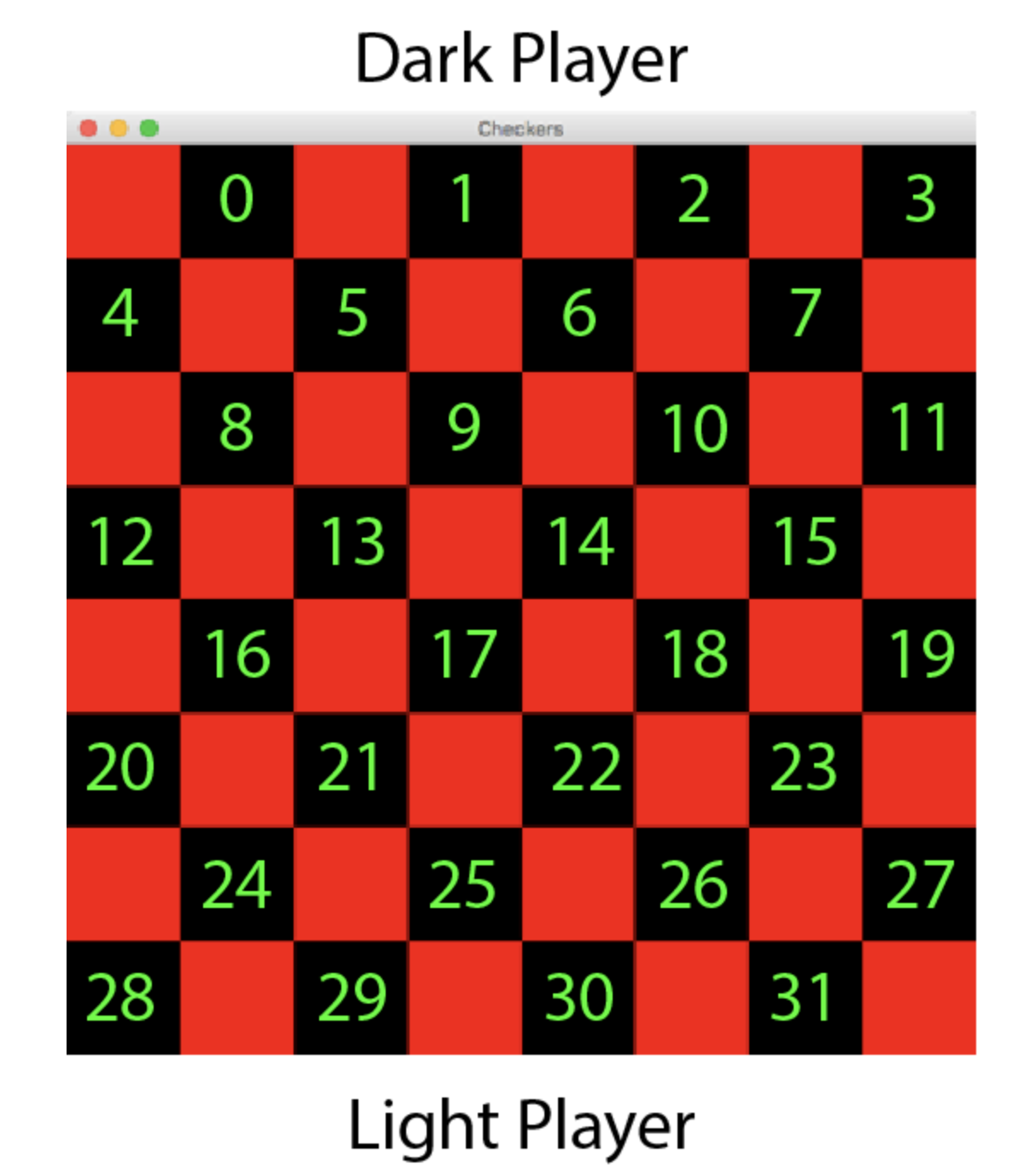
David Emily

CS4330

Checkers Data Model Concept

The checkers game can make use of Java’s OO structure by relying on objects to handle the checkers. On a new game, the checkers will need to be placed on squares 0-7 for dark and 24-31 for light. 

A simple counter can be used to keep track if it the dark player’s turn or the light player’s turn. Simply bumping the counter and then added the modulus operator can handle this. I.E. counter % 2 == 1 means it’s the light player’s turn else dark player’s turn. The dark player will have choices to move each piece to a position either +3 or +4 their current space; the light player will be -3 or -4. Before each turn, a test should be running to see if there are any playable moves. To check the eligible moves, each piece should be checked to see if there is a move opportunity. If the move opportunity is blocked by the opponent’s piece, check for a jump opportunity. If there is no jump or move opportunity, that piece cannot be moved. If the player cannot move any of their pieces, it should be declared a winner for the opponent.

If the current player can move a piece, give them the option to do that. If the chosen position is occupied by the opponents piece (+3, +4 for dark -3 -4 for light), check to see if the jump position is open (+7 +9 for dark, -7 -9 for light). If a jump is chosen, remove the opponent’s piece and place the player’s piece in the chosen position. If this was the opponent’s last piece, than the current player wins and the board is reset.

If the spot is the opponent’s final row (0-3 for light, 28-31 for dark) then the piece that lands there needs to be changed into royalty. The royal piece should have a different appearance to differentiate it from the other pieces, and have a different move set. The royal piece can move to any directly connecting black square, either front or back -- IE.-5, -4, +4, +5.

If the opponent’s last piece was not taken, the turn counter is bumped and the other player has the chance to go.

