**UML Class Diagram**

**Collections**

We used a Map (HashMap<String, Boolean> information) in our piece class that stores the Boolean values for kingness, redness, and aliveness.

We also used a Set (ArrayList<Piece> pieces) in order to store pieces in our Board class.

**Design Changes**

We removed the AI class. Setting up the graphic interface, implementing an undo button, allowing for multiple jumps, and making sure the program would reject any invalid moves took up the majority of our time and weren’t up to trying to implement a recursive function and use logic trees to create an AI.

We added the method isValidMove to our Board class. This method took up the majority of our time spent on the logical functionality of the Checkers game. We decided that we could not trust the user to make a valid move, as even one of our group members didn’t know how to play checkers, so we created the method that make MANY checks to make sure that the user does not portal to places it isn’t allowed to go.

We added the method mousePressed in our game class. We realized that in order to properly implement java graphics we needed this method in the same class as main. This method also is in charge of allowing for undo’s and multiple jumps. Since most board games have an undo function, we decided it would be wise to add one for our own game.

We added a copyBoard and copyPiece to our Board and Piece class. copyPiece was very useful for the move functionality, and copyBoard’s deep copy became vital for our stack of Boards that we used for our undo function.

We added a capture method in board that allows us to separate the capture action from isValidMove. This allows for a future addition of an AI so it can check if a move is valid without committing to capturing a piece.