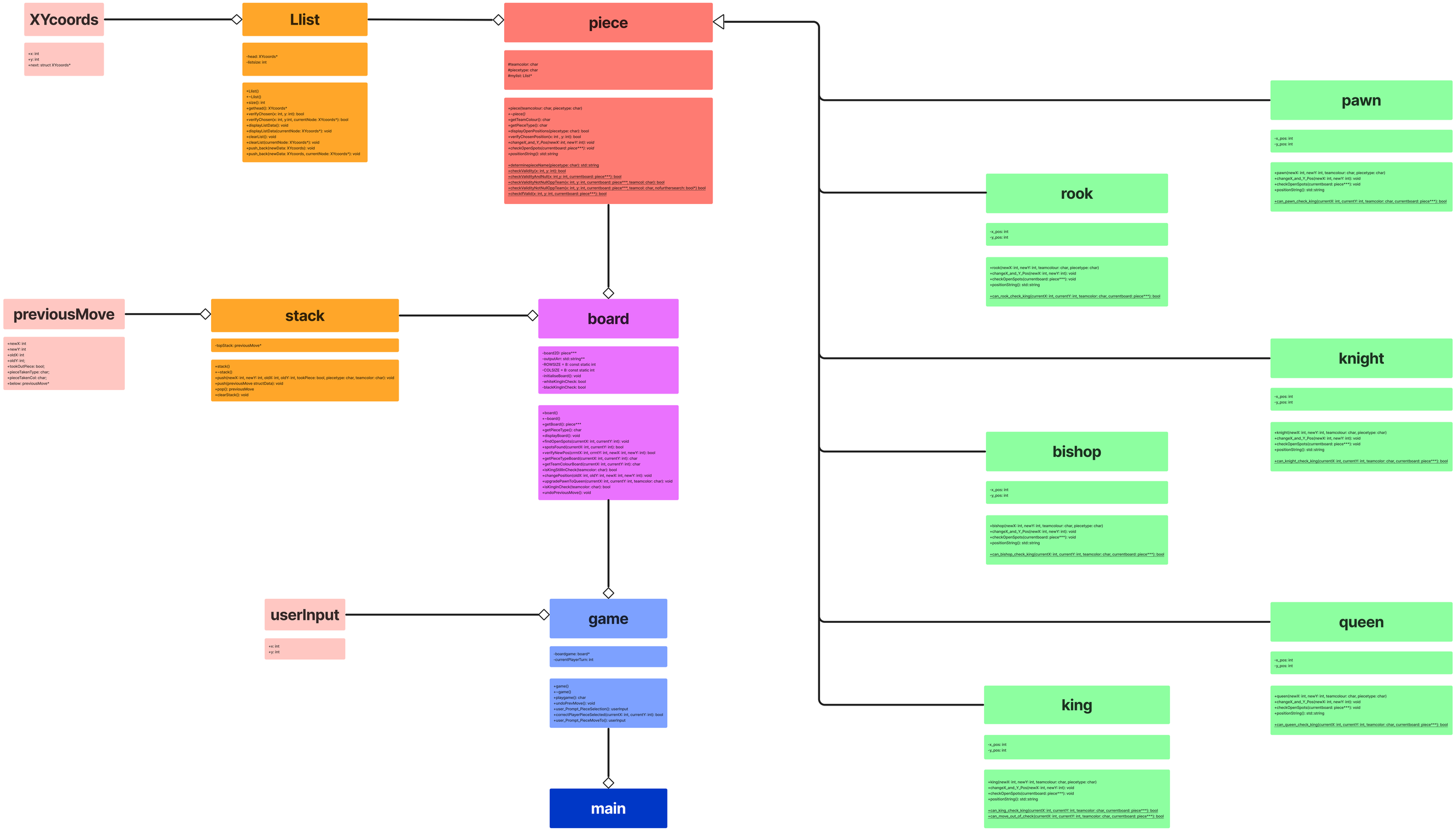
XYcoords
+x: int
+y: int
+next: struct XYcoords*
Llist
-head: XYcoords*
-listsize: int
+Llist()
+~Llist()
+size(): int
+gethead(): XYcoords*
+verifyChosen(x: int, y: int): bool
+verifyChosen(x: int, y:int, currentNode: XYcoords*): bool
+displayListData(): void
+displayListData(currentNode: XYcoords*): void
+clearList(): void
+clearList(currentNode: XYcoords*): void
+push_back(newData: XYcoords): void
+push_back(newData: XYcoords, currentNode: XYcoords*): void
piece
#teamcolor: char
#piecetype: char
#mylist: Llist*
+piece(teamcolour: char, piecetype: char)
+~piece()
+getTeamColour(): char
+getPieceType(): char
+displayOpenPositions(piecetype: char): bool
+verifyChosenPosition(x: int , y: int): bool
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+determinepieceName(piecetype: char): std::string
+checkValidity(x: int, y: int): bool
+checkValidityAndNull(x: int,y: int, currentboard: piece***): bool
+checkValidityNotNullOppTeam(x: int, y: int, currentboard: piece***, teamcol: char): bool
+checkValidityNotNullOppTeam(x: int, y: int, currentboard: piece***, teamcol: char, nofurthersearch: bool*) bool
+checkIfValid(x: int, y: int, currentboard: piece***): bool
pawn
-x_pos: int
-y_pos: int
+pawn(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_pawn_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
rook
-x_pos: int
-y_pos: int
+rook(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_rook_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
knight
-x_pos: int
-y_pos: int
+knight(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_knight_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
bishop
-x_pos: int
-y_pos: int
+bishop(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_bishop_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
queen
-x_pos: int
-y_pos: int
+queen(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_queen_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
king
-x_pos: int
-y_pos: int
+king(newX: int, newY: int, teamcolour: char, piecetype: char)
+changeX_and_Y_Pos(newX: int, newY: int): void
+checkOpenSpots(currentboard: piece***): void
+positionString(): std::string
+can_king_check_king(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
+can_move_out_of_check(currentX: int, currentY: int, teamcolor: char, currentboard: piece***): bool
previousMove
+newX: int
+newY: int
+oldX: int
+oldY: int;
+tookOutPiece: bool;
+pieceTakenType: char;
+pieceTakenCol: char;
+below: previousMove*
stack
-topStack: previousMove*
+stack()
+~stack()
+push(newX: int, newY: int, oldX: int, oldY: int, tookPiece: bool, piecetype: char, teamcolor: char): void
+push(previousMove structData): void
+pop(): previousMove
+clearStack(): void
board
-board2D: piece***
-outputArr: std::string**
-ROWSIZE = 8: const static int
-COLSIZE = 8: const static int
-initialiseBoard(): void
-whiteKinginCheck: bool
-blackKinginCheck: bool
+board()
+~board()
+getBoard(): piece***
+getPieceType(): char
+displayBoard(): void
+findOpenSpots(currentX: int, currentY: int): void
+spotsFound(currentX: int, currentY: int): bool
+verifyNewPos(crrntX: int, crrntY: int, newX: int, newY: int): bool
+getPieceTypeBoard(currentX: int, currentY: int): char
+getTeamColourBoard(currentX: int, currentY: int): char
+isKingStillinCheck(teamcolor: char): bool
+changePosition(oldX: int, oldY: int, newX: int, newY: int): void
+upgradePawnToQueen(currentX: int, currentY: int, teamcolor: char): void
+isKinginCheck(teamcolor: char): bool
+undoPreviousMove(): void
userInput
+x: int
+y: int
game
-boardgame: board*
-currentPlayerTurn: int
+game()
+~game()
+playgame(): char
+undoPrevMove(): void
+user_Prompt_PieceSelection(): userInput
+correctPlayerPieceSelected(currentX: int, currentY: int): bool
+user_Prompt_PieceMoveTo(): userInput
main