**Checkers Game Data Model Concept**  
DARK PLAYER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1 O** |  | **2 O** |  | **3 O** |  | **4 O** |
| **5 O** |  | **6 O** |  | **7 O** |  | **8 O** |  |
|  | **9 O** |  | **10 O** |  | **11 O** |  | **12 O** |
| **13** |  | **14** |  | **15** |  | **16** |  |
|  | **17** |  | **18** |  | **19** |  | **20** |
| **21  O** |  | **22  O** |  | **23  O** |  | **24  O** |  |
|  | **25  O** |  | **26  O** |  | **27  O** |  | **28  O** |
| **29** |  | **30** |  | **31  O** |  | **32 O** |  |

LIGHT PLAYER

**public class Game** {

private Board board;

public Game() {}  
 public boolean initializeBoard () {}   
}  
  
**public class Player** {

public boolean isLight;

public int score;

}  
  
**public class Board** {

private Checker[][] checkers;

private Square squares;  
public void setSquares(Square square) {}

public void setCheckers(Checker checker) {}

public Board(){}  
}  
  
**public class Square** {  
 int x;

int y;

Checker checker;

public Square(int, int) {}  
 public void isOccupied(){}  
 public void isAllowedMove(Player player){}  
 public void setCheckerOnSquare(Checker checker){}  
 public Checker removeChecker(){}

}  
**public class Checker {** public boolean isKing;public int currentPosition[][];

public int attemptedPosition[][];  
public int newPosition[][];

public bool isMoveLegal;

public void getColor(Player player); **}**

To implement checkers in Java, I would keep the numbering notation standard according to the game notation, as pictured above, starting at one on black at the top with a white top left corner, and continuing each dark space in order left to right. I assume logic for moving pieces will be adding or subtracting to determine allowed directions for the checker being moved based on the player color. I want to keep track of where it is (in case move is invalid and need to return it until moved again), if it is a king or regular checker (so I know legal moves) where they want to move it. If the player lands on an unoccupied, legal space, then a counter for players score for each piece jumped over will be increased and the checker on the square that was jumped over will be removed. The next players turn would then begin. If the piece lands in the corresponding opposite end, it is kinged, and then can move any direction. If there are no legal moves, the player with the highest score should be displayed as the winner.