Refactoring and Design Pattern List

Design Patterns Used:

-Polymorphism: We used this GRASP pattern when designing the pieces for the game board. This pattern allowed us to create different pieces with variations of behavior, such as movement, while writing methods that handle them all the same.

-Controller Observer: We used the Observer pattern throughout this project as way to update state/dependent objects automatically when certain methods of subject objects were executed.

-Façade: We used the Façade pattern to simplify the interface between the Model and Server

Refactoring:

Pull request #39 Commit "Changing from y,x to row,col":
-To avoid indexing double arrays as arr[x][y], we changed x,y to col,row so we now index double arrays with arr[row][col], which is easier to understand.

Pull request #40 Commit "Constructor in piece was added":
-Pull up method used to set isFlipped variable in superclass instead of in all of the subclasses

Pull request #50 Commit "Now cannon can move without capture":
-Extracted code from isMoveValid in the Cannon class into the method decideBetweenMoveOrCapture()
-Extracted code from isMoveValid in the Cannon class which makes sure pieces passed in are of different colors into the method arePiecesDifferentColors
-Pull up method used to put arePiecesDifferentColors into the Piece superclass

Pull request #51 Commit "Refactoring":

-Restructured project so all test classes are in the test folder, all pieces are in the pieces folder, all controllers are in the controllers folder, and all exceptions are in the exceptions folder.

Pull request #59 Commit “Develop API”:
-Extracted username validation and setting class variable username into method setUsername