**JAVA CODE for a basic Chess Game by Sai Charan Reddy   
Mutyala**

public class Game {

private Board board = new Board();

private Player white;

private Player black;

public Game() {

super();

}

public void setColorWhite(Player player) {

this.white = player;

}

public void setColorBlack(Player player) {

this.black = player;

}

public Board getBoard() {

return board;

}

public void setBoard(Board board) {

this.board = board;

}

public Player getWhite() {

return white;

}

public void setWhite(Player white) {

this.white = white;

}

public Player getBlack() {

return black;

}

public void setBlack(Player black) {

this.black = black;

}

public boolean initializeBoardGivenPlayers() {

if(this.black == null || this.white == null)

return false;

this.board = new Board();

for(int i=0; i<black.getPieces().size(); i++){

board.getSpot(black.getPieces().get(i).getX(), black.getPieces().get(i).getY()).occupySpot(black.getPieces().get(i));

}

return true;

}

}

Player.java

public class Player {

public final int PAWNS = 8;

public final int BISHOPS = 2;

public final int ROOKS = 2;

public boolean white;

private List<Piece> pieces = new ArrayList<>();

public Player(boolean white) {

super();

this.white = white;

}

public List<Piece> getPieces() {

return pieces;

}

public void initializePieces(){

if(this.white == true){

for(int i=0; i<PAWNS; i++){ // draw pawns

pieces.add(new Pawn(true,i,2));

}

pieces.add(new Rook(true, 0, 0));

pieces.add(new Rook(true, 7, 0));

pieces.add(new Bishop(true, 2, 0));

pieces.add(new Bishop(true, 5, 0));

pieces.add(new Knight(true, 1, 0));

pieces.add(new Knight(true, 6, 0));

pieces.add(new Queen(true, 3, 0));

pieces.add(new King(true, 4, 0));

}

else{

for(int i=0; i<PAWNS; i++){ // draw pawns

pieces.add(new Pawn(true,i,6));

}

pieces.add(new Rook(true, 0, 7));

pieces.add(new Rook(true, 7, 7));

pieces.add(new Bishop(true, 2, 7));

pieces.add(new Bishop(true, 5, 7));

pieces.add(new Knight(true, 1, 7));

pieces.add(new Knight(true, 6, 7));

pieces.add(new Queen(true, 3, 7));

pieces.add(new King(true, 4, 7));

}

}

}

Board.java

public class Board {

private Spot[][] spots = new Spot[8][8];

public Board() {

super();

for(int i=0; i<spots.length; i++){

for(int j=0; j<spots.length; j++){

this.spots[i][j] = new Spot(i, j);

}

}

}

public Spot getSpot(int x, int y) {

return spots[x][y];

}

}

Spot.java

public class Spot {

int x;

int y;

Piece piece;

public Spot(int x, int y) {

super();

this.x = x;

this.y = y;

piece = null;

}

public void occupySpot(Piece piece){

//if piece already here, delete it, i. e. set it dead

if(this.piece != null)

this.piece.setAvailable(false);

//place piece here

this.piece = piece;

}

public boolean isOccupied() {

if(piece != null)

return true;

return false;

}

public Piece releaseSpot() {

Piece releasedPiece = this.piece;

this.piece = null;

return releasedPiece;

}

}

Piece.java

public class Piece {

private boolean available;

private int x;

private int y;

public Piece(boolean available, int x, int y) {

super();

this.available = available;

this.x = x;

this.y = y;

}

public boolean isAvailable() {

return available;

}

public void setAvailable(boolean available) {

this.available = available;

}

public int getX() {

return x;

}

public void setX(int x) {

this.x = x;

}

public int getY() {

return y;

}

public void setY(int y) {

this.y = y;

}

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY){

if(toX == fromX && toY == fromY)

return false; //cannot move nothing

if(toX < 0 || toX > 7 || fromX < 0 || fromX > 7 || toY < 0 || toY > 7 || fromY <0 || fromY > 7)

return false;

return true;

}

}

King.java

public class King extends Piece{

public King(boolean available, int x, int y) {

super(available, x, y);

// TODO Auto-generated constructor stub

}

@Override

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY) {

if(super.isValid(board, fromX, fromY, toX, toY) == false)

return false;

if(Math.sqrt(Math.pow(Math.abs((toX - fromX)),2)) + Math.pow(Math.abs((toY - fromY)), 2) != Math.sqrt(2)){

return false;

}

return false;

}

}

Knight.java

public class Knight extends Piece{

public Knight(boolean available, int x, int y) {

super(available, x, y);

}

@Override

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY) {

if(super.isValid(board, fromX, fromY, toX, toY) == false)

return false;

if(toX != fromX - 1 && toX != fromX + 1 && toX != fromX + 2 && toX != fromX - 2)

return false;

if(toY != fromY - 2 && toY != fromY + 2 && toY != fromY - 1 && toY != fromY + 1)

return false;

return true;

}

}

Bishop.java

public class Bishop extends Piece{

public Bishop(boolean available, int x, int y) {

super(available, x, y);

// TODO Auto-generated constructor stub

}

@Override

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY) {

if(super.isValid(board, fromX, fromY, toX, toY) == false)

return false;

if(toX - fromX == toY - fromY)

return true;

return false;

}

}

Rook.java

public class Rook extends Piece{

public Rook(boolean available, int x, int y) {

super(available, x, y);

// TODO Auto-generated constructor stub

}

@Override

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY) {

if(super.isValid(board, fromX, fromY, toX, toY) == false)

return false;

if(toX == fromX)

return true;

if(toY == fromY)

return true;

return false;

}

}

Queen.java

public class Queen extends Piece{

public Queen(boolean available, int x, int y) {

super(available, x, y);

}

@Override

public boolean isValid(Board board, int fromX, int fromY, int toX, int toY) {

if(super.isValid(board, fromX, fromY, toX, toY) == false)

return false;

//diagonal

if(toX - fromX == toY - fromY)

return true;

if(toX == fromX)

return true;

if(toY == fromY)

return true;

return false;

}

}