Name: Elgene Menon Leo Anthony
Student ID: 300492604

Reflection – Cluedo Assignment 2

GUI Discussion

Our team strategized the creation of the Graphic User Interface (GUI) by deciding how many different stages our program would have, what each stage would look and what it would look like as it changed from one stage to another. We chose the appearance of the GUI and how it would cooperate with other portions of the program including the model code (which we had created in assignment 1). We decided to use a Model View Controller design. This made the GUI easier to use because the model did not need to be changed. To ensure that the program worked correctly we used JUnit to test it.

We divided the portions of the code into different groups. One group had the view and control portions of the program and another contained the model code. The view and control group were written in the GUI class. This was divided into methods for each portion of our GUI. We also decided which swing components to use and where they would be used.

The following swing components were used - To exhibit everything in the program we used JFrame. To exhibit the suggest and accuse buttons, the quit button, the restart button, and the end turn we used the JMenuBar and JMenuItems. To select a character and keep characters from being selected twice we used JRadioButtons. Characters could not be selected twice because those already chosen were switched to gray. The selection of game characters, the room cards, the person cards and the weapons were done with JComboBox. Each gamers name was exhibited using JTextField. Suggestions and accusations were exhibited using JDialog. Roll dice, accuse, suggest, and end turn controls were done using JButton. To pick objects used in suggestions and accusations JOptionPane was used. The MouseListener swing component was used to verify if a move was valid after a player had rolled by placing their mouse over it or to move to a valid tile after clicking it with the mouse. The frame was separated into different areas using JPanel.

Name: Elgene Menon Leo Anthony
Student ID: 300492604

Through the use of action listeners swing components would initiate the correct function which would then alter (or “paint”) the JFrame. Every assignment requirement was accomplished through the swing components. This is also true of the mouse hovering extension requirement. The swing components listed above allowed our MVC design to control the game. We planned our game functionality using a state diagram. This state diagram showed when each swing component would appear and allowed us to check we had included all required functions.

Design Discussion

We had two challenges. Firstly, the toString() method for some classes from the Assignment One code was not consistent with the code from Assignment 2. This meant it was more difficult to fix the bugs. We were able to fix these bugs, but it took some time. Secondly, most of the game logics were implemented in the Player Class instead of the Game Class. This meant that we struggled to access some states for example a player or game state. However, we solved this problem by collaborating the game logic with the GUI class, game class and players.

One major change our design made to accommodate using a GUI was the Suggestion and Accusation method. The code from Assignment 1 for Suggestion and Accusation was text based which did not interact well with GUI. So, we had to change this code to make it compliant with “Swing” in GUI.

The design our team used to add the GUI to the code from Assignment 1 was a good design. Our biggest advantage was that our classes and methods could be reused. Therefore, when we had problems using GUI, we discovered our code could be easily adapted to solve the problem because of its reusability.

Name: Elgene Menon Leo Anthony
Student ID: 300492604

Discussion of the State Diagram

Our team utilized the website draw.io to create a state diagram for several purposes. First, it made sure everyone understood how the program worked and how it would work when different functions were activated. We also used the diagram to assign the creation of separate parts of the program to each team member therefore ensuring that team members code did not cause other peoples code to malfunction.

Our state diagram has two super states – these are the pre-game state and the in-game state. Within the pre-game super state are two states, each with a different function. The first of these was entitled “Asking for player count.” It showed the GUI and allowed for the selection of the number of players within the game. It had only one transition.

This transition takes you to the second state within the pre-game super state. This state was entitled “player assigning.” Within this state, it is possible to give each game player a character and link that character to the game player’s name. The transition from this state goes to the state entitled “turn cycling state.” This is inside the in-game state and contains a method in the Game class which triggers the GUI for the following players turn if they have not yet left the game.

Once the GUI has been triggered it goes into the next state entitled “player-turn state”. The player-turn state is a super state sitting within the larger super state of the in-game super state. The player-turn state encompasses all the things that can happen to a player during their turn. The player can use different choices to move into different states for example, making a suggestion or an accusation. The player ends their turn by clicking a button or finishing their accusation. If an accusation results in the game ending, then the player leaves the in-game super state and smaller players turn super state it contains. The player is taken to another state entitled the “game ended state.” Within this state the player is asked if they wish to play again. If they select “Yes” then they are taken back to the beginning of the game the display player count GUI. If the player selects “No” the game will end completely.