package battleship;

import java.util.ArrayList;

public class AI extends Side {

private int difficulty=0; //0 = easy, 1 = medium, 2 = hard

private int pX, pY, fX, fY;

private int[] fire = new int[] {1,0};

private boolean found=false;

private boolean target=false;

private ArrayList<int[]> misses = new ArrayList<int[]>();

private Player p;

public AI(Player pl) {

board=new int[8][8];

oBoard = new int[8][8];

difficulty=0;

p=pl;

generateShips();

} public void setDifficulty(int d) {

difficulty=d;

} private void fireAtRandom(int strategy) {

int x=-1, y=-1;

do {

x = (int)(Math.random()\*8);

y = (int)(Math.random()\*8);

} while (!(oBoard[x][y]==0 && (strategy==0 || strategy>0 && x%2!=y%2) && (strategy<2 || strategy==2 && shipFit(x+fire[0],y+fire[1]))));

pX=x;

pY=y;

update(x,y);

} public int[][] getOBoard() {

return oBoard;

} public int[] fire() {

if (difficulty==0) {

if (!target) {

fireAtRandom(0);

} else {

followFire(false);

}

} else if (difficulty==1) {

if (!target) {

fireAtRandom(1);

} else {

followFire(false);

}

} else {

if (!target) {

fireAtRandom(2);

} else {

followFire(true);

}

}

// System.out.println(fX+", "+fY);

return new int[] {fX,fY};

}

private void followFire(boolean strategy) {

int y, f=0;

boolean cont=false;

while (true) {

while (pX+fire[0]>7 || pY+fire[1]>7 || pX+fire[0]<0 || pY+fire[1]<0 || oBoard[pX+fire[0]][pY+fire[1]]!=0) {

System.out.println("back here!");

cont=false;

if (!found) {

if (fire[0]==1) {

fire[0]--;

fire[1]--;

} else if (fire[1]==-1) {

fire[0]--;

fire[1]++;

} else if (fire[0]==-1) {

fire[0]++;

fire[1]++;

} else {

fire[0]++;

fire[1]--;

}

} else {

fire[0]\*=-1;

fire[1]\*=-1;

f=0;

while (true) {

f++;

if (pX+f\*fire[0]>7 || pY+f\*fire[1]>7 || pX+f\*fire[0]<0 || pY+f\*fire[1]<0 || oBoard[pX+f\*fire[0]][pY+f\*fire[1]]==1) {

System.out.println("HERE!");

found=false;

pX=misses.get(0)[0];

pY=misses.get(0)[1];

cont=true;

break;

} if (oBoard[pX+f\*fire[0]][pY+f\*fire[1]]==0) {

break;

}

}

}

}

if (cont) {

continue;

}

while (oBoard[pX+fire[0]][pY+fire[1]]!=0) {

if (pX+fire[0]>7 || pX+fire[0]<0 || pY+fire[1]>7 || pY+fire[1]<0) {

fire[0]\*=-1;

fire[1]\*=-1;

}

pX+=fire[0];

pY+=fire[1];

}

y=update(pX+fire[0],pY+fire[1]);

if (y>0) {

if (y!=2) {

pX+=fire[0];

pY+=fire[1];

}

if (y==1) {

found=true;

}

break;

}

break;

}

}

private int update(int x, int y) {

fX=x;

fY=y;

p.decrement(x,y);

int z = p.hitOrMiss(x, y);

System.out.println(z);

lookForMissed();

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

System.out.print("("+misses.get(i)[0]+", "+misses.get(i)[1]+") ");

}

if (misses.size()==0) {

target=false;

}

if (z>0) {

oBoard[x][y]=2;

target=true;

if (z>1) {

oBoard[x][y]=z;

for (int i=0; i<z-9; i++) {

oBoard[x-fire[0]\*i][y-fire[1]\*i]=z;

lookForMissed();

}

if (misses.size()==0) {

target=false;

} else {

target=true;

pX=misses.get(0)[0];

pY=misses.get(0)[1];

}

found=false;

oBoats[z-11]=true;

return 2; //SUNK

}

return 1; //HIT

} else {

oBoard[x][y]=1;

return 0; //RETURN FALSE

}

}

public boolean printOBoats() {

int ct=0;

for (int i=0; i<4; i++) {

if (oBoats[i])

ct++;

}if (ct==4) {

return true;

}return false;

}

private boolean shipFit(int x, int y) {

int size=0;

boolean z=true;

for (int i=0; i<4; i++) { //Go through the boats and find the smallest boat still unsunk, set its size to size

if (!oBoats[i]) {

size=i+2;

}

}

for (int i=x-size; i<=x; i++) {

z=true;

for (int j=i; j<size; j++) {

if (x<0 || x>7 || j<0 || j>7 || !(oBoard[x][j]==0 || oBoard[x][j]==2)) {

z=false;

}

}if (z) {

return true;

}

}

for (int i=y-size; i<=y; i++) {

z=true;

for (int j=i; j<size; j++) {

if (y<0 || y>7 || j<0 || j>7 || !(oBoard[j][y]==0 || oBoard[j][y]==2)) {

z=false;

}

}if (z) {

return true;

}

}

return false;

} private void lookForMissed() {

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

if (oBoard[misses.get(i)[0]][misses.get(i)[1]]!=2) {

misses.remove(i);

i--;

}

}

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

for (int j=0; j<oBoard[0].length; j++) {

if (oBoard[i][j]==2 && !contains(i,j)) {

misses.add(new int[] {i,j});

}

}

}

} private boolean contains(int x, int y) {

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

if (misses.get(i)[0]==x && misses.get(i)[1]==y) {

return true;

}

}return false;

}

public void generateShips() {

int x, y, ortn;

boolean valid=false;

for (int i=5; i>=2; i--) {

valid=false;

while (!valid) {

valid=true;

x = (int)(8 \* Math.random());

y = (int)(8 \* Math.random());

ortn = (int)(4 \* Math.random());

if (ortn==0) { //Right

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

if (x+j>7 || board[y][x+j]!=0) {

valid=false;

}

} if (valid) {

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

board[y][x+j]=i-1;

}

}

} else if (ortn==1) { //Down

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

if (y+j>7 || board[y+j][x]!=0) {

valid=false;

}

} if (valid) {

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

board[y+j][x]=i-1;

}

}

} else if (ortn==2) { //Left

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

if (x-j<0 || board[y][x-j]!=0) {

valid=false;

}

} if (valid) {

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

board[y][x-j]=i-1;

}

}

} else { //Up

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

if (y-j<0 || board[y-j][x]!=0) {

valid=false;

}

} if (valid) {

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

board[y-j][x]=i-1;

}

}

}

}

}

}

}