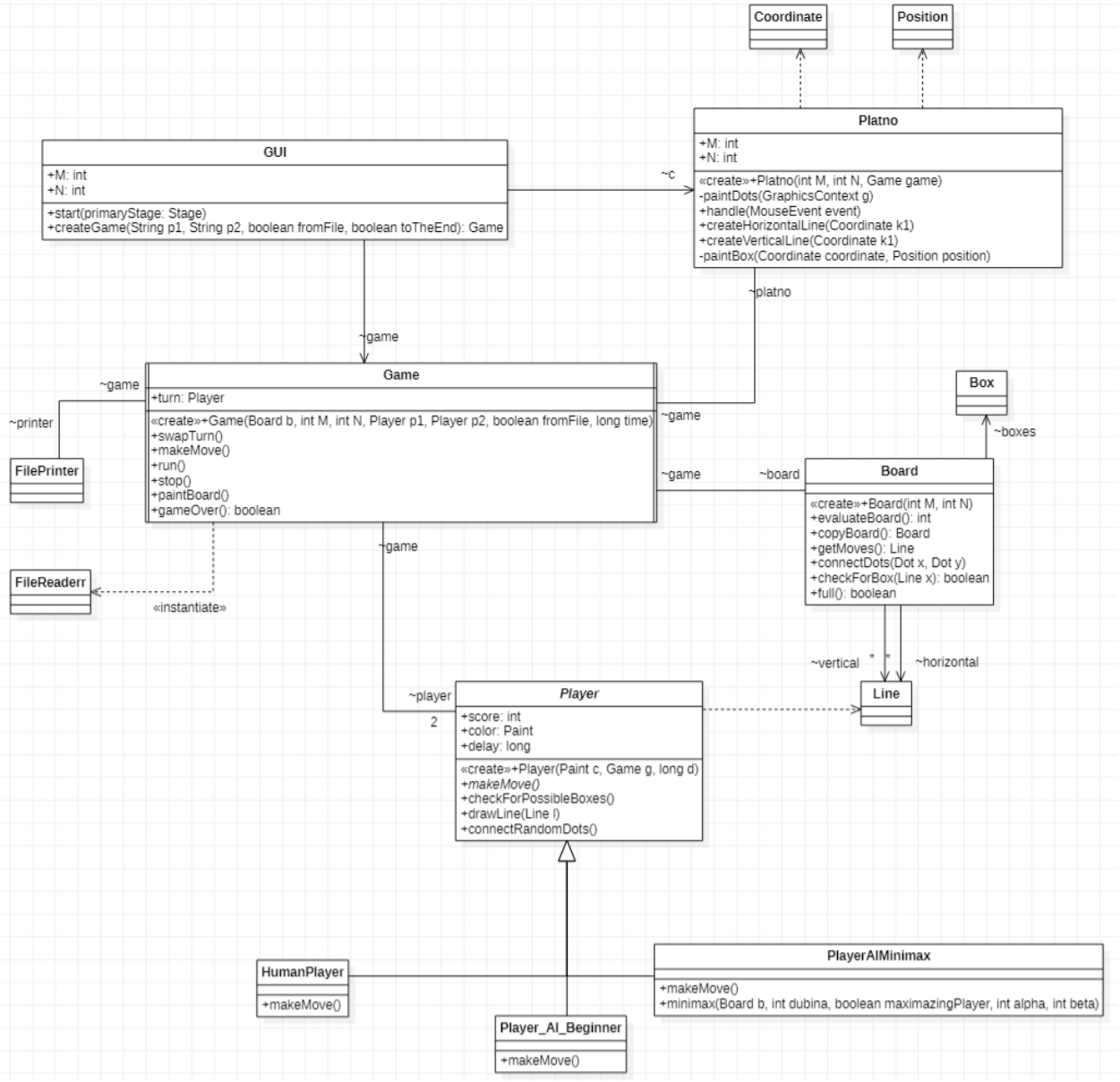
Izveštaj IS 2020 – Dots And Boxes

Ovaj izveštaj izdvaja najbitnije klase I algoritme, implementirane prilikom izrade domaćeg zadatka iz predmeta Inteligentni sistemi. Zahtevi domaćeg zadatka su obuhvatali implementaciju igrice Dots And Boxes u Java programskom jeziku sa akcentom na Minimax algoritam. U nastavku je priložen UML dijagram klasa I ukratko su objasnjeni bitniji algoritmi.



Slika 1. UML dijagram klasa

Od interesa su metode makeMove () podklasa klase Player.

- U klasi HumanPlayer metoda makeMove () vrši uposleno čekanje na potez korisnika.
- U klasi Player_AI_Beginner metoda makeMove() uvek pokusava da postigne poen, ukoliko to nije moguće povlači nasumičan potez
- U klasi PlayerAIMinimax metoda makeMove() koristi Minimax algoritam sa alfa-beta odsecanjem zadate dubine I na osnovu njegovog rezultata povlači potez. Na slici 2 je prikazan Java kod pomenutog Minimax algoritma

```
public MinimaxReturn minimax(Board b, int dubina, boolean maximazingPlayer, int alpha, int beta) {
    int bestScore, currentScore;
    Moves bestMove = null;
    MinimaxReturn pom = null;
    ArrayList<Moves> moves;
    moves = b.getMoves();
    if (b.full() || dubina == 0 || moves.size() == 0) {
        return new MinimaxReturn(null, b.evaluateBoard());
    }
    if (maximazingPlayer) {
        bestScore = Integer.MIN_VALUE;
    } else {
        bestScore = Integer.MAX_VALUE;
    }
    Collections.shuffle(moves);
    for (Moves l : moves) {
        Board newBoard = b.copyBoard();
        newBoard.addLine(l);
        if (maximazingPlayer) {
            pom = minimax(newBoard, dubina - 1, false, alpha, beta);
            currentScore = pom.getValue();
            if (currentScore > bestScore) {
                bestScore = currentScore;
                bestMove = l;
                if (bestScore >= beta) {// odsecanje, ako smo izasli iz prozora
                    return new MinimaxReturn(bestMove, bestScore);
                }
                alpha = Math.max(alpha, bestScore);
            }
        } else {
            pom = minimax(newBoard, dubina - 1, true, alpha, beta);
            currentScore = pom.getValue();
            if (currentScore < bestScore) {
                bestScore = currentScore;
                bestMove = l;
                if (bestScore <= alpha) {// odsecanje, ako smo izasli iz prozora
                    return new MinimaxReturn(bestMove, bestScore);
                }
                beta = Math.min(beta, bestScore);
            }
        }
    }
    return new MinimaxReturn(bestMove, bestScore);
```

Slika 2. Implementacija Minimax algoritma sa alfa-beta odsecanjem