In our implementation of Deadwood, my partner and I used the following design patterns: Model View Controller, Singleton, and Observer.

The reason we chose to use Model View Controller was because our end goal was to create a boardgame with a working GUI, which meant that we needed to create something where a user could take an action, have that action be reflected on the screen, and then (potentially) have some stored data change. Using MVC to create a set of model classes, a set of view classes, and then a controller class allowed for our classes to be separated in a logical manner and created code that was easy to parse and intuitively understandable.

We chose to implement the Singleton design pattern on two of our classes: Board and Game. Our reasoning was that Board and Game are initialized only once, and afterwards we would only want references to the original object.

Finally, we decided to use the Observer design pattern in order to implement our Model View Controller design pattern. We needed a way for our Model classes to notify our View classes when data was changed and for our View classes to update the GUI when they were notified by the Model classes.