Shane Bishop
CS4330 OO Design
Checkers Game Data Model Concept

User Actions

1. The first thing that would be asked of the users would be what kind of game they would be playing, as well as defining the number of human players and computer players.
2. Once the game type and number of players is decided names would be given to each player, both computer and human types of players would have names applied.
3. The players would then be offered a choice on a coin flip in order to choose sides, using random number generation to produce a result and giving the winner choice of color.

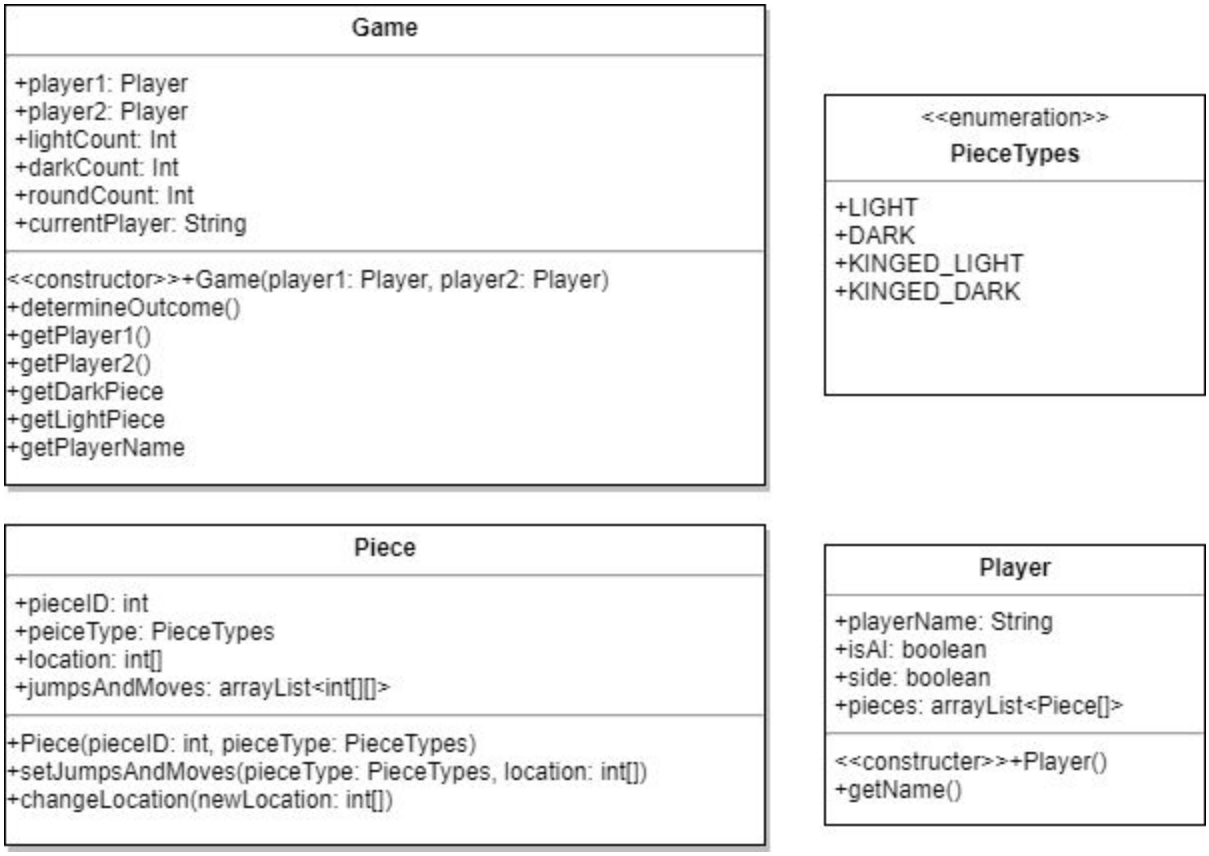
Game Board

1. The board would be an eight by eight grid, with each subsection being given an x-coordinate and y-coordinate to determine position to be used for game movement and data. The purpose of this design is to provide the most flexibility and would allow for the adoption of further game modes easily.
2. Each side would be filled with 12 pieces each, filling the first three rows of dark spaces with the corresponding sides' colored pieces.

Move Rules / Player Choices

1. Pieces should never populate lighter spaces, thusly spaces with even x-coordinates and even y-coordinates and odd x-coordinates and odd y-coordinates will always be considered out of bounds (ex. 2,4 or 3,7).
2. In the case of a two person game, players will only be able to move their pieces on their own turns.
3. If a dark piece reaches the bottom of the grid, it will be made a king, by replacing the icon with a crown, and the same will happen to a light piece if it reaches the top of the grid.
4. Pieces can only move diagonally, and non king pieces may only move away from their starting side.

UML


Game
+player1: Player
+player2: Player
+lightCount: Int
+darkCount: Int
+roundCount: Int
+currentPlayer: String
<<constructor>>+Game(player1: Player, player2: Player)
+determineOutcome()
+getPlayer1()
+getPlayer2()
+getDarkPiece
+getLightPiece
+getPlayerName
<<enumeration>>
PieceTypes
+LIGHT
+DARK
+KINGED_LIGHT
+KINGED_DARK
Piece
+pieceID: int
+peiceType: PieceTypes
+location: int[]
+jumpsAndMoves: arrayList<int[][]>
+Piece(pieceID: int, pieceType: PieceTypes)
+setJumpsAndMoves(pieceType: PieceTypes, location: int[])
+changeLocation(newLocation: int[])
Player
+playerName: String
+isAI: boolean
+side: boolean
+pieces: arrayList<Piece[]>
<<constructer>>+Player()
+getName()