user plays a card as specified by section 3.1.3 of the requirements document. A successful play action will consist of 6 units test including

- the PlayField opponent Successful play test,
- the PlayField opponent Failed play test,
- the played card's update test,
- the player's hand update test, and
- the Fuse Tokens update test.

3.7 Opponent Discard Card Test

The Opponent Discard Card integration test will demonstrate the correct updating of the game when the user discards a card as specified by 3.1.3 of the requirement documents. A successful discard action will consist 4 unit tests including

- the PlayField opponent Successful discard test,
- the Discarded Cards update test,
- the Player's Hand update test, and
- the Info Token update test.

3.8 Opponent Giving Hint Test

The opponent hint integration test will be responsible for verifying the correct behaviour of the program in regards to a hint that is given to the user by an opponent. The test will test the hint action

described in the requirements document in section 3.1.3. The unit tests which will be composing this integration test are:

- the PlayField Opponent Hint test,
- the Info Tokens update test, and
- the Players Hand update test.

The integration test will only be considered successful if all of these unit tests succeed.

3.9 Display Timeout Message Test

The “Take time exceed given turn length” integration test will demonstrate the correct display of the message when the player takes too long to take a valid action as specified by section 2 of the design document. This test will be considered successful if the Display Timeout Message unit test succeeds.

3.10 End Game Display Test

The “End game display” integration test will demonstrate the correct display of the game when the game ends as specified by section 2 of the design document. A successful end game display will consist of four unit tests including

- The Receive the String unit test
- The Parse into 5 Method Calls unit test
- The Display Fireworks unit test
- The Display Message unit test

This test will be considered successful if all of the unit tests are successful.

3.11 Join A Game Test

The Join a Game integration test will demonstrate the correct processes of the game when the user joins a game as specified by the Join Menu section in section 1.1 of the Design document. A successful Join Game action will consist of three unit tests including

- the Message Sent to Server Test,
- the Successful Message Received Test, and
- the Check PlayField Start-up Values Test.

3.12 Leave A Game Test

The Leave a Game integration test will demonstrate the correct processes of the game when a user leaves a game in progress as specified by the Leave Game section in section 1.1 of the Design Document. A successful Leave Game action will consist of two unit tests including

- the End Game Test, and
- the Return to Start Menu Test.

4. Unit Tests

4.1 Send Message to Server Test

The Send Message to Server test will demonstrate the correctness of the message sent to the server when creating a game as described by section 3.1.2 of the requirements document. A mock will be used to create a scenario test whether the information will be sent successfully. This scenario will

include “number of players” as 3 and “timeout length” as 30s. The scenario will send the number of players and timeout length to the create class and receive the output. This test will be considered successful if the output of createServer(int , int) matches the expect output which is 3 player and 30s as timeout length in the correct message format.

4.2 Receive Json Message Test

The Successfully Received Message unit test will demonstrate the correctness of the method receive( ) of the class JSON as specified by section 2 of the design document. This unit test will test whether the method receive( ) can handle the JSON package that is sent by the server and whether the method can correctly transform the JSON package to a string or not. A mock will be used to test the actual string output with the expected string output. The test will be considered successful if the actual string output matches the expected string output.

4.3 Check PlayField Start-up Values Test

The PlayField Start-up Values test will demonstrate the correct functioning of the create as specified in section 3.1.2 of the requirement documents. The test will be considered successful if the values for the fields in the PlayField match their defaults which are

- an empty list for the player’s hands,
- three fuse tokens,
- eight information tokens,
- no cards in the played stacks, and
- no cards in the discarded array.

4.4 PlayField – Successful Play Test

The PlayField Successful Play unit test will demonstrate the correct updating of the fields in PlayField when a card is played as specified by section 3.1.3 of the requirements document. A mock will be used to create a scenario where the first card in a player’s hand can be played successfully. This scenario will include completing a stack by placing the red “4” on the red “5”. Additionally, the player will be able to see the value of this card as though they had been given hints about both its suit and rank. This test will be considered successful if

- the player’s card list field updates to show an unknown card in the slot for the first card,
- the played pile field updates to show the “5” instead of the “4” previously displayed, and
- the information tokens counter is increased by one.

This test requires the fields of the PlayField class be visible and accessible.

4.5 Played Cards Stack Update Test

The Played Cards Update unit test will demonstrate the correct functioning of the played cards view as specified in section 1.2 of our design document. The test will be considered successful if the played cards view updates to show a card played successfully. The success of the test will be determined visually since this is a graphical user interface. This test requires knowledge of the cards in

the played cards pile and the card being played to ensure the pile updates to show the card being played.

4.6 Player's Hand Update on Card Play Test

The Player's Hand Update unit test will demonstrate the correct functioning of the player's hand view as specified in section 1.2 of our design document. The test will be considered successful if the player's hand view updates to show a new card of unknown suit and rank in the position of the card that was previously played. The success of the test will be determined visually since this is a graphical user interface. This test requires knowledge of the position of the card being played to ensure that the position being replaced is the correct one.

4.7 Information Token Update on Stack Complete Test

The Information Tokens Update unit test will demonstrate the correct functioning of the Information Tokens view as specified in section 1.2 of our design document. This test will be considered successful if the information tokens view updates to return one token on the successful completion of a stack (playing a "5" on a "4"). The success of this test will be determined visually since this is a graphical component. This test requires knowledge of a stack with a "4" and a "5" in the player's hand.

4.8 PlayField – Unsuccessful Play Test

The PlayField Unsuccessful Play unit test will demonstrate the correct updating of the fields in PlayField when a card is played as specified by section 3.1.3 of the requirements document. A mock will be used to create a scenario where the first card in a player's hand can be played unsuccessfully. This scenario will involve the user playing the first card in their hand, which will be stacked to be a red "4" when the played stack has only a "1". The test will be considered successful if

- the player's card list field updates to show an unknown card in the slot for the first card,
- the discarded card display increases the count of red "4"s discarded, and
- the fuse token count decreases by 1.

This test requires the fields of the PlayField class be visible and accessible.

4.9 Fuse Token Update Test

The Fuse Tokens Update unit test will demonstrate the correct functioning of the Fuse Tokens view as specified in section 1.2 of our design document. This test will be considered successful if the fuse tokens view updates to remove one token on the unsuccessful playing of a card. The success of this test will be determined visually since this is a graphical component. This test requires knowledge of the player's hand to ensure that the card being played will not add to a stack.

4.10 Discarded Cards Update on Unsuccessful Card Play Test

The Discarded Cards Update unit test will demonstrate the correct functioning of the discarded cards view as specified in section 1.2 of our design document. The test will be considered successful if the discarded cards view updates to show a card played unsuccessfully. The success of the test will be determined visually since this is a graphical user interface. This test requires knowledge of the cards in the played cards pile and the card being played to ensure the card being played will not go the played cards stack.

4.11 PlayField Discard Test

The purpose of the PlayField Discard test is to ensure that our program properly updates the fields which are changed when a card is discarded in line with the rules of Hanabi. In this test we will create a mock play field in which there are two user hands. On player 1's turn we will have the player discard a card. For the test to be successful the play field must:

- replace the field in the player's hand list with an empty string,
- increment the information token counter, and
- add the discarded card to the field containing other discarded cards.

4.12 Gray Out Invalid Moves Discard Test

The purpose of this test is to ensure that the user has no option to discard a card when the play field has all 8 information tokens available. More information on the action is available in the requirements document in section 3.1.3. For the test we will need to create a play field with two hands. We will have 8 information tokens in this play field. For a successful test the user should not be able to press the button that allows for a discard.

4.13 Player's Hand on Discard Update Test

In this test we will view the user interface to see if the players hand has updated after the player discarded a card. The test will require a mock play field with two hands and a number of information tokens less than eight. We will have the player discard a card. For the test to be considered successful the player's hand will have a replacement card with no visible suit or rank in the position of the card that was discarded.

4.14 Discard Pile Update Test

The discard pile updates test will be responsible to ensure that the program will properly update the graphical user interface when a player has decided to discard a card. The test will require a mock play field with a number of information tokens less than eight. We will have the Player discard a card to start the test. For a the test to be successful the user interface should reflect the card selected by the user being added to the discard pile view.

4.15 Information Token Update on Discard Test

Info tokens update test will be a visual confirmation that the graphical user interface will update to reflect an increase in information tokens due to discarding a card. For this test we will need a play field with two hands and less than eight information tokens so that the player can discard a card. We will then have a player discard a card from the hand. For a successful test the information token view should show an increase of one token.

4.16 PlayField Update on Player Hint Test

To ensure proper functionality in giving a hint to another player we must ensure that the proper fields in the play field are updated for an accurate representation of the game state. We will form a game state which will contain players and a play field. The play field will contain a non-zero number of information tokens to allow for a hint to be performed. A successful test will show the information token counter decreasing by one.

4.17 Gray Out Invalid Moves (No Information Tokens) Test

This test will require the play field with two opponents and no information tokens. This will have player 2 contain some cards in hand and allow player 1 to play out the turn. Player 1 should only have the option to discard or play a card as there are no information tokens available to spend on a hint. For a successful test the option to hint should be greyed out, disallowing any access to the hint menu to player 1. We will be viewing the graphical user interface to make sure that the option is inaccessible to the player.

4.18 Gray Out Invalid Moves (Invalid Hints) Test

For the Invalid Hints test we will have the hint receiving player (the opponent) hold a hand of various cards. Then we will allow the user to provide a hint for the opponent. For a successful test the options for giving a hint based on suits or ranks that are not contained in the opponent’s hand should be unable to be pressed.

4.19 Information Token Update on Hint Test

The purpose of this unit test is to ensure that the graphical user interface properly updates the number of information tokens in the view when a hint is given. To test for this, we will create a play field with two players. We will allow player 1 to hint at a suit or number in player 2’s hand. After a successful hint by player 1, the graphical user interface should reflect the change in information tokens.

4.20 PlayField Update on Successful Opponent Play Test

The PlayField Successful Play unit test will demonstrate the correct updating of the fields in PlayField when a card is played by an opponent as specified by section 3.1.3 of the requirements document. A mock will be used to create a scenario where the first card in an opponent hand can be played successfully. This scenario will include completing a stack by placing the red “4” on the red “5”.

Additionally, the player will be able to see the value of this card as though they had been given hints about both its suit and rank. This test will be considered successful if

- The player's card list field updates to show an unknown card in the slot for the first card,
- The play pile field update to show the "5" instead of the "4" previously displayed, and
- The information counter will increase by one.

4.21 PlayField Update on Unsuccessful Opponent Play Test

The PlayField Failed Play unit test will demonstrate the correct updating of the field in PlayField when a card being played as specified by section 3.1.3 of the requirements document. A mock will be used to create a scenario where the first card in an opponent hand can be play failed. This scenario will entail placing the red "4" on the red "4". Additionally, the player will be able to see the value of this card as though they had been given hints about both its suit and rank. This test will be considered successful if the player's card list field updates to show an unknown card in the slot for the first card, and the discarded cards field updates to show the red "4" go to the discard pile.

4.22 Played Cards Update on Opponent Play Test

The Played card update test will demonstrate the correct functioning of the opponent card being played as specified in section 3.1.3 of the requirements document. The test will be considered successful if the played card is displayed in the correct pile. The success of the test will be determined by visually inspecting the display, which show the successfully played card in the played pile and the failed played card displayed in the discard pile. This test requires the player to make 2 moves, one for a successfully played card, and one for an unsuccessfully played card.

4.23 Player's Hand Update On Opponent Move Test

The player's hand update test will demonstrate the correct functioning of the played card as specified in the section 3.1.3 of the requirement document. The test will be considered successful if a new card shows in opponent's hands after they make a move. The success of the test will determined by visually inspecting the display, which should show the card in player's hand replaced with a new card after they play or discard a card.

4.24 Fuse Token Update on Opponent Play Test

The Fuse Tokens Update test will demonstrate the correct functioning of the card being unsuccessfully being played as specified in the section 3.1.3 of the requirement document. The test will be considered successful if one fuse token is removed from the display when an opponent plays an invalid card. The success of the test will be determined by visually inspecting the display, which should show one fuse token disappearing when an invalid card is played. This test requires one opponent play a a card that does not add to a current stack.

4.25 PlayField Update On Opponent Discard Test

The PlayField Opponent Discard unit test will demonstrate the correct updating of the fields in PlayField when a card is discard as specified by section 3.1.3 of the requirements document. A mock will be used to create a scenario where the first card in an opponent's hand can be discarded successfully. This test will be considered successful if

- The player's card list field updates to show a new card in the slot for the first card,
- The discard pile field updates to show the card discarded, and
- The information token counter will increases by one.

4.26 Discarded Cards Update On Opponent Discard Test

The Discarded card update test will demonstrate the correct functioning of the discarded cards view as specified in section 3.1.3 of the requirements document. The test will be considered successful if the discarded card is display in the discard pile. The success of the test will be determined by visually inspecting the display, which should show the discarded card in the discard pile.

4.27 Information Token Update on Opponent Discard Test

The Opponent Discard Information Token test will demonstrate the correcting functioning of the information tokens as a result of an opponent's discard as specified in the section 3.1.3 of the requirement document. The test will consider if one Info Token being added on when an opponent discards a card. The success of the test will determined by visually inspecting the display, which should show one Info Token appearing when a card is discarded by an opponent. This test requires one opponent to discard a card.

4.28 PlayField Opponent Hint Test

The purpose of the PlayField Opponent Hint test is the validation of the programs ability to properly update the appropriate fields in the playField. For this test we will generate a play field with two hands and a number of information tokens greater than zero. We will then have the other player give a hint to the user. The test is successful if the information tokens counter increases by one and the player's hand updates to contain the suit or rank of the cards hinted at.

4.29 Information Token Update on Opponent Hint Test

The Information Token update test will be checking whether the graphical user interface of the user is correctly removing an information token after a hint is given to the user. For the test we will generate a playField with two hands. In the playField, we will have more than zero information tokens. We will then have a player give a hint to the user. For a successful test the graphical user interface will update the new number of information tokens which should be 1 less than the previous value.

4.30 User's Hand Update on Opponent Hint Test

The player's hand updates test is testing if the graphical user interface reflects the changes of the values in the users hand after a player has given a hint to the user. A playField with two hands and more than zero information tokens will be needed. We will then have the player give the user a hint. For a successful test the graphical user interface should update the display of the user's hand to show the suit or rank of the cards hinted at.

4.31 Display Timeout Message Test

The "Display Timeout Message" unit test will demonstrate the correct functionality of the method displayMessage( ) in the class PlayField as specified by section 2 of the design document. This test will be considered successful if the message for the exceeding turn length warning have been displayed on the screen when the player takes too long to take an action. The success of the test will be determined by visually inspecting the display.

4.32 Display Fireworks Test

The Display Fireworks unit test will demonstrate the correct functionality of class Firework as specified by section 2 of the design document. This test will be considered successful if the fireworks for each color have been displayed on the screen at the end of the game. The success of the test will be determined by visually inspecting the display, which should show the play field screen.

4.33 Display Message Test

The Display Message unit test will demonstrate the correct functionality of method displayScore( ) in class EndGame as specified by section 2 of the design document. This test will be considered successful if the score that achieved by the players is displayed on the screen at the end of the game. The success of the test will be determined by visually inspecting the display, which should show the play field screen.

4.34 Receive Firework String Test

The Receive the String unit test will demonstrate the correct functionality of passing the score for each color as a string into the class Firework as specified in section 2 of the design document. A mock will be used to create a scenario test when the scores of each color will be given as the specific input. The test will be considered successful if the Firework class successfully receives the given string as the input and stores the scores in the corresponding fields.

4.35 Parse Firework String Test

The "Parse it into 5 method calls" unit test will demonstrate the correct functionality of the method displayFirework( ) in the class Firework as specified in section 2 of the design document. A mock will be used to create a scenario test when the scores of each color will be given as the specific input to pass into the method. The test will be considered successful if the Firework class successfully uses the fields that store the string and displays the fireworks successfully.

4.36 Send Join Game Message Test

The Message Sent to Server unit test will demonstrate the correctness of the method sendGameInfo( ) in class JoinMenu that pass the user input to class JSON as specified in section 2 of the design document. A mock will be used to create a scenario test whether the information will be transformed into a JSON package and sent the package to JSON class successfully. This scenario will include “number of players” as 4 and “turn length” as 60 seconds. The test will be considered successful if the JSON package that is created with the given input matches the expected output.

4.37 AI Player Test

The AiPlayer unit test will demonstrate the correct initiation of the Ai player taking over for the human player as a “break” feature as specified by section 2.11 of the design document. A scenario will be created where the human player initiates the break function and the Ai will take over playing for the human player. The Ai should be able to make any type of move(play, discard and hint). Additionally the feature should also be able to be turned off. Thus the Ai would cease to make move decisions and the human player would be able to resume making plays. This test will be considered successful if:

- when the break button is pressed the Ai instantly begins to gather the necessary information to make the next play decision for the human player in the simulated scenario and thus there is a seamless takeover of control in the game.

- after the break button is selected the only option available to the human player for selection in the play field GUI will be to deselect the break button(and leave game). This will be able to be seen visually as well it will be able to be seen as a boolean value of isSelectable/notSelectable

- the Ai as it plays through the simulated scenarios shows that it is able to make each type of move(play, discard, hint)

- when the correct play is made it should be seen visually on the play field GUI and also it should change the corresponding field values when an action is made

- the Ai makes the “logically correct move” ie: 1)when there is a card in the Ai Players hand that can be played onto a play stack it is played and no other action is made instead. 2) If the next player has a card that is playable onto a play stack the Ai Player will hint at that player to play the card unless a play action can be made instead.

- when the break button is pressed again thus deselecting it the functionality of the Ai will be turned off. This will then cease any actions that are made by the Ai and the human player will then have the ability to make play decisions again thus the action buttons will once again be available for selection in the play field GUI . This will be able to be seen visually as well it will be able to be seen as a boolean value of isSelectable/notSelectable

For this fields of the PlayField class must be visible and accessible.

4.38 Leave Game Test

The Leave Game unit test will demonstrate the correct functioning of the player's ability to leave a game in progress as specified in section 3.1.3 of the requirements document. This test will require a mock of a game in progress. The success of this test will be determined on the correct message being sent to the server, and by the Start Menu user interface returning. The former will be tested with a hard-coded test, but since the latter pertains to a graphical interface its success will be determined visually.

4.39 Tutorial Demonstration Test

The Tutorial unit test will demonstrate the correct functioning of the tutorial as specified in section 3.1.1 of the requirements document. The test will be considered successful if all of the expected pictures are displayed in the expected order. The success of the test will be determined by visually inspecting the display, which should show a series of pictures in order. This test requires a copy of the pictures and the order they will be displayed in to compare with what is displayed.

4.40 Check Time Test

The "Check time" unit test will demonstrate the correct functionality of the class Time as specified in section 2 of the design document. The test will be considered successful if the class successfully gets the real time and tracks the time as the program runs, and the class records and track various time intervals.

4.41 Launch Settings Test

The Launch Settings test will demonstrate the successful launching of the Settings menu as described by the use-case diagram displayed in Figure 3.1 of the requirements document. A mock will be used to create a scenario where the player is taking action from the starting menu. This test will be considered successful if

- the player is successfully redirected to the correct screen upon selecting the SETTINGS command that is presented in the starting menu,
- the display is correctly updated to represent the visual information and actions that are associated with the Settings menu, and
- the player is successfully redirected back to the starting menu upon selecting the BACK command that is presented in the settings display.

This test requires the fields of the start menu must be visible and accessible.

5. Summary

The tests detailed in this test plan are designed to test the main functionality of our Hanabi client. The plan consists of two end to end system tests, fourteen integration tests and 50 unit tests. Not every method can be tested extensively due to time constraints, but if the tests contained within this test plan succeed the client will meet the minimum requirements to be functional. The tests contained within this plan are designed with the intent to ensure that the game is playable at minimum.