Source Code

1. AI.java

package main;

import java.util.Arrays;

import java.util.Random;

import java.util.concurrent.TimeUnit;

public class AI {

private final TicTacToe game;

public AI(TicTacToe game) {

this.game = game;

}

private void randomMove() {

Tile[] possible = new Tile[0];

Tile[] tiles = game.getTiles();

Random random = new Random();

for (Tile t : tiles) {

if (!t.isClaimed()) {

possible = Arrays.copyOfRange(possible, 0, possible.length + 1);

possible[possible.length - 1] = t;

}

}

if (possible.length > 0) {

int toClaim = random.nextInt(possible.length);

game.attemptClaim(possible[toClaim].getX(),

possible[toClaim].getY());

}

}

public void aiMove() {

Random random = new Random();

int[][] wins = game.getWins();

Tile[] tiles = game.getTiles();

try {

TimeUnit.SECONDS.sleep(random.nextInt(3));

} catch (InterruptedException e) {

e.printStackTrace();

}

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

if (game.toWin(game.getTurn(), wins[i], tiles) == 1) {

Tile winningTile = game.getWin(game.getTurn(), wins[i], tiles);

game.attemptClaim(winningTile.getX() + 1,

winningTile.getY() + 1);

System.out.println("Claiming win...");

return;

}

if (game.toWin(game.getTurn().getOpposite(), wins[i], tiles) == 1) {

Tile defendingTile = game.getWin(game.getTurn().getOpposite(),

wins[i], tiles);

game.attemptClaim(defendingTile.getX() + 1,

defendingTile.getY() + 1);

System.out.println("Defending...");

return;

}

}

randomMove();

}

}

1. ClickHandler.java

package main;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

public class ClickHandler implements MouseListener {

private final TicTacToe game;

public ClickHandler(TicTacToe game) {

this.game = game;

}

public void mouseReleased(MouseEvent e) {

if (!game.isAiTurn()) {

game.attemptClaim(e.getX() - 3, e.getY() - 26);

game.getGameframe().repaint();

}

}

@Override

public void mouseClicked(MouseEvent e) {

}

@Override

public void mousePressed(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}

1. Holder.java

package main;

import java.awt.Color;

public enum Holder {

X, O, GAME, ANY;

public String getText() {

switch (this) {

case X:

return "X";

case O:

return "O";

default:

return " ";

}

}

public Holder getOpposite() {

switch (this) {

case X:

return O;

case O:

return X;

default:

return GAME;

}

}

public Color getColor() {

switch (this) {

case X:

return Color.LIGHT\_GRAY;

case O:

return Color.DARK\_GRAY;

default:

return Color.WHITE;

}

}

public Color getTextColor() {

switch (this) {

case X:

return Color.BLACK;

case O:

return Color.WHITE;

default:

return Color.MAGENTA;

}

}

}

1. TicTacToe.java

package main;

import java.awt.Dimension;

import java.util.concurrent.TimeUnit;

import javax.swing.JApplet;

import javax.swing.JFrame;

import javax.swing.JLabel;

public final class TicTacToe extends JApplet {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private final Tile[] TILES = new Tile[9];

private final int TILE\_SPACING = 96;

private final int WIDTH = 96, HEIGHT = 96;

private final JFrame GAMEFRAME = new JFrame("Tic-Tac-Toe");

private final TilePainter PAINTER = new TilePainter(this);

private final ClickHandler CLICK\_HANDLER = new ClickHandler(this);

private final boolean AI;

private boolean aiTurn = false;

private Holder turn = Holder.X;

private int whoseTurn = 0;

private final Dimension FRAME\_SIZE = new Dimension(295, 304);

private final int FONT\_SIZE = 64;

private int oWins = 0;

private int xWins = 0;

private boolean gameOver = false;

private boolean nextTurn = false;

public final AI GAME\_AI = new AI(this);

public void init() {

TicTacToe game = new TicTacToe(true);

game.newGame();

}

private final int[][] wins = { { 1, 1, 1, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 1, 1, 1, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 1, 1, 1 },

{ 1, 0, 0, 0, 1, 0, 0, 0, 1 },

{ 1, 0, 0, 1, 0, 0, 1, 0, 0 },

{ 0, 1, 0, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 1, 0, 0, 1, 0, 0, 1 },

{ 0, 0, 1, 0, 1, 0, 1, 0, 0 } };

public boolean allFull() {

for (Tile t : TILES) {

if (!t.isClaimed()) {

return false;

}

}

return true;

}

public boolean hasWon(Holder h) {

boolean hasWon;

for (int[] i : wins) {

hasWon = true;

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

if (i[j] == 1) {

if (TILES[j].getHolder() != h) {

hasWon = false;

j = i.length;

}

}

}

if (hasWon)

return true;

}

return false;

}

public int toWin(Holder h, int[] win, Tile[] tiles) {

int total = 0;

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

if (win[j] == 1) {

total++;

if (tiles[j].getHolder() == h) {

total--;

} else if (tiles[j].getHolder() != Holder.GAME) {

return -1;

}

}

}

return total;

}

public Tile getWin(Holder h, int[] win, Tile[] tiles) {

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

if (win[j] == 1) {

if (tiles[j].getHolder() == Holder.GAME) {

return TILES[j];

}

}

}

return null;

}

public void endGame() {

gameOver = true;

}

public int getFontSize() {

return FONT\_SIZE;

}

public TicTacToe(boolean ai) {

this.AI = ai;

PAINTER.setSize(FRAME\_SIZE);

buildFrame();

loadTiles();

}

public void loadTiles() {

int tile = 0;

for (int i = 0; i < TILES.length / 3; i++) {

for (int j = 0; j < TILES.length / 3; j++) {

TILES[tile] = new Tile(i \* this.TILE\_SPACING, j

\* this.TILE\_SPACING, this.WIDTH, this.HEIGHT, this);

tile++;

}

}

}

private void nextTurn() {

if (hasWon(turn)) {

gameOver = true;

sendWin(turn);

return;

}

if (allFull()) {

gameOver = true;

sendDraw();

return;

}

turn = turn.getOpposite();

if (aiTurn && AI) {

aiTurn = false;

} else if (!aiTurn && AI) {

aiTurn = true;

GAME\_AI.aiMove();

} else {

aiTurn = false;

}

}

public void attemptClaim(int x, int y) {

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

if (!TILES[i].isClaimed() && TILES[i].inArea(x, y)) {

TILES[i].claim(turn);

nextTurn = true;

return;

}

}

}

public boolean isAiTurn() {

return aiTurn;

}

private void buildFrame() {

getGameframe().addMouseListener(CLICK\_HANDLER);

getGameframe().setSize(FRAME\_SIZE);

getGameframe().setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getGameframe().setResizable(false);

getGameframe().setMaximumSize(FRAME\_SIZE);

getGameframe().setMinimumSize(FRAME\_SIZE);

getGameframe().add(PAINTER);

getGameframe().pack();

}

private JFrame outcome = new JFrame();

private void sendWin(Holder winner) {

outcome.setVisible(false);

outcome.dispose();

outcome = null;

if (winner == Holder.X)

xWins++;

else if (winner == Holder.O)

oWins++;

outcome = new JFrame(winner.getText() + " has won!");

JLabel winMessage = new JLabel(" " + winner.getText()

+ " has won! Score is X: " + xWins + ", O: " + oWins);

outcome.add(winMessage);

outcome.setResizable(false);

outcome.setAlwaysOnTop(true);

outcome.pack();

outcome.setVisible(true);

}

private void sendDraw() {

outcome.setVisible(false);

outcome.dispose();

outcome = null;

outcome = new JFrame("Draw!");

JLabel drawMessage = new JLabel(" Its a Draw! Score is X: " + xWins

+ ", O: " + oWins);

outcome.add(drawMessage);

outcome.setResizable(false);

outcome.setAlwaysOnTop(true);

outcome.pack();

outcome.setVisible(true);

}

private void resetTiles() {

for (Tile t : TILES)

t.reset();

}

public void newGame() {

while (true) {

gameOver = false;

resetTiles();

GAMEFRAME.setVisible(true);

if (whoseTurn == 1 && this.AI) {

whoseTurn = 0;

turn = Holder.O;

aiTurn = true;

} else if (this.AI) {

turn = Holder.X;

aiTurn = false;

whoseTurn = 1;

} else {

turn = Holder.X;

}

if (aiTurn)

GAME\_AI.aiMove();

while (!gameOver) {

if (nextTurn) {

nextTurn = false;

nextTurn();

}

try {

TimeUnit.MILLISECONDS.sleep(25);

} catch (InterruptedException e1) {

e1.printStackTrace();

}

GAMEFRAME.repaint();

}

try {

TimeUnit.SECONDS.sleep(5);

} catch (InterruptedException e) {

e.printStackTrace();

}

getGameframe().setVisible(false);

}

}

public int[][] getWins() {

return wins;

}

public Holder getTurn() {

return turn;

}

public Tile[] getTiles() {

return TILES;

}

public static void main(String[] args) {

TicTacToe game;

if (args.length >= 1) {

game = new TicTacToe(false);

game.newGame();

} else {

game = new TicTacToe(true);

game.newGame();

}

}

public JFrame getGameframe() {

return GAMEFRAME;

}

}

1. Tile.java

package main;

import java.awt.Graphics;

public final class Tile {

private final int x, y, width, height;

private boolean claimed = false;

private Holder heldBy = Holder.GAME;

private final TicTacToe game;

public Tile(int x, int y, int width, int height, TicTacToe game) {

this.x = x;

this.y = y;

this.width = width;

this.height = height;

this.game = game;

}

public void claim(Holder h) {

if (heldBy == Holder.GAME && !claimed) {

claimed = true;

heldBy = h;

}

}

public Holder getHolder() {

return heldBy;

}

public void paint(Graphics g) {

g.setColor(heldBy.getColor());

g.fillRect(x, y, width, height);

g.setColor(heldBy.getTextColor());

g.drawString(heldBy.getText(), middleX() - (game.getFontSize() / 3),

middleY() + (game.getFontSize() / 3));

}

private int middleX() {

return (x + width / 2);

}

private int middleY() {

return (y + height / 2);

}

public void reset() {

this.heldBy = Holder.GAME;

this.claimed = false;

}

public boolean inArea(int x, int y) {

return (this.x <= x && this.x + this.width >= x)

&& (this.y <= y && this.y + this.height >= y);

}

public boolean isClaimed() {

return claimed;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public int getWidth() {

return width;

}

public int getHeight() {

return height;

}

}

1. TilePainter.java

package main;

import java.awt.Color;

import java.awt.Component;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

public final class TilePainter extends Component {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private final TicTacToe game;

public TilePainter(TicTacToe game) {

this.game = game;

}

public void paint(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.setFont(new Font("Dialogue", Font.BOLD, game.getFontSize()));

for (Tile t : game.getTiles()) {

g.setColor(Color.BLACK);

g.drawRect(t.getX() - 1, t.getY() - 1, t.getWidth() + 1,

t.getHeight() + 1);

t.paint(g2d);

}

}

}