Design Pattern Report

Group: Archan Rupela, Ivan Chuprinov

In assignment 3 we implemented a Model-View-Controller design pattern because of its clean and easily upgradable style. The user, when playing the game, will always only interact with the 'GUIDeadwoodView" class and nothing else. It communicates directly with the "DeadwoodController" class to receive player progress and information about their character while playing. The "DeadwoodModel" class acts as our model, it communicates with the controller and receives tasks sent by it to handle all internal game logic. This design style is especially useful when we created our GUI because little to no changes had to be made in our model.

Another design pattern we had used when creating this game was Singleton; in our view and controller class (and many others) we added functionality to have at most one instance of that class so that information can only be flowed one way and we don't create duplicate objects that would not communicate with others. This new design pattern was useful because it is easier to organize, increases code readability and allows more code reusability. However, the downsides are the fact that it creates heavier coupling and is less memory efficient than an optimal solution.