Game Project

จัดทำโดย

6504062630235 นางสาวมัญชุธร ใจตรง

เสนอ

ผู้ช่วยศาสตราจารย์ สถิตย์ ประสมพันธ์

วิชา Object Oriented Programming

คณะวิทยาศาสตร์ประยุกต์ สาขาวิทยาการคอมพิมเตอร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2566 ภาคเรียนที่ 1

สารบัญ

เรื่อง ..... หน้า

บทที่ 1 บทนำ 1

1.1 ที่มาและความสำคัญของโครงการ 1

1.2 ประเภทโครงงาน 1

1.3 ประโยชน์ 1

1.4 ขอบเขตของโครงการ 1

บทที่ 2 ส่วนการพัฒนา 2

2.1 วิธีการเล่น 2

2.2 Class Diagram 3

2.3 หน้าจอ GUI 12

2.4 Event Handling 12

2.5 อัลกอริทึมที่สำคัญ 12

บทที่ 3 สรุป 15

3.1 ปัญหาที่พบในระหว่างการพัฒนา 15

3.2 จุดเด่นของโปรแกรม 15

3.3 คำแนะนำสำหรับผู้สอนที่อยากให้อธิบาย หรือที่เรียนแล้วไม่เข้า หรืออยากให้เพิ่มสำหรับน้องรุ่นต่อไป 15

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงการ

โครงงานนี้เกิดขึ้นจากความรู้ที่ได้จากการเรียนในห้องเรียนและการนำมาประยุกต์ใช้ในการสร้างเกม เพื่อพัฒนาและต่อยอดความรู้ที่ได้จากการเรียนในห้องเรียน

1.2 ประเภทโครงงาน

โครงงานโปรแกรมรูปแบบเกม

1.3 ประโยชน์

1.3.1 เพื่อนำความรู้ที่ได้จากห้องเรียนมาพัฒนาต่อ

1.3.2 เพื่อนำความรู้มาประยุกต์ใช้ในการทำเกม

1.3.3 เพื่อฝึกการวางแผนให้เป็นระบบ

1.4 ขอบเขตของโครงการ

ตารางแผนงานการทำงาน

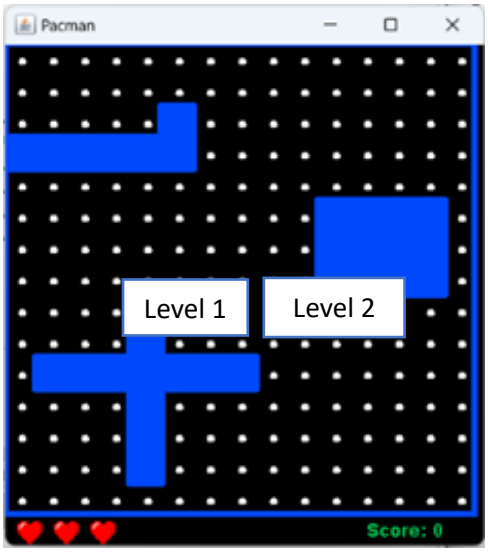
ลำดับ	รายการ	5 - 9	10-14	15-21	22-25	26-29
1	ทำกราฟิกตัวละครและออกแบบ					
2	ศึกษาวิธรการเขียนโปรแกรม					
3	ลงมือเขียนโปรแกรม					
4	จัดทำเอกสาร					
5	ตรวจสอบและแก้ไขข้อผิดพลาด					

บทที่ 2

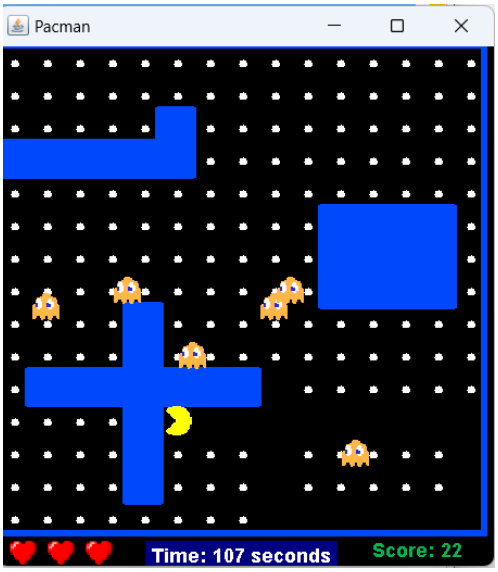
ส่วนการพัฒนา

2.1 วิธีการเล่น

Step 1 : Click ปุ่ม Level ให้เลือก



Step 2: ใช้ปุ่ม A , S , W , D ในการเคลื่อนที่ตัว Pacman เพื่อเก็บเหรียญ



2.2 Class Diagram

คลาส Model เป็นคลาสหลักที่ใช้สำหรับการควบคุมและแสดงหน้าต่างเกมของ Pac-Man และสร้างหน้าต่าง GUI สำหรับเล่นเกม Pacman คลาสนี้มีรายละเอียดต่าง ๆ ดังนี้

1. เมธอด Constructor (Model()): สร้างคอนสตรักเตอร์สำหรับคลาส Model ซึ่งเป็นที่ตั้งสำหรับเริ่มต้นเกมและการกำหนดค่าเริ่มต้นอื่น ๆ เช่น ชีวิตของผู้เล่น คะแนน และเวลาเริ่มต้น. โดยในคอนสตรักเตอร์นี้มีการสร้างปุ่ม "Level 1" และ "Level 2" สำหรับเลือกระดับเกม, และเรียกเมธอด initGame() เพื่อเตรียมค่าเริ่มต้น.

```
        setFocusable(focusable:true);

        JButton level1Button = new JButton(text: "Level 1");
        JButton level2Button = new JButton(text: "Level 2");
        // Level 1
        level1Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                currentLevel = 1;
                startGame();
                level1Button.setVisible(aFlag: false);
                level2Button.setVisible(aFlag: false);
            }
        });
        // Level 2
        level2Button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                currentLevel = 2;
                startGame();
                level1Button.setVisible(aFlag: false);
                level2Button.setVisible(aFlag: false);
            }
        });
        // Arrange buttons
        JPanel buttonPanel = new JPanel();
        buttonPanel.add(comp: level1Button);
        buttonPanel.add(comp: level2Button);
        add(comp: buttonPanel);
```

```
        // create Layout Manager
        setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(top:10, left: 10, bottom:10, right: 10);
        // Set the position and size of the button.
        c.gridx = 0;
        c.gridy = 0;
        c.gridwidth = 1;
        c.fill = GridBagConstraints.HORIZONTAL;
        add(comp: level1Button, constraints: c);
        c.gridx = 1;
        c.gridy = 0;
        c.gridwidth = 1;
        c.fill = GridBagConstraints.HORIZONTAL;
        add(comp: level2Button, constraints: c);
        initGame();
        // Initialize the timer with 2 minutes (120 seconds)
        timeLeft = 120;
        // Create a Timer with a 1-second delay that counts down the time
        countdownTimer = new Timer(delay: 1000, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                timeLeft--;

                if (timeLeft < 0) {
                    timeLeft = 0;
                    // Handle game over or timeout logic here
                }
            }
        });
        // Start the countdown timer
        countdownTimer.start();
```

2. เมธอด startGame(): ใช้สำหรับเริ่มเกมใหม่โดยกำหนดค่าเริ่มต้นของชีวิต, คะแนน, และเลเวลของเกมและเริ่มต้นการเคลื่อนที่ของ Pacman และ Ghosts.

```
private void startGame() {
    lives = 3;
    score = 0;
    initLevel();
    N_GHOSTS = 6;
    currentSpeed = 3;
    inGame = true;
    timer.restart();
}
```

3. เมธอด loadImages(): โหลดรูปภาพต่าง ๆ จากไฟล์ภาพบนดิสก์ สำหรับ Pacman, Ghosts, และของเล่นอื่น ๆ ที่ใช้ในเกม.

```
private void loadImages(){
    down = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\down.gif").getImage();
    up = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\up.gif").getImage();
    left = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\left.gif").getImage();
    right = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\right.gif").getImage();
    ghost = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\ghost.gif").getImage();
    heart = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\heart.png").getImage();
}
```

4. เมธอด initVariables(): กำหนดค่าเริ่มต้นในตัวแปรต่าง ๆ ที่ใช้ในเกม เช่น ตำแหน่งของ Pacman และ Ghosts, ความเร็ว, และตัวแปรอื่น ๆ.

```
private void initVariables() {
screenData = new short[N_BLOCKS * N_BLOCKS];
d = new Dimension(width: 400, height: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(delay: 40, listener: this);
```

5. เมธอด playGame(): ใช้สำหรับการเล่นเกม Pacman โดยรวมการเคลื่อนที่ของ Pacman และ Ghosts, ตรวจสอบเงื่อนไขของเกม, และแสดงเวลาที่เหลืออยู่บนหน้าจอ.

```
private void playGame(Graphics2D g2d) {
    if (dying) {
        death();
    } else {
        movePacman();
        drawPacman(g2d);
        moveGhosts(g2d);
        checkMaze();
    }
    // Calculate the width of the time string to center it
    FontMetrics fontMetrics = g2d.getFontMetrics(f: smallFont);
    int timeStringWidth = fontMetrics.stringWidth("Time: " + timeLeft + " seconds");
    // Calculate the x-coordinate for centering the time string
    int centerX = (SCREEN_SIZE - timeStringWidth) / 2;
    // Create a blue rectangle background for the time
    g2d.setColor(new Color(r: 0, g: 0, b: 255, a: 128)); // Blue color with alpha transparency
    g2d.fillRect(centerX - 5, SCREEN_SIZE + 5, timeStringWidth + 10, height:25);
    // Draw the time string on top of the blue rectangle
    g2d.setColor(new Color(r: 255, g: 255, b: 255));
    g2d.drawString("Time: " + timeLeft + " seconds", x: centerX, SCREEN_SIZE + 20);
    // Display the score and heart images as you were doing before
    drawScore(g: g2d);
}
```

6. เมธอด isGameOver(): ใช้สำหรับตรวจสอบว่าเกมจบหรือยัง ถ้าเวลาหมดลง 0 แล้วจะถือว่าเกมจบ.

```
private boolean isGameOver() {
    return timeLeft <= 0;
}
```

7. เมธอด showIntroScreen(): แสดงหน้าจอเริ่มต้นของเกม ซึ่งแสดงข้อความ "Press SPACE to start".

```
private void showIntroScreen(Graphics2D g2d) {

    //String start = "Press SPACE to start";
    String start = "";
    g2d.setColor(c: Color.yellow);
    g2d.drawString(str: start, (SCREEN_SIZE)/4, y: 150);
}
```

8. เมธอด drawScore(): ใช้สำหรับการแสดงคะแนนและหัวใจที่แสดงจำนวนชีวิตที่เหลือของผู้เล่น.

```
private void drawScore(Graphics2D g) {
    g.setFont(font: smallFont);
    g.setColor(new Color(r: 5, g: 181, b: 79));
    String s = "Score: " + score;
    g.drawString(str: s, SCREEN_SIZE / 2 + 96, SCREEN_SIZE + 16);

    for (int i = 0; i < lives; i++) {
        g.drawImage(img: heart, i * 28 + 8, SCREEN_SIZE + 1, observer: this);
    }
}
```

9. เมธอด checkMaze(): ใช้สำหรับตรวจสอบว่า Pacman ได้กินจุด (dots) หมดหรือยัง ถ้ากินจุดหมดแล้วจะนับคะแนน, เพิ่มความเร็วของ Ghosts, และเริ่มระดับใหม่.

```
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();
    }
}
```

10. เมธอด death(): ใช้สำหรับการจัดการเมื่อ Pac-Man ตายโดยลดจำนวนชีวิตของผู้เล่นและดำเนินการตามความเหมาะสม หากจำนวนชีวิตเหลือเป็น 0 จะจบเกม.

```
private void death() {
    lives--;
    if (lives == 0) {
        inGame = false;
    }
    continueLevel();
}
```

11. เมธอด moveGhosts(): ใช้สำหรับควบคุมเคลื่อนที่ของ Ghosts และตรวจสอบการชนกับ Pac-Man.

```
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;
```

12. เมธอด drawGhost(): ใช้สำหรับวาดรูปภาพของ Ghosts บนหน้าจอ.

```
private void drawGhost(Graphics2D g2d, int x, int y) {
    g2d.drawImage(img:ghost, x, y, observer: this);
    }
```

13. เมธอด movePacman(): ใช้สำหรับควบคุมเคลื่อนที่ของ Pac-Man ซึ่งรวมการตรวจสอบขอบเขตของ Maze และการแจ้งเหตุการณ์เมื่อ Pac-Man กินจุด.

```
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;
}
```

14. เมธอด drawPacman(): ใช้สำหรับวาดรูปภาพของ Pac-Man บนหน้าจอตามทิศทางการเคลื่อนที่.

```
private void drawPacman(Graphics2D g2d) {

    if (req_dx == -1) {
            g2d.drawImage(img:left, pacman_x + 1, pacman_y + 1, observer: this);
    } else if (req_dx == 1) {
            g2d.drawImage(img:right, pacman_x + 1, pacman_y + 1, observer: this);
    } else if (req_dy == -1) {
            g2d.drawImage(img:up, pacman_x + 1, pacman_y + 1, observer: this);
    } else {
            g2d.drawImage(img:down, pacman_x + 1, pacman_y + 1, observer: this);
    }
}
```

15. เมธอด drawMaze(): ใช้สำหรับวาดขอบเขตและจุด (dots) ใน Maze บนหน้าจอ.

```
private void drawMaze(Graphics2D g2d) {
    short i = 0;
    int x, y;
    for (y = 0; y < SCREEN_SIZE; y += BLOCK_SIZE) {
        for (x = 0; x < SCREEN_SIZE; x += BLOCK_SIZE) {
            g2d.setColor(new Color(r: 0,g: 72,b: 251));
            g2d.setStroke(new BasicStroke(width: 5));
            if ((levelData[i] == 0)) {
                    g2d.fillRect(x, y, width: BLOCK_SIZE, height:BLOCK_SIZE);
             }
            if ((screenData[i] & 1) != 0) {
                g2d.drawLine(x1: x, y1: y, x2: x, y + BLOCK_SIZE - 1);
            }
            if ((screenData[i] & 2) != 0) {
                g2d.drawLine(x1: x, y1: y, x + BLOCK_SIZE - 1, y2: y);
            }
            if ((screenData[i] & 4) != 0) {
                g2d.drawLine(x + BLOCK_SIZE - 1, y1: y, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }
            if ((screenData[i] & 8) != 0) {
                g2d.drawLine(x1: x, y + BLOCK_SIZE - 1, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }
            if ((screenData[i] & 16) != 0) {
                g2d.setColor(new Color(r: 255,g: 255,b: 255));
                g2d.fillOval(x + 10, y + 10, width: 6, height:6);
            }
            i++;
```

16. เมธอด initGame(): ใช้สำหรับการเริ่มเกมใหม่โดยกำหนดค่าเริ่มต้นของชีวิตและคะแนน.

```
private void initGame() {

    lives = 3;
    score = 0;
    initLevel();
    N_GHOSTS = 6;
    currentSpeed = 3;
}
```

17. เมธอด initLevel(): ใช้สำหรับเริ่มระดับเกมใหม่โดยรีเซ็ตค่าเริ่มต้นของ Maze และ Ghosts.

```
private void initLevel() {

    int i;
    for (i = 0; i < N_BLOCKS * N_BLOCKS; i++) {
        screenData[i] = levelData[i];
    }

    continueLevel();
}
```

18. เมธอด continueLevel(): ใช้สำหรับเริ่มระดับเกมใหม่หลังจากเกมก่อนหน้าเสร็จสิ้น โดยกำหนดตำแหน่งและความเร็วของ Ghosts ใหม่.

```
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;
    // Increase ghost speed in stage 1
    if (currentLevel == 1) {
        ghostSpeed[i] = 2; // speed in stage 1
    } else{
    // Increase ghost speed in stage 2
    if (currentLevel == 2) {
        ghostSpeed[i] = 6; // speed in stage 2
    } else {
        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;
}
```

19. เมธอด paintComponent(): ใช้สำหรับวาดกราฟิกบนหน้าต่างเกม และแสดงข้อมูลเกมทั้งหมดเช่น Maze, Pac-Man, Ghosts, คะแนน, และเวลา.

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g;
    g2d.setColor(c: Color.black);
    g2d.fillRect(x: 0, y: 0, width: d.width, height:d.height);
    drawMaze(g2d);
    drawScore(g: g2d);
    if (inGame) {
        playGame(g2d);
    } else {
        showIntroScreen(g2d);
    }
    Toolkit.getDefaultToolkit().sync();
    g2d.dispose();
}
```

20. คลาส TAdapter: เป็นคลาสภายในซึ่งเป็น KeyAdapter และใช้สำหรับการจัดการกับเหตุการณ์การกดปุ่มบนคีย์บอร์ด เช่น การควบคุมเคลื่อนที่ของ Pac-Man.

```
class TAdapter extends KeyAdapter {
    @Override
    public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode();
        if (inGame) {
            if (key == KeyEvent.VK_A) {
                req_dx = -1;
                req_dy = 0;
            } else if (key == KeyEvent.VK_D) {
                req_dx = 1;
                req_dy = 0;
            } else if (key == KeyEvent.VK_W) {
                req_dx = 0;
                req_dy = -1;
            } else if (key == KeyEvent.VK_S) {
                req_dx = 0;
                req_dy = 1;
            } else if (key == KeyEvent.VK_ESCAPE && timer.isRunning()) {
                inGame = false;
```

การสร้างเกม Pac-Man ซึ่งเป็นเกมแบบ Applet โดยมีการสร้างคลาส Model ซึ่งเป็นคลาสที่ Extends มาจาก JPanel และทำการ Override Method paintComponent เพื่อวาดกราฟิกของเกมบน GUI

1. **Constructor:** ส่วนที่ถูกเรียกเมื่อคลาส Model ถูกสร้างขึ้น และใน constructor นี้มีการเรียกเมธอด loadImages และ initVariables และกำหนดปุ่ม "Level 1" และ "Level 2" สำหรับเริ่มเกมในระดับต่าง ๆ ให้กับ Applet.

2. **Encapsulation:** การป้องกันการเข้าถึงตัวแปรโดยตรงด้วยการประกาศตัวแปรในคลาส Model ให้เป็น private และใช้ Getters และ Setters เพื่ออ่านและกำหนดค่าของตัวแปรนี้ ในตัวเกมนี้ เราสามารถเห็นการใช้ตัวแปร private อย่างเช่น lives, score, inGame และอื่น ๆ โดยใช้เมธอด Getters และ Setters สำหรับการเข้าถึงและกำหนดค่า เป็นตัวอย่างCodeแค่บางส่วน

```
    private Dimension d;
private final Font smallFont = new Font(name: "Arial", style: Font.BOLD, size: 14);
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 = 6;
private int lives, score;
private int[] dx, dy;
private int[] ghost_x, ghost_y, ghost_dx, ghost_dy, ghostSpeed;

private Image heart, ghost;
private Image up, down, left, right;
```

3. **Composition:** มีการใช้คลาส ImageIcon เพื่อโหลดรูปภาพที่ใช้ในเกม และส่วนอื่น ๆ ของโปรแกรม ในเกมนี้ ใช้ ImageIcon ในการโหลดรูปภาพของ Pacman, Ghosts, heart นำมาใช้ในการวาดบน GUI.

```
private void loadImages(){
    down = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\down.gif").getImage();
    up = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\up.gif").getImage();
    left = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\left.gif").getImage();
    right = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\right.gif").getImage();
    ghost = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\ghost.gif").getImage();
    heart = new ImageIcon(filename: "C:\\Users\\MJAm\\Desktop\\character\\heart.png").getImage();
}
```

4. **Polymorphism:** เกมนี้ไม่ได้ใช้ Polymorphism
5. **Abstract:** เกมนี้ไม่มีการใช้ Abstract class หรือ Abstract method ที่ถูกประกาศ

6. **Inheritance:** ในเกมนี้ คลาส Model Extend จากคลาส JPanel และนำไปใช้ในการสร้าง Applet ที่ใช้ในการเล่นเกม Pacman.

```
public class Model extends JPanel implements ActionListener {

    private Dimension d;
    private final Font smallFont = new Font(name: "Arial", style: Font.BOLD, size: 14);
    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 = 6;
    private int lives, score;
    private int[] dx, dy;
    private int[] ghost_x, ghost_y, ghost_dx, ghost_dy, ghostSpeed;

    private Image heart, ghost;
    private Image up, down, left, right;
```

2.3 หน้าจอ GUI

แนวคิดหลักของ GUI ในเกม Pac-Man คือการสร้างหน้าจอที่มีพื้นหลังสีดำ และใช้กราฟิกสีในการวาดสิ่งต่าง ๆ บนหน้าจอ แสดงด่านของเกมและคะแนน รูปของ Pac-Man และ Ghosts, และหัวใจที่แสดงจำนวนชีวิตที่เหลือ

2.4 Event Handling

ในเกมนี้ มีการใช้ **KeyAdapter** และฟังก์ชัน **keyPressed** เพื่อจัดการกับการกดปุ่มบนคีย์บอร์ด เพื่อเปลี่ยนทิศทางการเคลื่อนที่ของ Pacman นอกจากนี้ ยังมีการใช้ **ActionListener** เพื่อจัดการกับการทำงานของเกมที่เกิดขึ้นในแต่ละรอบเวลา

2.5 อัลกอริทึมที่สำคัญ

1. **initVariables()**:

ฟังก์ชันนี้ใช้ในการกำหนดค่าตัวแปรและสร้างอ็อบเจกต์เริ่มต้น เช่น อ็อบเจกต์ต่าง ๆ ที่ใช้ในเกม เช่น ตำแหน่งแต่ละองค์ประกอบในเกมและตำแหน่งของแง่ม.

2. **playGame(Graphics2D g2d)**:

ฟังก์ชันนี้ควบคุมกระบวนการที่เกิดขึ้นในเกม รวมถึงการเคลื่อนที่ของผู้เล่นและสร้างภาพของแง่ม และตรวจสอบว่าผู้เล่นเข้าสู่สภาพตายหรือไม่.

3. **drawMaze(Graphics2D g2d)**:

ฟังก์ชันนี้ใช้ในการวาดข้อมูลของแผนที่เกม Pac-Man, รวมถึงผนังและลูกศรที่บอกทิศทาง.

4. **initGame()**:

ฟังก์ชันนี้ใช้ในการเริ่มเกม ค่าตัวแปรสำคัญเช่น จำนวนชีวิต และคะแนนจะถูกตั้งค่าเป็นค่าเริ่มต้นและเกมจะถูกเริ่มใหม่.

5. **initLevel()**:

ฟังก์ชันนี้ใช้ในการกำหนดระดับเริ่มต้นของเกม, รวมถึงตำแหน่งเริ่มต้นของผู้เล่นและแง่ม.

6. **movePacman()**:

ฟังก์ชันนี้ควบคุมการเคลื่อนที่ของผู้เล่น Pac-Man และตรวจสอบว่าผู้เล่นกินแต้มหรือชนขอบของผนัง.

7. **drawPacman(Graphics2D g2d)**:

ฟังก์ชันนี้ใช้ในการวาด Pac-Man บนหน้าจอตามทิศทางที่ผู้เล่นกำหนด.

8. **moveGhosts(Graphics2D g2d)**:

ฟังก์ชันนี้ควบคุมการเคลื่อนที่ของแง่มและตรวจสอบว่าผู้เล่นชนกับแง่มหรือไม่.

9. **drawGhost(Graphics2D g2d, int x, int y)**:

ฟังก์ชันนี้ใช้ในการวาดแง่มบนหน้าจอ.

10. **checkMaze()**:

ฟังก์ชันนี้ใช้ในการตรวจสอบว่าผู้เล่นได้เคลื่อนที่ทุกจุดในแผนที่และได้กินแต้มหมดหรือไม่.

11. **death()**:

ฟังก์ชันนี้ใช้ในกรณีที่ผู้เล่นต้องเสียชีวิต จะลดจำนวนชีวิตอีก 1 ชีวิต และตรวจสอบว่าเกมจบหรือไม่.

12. **showIntroScreen(Graphics2D g2d)**:

ฟังก์ชันนี้ใช้ในการแสดงหน้าจอเริ่มต้นของเกม และรอผู้เล่นกดปุ่มเพื่อเริ่มเล่น.

13. **paintComponent(Graphics g)**:

ฟังก์ชันนี้ใช้ในการวาดสิ่งต่าง ๆ บนหน้าจอ รวมถึงแผนที่, ผู้เล่น, แง่ม และคะแนน.

บทที่ 3

สรุป

3.1 ปัญหาที่พบในระหว่างการพัฒนา

1.ตัวเกมที่ออกมายังไม่เป็นไปตามที่คาดหวัง

2.ใช้เวลานานกว่าแผนที่กำหนดไว้

3.2 จุดเด่นของโปรแกรม

เป็นเกมที่เข้าใจง่าย แต่สามารถผ่านได้ยาก

3.3 คำแนะนำสำหรับผู้สอนที่อยากให้อธิบาย หรือที่เรียนแล้วไม่เข้า หรืออยากให้เพิ่มสำหรับน้องรุ่นต่อไป

การสอนในบางคาบอาจจะสอนเร็วไป ตัว lab มีความยากกว่าที่สอนในบางข้อ เพราะบางคนอาจจะยังไม่มีพื้นฐานที่เน้นมากพอ ทำให้ตามไม่ทัน