Class gameboard

Private int score =0;

Private int hightScore = 0;

Private Font scoreFont;

Private String saveDataPath;

Private String fileName= "saveData";

private long elapsedMS;

private long fastestMS;

private long startTime;

private boolean hasStarted;

private String formattedTime = "00:00:000";

public GameBoard(int x, int y){

//add them try catch

try{

saveDataPath = GameBoard.class.getProtectionDomain().getCodeSource().getLocation().toURI().getPath();

}

catch(Exception e){

e.printStackTrace();

}

scoreFont = Game.main.deriveFont(24f);

this.x=x;

this.y = y;

board = new Tile[ROWS][COLS];

gameBoard = new BufferedImage(BOARD\_WIDTH, BOARD\_HEIGHT,

BufferedImage.TYPE\_INT\_RGB);

finalBoard =new BufferedImage(BOARD\_WIDTH, BOARD\_HEIGHT,

BufferedImage.TYPE\_INT\_RGB);

startTime=system.nanoTime();

loadHighScore();

createBoardImage();

start();

}

private void createSaveData(){

try{

File file new File(saveDataPath, fileName);

FileWriter output = new FileWriter(file);

BufferedWriter writer = new BufferedWriter(output);

writer.write(""+ 0);

writer.newLine();

writer.write(""+Integer.Max\_Value);

writer.close();}

Catch(Exception e){

e.printStackTrace();

}

}

private void loadHighScore(){

try{

File f = new File(saveDataPath, fileName);

if(If.isFile()){

createSaveData();

}

BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(f)));

highScore = Integer.parseInt(reader.readLine());

fastestMS = Long.parseLong(reader.readLine());

reader.close();

}

catch (Exception e){

e.printStackTrace();

}

private void setHighScore(){

FileWriter output = null;

try{

File f = new File(saveDataPath, fileName);

output = new FileWriter (f);

BufferedWriter writer = new BufferedWriter(output);

writer.write("" + highScore);

writer.newLine();

if(elapsedMS <- fastestMS && won){

writer.write("" + elapsedMS);

}

else

writer.write(+ fastestMS);

}

writer.close();

catch(Exception e){

e.printStackTrace();

}

//add if vao update

If(score>= highScore){

highScore=score;}

//add vao render

g.setColor(Color.lightGray);

g.setFont(scoreFont);

g.drawString(" + score, 30, 40);

g.setColor(Color.red);

g.drawString("Best: " + highScore, Game. WIDTH - DrawUtils.getMessageWidth("Best: " + highScore, scoreFont, g) - 20, 40);

g.setColor(Color.black);

g.drawString("Time: g.setColor(Color.red);

formatted Time, 30, 90);

g.drawString("Fastest: " +

formattedTime(fastestMS), Game. WIDTH - DrawUtils.getMessageWidth("Fastest: " + formattedTime(fastestMS), scoreFont, g)-20,90);

//add vao move

Score+= board[newRom][newCol].getValue();

//add vao checkDead

setHighScore();

private String formatTime(long millis){

String formattedTime;

String hourFormat = "";

int hours = (int)(millis / 3600000);

if(hours >= 1){

millis -=hours 3600000;

if (hours < 10){

hourFormat = "0" + hours;

}

else{

hourFormat = "" + hours;

}

}

hourFormat+=":";

String minuteFormat;

int minutes = (int)(millis / 60000);

if(minutes >=1){

millis -= minutes \* 60000;

if(minutes < 10){

minuteFormat = "0" + minutes;

}

else{

minuteFormat = "" + minutes;

}

}

else{

minuteFormat = "00";

}

String secondFormat;

int seconds = (int)(millis / 1000);

if(seconds >=1){

millis -= seconds \* 1000;

if(seconds < 10){

secondFormat = "0" + seconds;

}

else{

secondFormat = "" + seconds;

}

}

else{

secondFormat = "00";

}

String milliFormat;

if(millis > 99){

milliFormat = "" + millis;}

else if(millis > 9){

milliFormat = "0" + millis;

}

else{

milliFormat = "00" + millis;

}

formattedTime =hourFormat + minuteFormat+ ":" + secondFormat+ ":" miliFormat;

return formattedTime;

}

public void update(){

//add

if (!won && !dead) {

            if (hasStarted) {

                elapsedMS = (System.nanoTime() - startTime) / 1000000;

                formattedTime = formatTime(elapsedMS);

            }

            else {

                startTime = System.nanoTime();

            }