Our implementation of the GUI we used for our Deadwood game, we tried to use the MVC pattern but ultimately just used a model and view. Our model that contains the information about the state of the game includes all of the classes we wrote for A4. This way, we were able to keep track of data like the number of days left in the game, the number of scenes left on the board, the location and rank of players, and many other things. Our view handles all of the visual aspects of the display that the user sees while playing the game. Within our BoardView class, there are methods responsible for displaying images of the board, scene cards, shot markers, and player die. There are also methods we use to remove those images for scenarios like when a player moves to a new set or when a scene card is flipped over.

There are a few small bugs in our implementation of Deadwood that occur after the first day has ended. All of the data and logic in the model component still works, but the view does not remove the backs of every scene card on the second day of the game, despite there being a valid scene for the player to act on. In fact, despite the scene card back not getting removed, the player can still take a role and act as normal despite the visual bug. The other bug with the second day of the game is that certain scene cards do not visually get updated despite the model containing the new scene card. The scene card from the previous day is still displayed. Despite both of these bugs with the view, the game logic remains intact and the game is still very playable. (So players can still act as normal, ignoring the outdated view). We worked for hours trying to understand why the cards were not being removed as they should have been, with little success. We think the reason the cards are not being updated correctly has something to do with creating a new JLabel in the setup and then trying to remove them is removing the different instance of the JLabels. We tried writing new methods to remove scene card backs that got called specifically after a new day but this still didn’t work, we added various printlns to the methods that weren’t working to make sure the methods were still being called. The methods were being called correctly but the view was still not updating the way it should be. If we had more time to work on this project, we would have added new methods that placed the scene cards by re-creating the scene cards completely so they we being placed the initial way that we know works, or create arrays the handle each JLabel instantiation.

One assumption that we made is that player’s could be automatically assigned a colored die based on what player number they were. To elaborate, player 1 will always have a blue player die, player 2 will always have a cyan die, player 3 will always have a green die, etc. This was mainly to simplify the code since selecting specific colors would have required more work than was necessary to make the game functional. Given more time, we could have added code that allowed players to select which color they wanted their player die to be. Another assumption that we made was that when a player makes an invalid choice on their turn, they forfeit their turn and play continues with the next available player. For example, if a player tries to act if they don’t have a role, their turn will automatically end.

To test our code, we created a separate Deadwood.java file outside of our model and view. As we added methods to our BoardView class, we would call those methods from Deadwood.java to see if our methods were functioning in the way that we wanted them to, as well as to see what the game display looked like so we could tweak the spacing of the images and text if needed.

The most challenging part of this assignment was with the View aspect of this assignment. For example, having images and players move to the correct locations on the board was very tricky and we had issues getting images to be replaced after each day, which is where our bugs appear.

An interesting variant of this assignment could be to make the player die draggable. This way, players could drag their die into an adjacent set or onto a role either on or off of the scene card. While the functionality of the move and take role options would be the same, the ways to implement them would be different. The time frame for completing this assignment is probably too short to make dragging the player die a requirement of the implementation, but it could be added an option for extra credit for students that have more time and want a challenge.