**-don’t show private fields/private methods in interface design (GRADER file)**

**-final static constant for colors (black, white) and empty square**

**-what is getValue() doing??**

**-just need alphaBeta() method to search to depth and pass good moves up while pruning**

**-evaluation function assigns scores**

**-changed return type of alphaBeta() to Move instead of int**

**-DList probably better to keep track of list of valid moves and connected chips instead of array**

**-for step move, need to specify where you’re moving (make distinction between step/add**

**moves**

**Square(int x, int y)**

Fields:

Chip chip = null;

int x, y;

Methods:

int getX()

int getY()

Chip getChip()

**Chip(int color, Square location)**

Fields:

int color;

Square location;

Methods:

int getColor()

     -returns the color of "this" Chip

Square getLocation()

     -returns the location of "this" Chip

**GameBoard()**

Fields:

Square[][] board;

Methods:

private boolean hasNetwork(int color)

     -checks if "color" has valid Network

private boolean isValidMove(int color, Square location)

     -checks if "location" is a valid move for "color" on "this" GameBoard

public DList validMoves(int color)

     -returns a list of Squares that are valid moves on "this GameBoard

Chip[] connectedChips(Chip chip)

     -returns a list of chips connected to "chip" on "this" GameBoard

boolean isConnected(Chip a, Chip b)

     -checks to see if "a" and "b" are connected on "this" GameBoard

**MachinePlayer()**

Additional Methods:

int getValue(GameBoard board, int depth, int alpha, int beta)

     -returns the max score for the maximizer and min score for the minimizer

Move alphaBeta(GameBoard board, int depth, int alpha, int beta)

     -performs minimax algorithm with alpha-beta pruning for getValue()

int evalFxn(GameBoard board)

     -assigns a score to each possible board