Project Timeline

Work	Dates
Create project file structure and Github repo	December 1
Construct the function headers	December 2 - 3
Implement the header functions and main file	December 4 - 9
Create test suite	December 10 - 11
Test program and fix errors in implementation	December 11 - 13
Update UML, make demo plan and design doc	December 13 - 14
Finishing touches and submission	December 14 - 15

Question: What sort of class design or design pattern should you use to structure your game classes so that changing the user interface from text-based to graphical, or changing the game rules, would have as little impact on the code as possible? Explain how your classes fit this framework.

Answer: I've divided the game into a Table class which handles most of the game and a Player class that deals with an individuals moves. Doing so allows for easy changes to the interface and game rules. If a legal game move needs to be changed, I only need to change part of the Player class to allow for this. Similarly, to handle the interface of the game, I need only implement an observer and create multiple forms of interface and the rest of the program remains the same. Splitting classes into smaller and more specific classes reduces coupling and allows for fewer coding changes to be made to implement a change to the game itself.

Question: Consider that different types of computer players might also have differing play strategies, and that strategies might change as the game progresses i.e. dynamically during the play of the game. How would that affect your class structures?

Answer: All types of players inherit from the Player class and so implementing different computer strategies would not be difficult. All information needed for a given strategy is stored in the Player class. Implementing multiple computer sub-classes with different playing strategies would require implementing new sub-classes of Player and would not change the Player class itself. For the computer players to dynamically change based on the progression of the game, only a change to the given computer player sub-class needs to be made. This change will allow the computer player to dynamically change their strategy without affecting the parent Player class.

Question: How would your design change, if at all, if the two Jokers in a deck were added to the game as wildcards i.e. the player in possession of a Joker could choose it to take the place of any card in the game except the `7s`?

Answer: I would make a change to the function that checks which cards can be played in the Moves class. As the Joker can replace any card (aside from the 7s), overriding the legal playing card function would be easy, and allow the given player to treat the card as any of the legally playable cards. A change would also need to be made to the function that plays a card, where the player that posses the wildcard would need to indicate which card they are playing it as when it is played.