National Swag Agency APCS pd 6
Nora Miller, Sophia Eiden, Ameer Alnasser Spring 2022

Driver	Board	Piece:
private boolean _over private int _whiteTurn - either 1 or -1 private Board gameBoard public Driver() - initializes board with pieces in standard configuration private boolean checkDanger(String color, String where) - checks if a king of specified color at specified location is in danger private void go() - invokes turn() if _over == false private void turn() - boolean inDanger - Piece captured - Piece moving - checks if king is in danger. - prompts user for input; assign moving - if king is in danger and moving != king, turn() - handles castling case - if moving is a king, checks if new location is in danger. If so, turn() - if moving is a pawn, and it is en passant, set captured to piece captured. If pawn is moving to opposite end of board, promote by typecasting - In other cases, if new location is occupied, set captured to the occupant - gameBoard.pieceDies(captured) - gameBoard.relocatePiece(moving) - are both kings on board? if not, _over = true - _whiteTurn *= -1	private Piece[][] _contents private Piece[] _takenBlack private Piece[] _takenWhite public Board() - initializes _contents with pieces in standard starting configuration public String toString() - displays board with pieces and row and file numbers in terminal private String relocatePiece(Piece which, String where) - invokes which.move(where) - returns old location private String pieceDies (Piece which) - invokes which.move(null) - returns former location	protected ArrayList<String> _possMoves protected String _location public Piece() public getLocation() protected void checkMoves() - updates _possMoves. If _location = null, _possMoves is empty. public String move(String where) - updates _location - invokes checkMoves() - returns old location public boolean isMove(String move) - returns if move is in _possMoves

Pawn	Knight	Bishop	King	Queen
private void checkMoves() - if not blocked by another piece, the square ahead - if in the starting row, also the square two ahead of it - If there is a piece of the opposite color in the same row as itself in an adjacent file, it can move diagonally to the square ahead of the piece it would capture (en passant)	private void checkMoves() - all possible L-shaped moves from current location, including squares occupied by pieces of the other color (but not your own color)	private void checkMoves() - for each diagonal direction, include moves up to the first occupied square. If that square is occupied by a piece of the opposite color, include that as a possible move too.	private void checkMoves() - by default, “C”, or castling, is in the moveset. - use the same algorithm as in the bishop class for all other moves, except in the same row or file as the rook	private void checkMoves() - same algorithm as the bishop class; both orthogonal and diagonal moves