**DEPARTMENT OF COMPUTER TECHNOLOGY**

**MADRAS INSTITUTE OF TECHNOLOGY**

**CS6308 - JAVA PROGRAMMING**

**MINI PROJECT REPORT**

**LEARN WITH PAC-MAN GAME**



**MENTOR: Dr. V. P. Jayachitra**

BY

Kartikeyan TR - 2020503520

Suriyaa V - 2020503550

Guru Raman C - 2020503510

**ABSTRACT**

"Learn with Pac-Man" is an action maze chase video game; the player controls the eponymous Pac-Man character through an enclosed maze. The objective of the game is to eat all of the HEALTHY FRUITS placed in the maze while avoiding JUNK FOODS — Cola, Pizza, Burger — that pursue Pac-Man and also enable people(mostly children) to realize that eating junk food is harmful.

**INTRODUCTION**

When Pac-Man eats all of the fruits, the player wins the level. Levels can be chosen from the Menu screen. Each level provides 3 lives for Pac-Man. If Pac-Man is caught by a Junk Food, he will lose a life; the game ends when all the three lives are lost. Eating healthy food increases the score of the person.

**PROPOSED WORK**

* Designing the game maze where the pacman moves around eating healthy fruits.
* Creating Junk Foods which appear as ghosts that chases Pacman all throughout the game.
* Designing an algorithm for the pacman to navigate around walls and also for the junk ghosts.
* Creating a database table to store the scores.
* Updation of each game score in the database along with the time and date of game played.
* Presenting an option to choose levels with an initial build of providing availability of 3 levels with 3 different mazes, food and junk ghosts.
* Creation of an about page which has the details of the creators.
* Creating an Exit Button.
* Making children understand that junk food is bad for health and fruits are good for health

**IMPLEMENTATION**

**CODE:**

**Intro.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.sql.SQLException;

import javax.imageio.ImageIO;

public class Intro {

public static void main(String[] args) throws IOException, FontFormatException {

ShowIntro w = new ShowIntro();

}

}

class ShowIntro extends JFrame{

Font Lot;

static Font CrackMan;

Font video;

public ShowIntro() throws IOException, FontFormatException {

//Pacman Font 1

Lot = Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf"));

GraphicsEnvironment g = GraphicsEnvironment.getLocalGraphicsEnvironment();

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf")));

//Pacman Font2

CrackMan = Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF")));

video = Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF")));

//Title

JLabel q = new JLabel("Learn with PAC-MAN");

q.setFont(new Font("CrackMan", Font.PLAIN, 35));

q.setBackground(Color.black);

q.setForeground(Color.yellow);

q.setBounds(410 , 20 , 500 , 100);

//LeaderBoard

JButton leaderboard = new JButton("LeaderBoard");

leaderboard.setFont(new Font("Lot", Font.PLAIN, 20));

leaderboard.setBackground(Color.black);

leaderboard.setForeground(Color.yellow);

leaderboard.setBounds(470 , 360 , 300 , 90);

//New Game

JButton newgame = new JButton("New Game");

leaderboard.setFont(new Font("Lot", Font.PLAIN, 20));

newgame.setFont(new Font("Lot" , Font.PLAIN, 20));

newgame.setBackground(Color.black);

newgame.setForeground(Color.yellow);

newgame.setBounds(470 , 250 , 300 , 90);

//Exit

JButton ext = new JButton("Exit");

ext.setFont(new Font("Lot" , Font.PLAIN, 20));

ext.setBackground(Color.black);

ext.setForeground(Color.yellow);

ext.setBounds(470 , 470 , 300 , 50);

//About

JButton about = new JButton("About");

about.setFont(new Font("Lot" , Font.PLAIN, 15));

about.setBackground(Color.black);

about.setForeground(Color.yellow);

about.setBounds(1000 , 500 , 200 , 50);

BufferedImage myPicture = ImageIO.read(new File("images/maze.jpg"));

JLabel picLabel = new JLabel(new ImageIcon(myPicture));

picLabel.setBounds(5, 220 , 300 , 300);

BufferedImage burger = ImageIO.read(new File("images/burg.png"));

JLabel junkfood = new JLabel(new ImageIcon(burger));

junkfood.setBounds(895 , 250 , 300 , 300);

BufferedImage fruits = ImageIO.read(new File("images/healthy.jpg"));

JLabel healthy = new JLabel(new ImageIcon(fruits));

healthy.setBounds(450, 20 , 300 , 300);

BufferedImage chery = ImageIO.read(new File("images/cherry.png"));

JLabel cherry = new JLabel(new ImageIcon(chery));

cherry.setBounds(50 , 100 , 200 , 200);

BufferedImage appl = ImageIO.read(new File("images/apple.png"));

JLabel apple = new JLabel(new ImageIcon(appl));

apple.setBounds(930 , 100 , 200 , 200);

setVisible(true);

setLayout(null);

add(picLabel);

add(junkfood);

add(healthy);

add(cherry);

add(apple);

add(q);

add(leaderboard);

add(newgame);

add(ext);

add(about);

setSize(1280,600);

getContentPane().setBackground(Color.black);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

leaderboard.addActionListener(e -> {

Leaderboard t = null;

try {

t = new Leaderboard();

} catch (Exception r) {

}

});

newgame.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

EventQueue.invokeLater(() -> {

try {

new ShowLevels();

} catch (IOException ex) {

throw new RuntimeException(ex);

} catch (FontFormatException ex) {

throw new RuntimeException(ex);

}

});

dispose();

}

});

about.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

EventQueue.invokeLater(() -> {

try {

new about();

} catch (IOException ex) {

throw new RuntimeException(ex);

} catch (FontFormatException ex) {

throw new RuntimeException(ex);

}

});

dispose();

}

});

ext.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

}

}

**Pacman.java**

import java.awt.\*;

import javax.swing.JFrame;

public class Pacman extends JFrame {

public Pacman(int level) {

initUI(level);

}

private void initUI(int level) {

if(level == 1) {

add(new Board());

}

if(level == 2){

add(new Board2());

}

if(level == 3){

add(new Board3());

}

setTitle("Pacman");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize( 760, 435);

setResizable(false);

setLocationRelativeTo(null);

}

}

**Board.java**

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.sql.SQLException;

import javax.swing.\*;

public class Board extends JPanel implements ActionListener {

private Dimension d;

private final Font smallFont = new Font("Arial", Font.BOLD, 14);

private final Font factfont = new Font("Arial", Font.BOLD, 18);

private final Font qfont = new Font("Arial", Font.BOLD, 25);

private boolean inGame = false;

private boolean dying = false;

private final int BLOCK\_SIZE = 24;

private final int N\_BLOCKS = 15;

private final int SCREEN\_SIZE = N\_BLOCKS \* BLOCK\_SIZE;

private final int MAX\_GHOSTS = 12;

private final int PACMAN\_SPEED = 6;

private int N\_GHOSTS = 12;

private int lives, score;

private int[] dx, dy;

private int[] ghost\_x, ghost\_y, ghost\_dx, ghost\_dy, ghostSpeed;

private Image heart, pizza, grapes;

private Image up, down, left, right;

private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;

private int req\_dx, req\_dy;

//MAZE1

private final short levelData[] = {

19, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 18, 22,

17, 16, 16, 16, 16, 24, 16, 16, 16, 16, 16, 16, 16, 16, 20,

25, 24, 24, 24, 28, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20,

0, 0, 0, 0, 0, 0, 17, 16, 16, 16, 16, 16, 16, 16, 20,

19, 18, 18, 18, 18, 18, 16, 16, 16, 16, 24, 24, 24, 24, 20,

17, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,

17, 16, 16, 16, 16, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,

17, 16, 16, 16, 24, 16, 16, 16, 16, 20, 0, 0, 0, 0, 21,

17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 18, 18, 18, 18, 20,

17, 24, 24, 28, 0, 25, 24, 24, 16, 16, 16, 16, 16, 16, 20,

21, 0, 0, 0, 0, 0, 0, 0, 17, 16, 16, 16, 16, 16, 20,

17, 18, 18, 22, 0, 19, 18, 18, 16, 16, 16, 16, 16, 16, 20,

17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 20,

17, 16, 16, 20, 0, 17, 16, 16, 16, 16, 16, 16, 16, 16, 20,

25, 24, 24, 24, 26, 24, 24, 24, 24, 24, 24, 24, 24, 24, 28

};

JButton p = new JButton("LeaderBoard");

private final int validSpeeds[] = {1, 2, 3, 4, 6, 8};

private final int maxSpeed = 6;

private int currentSpeed = 3;

private short[] screenData;

private Timer timer;

public Board() {

loadImages();

initVariables();

addKeyListener(new TAdapter());

setFocusable(true);

initGame();

}

private void loadImages() {

down = new ImageIcon("images/down.gif").getImage();

up = new ImageIcon("images/up.gif").getImage();

left = new ImageIcon("images/left.gif").getImage();

right = new ImageIcon("images/right.gif").getImage();

heart = new ImageIcon("images/heart.png").getImage();

pizza = new ImageIcon("images/pizza.png").getImage();

grapes = new ImageIcon("images/grapes.png").getImage();

}

private void initVariables() {

screenData = new short[N\_BLOCKS \* N\_BLOCKS];

d = new Dimension(760, 400);

ghost\_x = new int[MAX\_GHOSTS];

ghost\_dx = new int[MAX\_GHOSTS];

ghost\_y = new int[MAX\_GHOSTS];

ghost\_dy = new int[MAX\_GHOSTS];

ghostSpeed = new int[MAX\_GHOSTS];

dx = new int[4];

dy = new int[4];

timer = new Timer(40, this);

timer.start();

}

private void playGame(Graphics2D g2d) throws SQLException, ClassNotFoundException {

if (dying) {

death();

} else {

movePacman();

drawPacman(g2d);

moveGhosts(g2d);

checkMaze();

}

}

private void showIntroScreen(Graphics2D g2d) {

String start = "Press SPACE to start";

g2d.setColor(Color.yellow);

g2d.drawString(start, (SCREEN\_SIZE)/4, 150);

}

private void drawScore(Graphics2D g) {

g.setFont(smallFont);

g.setColor(new Color(5, 181, 79));

String s = "Score: " + score;

g.drawString(s, SCREEN\_SIZE / 2 + 96, SCREEN\_SIZE + 16);

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

g.drawImage(heart, i \* 28 + 8, SCREEN\_SIZE + 1, this);

}

}

private void drawFact(Graphics2D g) {

g.setFont(factfont);

g.setColor(new Color(5, 181, 79));

String s = "HOW PACMAN CAN STAY HEALTHY";

g.drawString(s, SCREEN\_SIZE / 2 + 220, 30);

String s1 = "if he eats healthy fruits";

String s12 = "his life won't decrease";

g.drawString(s1, SCREEN\_SIZE / 2 + 270, 75);

g.drawString(s12 , SCREEN\_SIZE / 2 + 272 , 95);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 270 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 290 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 310 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 330 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 350 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 370 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 390 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 410 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 430 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 450 , 115 , this);

g.drawImage(grapes, SCREEN\_SIZE / 2 + 470 , 115 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 270 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 290 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 310 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 330 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 350 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 370 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 390 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 410 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 430 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 450 , 140 , this);

g.drawImage(grapes , SCREEN\_SIZE / 2 + 470 , 140 , this);

g.setColor(Color.red);

String s2 = "If he eats junk food";

String s3 = "his life decreases";

g.drawString(s2, SCREEN\_SIZE / 2 + 285, 200);

g.drawString(s3 , SCREEN\_SIZE / 2 + 290 , 220);

g.drawImage(pizza, SCREEN\_SIZE/2 + 265 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 295 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 325 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 355 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 385 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 415 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 445 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 475 , 240 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 265 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 295 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 325 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 355 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 385 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 415 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 445 , 270 , this);

g.drawImage(pizza, SCREEN\_SIZE/2 + 475 , 270 , this);

String s5 = "(It is true in real life)";

g.setColor(Color.green);

g.setFont(qfont);

g.drawString(s5 , SCREEN\_SIZE/2 + 265 , 330);

String s6 = "Press L for LeaderBoard";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s6 , SCREEN\_SIZE/2 + 270 , 360);

String s7 = "Press G for Main Menu";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s7 , SCREEN\_SIZE/2 + 270 , 380);

}

private void checkMaze() {

int i = 0;

boolean finished = true;

while (i < N\_BLOCKS \* N\_BLOCKS && finished) {

if ((screenData[i]) != 0) {

finished = false;

}

i++;

}

if (finished) {

score += 50;

if (N\_GHOSTS < MAX\_GHOSTS) {

N\_GHOSTS++;

}

if (currentSpeed < maxSpeed) {

currentSpeed++;

}

initLevel();

}

}

private void death() throws SQLException, ClassNotFoundException {

lives--;

if (lives == 0) {

System.out.println("1111111");

jdbc q = new jdbc(score);

inGame = false;

}

continueLevel();

}

private void moveGhosts(Graphics2D g2d) {

int pos;

int count;

for (int i = 0; i < N\_GHOSTS; i++) {

if (ghost\_x[i] % BLOCK\_SIZE == 0 && ghost\_y[i] % BLOCK\_SIZE == 0) {

pos = ghost\_x[i] / BLOCK\_SIZE + N\_BLOCKS \* (int) (ghost\_y[i] / BLOCK\_SIZE);

count = 0;

if ((screenData[pos] & 1) == 0 && ghost\_dx[i] != 1) {

dx[count] = -1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 2) == 0 && ghost\_dy[i] != 1) {

dx[count] = 0;

dy[count] = -1;

count++;

}

if ((screenData[pos] & 4) == 0 && ghost\_dx[i] != -1) {

dx[count] = 1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 8) == 0 && ghost\_dy[i] != -1) {

dx[count] = 0;

dy[count] = 1;

count++;

}

if (count == 0) {

if ((screenData[pos] & 15) == 15) {

ghost\_dx[i] = 0;

ghost\_dy[i] = 0;

} else {

ghost\_dx[i] = -ghost\_dx[i];

ghost\_dy[i] = -ghost\_dy[i];

}

} else {

count = (int) (Math.random() \* count);

if (count > 3) {

count = 3;

}

ghost\_dx[i] = dx[count];

ghost\_dy[i] = dy[count];

}

}

ghost\_x[i] = ghost\_x[i] + (ghost\_dx[i] \* ghostSpeed[i]);

ghost\_y[i] = ghost\_y[i] + (ghost\_dy[i] \* ghostSpeed[i]);

drawGhost(g2d, ghost\_x[i] + 1, ghost\_y[i] + 1);

if (pacman\_x > (ghost\_x[i] - 12) && pacman\_x < (ghost\_x[i] + 12)

&& pacman\_y > (ghost\_y[i] - 12) && pacman\_y < (ghost\_y[i] + 12)

&& inGame) {

dying = true;

}

}

}

private void drawGhost(Graphics2D g2d, int x, int y) {

g2d.drawImage(pizza, x, y, this);

}

private void movePacman() {

int pos;

short ch;

if (pacman\_x % BLOCK\_SIZE == 0 && pacman\_y % BLOCK\_SIZE == 0) {

pos = pacman\_x / BLOCK\_SIZE + N\_BLOCKS \* (int) (pacman\_y / BLOCK\_SIZE);

ch = screenData[pos];

if ((ch & 16) != 0) {

screenData[pos] = (short) (ch & 15);

score++;

}

if (req\_dx != 0 || req\_dy != 0) {

if (!((req\_dx == -1 && req\_dy == 0 && (ch & 1) != 0)

|| (req\_dx == 1 && req\_dy == 0 && (ch & 4) != 0)

|| (req\_dx == 0 && req\_dy == -1 && (ch & 2) != 0)

|| (req\_dx == 0 && req\_dy == 1 && (ch & 8) != 0))) {

pacmand\_x = req\_dx;

pacmand\_y = req\_dy;

}

}

// Check for standstill

if ((pacmand\_x == -1 && pacmand\_y == 0 && (ch & 1) != 0)

|| (pacmand\_x == 1 && pacmand\_y == 0 && (ch & 4) != 0)

|| (pacmand\_x == 0 && pacmand\_y == -1 && (ch & 2) != 0)

|| (pacmand\_x == 0 && pacmand\_y == 1 && (ch & 8) != 0)) {

pacmand\_x = 0;

pacmand\_y = 0;

}

}

pacman\_x = pacman\_x + PACMAN\_SPEED \* pacmand\_x;

pacman\_y = pacman\_y + PACMAN\_SPEED \* pacmand\_y;

}

private void drawPacman(Graphics2D g2d) {

if (req\_dx == -1) {

g2d.drawImage(left, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dx == 1) {

g2d.drawImage(right, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dy == -1) {

g2d.drawImage(up, pacman\_x + 1, pacman\_y + 1, this);

} else {

g2d.drawImage(down, pacman\_x + 1, pacman\_y + 1, this);

}

}

private void drawMaze(Graphics2D g2d) {

short i = 0;

int x, y;

int cnt = 1;

for (y = 0; y < SCREEN\_SIZE; y += BLOCK\_SIZE) {

for (x = 0; x < SCREEN\_SIZE; x += BLOCK\_SIZE) {

g2d.setColor(new Color(0,72,251));

g2d.setStroke(new BasicStroke(5));

if ((levelData[i] == 0)) {

g2d.fillRect(x, y, BLOCK\_SIZE, BLOCK\_SIZE);

}

if ((screenData[i] & 1) != 0) {

g2d.drawLine(x-2, y-2, x-2, y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 2) != 0) {

g2d.drawLine(x-2, y-2, x-2 + BLOCK\_SIZE + 2, y);

}

if ((screenData[i] & 4) != 0) {

g2d.drawLine(x-2 + BLOCK\_SIZE + 2, y-2, x-2 + BLOCK\_SIZE + 2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 8) != 0) {

g2d.drawLine(x-2, y-2 + BLOCK\_SIZE + 2, x-2 + BLOCK\_SIZE +2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 16) != 0) {

g2d.drawImage(grapes, x, y, this);

cnt++;

}

i++;

}

}

}

private void initGame() {

lives = 3;

score = 0;

initLevel();

N\_GHOSTS = 6;

currentSpeed = 3;

}

private void initLevel() {

int i;

for (i = 0; i < N\_BLOCKS \* N\_BLOCKS; i++) {

screenData[i] = levelData[i];

}

continueLevel();

}

private void continueLevel() {

int dx = 1;

int random;

for (int i = 0; i < N\_GHOSTS; i++) {

ghost\_y[i] = 4 \* BLOCK\_SIZE; //start position

ghost\_x[i] = 4 \* BLOCK\_SIZE;

ghost\_dy[i] = 0;

ghost\_dx[i] = dx;

dx = -dx;

random = (int) (Math.random() \* (currentSpeed + 1));

if (random > currentSpeed) {

random = currentSpeed;

}

ghostSpeed[i] = validSpeeds[random];

}

pacman\_x = 7 \* BLOCK\_SIZE; //start position

pacman\_y = 11 \* BLOCK\_SIZE;

pacmand\_x = 0; //reset direction move

pacmand\_y = 0;

req\_dx = 0; // reset direction controls

req\_dy = 0;

dying = false;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setColor(Color.black);

g2d.fillRect(0, 0, d.width, d.height);

drawMaze(g2d);

drawScore(g2d);

drawFact(g2d);

if (inGame) {

try {

playGame(g2d);

} catch (SQLException e) {

throw new RuntimeException(e);

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

} else {

showIntroScreen(g2d);

}

Toolkit.getDefaultToolkit().sync();

g2d.dispose();

}

//controls

class TAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (inGame) {

if (key == KeyEvent.VK\_LEFT) {

req\_dx = -1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_RIGHT) {

req\_dx = 1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_UP) {

req\_dx = 0;

req\_dy = -1;

} else if (key == KeyEvent.VK\_DOWN) {

req\_dx = 0;

req\_dy = 1;

} else if (key == KeyEvent.VK\_ESCAPE && timer.isRunning()) {

inGame = false;

}

} else {

if (key == KeyEvent.VK\_SPACE) {

inGame = true;

initGame();

}

if (key == KeyEvent.VK\_L) {

try {

new Leaderboard();

} catch (Exception qq) {

}

}

if (key == KeyEvent.VK\_G) {

try {

new ShowIntro();

} catch (Exception qq) {

}

}

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

repaint();

}

}

**Board2.java**

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.sql.SQLException;

import javax.swing.\*;

public class Board2 extends JPanel implements ActionListener {

private Dimension d;

private final Font smallFont = new Font("Arial", Font.BOLD, 14);

private final Font factfont = new Font("Arial", Font.BOLD, 18);

private final Font qfont = new Font("Arial", Font.BOLD, 25);

private boolean inGame = false;

private boolean dying = false;

private final int BLOCK\_SIZE = 24;

private final int N\_BLOCKS = 15;

private final int SCREEN\_SIZE = N\_BLOCKS \* BLOCK\_SIZE;

private final int MAX\_GHOSTS = 12;

private final int PACMAN\_SPEED = 6;

private int N\_GHOSTS = 12;

private int lives, score;

private int[] dx, dy;

private int[] ghost\_x, ghost\_y, ghost\_dx, ghost\_dy, ghostSpeed;

private Image heart, ghost , cherry ;

private Image up, down, left, right;

private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;

private int req\_dx, req\_dy;

private final short levelData[] = {

19, 26, 26, 26, 26, 26, 26, 26, 26, 18, 26, 26, 26, 26, 22,

21, 0, 0, 0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 21,

17, 18, 26, 18, 18, 18, 18, 26, 18, 16, 18, 18, 26, 18, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 24, 24, 28, 0, 25, 24, 24, 20, 0, 17, 20,

17, 16, 26, 20, 0, 0, 0, 0, 0, 0, 0, 17, 26, 16, 20,

17, 20, 0, 17, 18, 18, 22, 0, 19, 18, 18, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 24, 26, 24, 24, 24, 24, 18, 24, 24, 24, 24, 26, 24, 20,

21, 0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 0, 0, 21,

17, 18, 18, 26, 18, 18, 18, 24, 18, 18, 18, 26, 18, 18, 20,

17, 16, 20, 0, 17, 16, 20, 0, 17, 16, 20, 0, 17, 16, 20,

25, 24, 24, 26, 24, 24, 24, 26, 24, 24, 24, 26, 24, 24, 28

};

JButton p = new JButton("LeaderBoard");

private final int validSpeeds[] = {1, 2, 3, 4, 6, 8};

private final int maxSpeed = 6;

private int currentSpeed = 3;

private short[] screenData;

private Timer timer;

public Board2() {

loadImages();

initVariables();

addKeyListener(new TAdapter());

setFocusable(true);

initGame();

}

private void loadImages() {

down = new ImageIcon("images/down.gif").getImage();

up = new ImageIcon("images/up.gif").getImage();

left = new ImageIcon("images/left.gif").getImage();

right = new ImageIcon("images/right.gif").getImage();

ghost = new ImageIcon("images/qqq.png").getImage();

heart = new ImageIcon("images/heart.png").getImage();

cherry = new ImageIcon("images/cherry1.png").getImage();

}

private void initVariables() {

screenData = new short[N\_BLOCKS \* N\_BLOCKS];

d = new Dimension(760, 400);

ghost\_x = new int[MAX\_GHOSTS];

ghost\_dx = new int[MAX\_GHOSTS];

ghost\_y = new int[MAX\_GHOSTS];

ghost\_dy = new int[MAX\_GHOSTS];

ghostSpeed = new int[MAX\_GHOSTS];

dx = new int[4];

dy = new int[4];

timer = new Timer(40, this);

timer.start();

}

private void playGame(Graphics2D g2d) throws SQLException, ClassNotFoundException {

if (dying) {

death();

} else {

movePacman();

drawPacman(g2d);

moveGhosts(g2d);

checkMaze();

}

}

private void showIntroScreen(Graphics2D g2d) {

String start = "Press SPACE to start";

g2d.setColor(Color.yellow);

g2d.drawString(start, (SCREEN\_SIZE)/4, 150);

}

private void drawScore(Graphics2D g) {

g.setFont(smallFont);

g.setColor(new Color(5, 181, 79));

String s = "Score: " + score;

g.drawString(s, SCREEN\_SIZE / 2 + 96, SCREEN\_SIZE + 16);

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

g.drawImage(heart, i \* 28 + 8, SCREEN\_SIZE + 1, this);

}

}

private void drawFact(Graphics2D g) {

g.setFont(factfont);

g.setColor(new Color(5, 181, 79));

String s = "HOW PACMAN CAN STAY HEALTHY";

g.drawString(s, SCREEN\_SIZE / 2 + 220, 30);

String s1 = "if he eats healthy fruits";

String s12 = "his life won't decrease";

g.drawString(s1, SCREEN\_SIZE / 2 + 270, 75);

g.drawString(s12 , SCREEN\_SIZE / 2 + 272 , 95);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 270 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 290 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 310 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 330 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 350 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 370 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 390 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 410 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 430 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 450 , 115 , this);

g.drawImage(cherry , SCREEN\_SIZE / 2 + 470 , 115 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 270 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 290 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 310 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 330 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 350 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 370 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 390 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 410 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 430 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 450 , 140 , this);

g.drawImage(cherry, SCREEN\_SIZE / 2 + 470 , 140 , this);

g.setColor(Color.red);

String s2 = "If he eats junk food";

String s3 = "his life decreases";

g.drawString(s2, SCREEN\_SIZE / 2 + 285, 200);

g.drawString(s3 , SCREEN\_SIZE / 2 + 290 , 220);

g.drawImage(ghost , SCREEN\_SIZE/2 + 265 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 295 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 325 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 355 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 385 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 415 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 445 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 475 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 265 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 295 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 325 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 355 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 385 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 415 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 445 , 270 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 475 , 270 , this);

String s5 = "(It is true in real life)";

g.setColor(Color.green);

g.setFont(qfont);

g.drawString(s5 , SCREEN\_SIZE/2 + 265 , 330);

String s6 = "Press L for LeaderBoard";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s6 , SCREEN\_SIZE/2 + 270 , 360);

String s7 = "Press G for Main Menu";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s7 , SCREEN\_SIZE/2 + 270 , 380);

}

private void checkMaze() {

int i = 0;

boolean finished = true;

int cnt = 0 ;

while (i < N\_BLOCKS \* N\_BLOCKS && finished) {

if ((screenData[i]) != 0) {

finished = false;

}

if(screenData[i] == 0){

cnt++;

}

i++;

}

if(cnt == N\_BLOCKS\*N\_BLOCKS -5){

finished = true;

}

if (finished) {

score += 50;

if (N\_GHOSTS < MAX\_GHOSTS) {

N\_GHOSTS++;

}

if (currentSpeed < maxSpeed) {

currentSpeed++;

}

initLevel();

}

}

private void death() throws SQLException, ClassNotFoundException {

lives--;

if (lives == 0) {

System.out.println("1111111");

jdbc q = new jdbc(score);

inGame = false;

}

continueLevel();

}

private void moveGhosts(Graphics2D g2d) {

int pos;

int count;

for (int i = 0; i < N\_GHOSTS; i++) {

if (ghost\_x[i] % BLOCK\_SIZE == 0 && ghost\_y[i] % BLOCK\_SIZE == 0) {

pos = ghost\_x[i] / BLOCK\_SIZE + N\_BLOCKS \* (int) (ghost\_y[i] / BLOCK\_SIZE);

count = 0;

if ((screenData[pos] & 1) == 0 && ghost\_dx[i] != 1) {

dx[count] = -1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 2) == 0 && ghost\_dy[i] != 1) {

dx[count] = 0;

dy[count] = -1;

count++;

}

if ((screenData[pos] & 4) == 0 && ghost\_dx[i] != -1) {

dx[count] = 1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 8) == 0 && ghost\_dy[i] != -1) {

dx[count] = 0;

dy[count] = 1;

count++;

}

if (count == 0) {

if ((screenData[pos] & 15) == 15) {

ghost\_dx[i] = 0;

ghost\_dy[i] = 0;

} else {

ghost\_dx[i] = -ghost\_dx[i];

ghost\_dy[i] = -ghost\_dy[i];

}

} else {

count = (int) (Math.random() \* count);

if (count > 3) {

count = 3;

}

ghost\_dx[i] = dx[count];

ghost\_dy[i] = dy[count];

}

}

ghost\_x[i] = ghost\_x[i] + (ghost\_dx[i] \* ghostSpeed[i]);

ghost\_y[i] = ghost\_y[i] + (ghost\_dy[i] \* ghostSpeed[i]);

drawGhost(g2d, ghost\_x[i] + 1, ghost\_y[i] + 1);

if (pacman\_x > (ghost\_x[i] - 12) && pacman\_x < (ghost\_x[i] + 12)

&& pacman\_y > (ghost\_y[i] - 12) && pacman\_y < (ghost\_y[i] + 12)

&& inGame) {

dying = true;

}

}

}

private void drawGhost(Graphics2D g2d, int x, int y) {

g2d.drawImage(ghost, x, y, this);

}

private void movePacman() {

int pos;

short ch;

if (pacman\_x % BLOCK\_SIZE == 0 && pacman\_y % BLOCK\_SIZE == 0) {

pos = pacman\_x / BLOCK\_SIZE + N\_BLOCKS \* (int) (pacman\_y / BLOCK\_SIZE);

ch = screenData[pos];

if ((ch & 16) != 0) {

screenData[pos] = (short) (ch & 15);

score++;

}

if (req\_dx != 0 || req\_dy != 0) {

if (!((req\_dx == -1 && req\_dy == 0 && (ch & 1) != 0)

|| (req\_dx == 1 && req\_dy == 0 && (ch & 4) != 0)

|| (req\_dx == 0 && req\_dy == -1 && (ch & 2) != 0)

|| (req\_dx == 0 && req\_dy == 1 && (ch & 8) != 0))) {

pacmand\_x = req\_dx;

pacmand\_y = req\_dy;

}

}

// Check for standstill

if ((pacmand\_x == -1 && pacmand\_y == 0 && (ch & 1) != 0)

|| (pacmand\_x == 1 && pacmand\_y == 0 && (ch & 4) != 0)

|| (pacmand\_x == 0 && pacmand\_y == -1 && (ch & 2) != 0)

|| (pacmand\_x == 0 && pacmand\_y == 1 && (ch & 8) != 0)) {

pacmand\_x = 0;

pacmand\_y = 0;

}

}

pacman\_x = pacman\_x + PACMAN\_SPEED \* pacmand\_x;

pacman\_y = pacman\_y + PACMAN\_SPEED \* pacmand\_y;

}

private void drawPacman(Graphics2D g2d) {

if (req\_dx == -1) {

g2d.drawImage(left, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dx == 1) {

g2d.drawImage(right, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dy == -1) {

g2d.drawImage(up, pacman\_x + 1, pacman\_y + 1, this);

} else {

g2d.drawImage(down, pacman\_x + 1, pacman\_y + 1, this);

}

}

private void drawMaze(Graphics2D g2d) {

short i = 0;

int x, y;

int cnt = 1;

for (y = 0; y < SCREEN\_SIZE; y += BLOCK\_SIZE) {

for (x = 0; x < SCREEN\_SIZE; x += BLOCK\_SIZE) {

g2d.setColor(new Color(0,72,251));

g2d.setStroke(new BasicStroke(5));

if ((levelData[i] == 0)) {

g2d.fillRect(x, y, BLOCK\_SIZE, BLOCK\_SIZE);

}

if ((screenData[i] & 1) != 0) {

g2d.drawLine(x-2, y-2, x-2, y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 2) != 0) {

g2d.drawLine(x-2, y-2, x-2 + BLOCK\_SIZE + 2, y);

}

if ((screenData[i] & 4) != 0) {

g2d.drawLine(x-2 + BLOCK\_SIZE + 2, y-2, x-2 + BLOCK\_SIZE + 2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 8) != 0) {

g2d.drawLine(x-2, y-2 + BLOCK\_SIZE + 2, x-2 + BLOCK\_SIZE +2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 16) != 0) {

g2d.drawImage(cherry, x, y, this);

cnt++;

}

i++;

}

}

}

private void initGame() {

lives = 3;

score = 0;

initLevel();

N\_GHOSTS = 6;

currentSpeed = 3;

}

private void initLevel() {

int i;

for (i = 0; i < N\_BLOCKS \* N\_BLOCKS; i++) {

screenData[i] = levelData[i];

}

continueLevel();

}

private void continueLevel() {

int dx = 1;

int random;

for (int i = 0; i < N\_GHOSTS; i++) {

ghost\_y[i] = 4 \* BLOCK\_SIZE; //start position

ghost\_x[i] = 4 \* BLOCK\_SIZE;

ghost\_dy[i] = 0;

ghost\_dx[i] = dx;

dx = -dx;

random = (int) (Math.random() \* (currentSpeed + 1));

if (random > currentSpeed) {

random = currentSpeed;

}

ghostSpeed[i] = validSpeeds[random];

}

pacman\_x = 7 \* BLOCK\_SIZE; //start position

pacman\_y = 11 \* BLOCK\_SIZE;

pacmand\_x = 0; //reset direction move

pacmand\_y = 0;

req\_dx = 0; // reset direction controls

req\_dy = 0;

dying = false;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setColor(Color.black);

g2d.fillRect(0, 0, d.width, d.height);

drawMaze(g2d);

drawScore(g2d);

drawFact(g2d);

if (inGame) {

try {

playGame(g2d);

} catch (SQLException e) {

throw new RuntimeException(e);

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

} else {

showIntroScreen(g2d);

}

Toolkit.getDefaultToolkit().sync();

g2d.dispose();

}

//controls

class TAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (inGame) {

if (key == KeyEvent.VK\_LEFT) {

req\_dx = -1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_RIGHT) {

req\_dx = 1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_UP) {

req\_dx = 0;

req\_dy = -1;

} else if (key == KeyEvent.VK\_DOWN) {

req\_dx = 0;

req\_dy = 1;

} else if (key == KeyEvent.VK\_ESCAPE && timer.isRunning()) {

inGame = false;

}

} else {

if (key == KeyEvent.VK\_SPACE) {

inGame = true;

initGame();

}

if (key == KeyEvent.VK\_L) {

try {

new Leaderboard();

} catch (Exception qq) {

}

}

if (key == KeyEvent.VK\_G) {

try {

new ShowIntro();

} catch (Exception qq) {

}

}

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

repaint();

}

}

**Board3.java**

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.io.IOException;

import java.sql.SQLException;

import javax.swing.\*;

public class Board3 extends JPanel implements ActionListener {

private Dimension d;

private final Font smallFont = new Font("Arial", Font.BOLD, 14);

private final Font factfont = new Font("Arial", Font.BOLD, 18);

private final Font qfont = new Font("Arial", Font.BOLD, 25);

private boolean inGame = false;

private boolean dying = false;

private final int BLOCK\_SIZE = 24;

private final int N\_BLOCKS = 15;

private final int SCREEN\_SIZE = N\_BLOCKS \* BLOCK\_SIZE;

private final int MAX\_GHOSTS = 12;

private final int PACMAN\_SPEED = 6;

private int N\_GHOSTS = 12;

private int lives, score;

private int[] dx, dy;

private int[] ghost\_x, ghost\_y, ghost\_dx, ghost\_dy, ghostSpeed;

private Image heart, ghost , cherry , banana;

private Image up, down, left, right;

private int pacman\_x, pacman\_y, pacmand\_x, pacmand\_y;

private int req\_dx, req\_dy;

private final short levelData[] = {

19, 26, 26, 26, 26, 26, 26, 26, 26, 18, 26, 26, 26, 26, 22,

21, 0, 0, 0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 21,

17, 18, 26, 26, 18, 26, 18, 26, 18, 16, 18, 18, 26, 18, 20,

17, 20, 0, 0, 21, 0, 21, 0, 17, 24, 16, 20, 0, 17, 20,

17, 16, 26, 26, 16, 26, 28, 0, 20, 0, 17, 20, 0, 17, 20,

17, 20, 0, 0, 21, 0, 0, 0, 25, 26, 24, 20, 0, 17, 20,

17, 16, 26, 18, 20, 0, 0, 0, 0, 0, 0, 17, 26, 16, 20,

17, 20, 0, 17, 16, 18, 22, 0, 19, 18, 18, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 20, 0, 17, 16, 16, 20, 0, 17, 16, 16, 20, 0, 17, 20,

17, 24, 26, 24, 24, 24, 24, 18, 24, 24, 24, 24, 26, 24, 20,

21, 0, 0, 0, 0, 0, 0, 21, 0, 0, 0, 0, 0, 0, 21,

17, 18, 18, 26, 18, 18, 18, 24, 18, 18, 18, 26, 18, 18, 20,

17, 16, 20, 0, 17, 16, 20, 0, 17, 16, 20, 0, 17, 16, 20,

25, 24, 24, 26, 24, 24, 24, 26, 24, 24, 24, 26, 24, 24, 28

};

JButton p = new JButton("LeaderBoard");

private final int validSpeeds[] = {1, 2, 3, 4, 6, 8};

private final int maxSpeed = 6;

private int currentSpeed = 3;

private short[] screenData;

private Timer timer;

public Board3() {

loadImages();

initVariables();

addKeyListener(new TAdapter());

setFocusable(true);

initGame();

}

private void loadImages() {

down = new ImageIcon("images/down.gif").getImage();

up = new ImageIcon("images/up.gif").getImage();

left = new ImageIcon("images/left.gif").getImage();

right = new ImageIcon("images/right.gif").getImage();

ghost = new ImageIcon("images/cola.png").getImage();

heart = new ImageIcon("images/heart.png").getImage();

banana = new ImageIcon("images/banana.png").getImage();

}

private void initVariables() {

screenData = new short[N\_BLOCKS \* N\_BLOCKS];

d = new Dimension(760, 400);

ghost\_x = new int[MAX\_GHOSTS];

ghost\_dx = new int[MAX\_GHOSTS];

ghost\_y = new int[MAX\_GHOSTS];

ghost\_dy = new int[MAX\_GHOSTS];

ghostSpeed = new int[MAX\_GHOSTS];

dx = new int[4];

dy = new int[4];

timer = new Timer(40, this);

timer.start();

}

private void playGame(Graphics2D g2d) throws SQLException, ClassNotFoundException {

if (dying) {

death();

} else {

movePacman();

drawPacman(g2d);

moveGhosts(g2d);

checkMaze();

}

}

private void showIntroScreen(Graphics2D g2d) {

String start = "Press SPACE to start";

g2d.setColor(Color.yellow);

g2d.drawString(start, (SCREEN\_SIZE)/4, 150);

}

private void drawScore(Graphics2D g) {

g.setFont(smallFont);

g.setColor(new Color(5, 181, 79));

String s = "Score: " + score;

g.drawString(s, SCREEN\_SIZE / 2 + 96, SCREEN\_SIZE + 16);

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

g.drawImage(heart, i \* 28 + 8, SCREEN\_SIZE + 1, this);

}

}

private void drawFact(Graphics2D g) {

g.setFont(factfont);

g.setColor(new Color(5, 181, 79));

String s = "HOW PACMAN CAN STAY HEALTHY";

g.drawString(s, SCREEN\_SIZE / 2 + 220, 30);

String s1 = "if he eats healthy fruits";

String s12 = "his life won't decrease";

g.drawString(s1, SCREEN\_SIZE / 2 + 270, 75);

g.drawString(s12 , SCREEN\_SIZE / 2 + 272 , 95);

g.drawImage(banana , SCREEN\_SIZE / 2 + 270 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 290 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 310 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 330 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 350 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 370 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 390 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 410 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 430 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 450 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 470 , 115 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 270 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 290 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 310 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 330 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 350 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 370 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 390 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 410 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 430 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 450 , 140 , this);

g.drawImage(banana , SCREEN\_SIZE / 2 + 470 , 140 , this);

g.setColor(Color.red);

String s2 = "If he eats junk food";

String s3 = "his life decreases";

g.drawString(s2, SCREEN\_SIZE / 2 + 285, 200);

g.drawString(s3 , SCREEN\_SIZE / 2 + 290 , 220);

g.drawImage(ghost , SCREEN\_SIZE/2 + 265 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 295 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 325 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 355 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 385 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 415 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 445 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 475 , 240 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 265 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 295 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 325 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 355 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 385 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 415 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 445 , 275 , this);

g.drawImage(ghost , SCREEN\_SIZE/2 + 475 , 275 , this);

String s5 = "(It is true in real life)";

g.setColor(Color.green);

g.setFont(qfont);

g.drawString(s5 , SCREEN\_SIZE/2 + 265 , 330);

String s6 = "Press L for LeaderBoard";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s6 , SCREEN\_SIZE/2 + 270 , 360);

String s7 = "Press G for Main Menu";

g.setColor(Color.yellow);

g.setFont(factfont);

g.drawString(s7 , SCREEN\_SIZE/2 + 270 , 380);

}

private void checkMaze() {

int i = 0;

boolean finished = true;

while (i < N\_BLOCKS \* N\_BLOCKS && finished) {

if ((screenData[i]) != 0) {

finished = false;

}

i++;

}

if (finished) {

score += 50;

if (N\_GHOSTS < MAX\_GHOSTS) {

N\_GHOSTS++;

}

if (currentSpeed < maxSpeed) {

currentSpeed++;

}

initLevel();

}

}

private void death() throws SQLException, ClassNotFoundException {

lives--;

if (lives == 0) {

System.out.println("1111111");

jdbc q = new jdbc(score);

inGame = false;

}

continueLevel();

}

private void moveGhosts(Graphics2D g2d) {

int pos;

int count;

for (int i = 0; i < N\_GHOSTS; i++) {

if (ghost\_x[i] % BLOCK\_SIZE == 0 && ghost\_y[i] % BLOCK\_SIZE == 0) {

pos = ghost\_x[i] / BLOCK\_SIZE + N\_BLOCKS \* (int) (ghost\_y[i] / BLOCK\_SIZE);

count = 0;

if ((screenData[pos] & 1) == 0 && ghost\_dx[i] != 1) {

dx[count] = -1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 2) == 0 && ghost\_dy[i] != 1) {

dx[count] = 0;

dy[count] = -1;

count++;

}

if ((screenData[pos] & 4) == 0 && ghost\_dx[i] != -1) {

dx[count] = 1;

dy[count] = 0;

count++;

}

if ((screenData[pos] & 8) == 0 && ghost\_dy[i] != -1) {

dx[count] = 0;

dy[count] = 1;

count++;

}

if (count == 0) {

if ((screenData[pos] & 15) == 15) {

ghost\_dx[i] = 0;

ghost\_dy[i] = 0;

} else {

ghost\_dx[i] = -ghost\_dx[i];

ghost\_dy[i] = -ghost\_dy[i];

}

} else {

count = (int) (Math.random() \* count);

if (count > 3) {

count = 3;

}

ghost\_dx[i] = dx[count];

ghost\_dy[i] = dy[count];

}

}

ghost\_x[i] = ghost\_x[i] + (ghost\_dx[i] \* ghostSpeed[i]);

ghost\_y[i] = ghost\_y[i] + (ghost\_dy[i] \* ghostSpeed[i]);

drawGhost(g2d, ghost\_x[i] + 1, ghost\_y[i] + 1);

if (pacman\_x > (ghost\_x[i] - 12) && pacman\_x < (ghost\_x[i] + 12)

&& pacman\_y > (ghost\_y[i] - 12) && pacman\_y < (ghost\_y[i] + 12)

&& inGame) {

dying = true;

}

}

}

private void drawGhost(Graphics2D g2d, int x, int y) {

g2d.drawImage(ghost, x, y, this);

}

private void movePacman() {

int pos;

short ch;

if (pacman\_x % BLOCK\_SIZE == 0 && pacman\_y % BLOCK\_SIZE == 0) {

pos = pacman\_x / BLOCK\_SIZE + N\_BLOCKS \* (int) (pacman\_y / BLOCK\_SIZE);

ch = screenData[pos];

if ((ch & 16) != 0) {

screenData[pos] = (short) (ch & 15);

score++;

}

if (req\_dx != 0 || req\_dy != 0) {

if (!((req\_dx == -1 && req\_dy == 0 && (ch & 1) != 0)

|| (req\_dx == 1 && req\_dy == 0 && (ch & 4) != 0)

|| (req\_dx == 0 && req\_dy == -1 && (ch & 2) != 0)

|| (req\_dx == 0 && req\_dy == 1 && (ch & 8) != 0))) {

pacmand\_x = req\_dx;

pacmand\_y = req\_dy;

}

}

// Check for standstill

if ((pacmand\_x == -1 && pacmand\_y == 0 && (ch & 1) != 0)

|| (pacmand\_x == 1 && pacmand\_y == 0 && (ch & 4) != 0)

|| (pacmand\_x == 0 && pacmand\_y == -1 && (ch & 2) != 0)

|| (pacmand\_x == 0 && pacmand\_y == 1 && (ch & 8) != 0)) {

pacmand\_x = 0;

pacmand\_y = 0;

}

}

pacman\_x = pacman\_x + PACMAN\_SPEED \* pacmand\_x;

pacman\_y = pacman\_y + PACMAN\_SPEED \* pacmand\_y;

}

private void drawPacman(Graphics2D g2d) {

if (req\_dx == -1) {

g2d.drawImage(left, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dx == 1) {

g2d.drawImage(right, pacman\_x + 1, pacman\_y + 1, this);

} else if (req\_dy == -1) {

g2d.drawImage(up, pacman\_x + 1, pacman\_y + 1, this);

} else {

g2d.drawImage(down, pacman\_x + 1, pacman\_y + 1, this);

}

}

private void drawMaze(Graphics2D g2d) {

short i = 0;

int x, y;

int cnt = 1;

for (y = 0; y < SCREEN\_SIZE; y += BLOCK\_SIZE) {

for (x = 0; x < SCREEN\_SIZE; x += BLOCK\_SIZE) {

g2d.setColor(new Color(0,72,251));

g2d.setStroke(new BasicStroke(5));

if ((levelData[i] == 0)) {

g2d.fillRect(x, y, BLOCK\_SIZE, BLOCK\_SIZE);

}

if ((screenData[i] & 1) != 0) {

g2d.drawLine(x-2, y-2, x-2, y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 2) != 0) {

g2d.drawLine(x-2, y-2, x-2 + BLOCK\_SIZE + 2, y);

}

if ((screenData[i] & 4) != 0) {

g2d.drawLine(x-2 + BLOCK\_SIZE + 2, y-2, x-2 + BLOCK\_SIZE + 2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 8) != 0) {

g2d.drawLine(x-2, y-2 + BLOCK\_SIZE + 2, x-2 + BLOCK\_SIZE +2,

y-2 + BLOCK\_SIZE + 2);

}

if ((screenData[i] & 16) != 0) {

g2d.drawImage(banana, x, y, this);

cnt++;

}

i++;

}

}

}

private void initGame() {

lives = 3;

score = 0;

initLevel();

N\_GHOSTS = 6;

currentSpeed = 3;

}

private void initLevel() {

int i;

for (i = 0; i < N\_BLOCKS \* N\_BLOCKS; i++) {

screenData[i] = levelData[i];

}

continueLevel();

}

private void continueLevel() {

int dx = 1;

int random;

for (int i = 0; i < N\_GHOSTS; i++) {

ghost\_y[i] = 4 \* BLOCK\_SIZE; //start position

ghost\_x[i] = 4 \* BLOCK\_SIZE;

ghost\_dy[i] = 0;

ghost\_dx[i] = dx;

dx = -dx;

random = (int) (Math.random() \* (currentSpeed + 1));

if (random > currentSpeed) {

random = currentSpeed;

}

ghostSpeed[i] = validSpeeds[random];

}

pacman\_x = 7 \* BLOCK\_SIZE; //start position

pacman\_y = 11 \* BLOCK\_SIZE;

pacmand\_x = 0; //reset direction move

pacmand\_y = 0;

req\_dx = 0; // reset direction controls

req\_dy = 0;

dying = false;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setColor(Color.black);

g2d.fillRect(0, 0, d.width, d.height);

drawMaze(g2d);

drawScore(g2d);

drawFact(g2d);

if (inGame) {

try {

playGame(g2d);

} catch (SQLException e) {

throw new RuntimeException(e);

} catch (ClassNotFoundException e) {

throw new RuntimeException(e);

}

} else {

showIntroScreen(g2d);

}

Toolkit.getDefaultToolkit().sync();

g2d.dispose();

}

//controls

class TAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (inGame) {

if (key == KeyEvent.VK\_LEFT) {

req\_dx = -1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_RIGHT) {

req\_dx = 1;

req\_dy = 0;

} else if (key == KeyEvent.VK\_UP) {

req\_dx = 0;

req\_dy = -1;

} else if (key == KeyEvent.VK\_DOWN) {

req\_dx = 0;

req\_dy = 1;

} else if (key == KeyEvent.VK\_ESCAPE && timer.isRunning()) {

inGame = false;

}

} else {

if (key == KeyEvent.VK\_SPACE) {

inGame = true;

initGame();

}

if (key == KeyEvent.VK\_L) {

try {

new Leaderboard();

} catch (Exception qq) {

}

}

if (key == KeyEvent.VK\_G) {

try {

new ShowIntro();

} catch (Exception qq) {

}

}

}

}

}

@Override

public void actionPerformed(ActionEvent e) {

repaint();

}

}

**Levels.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

public class levels {

public static void main(String[] args) throws IOException, FontFormatException {

ShowLevels w = new ShowLevels();

}

}

class ShowLevels extends JFrame{

Font Lot;

static Font CrackMan;

Font video;

public ShowLevels() throws IOException, FontFormatException {

Lot = Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf"));

GraphicsEnvironment g = GraphicsEnvironment.getLocalGraphicsEnvironment();

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf")));

CrackMan = Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF")));

video = Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF")));

JLabel q = new JLabel("Select Levels");

JButton level1 = new JButton("Level 1");

JButton level2 = new JButton("Level 2");

JButton level3 = new JButton("Level 3");

setVisible(true);

setLayout(null);

getContentPane().setBackground(Color.black);

level1.setFont(new Font("Lot", Font.PLAIN, 20));

level2.setFont(new Font("Lot" , Font.PLAIN, 20));

level1.setBackground(Color.black);

level1.setForeground(Color.yellow);

level2.setBackground(Color.black);

level2.setForeground(Color.yellow);

level3.setBackground(Color.black);

level3.setForeground(Color.yellow);

level3.setFont(new Font("Lot" , Font.PLAIN , 20));

//Select Levels

q.setFont(new Font("CrackMan", Font.PLAIN, 35));

q.setBackground(Color.black);

q.setForeground(Color.yellow);

q.setBounds(410 , 20 , 500 , 100);

level1.setBounds(470 , 250 , 300 , 50);

level2.setBounds(470 , 350 , 300 , 50);

level3.setBounds(470 , 450 , 300 , 50);

BufferedImage myPicture = ImageIO.read(new File("images/maze.jpg"));

JLabel picLabel = new JLabel(new ImageIcon(myPicture));

picLabel.setBounds(5, 220 , 300 , 300);

BufferedImage burger = ImageIO.read(new File("images/burg.png"));

JLabel junkfood = new JLabel(new ImageIcon(burger));

junkfood.setBounds(895 , 250 , 300 , 300);

BufferedImage fruits = ImageIO.read(new File("images/healthy.jpg"));

JLabel healthy = new JLabel(new ImageIcon(fruits));

healthy.setBounds(450, 20 , 300 , 300);

BufferedImage chery = ImageIO.read(new File("images/cherry.png"));

JLabel cherry = new JLabel(new ImageIcon(chery));

cherry.setBounds(50 , 100 , 200 , 200);

BufferedImage appl = ImageIO.read(new File("images/apple.png"));

JLabel apple = new JLabel(new ImageIcon(appl));

apple.setBounds(930 , 100 , 200 , 200);

add(picLabel);

add(junkfood);

add(healthy);

add(cherry);

add(apple);

add(q);

add(level1);

add(level3);

add(level2);

setSize(1280,600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

level1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Pacman ex = new Pacman(1);

ex.setVisible(true);

dispose();

}

});

level2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Pacman ex = new Pacman(2);

ex.setVisible(true);

dispose();

}

});

level3.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Pacman ex = new Pacman(3);

ex.setVisible(true);

dispose();

}

});

}

}

**Jdbc.java**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class jdbc {

public static int score = 0;

public jdbc(int score) throws ClassNotFoundException, SQLException {

System.out.println("2222222");

jdbc.score = score;

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

LocalDateTime now = LocalDateTime.now();

String time = now.format(dtf);

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "system";

String password = "system";

Connection con = DriverManager.getConnection(url, username, password);

System.out.println("Connection Established successfully");

Statement statement = con.createStatement();

System.out.println("insert");

String query = "INSERT INTO leaderboard (Score , data) values(" + score + " , '" + time + "')";

System.out.println("insert");

statement.executeUpdate(query);

}

}

**Leaderboard.java**

import java.awt.Color;

import java.awt.Font;

import java.awt.FontFormatException;

import java.awt.GraphicsEnvironment;

import java.awt.LayoutManager;

import java.io.File;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

public class Leaderboard {

JFrame frame = new JFrame();

Leaderboard() throws ClassNotFoundException, SQLException, IOException, FontFormatException {

Font smallFont = new Font("Arial", 1, 15);

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "system";

String password = "system";

Connection con = DriverManager.getConnection(url, username, password);

System.out.println("Connection Established successfully");

Statement statement = con.createStatement();

System.out.println("retrieve");

String query = "SELECT \* FROM leaderboard ORDER BY Score DESC LIMIT 10";

System.out.println("retrieve");

int i = 0;

String data[][] = new String[10][3];

ResultSet resultSet = statement.executeQuery(query);

while(resultSet.next()){

String score = Integer.toString(resultSet.getInt("SCORE"));

String date = resultSet.getString("DATA");

data[i] = new String[]{i+1+"" , score, date};

i++;

}

Font CrackMan = Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF"));

GraphicsEnvironment g = GraphicsEnvironment.getLocalGraphicsEnvironment();

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF")));

String[] columnnames = {"S.NO" , "SCORE" , "WHEN"};

JLabel t = new JLabel("Leaderboard");

t.setFont(new Font("CrackMan" , Font.BOLD , 50));

t.setForeground(Color.yellow);

t.setBounds(230,780,500,100);

t.setBackground(Color.black);

JTable j = new JTable(data , columnnames);

j.setBounds(200,200,0,0);

j.getColumnModel().getColumn(0).setPreferredWidth(0);

j.getColumnModel().getColumn(1).setPreferredWidth(100);

j.getColumnModel().getColumn(2).setPreferredWidth(100);

for(int ss = 0 ; ss < 10 ; ss++){

j.setRowHeight(ss,30);

}

j.setForeground(Color.yellow);

j.setFont(smallFont);

j.setGridColor(Color.blue);

j.setBackground(Color.black);

j.getTableHeader().setBackground(Color.black);

j.getTableHeader().setFont(smallFont);

j.getTableHeader().setForeground(Color.yellow);

JScrollPane sp = new JScrollPane(j);

frame.setSize(500,370);

frame.add(t);

frame.add(sp);

frame.setVisible(true);

frame.setBackground(Color.black);

frame.getContentPane().setBackground(Color.black);

frame.setLayout(null);

frame.setTitle("Leader Board");

frame.setDefaultCloseOperation(frame.DISPOSE\_ON\_CLOSE);

}

}

**About.java**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.sql.SQLException;

import javax.imageio.ImageIO;

class about extends JFrame{

Font Lot;

static Font CrackMan;

Font video;

public about() throws IOException, FontFormatException {

//Pacman Font 1

Lot = Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf"));

GraphicsEnvironment g = GraphicsEnvironment.getLocalGraphicsEnvironment();

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("Lot.otf")));

//Pacman Font2

CrackMan = Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("CrackMan.TTF")));

video = Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF"));

g.registerFont(Font.createFont(Font.TRUETYPE\_FONT , new File("videobeast80s.TTF")));

//Title

JLabel q = new JLabel("Learn with PAC-MAN");

q.setFont(new Font("CrackMan", Font.PLAIN, 35));

q.setBackground(Color.black);

q.setForeground(Color.yellow);

q.setBounds(410 , 20 , 500 , 100);

JButton ext = new JButton("Back");

ext.setFont(new Font("Lot" , Font.PLAIN, 20));

ext.setBackground(Color.black);

ext.setForeground(Color.yellow);

ext.setBounds(470 , 470 , 300 , 50);

JLabel names = new JLabel("This project was created by : ");

names.setFont(new Font("Lot", Font.PLAIN, 25));

names.setBackground(Color.black);

names.setForeground(Color.yellow);

names.setBounds(410 , 220 , 600 , 100);

JLabel name1 = new JLabel("Kartikeyan TR");

name1.setFont(new Font("CrackMan", Font.PLAIN, 25));

name1.setBackground(Color.black);

name1.setForeground(Color.yellow);

name1.setBounds(520 , 270 , 500 , 100);

JLabel name2 = new JLabel("Suriyaa V");

name2.setFont(new Font("CrackMan", Font.PLAIN, 25));

name2.setBackground(Color.black);

name2.setForeground(Color.yellow);

name2.setBounds(560 , 320 , 500 , 100);

JLabel name3 = new JLabel("Guru Raman C");

name3.setFont(new Font("CrackMan", Font.PLAIN, 25));

name3.setBackground(Color.black);

name3.setForeground(Color.yellow);

name3.setBounds(530 , 370 , 500 , 100);

BufferedImage myPicture = ImageIO.read(new File("images/maze.jpg"));

JLabel picLabel = new JLabel(new ImageIcon(myPicture));

picLabel.setBounds(5, 220 , 300 , 300);

BufferedImage burger = ImageIO.read(new File("images/burg.png"));

JLabel junkfood = new JLabel(new ImageIcon(burger));

junkfood.setBounds(895 , 250 , 300 , 300);

BufferedImage fruits = ImageIO.read(new File("images/healthy.jpg"));

JLabel healthy = new JLabel(new ImageIcon(fruits));

healthy.setBounds(450, 20 , 300 , 300);

BufferedImage chery = ImageIO.read(new File("images/cherry.png"));

JLabel cherry = new JLabel(new ImageIcon(chery));

cherry.setBounds(50 , 100 , 200 , 200);

BufferedImage appl = ImageIO.read(new File("images/apple.png"));

JLabel apple = new JLabel(new ImageIcon(appl));

apple.setBounds(930 , 100 , 200 , 200);

setVisible(true);

setLayout(null);

add(picLabel);

add(junkfood);

add(healthy);

add(cherry);

add(apple);

add(q);

add(ext);

add(names);

add(name1);

add(name2);

add(name3);

setSize(1280,600);

getContentPane().setBackground(Color.black);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

ext.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

new ShowIntro();

} catch (IOException ex) {

throw new RuntimeException(ex);

} catch (FontFormatException ex) {

throw new RuntimeException(ex);

}

dispose();

}

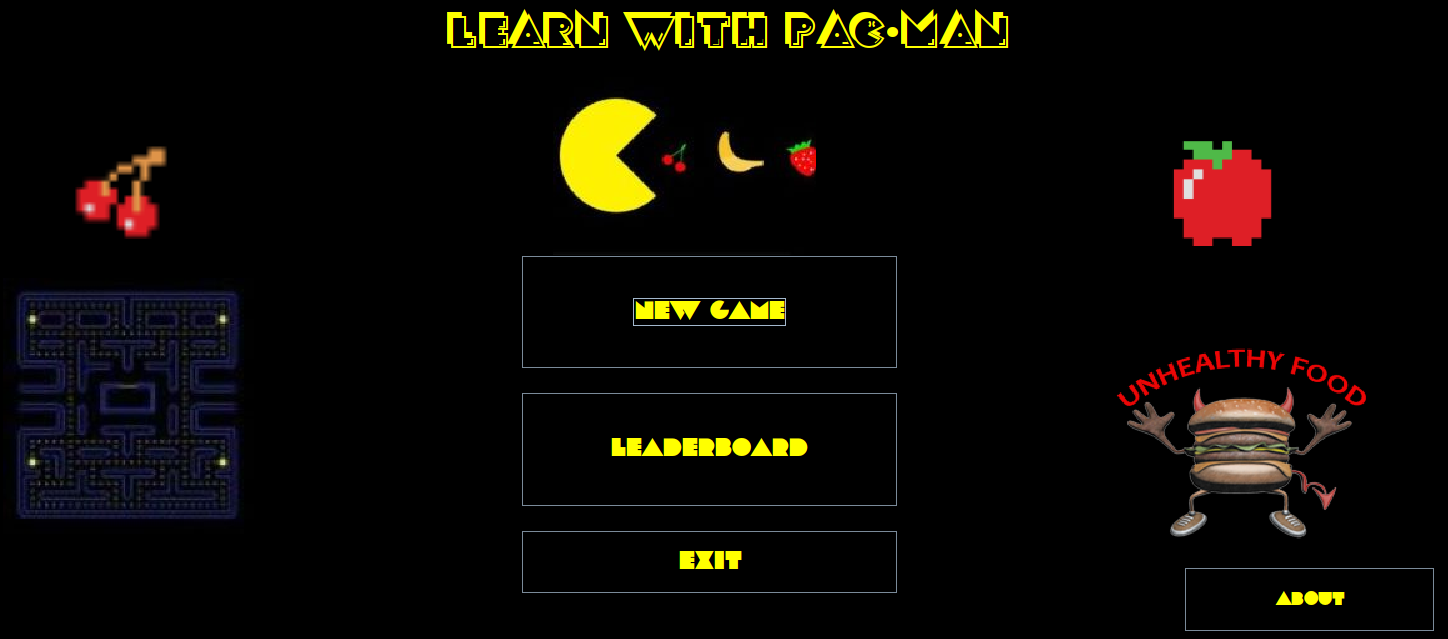
});

}

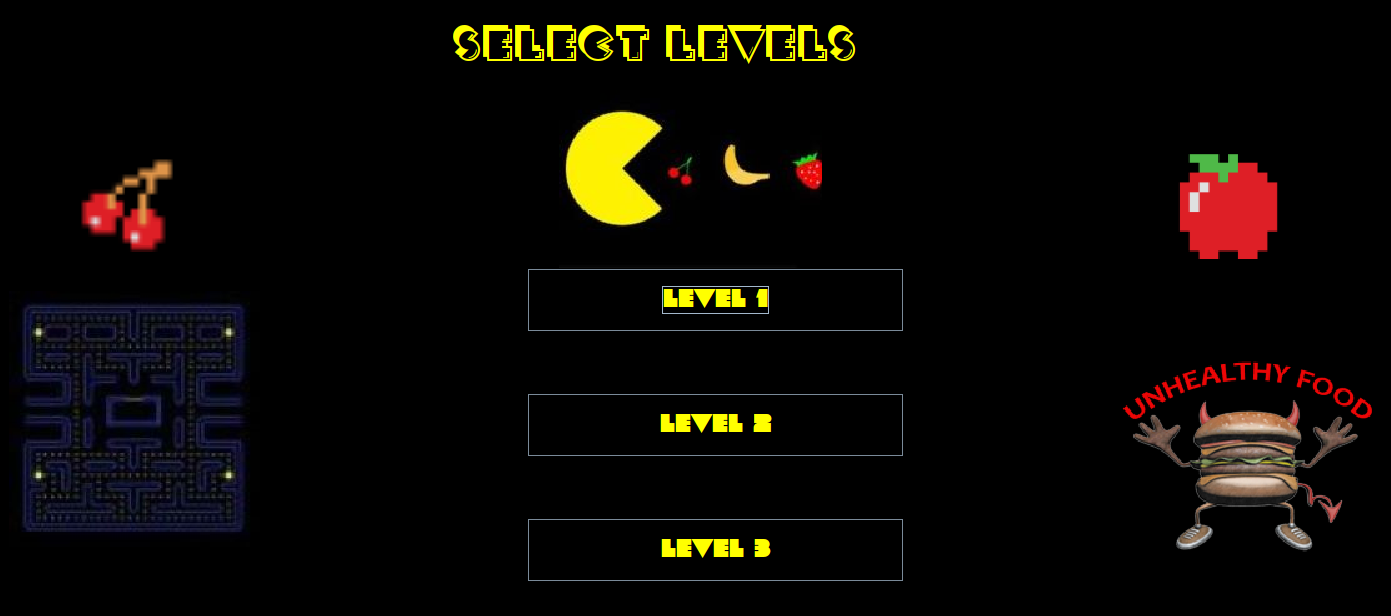
}

**APPLICATION SCREENSHOTS**

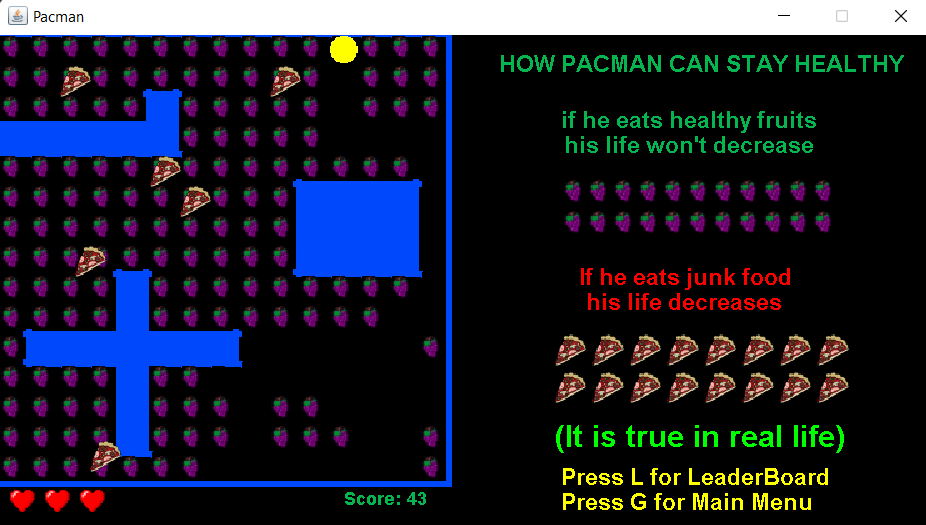
**Home**



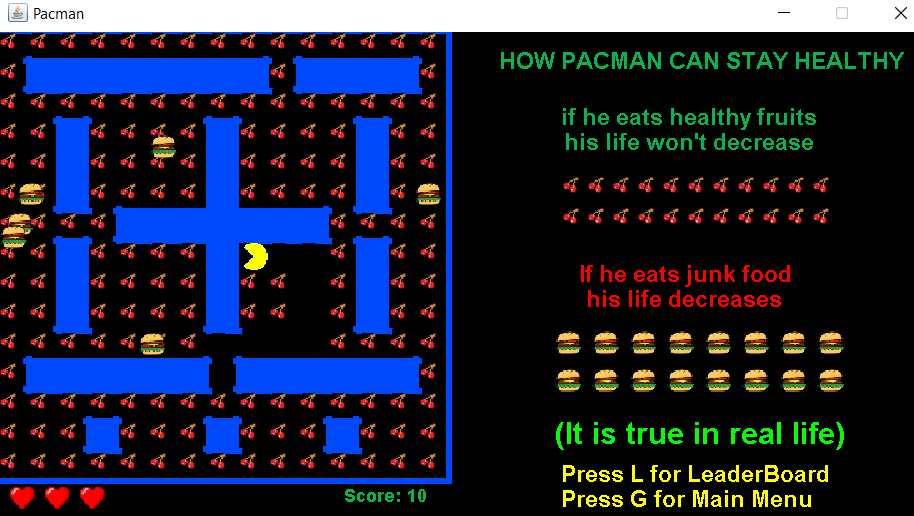
**Levels page**



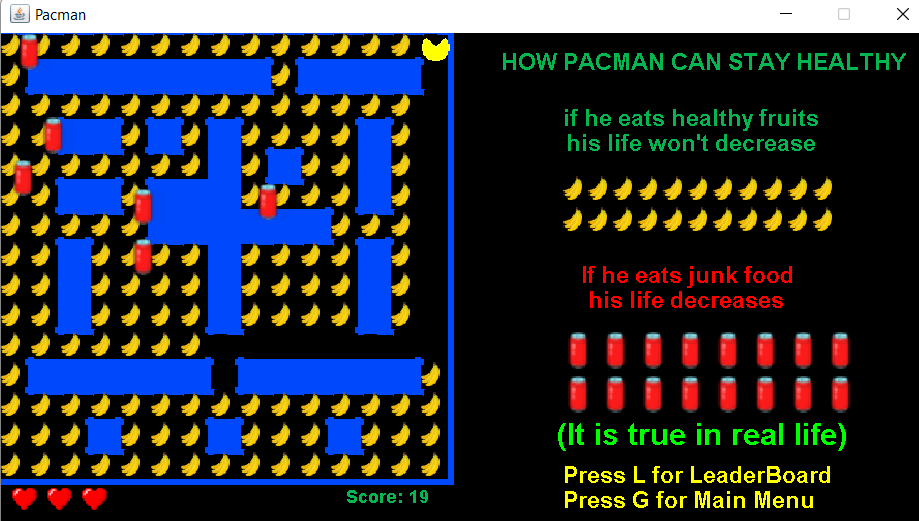
**Level 1**



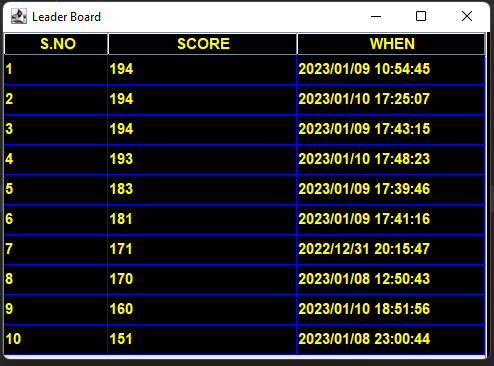
**Level 2**



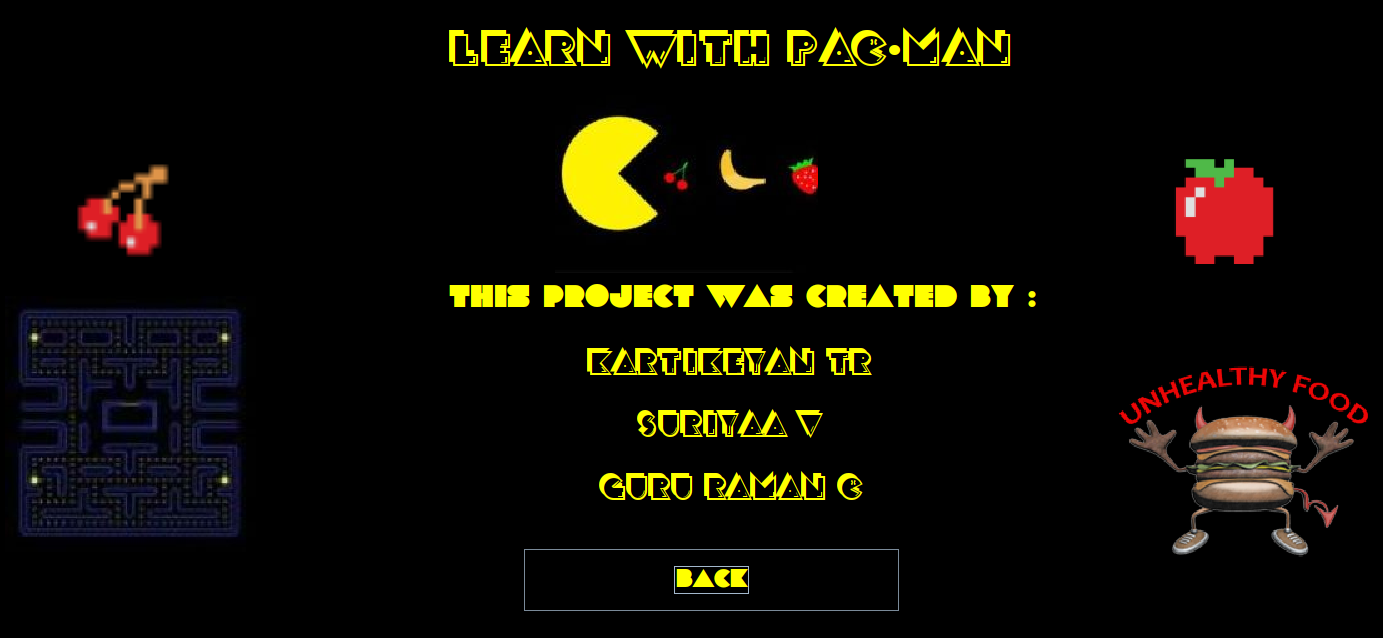
**Level 3**



**Leaderboard**



**About Page**



**RESULT**

Thus, the Game works well with different obstacles at different levels and it has low response time. The leaderboard helps to compare between different players. Switching between different levels makes the game harder.

**CONCLUSION**

The game has different characters and different levels that makes it attractive to the user and it has a leaderboard to track the user’s performance every time and has low response time making it user friendly. It also helps players realize that eating junk food is harmful.