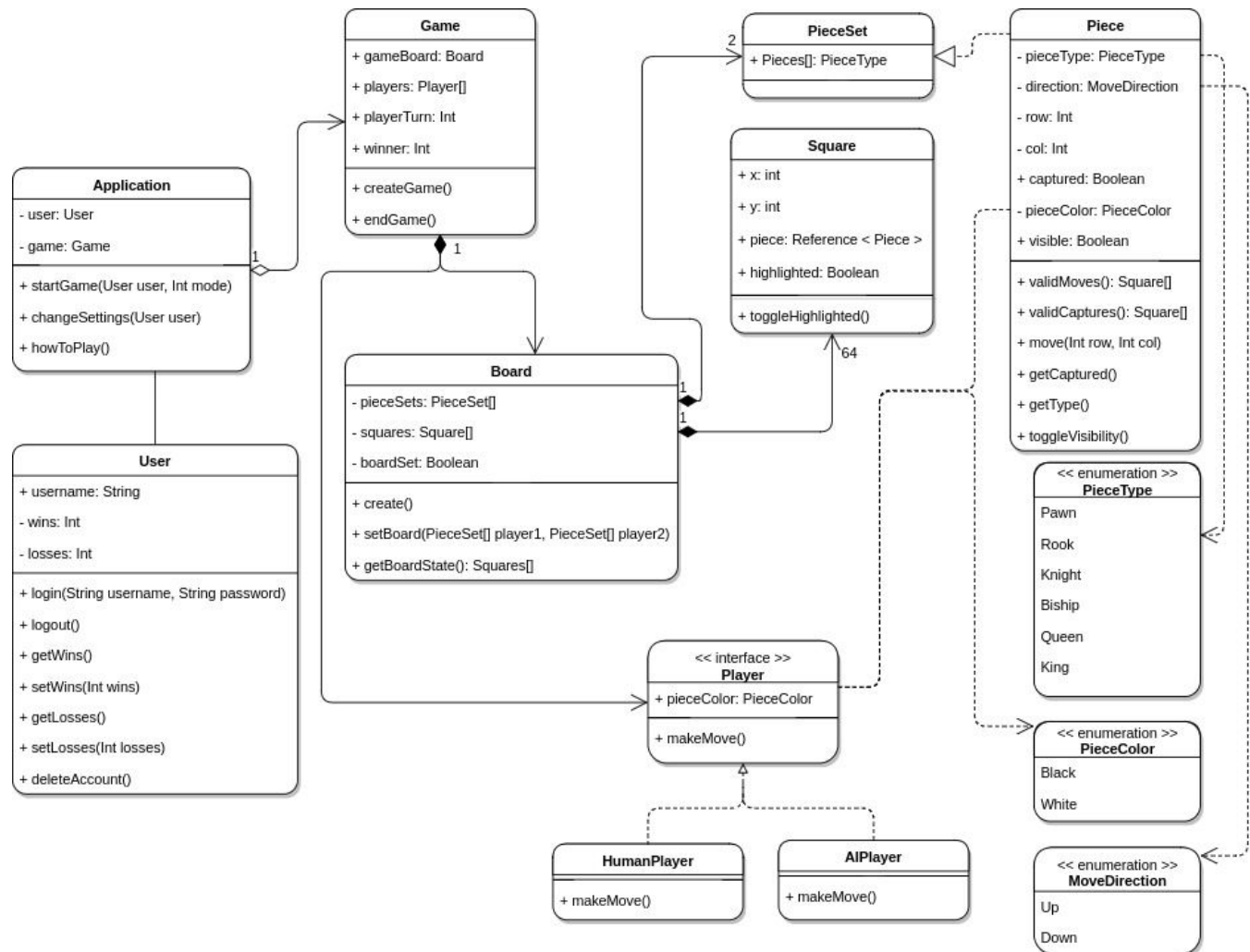


Team: Hunter Leise
Vincent Mahathirash
Raymond Duncan

Title: Chesspionage

Part 2 Class Diagram:

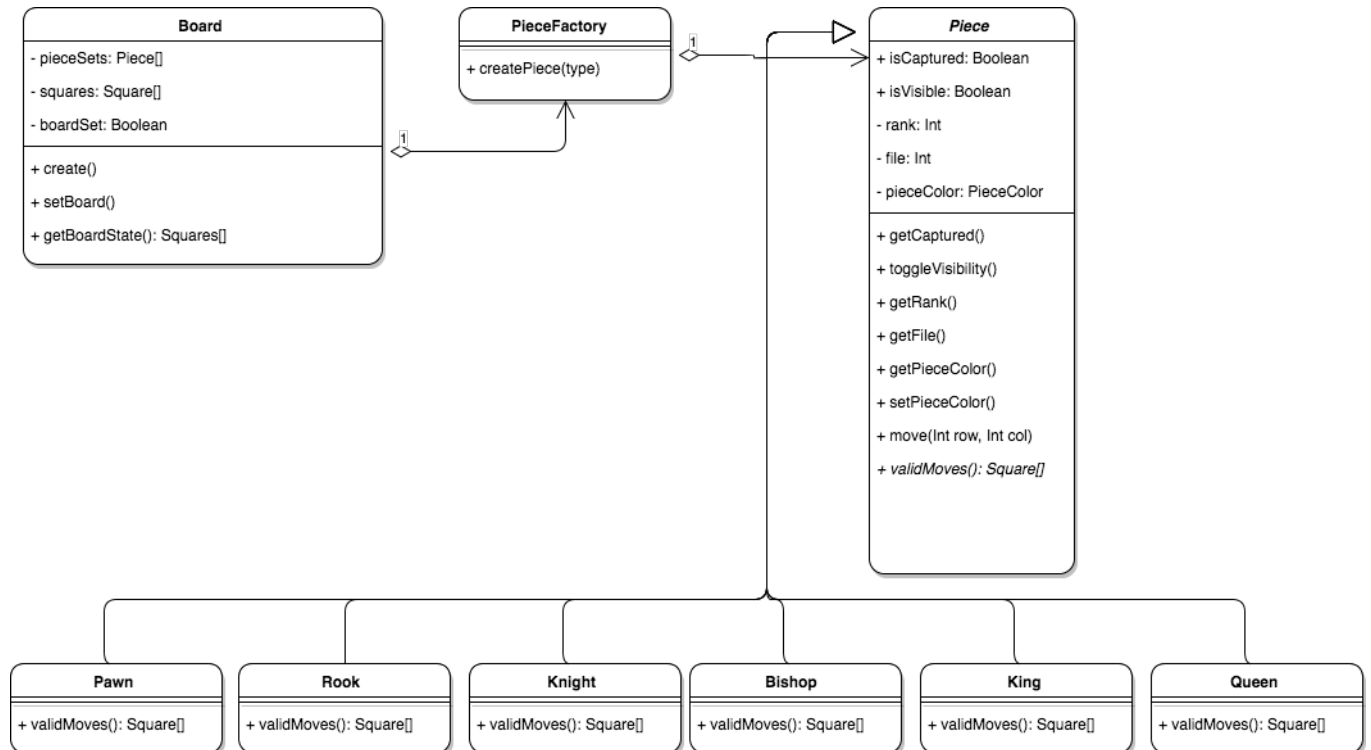


```
classDiagram
    class Application {
        - user: User
        - game: Game
        + startGame(User user, Int mode)
        + changeSettings(User user)
        + howToPlay()
    }
    class User {
        + username: String
        - wins: Int
        - losses: Int
        + login(String username, String password)
        + logout()
        + getWins()
        + setWins(Int wins)
        + getLosses()
        + setLosses(Int losses)
        + deleteAccount()
    }
    class Game {
        + gameBoard: Board
        + players: Player[]
        + playerTurn: Int
        + winner: Int
        + createGame()
        + endGame()
    }
    class Board {
        - pieceSets: Piece[]
        - squares: Square[]
        - boardSet: Boolean
        + create()
        + setBoard()
        + getBoardState(): Squares[]
    }
    class PieceFactory {
        + createPiece(type)
    }
    class Square {
        + x: int
        + y: int
        + isHighlighted: Boolean
        + piece: Reference < Piece >
        + toggleHighlighted()
    }
    class Piece {
        + isCaptured: Boolean
        + isVisible: Boolean
        - rank: Int
        - file: Int
        - pieceColor: PieceColor
        + getCaptured()
        + toggleVisibility()
        + getRank()
        + getFile()
        + getPieceColor()
        + setPieceColor()
        + move(Int row, Int col)
        + validMoves(): Square[]
    }
    class PieceColor {
        <<enumeration>>
        Black
        White
    }
    class Player {
        <<interface>>
        + makeMove()
    }
    class SkillLevel {
        <<enumeration>>
        Easy
        Average
    }
    class Strategy {
        <<interface>>
        + findMove()
    }
    class EasyImplementation {
        + findMove()
    }
    class AverageImplementation {
        + findMove()
    }
    class HumanPlayer {
        + makeMove()
    }
    class AIPlayer {
        + skillLevel: SkillLevel
        + makeMove()
        + calculateMove(skillLevel): Square
    }
    class Pawn
    class Rook
    class Knight
    class Bishop
    class King
    class Queen

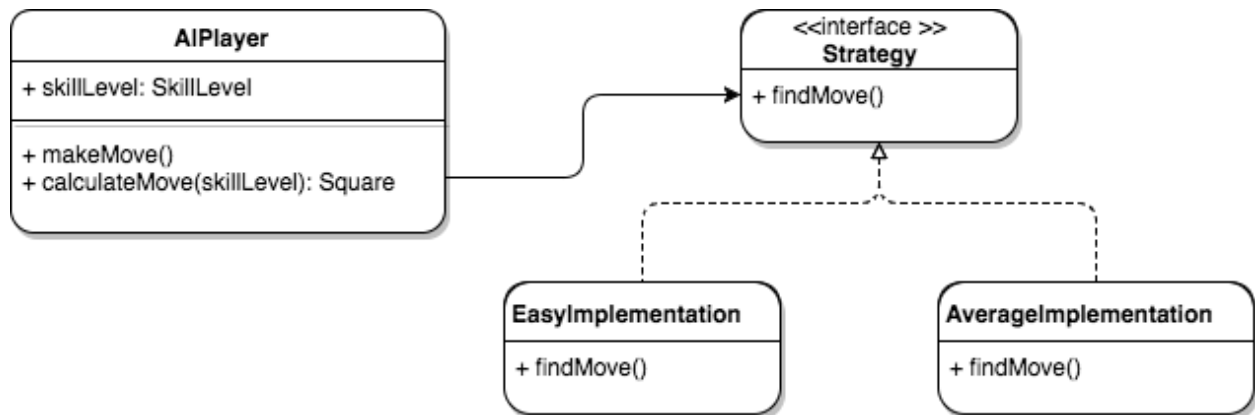
    Application "1" --> "1" Game
    User --> Game
    Game "1" *-- "1" Board
    Game "1" *-- "1" PieceFactory
    Board "1" *-- "64" Square
    PieceFactory --> PieceColor
    Square --> PieceColor
    Piece --> PieceColor
    Player <|-- HumanPlayer
    Player <|-- AIPlayer
    SkillLevel <|-- AIPlayer
    Strategy <|-- EasyImplementation
    Strategy <|-- AverageImplementation
    Pawn --> Piece
    Rook --> Piece
    Knight --> Piece
    Bishop --> Piece
    King --> Piece
    Queen --> Piece
```

Explanation:

We chose to implement a **factory method** design pattern for the creation of pieces because our original plan of using enumeration didn't easily allow us to customize the `validMoves` method for each piece type. By using enumeration, we would have needed to use a switch case statement inside the piece class's `validMoves` method to control how each type of piece moved. By factoring this out into specific piece subclasses, we not only make the code cleaner to read, but we also make it more extensible in the future.



We chose to implement a **strategy** design pattern for the AI player decision making because it allowed us to easily control the AI's difficulty by changing which strategy implementation we use at runtime. In doing so, we can easily extend the AI strategy to include more difficulty levels in the future.



Other than those two design patterns, we chose to get rid of the `PieceSet` class because it could be easily represented as an array of `Pieces` in the `Board` class instead, we added a few getters and setters for private variables that we forgot in our original class diagram, and we removed some unnecessary methods such as `ValidCaptures` in the `Piece` class. In addition to those changes, we fixed small issues mentioned in our project 2 feedback such as a few arrow notation errors and interfaces containing attributes.