SWEN 222 Report – Assignment 2: Cluedo (GUI Edition)

1 Brief

This current Assignment was an extension over the first assignment – where we had to implement a working GUI of our Cluedo program. It relies on mouse/keyboard input and event driven programming, where it relies on active user input to manipulate the games state.

I worked with my partner, Casey Huang over two weeks to build over our implementation of this game. Compared to our first assignment, we have added two new packages, namely ***cluedo.assets.tiles*** (to hold the tiles) and ***cluedo.gui*** to hold the View and the Controller classes.

We attempted to implement MVC for this assignment, where the CluedoView represents the view, and the CluedoGameController represents the Controller. The classes in ***cluedo.assets*** represents the model.

In total, our final design effort consists of **6** packages and **28 classes.** We also have **45** images, which represent various in game assets, such as room backgrounds, player colour icons, and game cards.

2 Design

2.1 Visual Design

The game board is represented by a two-dimensional array of tile objects. The Tile class is a superclass, which defines an x and y location, a default colour, and a player (if one is standing over it). To represent the different varieties of possible Tiles, we made the DoorTile, RoomTile, SolutionTile, StairsTile, StartTile, and WallTile classes, with their differing characteristics.

We drew background images on top of every room. That way, we feel like it improves the overall look of the game.

When the game is running, each player is represented by a coloured token on the board. For example, Colonel Mustard appears on the game as a yellow token, and Miss Scarlett appears as a Red token.

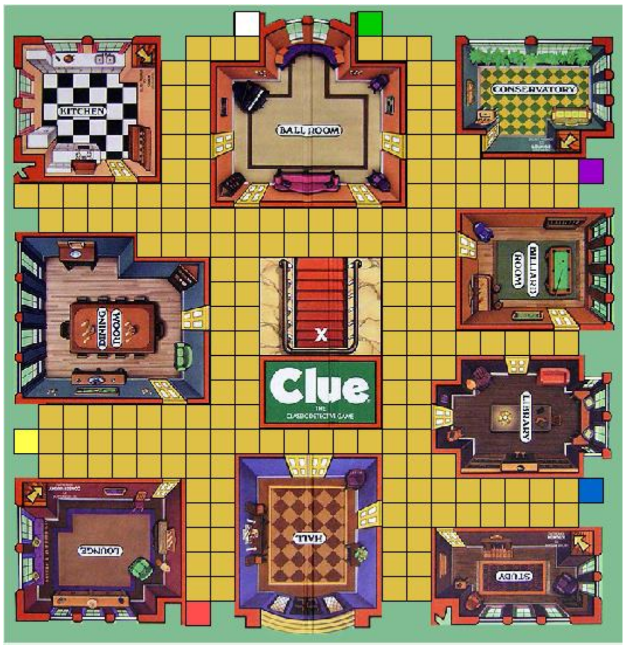


Figure 1 - A screenshot of the Board.

To the left of the board, are the player's controls. I have listed the current players name, along with his colour piece is listed at the very top. There are eight JButtons for the game which represent in-game actions, such as starting/ending your turn, rolling the dice, and making a suggestion or accusation.

The middle represents the current value of the dice roll, and the bottom represents the current list of players who are playing.

At the very top of the window, I have placed a JMenuBar for certain actions. The File menu has two menu items, which are "Start Game" and "Exit Game". The Help menu has two items, which are "Game Instructions", and "About CluedoGUI".

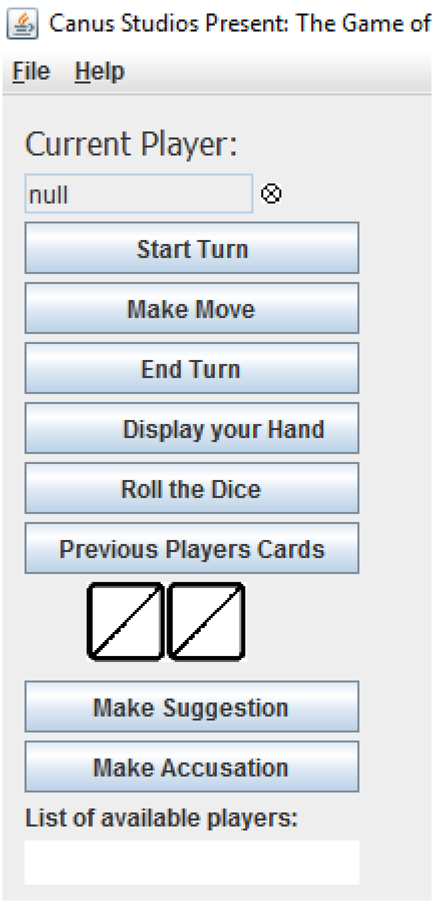


Figure 2 - The Controls of the Board

2.2 The Click Sequence

Mouse click events are delegated by the CluedoGameController class. We implemented the MouseListener and MouseMotionListener interfaces in order to make this possible.

To check if a mouse click is valid, the GameController calls the current player object, and checks if the players remaining moves is greater than 0. If so, it then finds the closest tile from the mouse click, and checks the subtype of that tile.

For example, if a player is in a room, and they clicked on a doorTile, it will call the exitRoom() method inside the CluedoCanvas, and redraw the canvas. Likewise, if a player was not in a room, it would call the move() method, which is also inside the CluedoCanvas class.

After calling the appropriate move method, the canvas then repaints itself, and then updates the board.

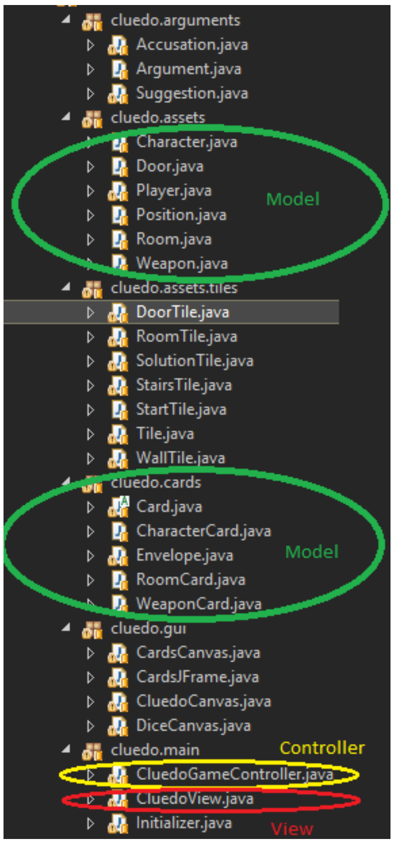
2.3 Class Hierarchy

We attempted to use Model-View-Controller design to make the game. For example, our Model is the classes in the assets/cards packages, the View is CluedoView, which is the JFrame holding the main window, and the Controller is the CluedoGame.

The Game is executed through the CluedoView. It sets up and displays the interface, and calls the CluedoGameController class and delegates the games logic. The actionListeners for the JButtons call methods in the GameController.

The Models represent the assets, such as the Cards, the game objects, and the Envelope data structure.

Figure 3 - Model-View-Controller setup.



```
cluedo.arguments
  Accusation.java
  Argument.java
  Suggestion.java
cluedo.assets
  Character.java
  Door.java
  Player.java
  Position.java
  Room.java
  Weapon.java
cluedo.assets.tiles
  DoorTile.java
  RoomTile.java
  SolutionTile.java
  StairsTile.java
  StartTile.java
  Tile.java
  WallTile.java
cluedo.cards
  Card.java
  CharacterCard.java
  Envelope.java
  RoomCard.java
  WeaponCard.java
cluedo.gui
  CardsCanvas.java
  CardsJFrame.java
  CluedoCanvas.java
  DiceCanvas.java
cluedo.main
  CluedoGameController.java
  CluedoView.java
  Initializer.java
```

3 Final thoughts and Conclusion

In conclusion, I feel quite happy/confident in our program. That said however, I feel like there could be a few things that could have been improved.

At first, I feel like the CluedoView class got a bit too bloated, and too big. At first, I was performing the game's logic inside of the View class, and I feel like some of the action Listener code was a bit too "meaty".

I also feel like for moving, that we should have implemented some sort of search algorithm, such as A*. Our implementation of moving relied on checking the difference between the players current square and the destination square. This means that pressing on an early square will use your turn up, even if you had some moves remaining.

All in all, I enjoyed this assignment, even though it was quite though to learn how to use Swing and JFrames at first.