Object Oriented Programming
Final Project
Pacman

Chellshe Love Simrochelle

Major: Computer Science

Course Code: COMP6699001

Student ID: 2502043040

Class: L2AC

Due date: 10th of June 2022 | Submission date: 9th June 2022

Presented to: Mr Jude Joseph Lamug Martinez

Plagiarism & Cheating

BINUS International Univerisity seriously regards all forms of plagiarism, cheating and collusion as academic offences which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BINUS International University's terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

Chellshe Love Simorchelle

Table of Contents

Project Specification

Pac-Man is a classic and enormously popular Japanese maze video game created in the 1980s. The yellow, pie-shaped Pac-Man character, travels around a maze trying to eat Pac-Man dots and avoid four hunting ghosts. After days of being in a dilemma of not knowing what to make for this final project, I decided to recreate my own version of Pac-Man using the Java programming language. I inputted a simple maze, a Pac-Man, the Pac-Man dots for it to eat and the ghosts to chase the Pac-Man. As a kid, playing Pac-Man remains in my core memory, creating my own take on Pac-Man reminded me of the past thrill and excitement every time I am playing the game with my friends or family.

Input

- The user uses the arrow keys to control the Pac-Man's directions (up, down, left and right).

Output

- the Pac-Man moves in their programmed restricted directions (up, down, left, right) and eats the Pac-Man dots.

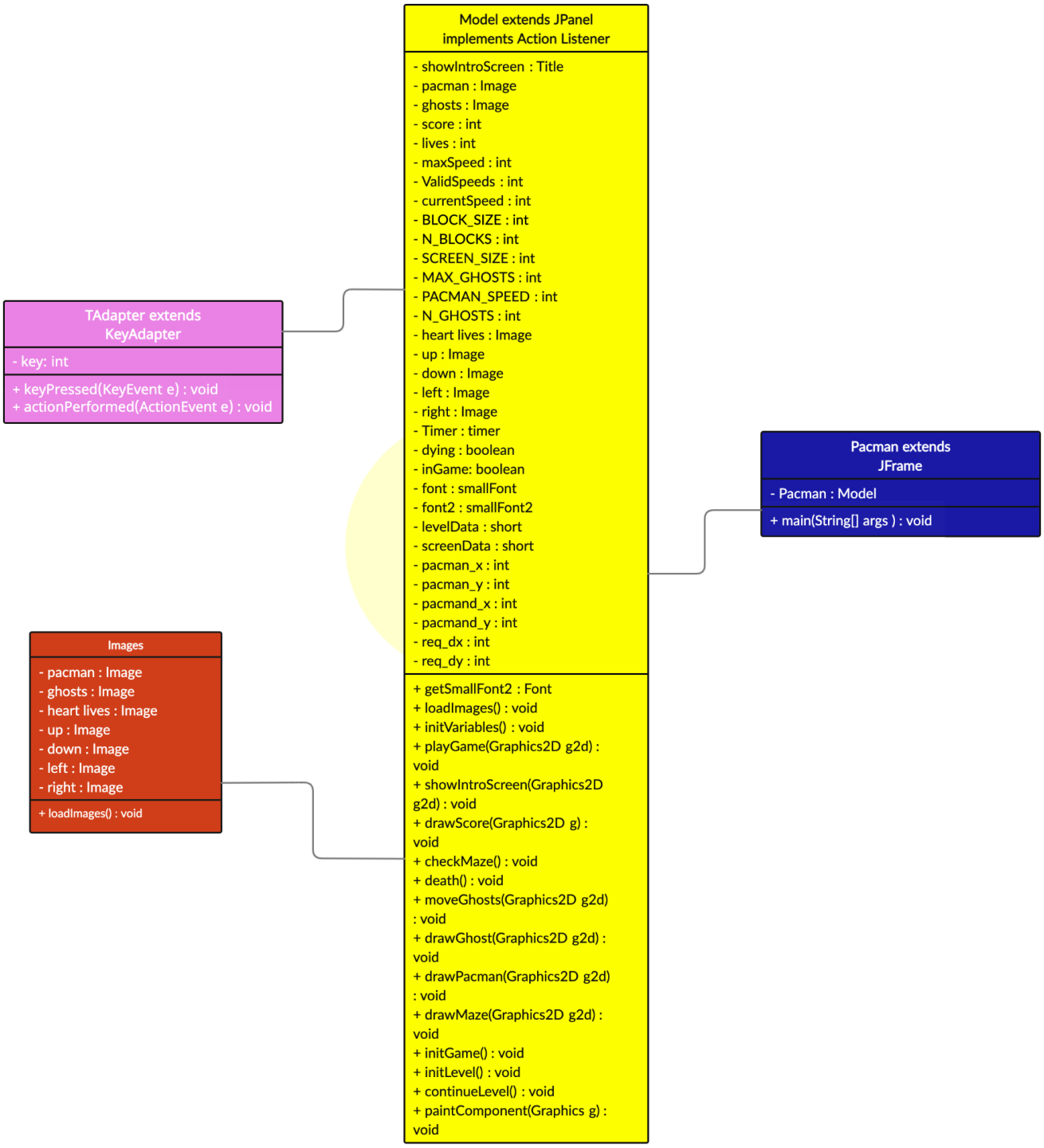
Solution Design

- My version of the classic Pac-Man game, made with JAVA programming language.

Class Diagram

- Parent class: Model, Pacman, module-info and TAdapter.
- Child class for Model and TAdapter: showIntroScreen, pacman, ghosts, score, lives, maxSpeed, ValidSpeeds, currentSpeed, BLOCK_SIZE, N_BLOCKS, SCREEN_SIZE, MAX_GHOSTS, PACMAN_SPEED, N_GHOSTS, heart lives, up, down, left, right, Timer, dying, inGame, font, font2, levelData, screenData, pacman_x, pacman_y, pacmand_x, pacmand_y, req_dx, req_dy, key.
- Chid class for Pacman: Pacman.

UML Diagram for Pac-Man



Hardware

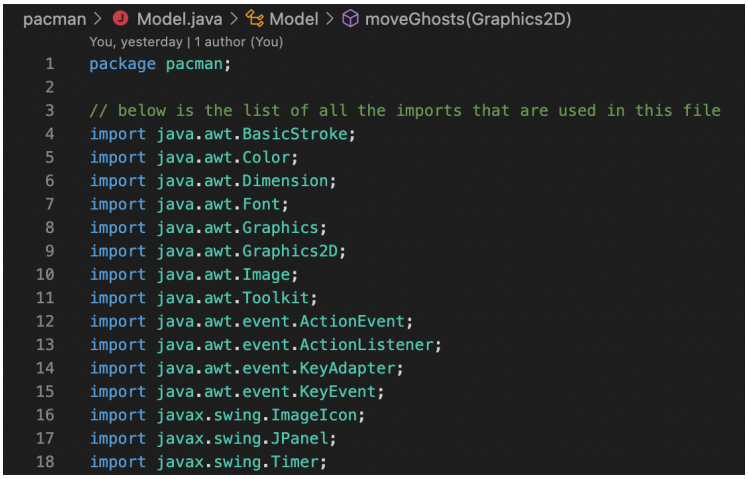
- MacBook Pro (13-inch, M1, 2020), macOS Monterey version 12.3.1
- Memory 8GB
- Apple M1 Chip
- 256GB SSD
- 2560 × 1600 screen resolution
- JDK 18

Software

- Visual Studio Code
- JAVA (programming language)

Implementation & Explanation of Code

For this project, I used a lot of packages that I have never heard of in the first place. I had to figure out what each package does in order to know which one to use and implement in my code respectively. I did not import all the packages in the very beginning. Rather, every time I think there needs to be a package imported, I will only then add it to the top with the rest while I am coding. There are a total of 463 lines on the Model.java file, 23 lines on the Pacman.java file and 3 lines on the module-info.java file which I will be explaining further in this report.



I started my code for my Model.java file with a public class for Model which extends JPanel and implements ActionListener, which will contain all the functions and prompts that I am going to use throughout this file. JPanel is a class that is used to draw on the screen and create a window and ActionListener is a class that is used to listen for events so that the game reacts when I am pressing the respective programmed buttons. Starting from the first section, dimension describes the height and width of the window or playing field. The font is used to display the text in the game with your chosen font and size. inGame checks if the game is running or not and dying checks if Pacman is alive or not. In the second section, BLOCK_SIZE describes the size of the blocks in the game, N_BLOCKS indicates the number of blocks in the game, SCREEN_SIZE is the size of the number of blocks and block size, MAX_GHOSTS is the maximum number of the ghosts in the game and PACMAN_SEED is the speed of the Pacman. In the third section, N_GHOSTS is the number of ghosts at the beginning of the game, lives and score is a simple integer countdown for lives and score, dx and dy are the numbers of lives and score of the player, for the position of the ghosts and ghost_x, ghost_y, ghost_dx, ghost_dy and ghostSpeed is used to determine the number and position of the ghosts. Finally the last section, the heart and ghost are images of the lives and the ghosts, up, down, left and right are images for the Pacman animation up, down, left and right, pacman_x, pacman_y, pacmand_x, pacmand_y are the coordinates of the Pacman sprite, the last two variables are the coordinates of the Pacman when it is moving and req_dx and req_dy are the determinants in the TAdapter in the class, these variables are controlled by the arrow keys.

```
public class Model extends JPanel implements ActionListener { // JPanel is a class that is used to draw on the screen and create a

    private Dimension d; //dimension describes the height and width of the window/playing field
    private final Font smallFont = new Font(name: "PacFont Good", Font.BOLD, size: 14); // font is used to dispay the text in the g
    private final Font smallFont2 = new Font(name: "Arial", Font.BOLD, size: 14); // font is used to dispay the text in the game wi
    private boolean inGame = false; // inGame checks if the game is running or not
    private boolean dying = false; // dying checks if pacman is alive or not

    private final int BLOCK_SIZE = 24; // BLOCK_SIZE descirbes the size of the blocks in the game
    private final int N_BLOCKS = 15; // N_BLOCKS indicates the number of blocks in the game
    private final int SCREEN_SIZE = N_BLOCKS * BLOCK_SIZE; // SCREEN_SIZE is the size of the number of blocks and block size
    private final int MAX_GHOSTS = 12; // maximum number of the ghosts in the game
    private final int PACMAN_SPEED = 6; // the speed of the pacman

    private int N_GHOSTS = 5; // the number of ghosts in the begining of the game
    private int lives, score; // simple integer countdowns for lives and score. it is the number of lives and score of the player
    private int[] dx, dy; // for the position of the ghosts
    private int[] ghost_x, ghost_y, ghost_dx, ghost_dy, ghostSpeed; // used to detemine the number and position of the ghosts

    private Image heart, ghost; // images for the heart lives and the ghosts
    private Image up, down, left, right; // images for the pacman animation up, down, left and right

    private int pacman_x, pacman_y, pacmand_x, pacmand_y; // pacman_x and pacman_y are the coordinates of the pacman sprite, the l
    private int req_dx, req_dy; // req_dx and req_dy are the determant in the TAdapter in the class, these variables are controlle
```

Below is the code for the maze where the Pacman and the ghost will roam around and the layout of the level that I wanted, the structure of the maze is that we have 225 numbers which represent 225 possible positions in the game, 15 columns and 15 rows. Each number represents a specific element to be displayed, 0 stands for the blue block obstacle, every 0 is a field, 1 is the left border, 2 is the top border, 4 is the right border and 8 is the bottom border, 16 are the white Pac-Man dots that the Pacman eats, the trick is for you to add the numbers together. For the top left number 19, you need a left border which is 1, a top border which is 2 and a white Pac-Man dot which is 16, adding them all together will result in the number 19. Another implementation for the bottom right number 28, you need a right border which is 4, a bottom border which is 8 and a white Pac-Man dot which is 16, adding them together again will result in the number 28. Some fields in the middle consist of only the number 16 due to the fact that it is just the plain white Pac-Man dot. One thing to keep in mind is that the obstacles which are represented by 0 must be surrounded by a border so the Pacman and ghosts are not able to go through them. At the cross obstacle, we have the number 22, which means you need a top border which is 2, a right border which is 4 and the white Pac-Man dot which is 16 resulting in a total of 22. Below the number 22 is the number 20, you only need the right border which is 4 and the white Pac-Man dot which is 16 and the total is 20.

```
private final short levelData[] = { // the level data
    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, 28
};
```

validSpeeds is an array of the valid speeds for the speed of the game, maxSpeed is the maximum speed the game can reach. The current speed is the speed at the starting point of the game. The array screenData will obtain the data from the level data which will be used to draw the game later on. lastly, the timer allows the animations to happen.

```
private final int validSpeeds[] = {1, 2, 3, 4, 6, 8}; //an array of the valid speeds for the speed
private final int maxSpeed = 6; //the maximum speed

private int currentSpeed = 3; // the current speed of the game
private short[] screenData; // the array screen data will take the data from the level data and will be used to draw the game
private Timer timer; // the timer allows anitmations to happen
```

The public Model is the constructor that should call different functions that are listed in it. loadImages is used to load the images or in my case gifs, initVariables is used to initialize the variables, addkeyListener(new TAdpater()) adds the key listener which will control the game using the arrow keys, setFocusable sets the focus to the game and initGame starts the game.

```
public Model() { // the constructor should call different funtions

    loadImages(); // loads the images
    initVariables(); // all variables are initialized
    addKeyListener(new TAdapter()); // adds the key listener to control the game
    setFocusable(focusable: true); // sets the focus to the game
    initGame(); // starts the game
}
```

The getSmallFont2 function will return the font that I have inputted and previously mentioned above, for the fonts I tried to look for Pacman themed fonts, therefore, it will tie in every component in the game more nicely and I eventually found the right one for my project.

```
public Font getSmallFont2() { // returns the font
    return smallFont2;
}
```

The next thing I did was I loaded and added all the gifs that I am going to be needing to make this Pacman game come alive using the function below which is new ImageIcon(filename).getImage(), the reason I used gifs instead of regular images was that I wanted to reduce the number of pictures I have to use and by using gifs I just need one gif for each Pacman either facing up, down, left or right rather than using 2 for each of them which will double the number of images needed to be loaded thus taking more time as well and using gifs for the ghosts and hearts made the whole game appear more immersive and lively from my personal opinion as the ghosts changes colors and the hearts beat.

```
private void loadImages() { // loads the images
    down = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/down.gif").getImage(); // loads the ima
    up = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/up.gif").getImage(); // loads the image f
    left = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/left.gif").getImage(); // loads the ima
    right = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/right.gif").getImage(); // loads the i
    ghost = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/ghost.gif").getImage(); // loads the i
    heart = new ImageIcon(filename: "/Users/chellshelove/Documents/GitHub/Pacman/images/heart.gif").getImage(); // loads the i

}
```

The initVariable function consists of the prompts that I have mentioned prior but they are a little different as each of them possesses more prompts and numbers to make the game good. The screenData is the array of the level data, d is the dimension of the game, the ghost x is the array of the ghost x, the ghost dx is the array of the ghost dx, the ghost y is the array of the ghost y, the ghost dy is the array of the ghost dy, the ghost speed is the array of the ghost speed, the dx is the array of the dx and the dy is the array of the dy respectively.

```
  private void initVariables() { // initializes all the variables

    screenData = new short[N_BLOCKS * N_BLOCKS]; // the screen data is the array of the level data
    d = new Dimension(width: 400, height: 400); // the dimension of the game
    ghost_x = new int[MAX_GHOSTS]; // the ghost x is the array of the ghost x
    ghost_dx = new int[MAX_GHOSTS]; // the ghost dx is the array of the ghost dx
    ghost_y = new int[MAX_GHOSTS]; // the ghost y is the array of the ghost y
    ghost_dy = new int[MAX_GHOSTS]; // the ghost dy is the array of the ghost dy
    ghostSpeed = new int[MAX_GHOSTS]; // the ghost speed is the array of the ghost speed
    dx = new int[4]; // the dx is the array of the dx
    dy = new int[4]; // the dy is the array of the dy

    timer = new Timer( delay: 50, this); //the timer takes care of the animation because it determines how often the image are
    timer.start(); // starts the timer
}
```

playGame(Graphics2D g2d) function is just a collection of other functions that are called to display the graphics. For the dying function, if the Pacman dies then the death function will be called other than that the movePacman, drawPacman, moveGhosts and the checkMaze function are called which I will be further explaining along in this report.

```
private void playGame(Graphics2D g2d) { // the function playGame is just a collection of other functions that are called to di

    if (dying) { // when pacman dies the death function is called
        death();

    } else {
        movePacman(); // the movePacman function is called
        drawPacman(g2d); // the drawPacman function is called
        moveGhosts(g2d); // the moveGhosts function is called
        checkMaze(); // the checkMaze function is called
    }
}
```

The next step that I did was to create the intro screen for the Pacman game so that the user will know which button to press in order for the game to start going. I used the showIntroScreen(Graphics2D g2d) function to draw it, the g2d.setFont to choose which font I am going to use first because I have two fonts inputted, and the String start to define the text that I would like to be displayed on the screen, the g2d.setColor to set the color of the text to Pacman yellow and finally the g2d.drawString to set where the text will be displayed on the playing field which in this case I displayed the text right in the center of the window.

```
private void showIntroScreen(Graphics2D g2d) { // this function is used to draw the intro screen
    g2d.setFont(smallFont); // sets the font
    String start = "Press SPACE to start"; // define the text that is displayed on the intro screen
    g2d.setColor(Color.yellow); // the color of the text is yellow
    g2d.drawString(start, (int) ((SCREEN_SIZE)/4.5), y: 175); // the text is displayed in the middle of the screen
}
```

I did the same process for the score of the Pacman after it eats the white Pac-Man dots, the differences are instead of (Graphics2D g2d) it is (Graphics2D g), and the font and text displayed are different because I used the second font for the score, the color is set to green and the placement for the score is on the bottom right corner. I also placed the heart lives gif on the bottom left corner using g.drawImage and made it into a loop so that the number of lives will decrease by 1 when the Pacman hits one of the ghosts and when all the lives are gone it will reload to the original amount which is 3 heart lives when the game restarts after the Pacman dies.

```
private void drawScore(Graphics2D g) { // this function is used to draw the score
    g.setFont(smallFont2); // sets the font
    g.setColor(new Color(r: 5, g: 181, b: 79)); // the color of the text is green
    String s = "Score: " + score; // define the text for the score
    g.drawString(s, SCREEN_SIZE / 2 + 100, SCREEN_SIZE + 20); // the text is displayed in the bottom right of the screen

    for (int i = 0; i < lives; i++) { // the loop is used to check how many heart lives the pacman has left and displays them
        g.drawImage(heart, i * 28 + 8, SCREEN_SIZE + 5, this); // the heart is displayed in the bottom left of the screen
    }
}
```

Here I am using the checkMaze function to check if there are any dot points left for the Pacman to eat. The boolean is used to check whether or not the game is finished, the while loop is used to check if there are any dot points left for the Pacman to eat, and the if statement is used when there are dot points left for the Pacman to eat and if so, that means the games is not finished. The next if statement is for when there are no dot points left for the Pacman to eat that means the game is won and the game restarts, however, the score will be increased by 50 points, the ghost and speed will be increased by 1 respectively and the level is initialized.

```
private void checkMaze() { // the function checkMaze checks if there are any dot points left for the pacman to eat

    int i = 0;
    boolean finished = true;

    while (i < N_BLOCKS * N_BLOCKS && finished) {

        if ((screenData[i]) != 0) {
            finished = false;
        }

        i++;
    }

    if (finished) { // if there are no dot points left for the pacman to eat the game is won and the game restarts

        score += 50; // the score is increased by 50

        if (N_GHOSTS < MAX_GHOSTS) { // the ghost is increased by 1
            N_GHOSTS++;
        }

        if (currentSpeed < maxSpeed) { // the speed is increased by 1
            currentSpeed++;
        }

        initLevel(); // the level is initialized
    }
}
```

I used the function death if the Pacman dies therefore one heart life will be deducted and the game will continue on until the Pacman has no more heart lives. If the Pacman has no more heart lives then the game is over and it will restart. Once the game restarts, the Pacman and ghosts will be reverted back to their original starting positions.

```
private void death() { // if the pacman dies then one heart lives is deducted and the game continues until the pacman has no mo

    lives--;

    if (lives == 0) {
        inGame = false;
    }

    continueLevel(); // the pacman and ghosts are put back to their starting positions
}
```

The moveGhosts function allows the ghosts to move automatically, I set the position of all the 6 ghosts using the block size and the number of ghosts, the ghost will move on one square then they will decide if they want to change directions, and the position of the ghost is the ghost_x divided by the block size plus the number of blocks times the ghost_y divided by the block size, then use the border information (1, 2, 4, 8) to determine how the ghosts can move, if the ghost is on a corner then the ghost will not move, for the else, it will determine where the ghost is located, in which position or square. There are 225 positions, the ghost cannot move over the borders, if there are no obstacles on the left and the ghost is already not moving to the right then the ghost will move to the left. Then the ghost will move in the direction that is determined by the count. Next, the ghost will move in the direction that is determined by the ghost_dx and ghost_dy, with the drawGhost function it loads the image of the ghost that is needed to be drawn, if the Pacman touches one of the ghosts then the Pacman loses a heart life until all of it is gone and the game is over.

```
private void moveGhosts(Graphics2D g2d) { // this function allows the ghosts to move automatically

    int pos; // the position of the ghost
    int count; // the count of the ghost

    for (int i = 0; i < N_GHOSTS; i++) { // set the position of all 6 ghosts again using block size and the number of ghosts
        if (ghost_x[i] % BLOCK_SIZE == 0 && ghost_y[i] % BLOCK_SIZE == 0) { // the ghost will move on one square then they will
            pos = ghost_x[i] / BLOCK_SIZE + N_BLOCKS * (int) (ghost_y[i] / BLOCK_SIZE); // the position of the ghost is the gho

            count = 0; // the count is set to 0

            if ((screenData[pos] & 1) == 0 && ghost_dx[i] != 1) { // use the border information (1, 2, 4, 8) to determine how t
                dx[count] = -1;
                dy[count] = 0;
                count++;
            }

            if ((screenData[pos] & 2) == 0 && ghost_dy[i] != 1) { // use the border information (1, 2, 4, 8) to determine how t
                dx[count] = 0;
                dy[count] = -1;
                count++;
            }
  You, 2 days ago • done
            if ((screenData[pos] & 4) == 0 && ghost_dx[i] != -1) { // use the border information (1, 2, 4, 8) to determine how
                dx[count] = 1;
                dy[count] = 0;
                count++;
            }

            if ((screenData[pos] & 8) == 0 && ghost_dy[i] != -1) { // use the border information (1, 2, 4, 8) to determine how
                dx[count] = 0;
                dy[count] = 1;
                count++;
            }

            if (count == 0) {

                if ((screenData[pos] & 15) == 15) { // if the ghost is on a corner then the ghost will not move
                    ghost_dx[i] = 0;
                    ghost_dy[i] = 0;
                } else {
                    ghost_dx[i] = -ghost_dx[i]; // this detemines where the ghost is located, in which position or square
                    ghost_dy[i] = -ghost_dy[i]; // this detemines where the ghost is located, in which position or square
                }
```

```
            } else {       You, 2 days ago • done

                count = (int) (Math.random() * count); // there are 225 positions, the ghost cannot move over the borders, if t

                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); // with the drawGhost function it loads the image of the ghost that is

        if (pacman_x > (ghost_x[i] - 12) && pacman_x < (ghost_x[i] + 12) // if the pacman touches one of the ghosts then the pa
                && pacman_y > (ghost_y[i] - 12) && pacman_y < (ghost_y[i] + 12)
                && inGame) {

            dying = true;
        }
    }
}
```

The drawGhost loads the image of the ghost that is needed to be drawn and with the g2d.drawImage the ghost is drawn.

```
private void drawGhost(Graphics2D g2d, int x, int y) { // this function loads the image of the ghost that is needed to be drawn
    g2d.drawImage(ghost, x, y, this); // the ghost is drawn
    }
```

Now the movePacman function is called, “pos” stands for the position of the Pacman and “ch” stands for the character of the dots that the Pacman eats. Then the starting position of the Pacman is determined, if the character is equal to 16 then the Pacman is able to eat the Pac-Man dot points, if the Pacman is in the field where the character 16 resides, the score will be increased by 1 point. With the req_dx and req_dy prompts, the Pacman is controlled by the arrow keys on your keyboard then it will check if the Pacman is in one of the borders, if it is then the Pacman will not be able to move further in the corresponding direction and it should change courses. The following step would be to check for a standstill and the last step is to adjust the speed of the Pacman accordingly to how fast you want it to be.

```
private void movePacman() { // the movePacman function is called

    int pos; // the pos is the position of the pacman
    short ch; // the ch is the character of the dots that the pacman eats

    if (pacman_x % BLOCK_SIZE == 0 && pacman_y % BLOCK_SIZE == 0) { // the position of the pacman is determined
        pos = pacman_x / BLOCK_SIZE + N_BLOCKS * (int) (pacman_y / BLOCK_SIZE);
        ch = screenData[pos];

        if ((ch & 16) != 0) { // if the character is equal to 16 then the pacman is able to eat the dot
            screenData[pos] = (short) (ch & 15); // if the pacman is in the on the field of 16, the score will increase
            score++; // the score is increased by one
        }

        if (req_dx != 0 || req_dy != 0) { // with req_dx and req_dy the pacman is controlled by the arrow keys
            if (!((req_dx == -1 && req_dy == 0 && (ch & 1) != 0) // checks if the pacman is in one of the borders then the pacm
                    || (req_dx == 1 && req_dy == 0 && (ch & 4) != 0) // checks if the pacman is in one of the borders then the
                    || (req_dx == 0 && req_dy == -1 && (ch & 2) != 0) // checks if the pacman is in one of the borders then the
                    || (req_dx == 0 && req_dy == 1 && (ch & 8) != 0))) { // checks if the pacman is in one of the borders then
                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_x is set to 0
            pacmand_y = 0; // pacmand_y is set to 0
        }
    }
    pacman_x = pacman_x + PACMAN_SPEED * pacmand_x; // the speed is adjusted accordingly
    pacman_y = pacman_y + PACMAN_SPEED * pacmand_y; // the speed is adjusted accordingly
}
```

The drawPacman will check which arrow key is pressed and draws the Pacman accordingly, and of course, the corresponding Pacman image is loaded for the different directions (up, down, left and right).

```
private void drawPacman(Graphics2D g2d) { // the drawPacman function is called

    if (req_dx == -1) { // it checks which arrow key is pressed and draws the pacman accordingly
        g2d.drawImage(left, pacman_x + 1, pacman_y + 1, this); // the corresponding pacman image is loaded for the different
    } else if (req_dx == 1) { // it checks which arrow key is pressed and draws the pacman accordingly
        g2d.drawImage(right, pacman_x + 1, pacman_y + 1, this); // the corresponding pacman image is loaded for the differen
    } else if (req_dy == -1) { // it checks which arrow key is pressed and draws the pacman accordingly
        g2d.drawImage(up, pacman_x + 1, pacman_y + 1, this); // the corresponding pacman image is loaded for the different d
    } else { // it checks which arrow key is pressed and draws the pacman accordingly
        g2d.drawImage(down, pacman_x + 1, pacman_y + 1, this); // the corresponding pacman image is loaded for the different
    }
}
```

In this drawMaze function, the game is drawn just as it is defined above with 225 numbers. i is the position of the Pacman, and the x and y are the coordinates of the dots that are needed to be drawn . It is iterated in two for loops with screen_size and block_size through the whole array, this will draw the x and y-axis of the array. Using g2d.setColor the color of the border and the obstacle blocks are set to blue and using g2d.setStroke the border is set to 5 pixels thick. If one field of the array is 0 it is colored blue, the blue block is then filled and drawn. If it is 1, 2, 4 and 8 the respective blue borders are drawn, if it is 16 then the white Pac-Man dot points are drawn, we need to set the color to white and also fill it in.

```
private void drawMaze(Graphics2D g2d) { // in this function the game is drawn just as it is defined above with 225 numbers

    short i = 0; // the i is the position of the pacman
    int x, y; // the x and y are the coordinates of the dots that are needed to be drawn

    for (y = 0; y < SCREEN_SIZE; y += BLOCK_SIZE) { // it is iterared in two for loops with screen_size and block_size through
        for (x = 0; x < SCREEN_SIZE; x += BLOCK_SIZE) {

            g2d.setColor(new Color(r: 0,g: 72,b: 251)); // the color of the border and blocks are set to blue
            g2d.setStroke(new BasicStroke(width: 5)); // the border is set to 5 pixels thick

            if ((levelData[i] == 0)) { // if one field of the array is 0 it is colored blue
                g2d.fillRect(x, y, BLOCK_SIZE, BLOCK_SIZE); // the blue block is filled and drawn
             }

            if ((screenData[i] & 1) != 0) { // if it is 1 the left border is drawn
                g2d.drawLine(x, y, x, y + BLOCK_SIZE - 1);
            }

            if ((screenData[i] & 2) != 0) { // if it is 2 the top border is drawn
                g2d.drawLine(x, y, x + BLOCK_SIZE - 1, y);
            }

            if ((screenData[i] & 4) != 0) { // if it is 4 the right border is drawn
                g2d.drawLine(x + BLOCK_SIZE - 1, y, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }

            if ((screenData[i] & 8) != 0) { // if it is 8 the bottom border is drawn
                g2d.drawLine(x, y + BLOCK_SIZE - 1, x + BLOCK_SIZE - 1,
                        y + BLOCK_SIZE - 1);
            }

            if ((screenData[i] & 16) != 0) { // if it is 16 the white dot is drawn
                g2d.setColor(new Color(r: 255,g: 255,b: 255)); // the color of the dot is set to white
                g2d.fillOval(x + 10, y + 10, width: 6, height: 6); // the dot is filled and drawn
            }

            i++;
        }
    }
}
```

initGame initializes the game variables, the starting values of the heart lives are 3, the starting value of the score is 0, I initialize the starting level, the starting number of ghosts roaming around in the game and lastly the speed of the ghosts.

```
private void initGame() { // initialize game variables
    lives = 3; // starting values for the lives
    score = 0; // starting values for the score
    initLevel(); // initialize the level
    N_GHOSTS = 5; // define the number of the ghosts
    currentSpeed = 2; // define the speed of the ghosts
}
```

initLevel will of course initialize the level and to initialize the level I created a loop and copied the whole playfield from the array levelData to the new array screenData.

```
private void initLevel() { // initialize the level

    int i;
    for (i = 0; i < N_BLOCKS * N_BLOCKS; i++) { // to initialize the level, create  for loop and copy the whole play field form
        screenData[i] = levelData[i];
    }

    continueLevel();
}
```

The function continueLevel defines the position of the ghosts, it also creates random speed for the ghosts. State the start position of the ghosts and they will move randomly as well as the speed of each ghost will be different, however, the speed may only have a value that is in the array validSpeeds. Then we state the start position of the Pacman, reset the direction move and reset the direction controls using the arrow keys.

```
private void continueLevel() { // the function continueLevel() defines the position of the ghosts, it also creates random speed

    int dx = 1;
    int random;

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

        ghost_y[i] = 4 * BLOCK_SIZE; //start position of the ghosts
        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]; // the speed may only have a value which is in the array validSpeeds
    }

    pacman_x = 7 * BLOCK_SIZE;  //start position of the pacman
    pacman_y = 11 * BLOCK_SIZE;
    pacmand_x = 0;  //reset direction move
    pacmand_y = 0;
    req_dx = 0;     // reset direction controls using the arrow keys
    req_dy = 0;
    dying = false;
}
```

To have the graphics component complete, the function paintComponent is used. Super is used to call the paintComponent method of the parent superclass, the next line is used to create a graphics context for the graphics component. I then set the background color to black and fill it in. Next, I had to draw the maze and score for the game information, the most important component is, of course, the Pacman and ghosts so I drew that as well, finally draw the intro screen and sync the graphics component but keep in mind to dispose of it.

```
public void paintComponent(Graphics g) { // to have the graphics component complete, the funtion paintComponent() is used
    super.paintComponent(g); // to call the paintComponent() method of the parent superclass

    Graphics2D g2d = (Graphics2D) g; // to create a graphics context for the graphics component

    g2d.setColor(Color.black); // to set the color of background
    g2d.fillRect(x: 0, y: 0, d.width, d.height); // to fill the background with the color black

    drawMaze(g2d); // to draw the maze for the game information
    drawScore(g2d); // to draw the score for the game information

    if (inGame) { // to draw the pacman and the ghosts in the game
        playGame(g2d); // to play the game
    } else {
        showIntroScreen(g2d); // to draw the intro screen for the game information
    }

    Toolkit.getDefaultToolkit().sync(); // to sync the graphics component
    g2d.dispose(); // to dispose the graphics context
}
```

Class TAdpater extends KeyAdpater is a function for the arrow keys. If the game is in progress then the Pacman will be controlled by the arrow keys, the variables req_dx and req_dy are used to control the x and y position, if the escape key is pressed while the timer is still running the game should end and if the space key is pressed the game should start. inGame means the game is in progress and the function initGame is called.

```
    You, 2 days ago | 1 author (You)
    //controls
    class TAdapter extends KeyAdapter { // function for the arrow keys

        @Override
        public void keyPressed(KeyEvent e) {

            int key = e.getKeyCode();

            if (inGame) { //if game is in progress then the pacman will be controlled by the arrow keys
                if (key == KeyEvent.VK_LEFT) {
                    req_dx = -1; // the variables req_dx and req_dy are used to control the x and y position
                    req_dy = 0;
                } else if (key == KeyEvent.VK_RIGHT) {
                    req_dx = 1; // the variables req_dx and req_dy are used to control the x and y position
                    req_dy = 0;
                } else if (key == KeyEvent.VK_UP) {
                    req_dx = 0; // the variables req_dx and req_dy are used to control the x and y position
                    req_dy = -1;
                } else if (key == KeyEvent.VK_DOWN) {
                    req_dx = 0; // the variables req_dx and req_dy are used to control the x and y position
                    req_dy = 1;
                } else if (key == KeyEvent.VK_ESCAPE && timer.isRunning()) { // if the escape key is pressed while the timer is run
                    inGame = false; // the game is no longer in progress
                }
            } else {
                if (key == KeyEvent.VK_SPACE) { // if the space key is pressed the game should start
                    inGame = true; // the game is now in progress
                    initGame(); // the function initGame() is called
                }
            }
        }
}
```

@Override is used to override the keyReleased method of the KeyListener interface, actionPerformed(ActionEvent e) is used to define the actionPerformed method of the ActionListener interface and repaint is used to repaint the graphics component.

```
@Override // to override the keyReleased() method of the KeyListener interface
public void actionPerformed(ActionEvent e) { // to define the actionPerformed() method of the ActionListener interface
    repaint(); // to repaint the graphics component      You, now • Uncommitted changes
}

}
```

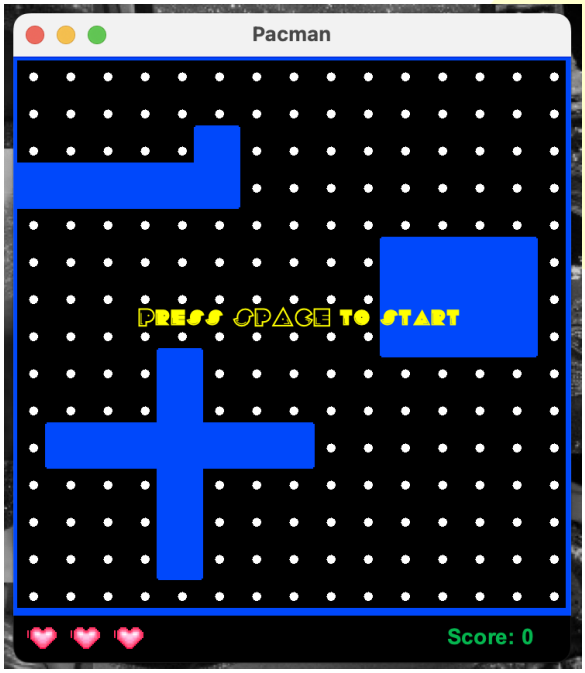
Moving on to my Pacman.java file, I only imported one package this time and I made the public class Pacman which extends to JFrame, I then added the Model.java file to the frame on the Pacman.java file so everything can run smoothly. Next, I created a new Pacman prompt called pac, I set the frame to visible, set the title of the frame in which I named Pacman, set the size of the frame that will pop up on my desktop, and set the default close operation of the frame which is done by pressing the red “X” button on the top left corner of the frame and lastly set the window position on the center of my desktop so that every time I run my code, my Pacman game window will automatically appear in the middle of my screen.

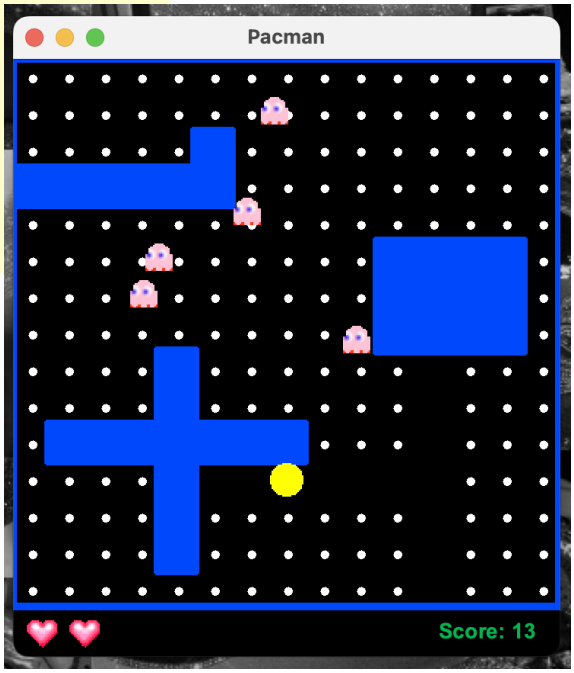
```
pacman > Pacman.java > Pacman > main(String[])
      You, 2 days ago | 1 author (You)
1     package pacman;
2
3     // the imports that is used in this file
4     import javax.swing.JFrame;
5
      You, 2 days ago | 1 author (You)
6     public class Pacman extends JFrame{ // the class that is used in this file with JFrame
7
8         public Pacman() {
9             add(new Model()); // add the model to the frame
10        }
11
12
          Run | Debug
13        public static void main(String[] args) {
14            Pacman pac = new Pacman(); // create a new pacman
15            pac.setVisible(b: true); // set the frame to visible
16            pac.setTitle(title: "Pacman"); // set the title of the frame
17            pac.setSize(width: 360, height: 420); // set the size of the frame      You, 2 days ago • done …
18            pac.setDefaultCloseOperation(EXIT_ON_CLOSE); // set the default close operation of the frame
19            pac.setLocationRelativeTo(c: null); // window position on the center of the screen
20
21        }
22
23    }
```

And last but not least my module-info.java file which contains the module Pacman which connects to both my other files (Model.java and Pacman.java)

```
pacman > module-info.java
        You, 2 days ago | 1 author (You)
1       module Pacman {
2           requires java.desktop;
3       }                You, 2 days ago • done …
```

Proof of Working Program





Reflection & Experiences

While developing this project I learned a lot of new knowledge and techniques to go about the Java programming language, nevertheless, it was still really complicated to put together and execute it into a fully working program. It took me less time to decide on what final project to make this time along as I have had the idea of making a Pacman game for last semester's algorithm and programming class but decided against it as it did not match my skill set at that current time, so I thought to myself, might as well attempt to do it this time around. Moreover, this project has helped me gain a better perspective on various aspects related to this class. I believe that Pacman is something that at least most people around the world have played once in their lifetime, it brought a lot of nostalgic memories of when I first played it when my family introduced it to me. I could play Pacman for hours to get past one difficult level so it kept me occupied when I was young.

Importing some of the packages and coming across errors in the code was really frustrating to me, especially with the ticking time of the due date but without the thought of giving up in mind, I managed to overcome those errors even though sometimes it took me hours to untangle one of each. I got confused at times on how I should go about writing the code because I wanted to make it as understandable as possible to both me and also the lecturers that will be reading and checking my project later on. Youtube and Stack Overflow was a crucial aspect in helping me complete this project as they guided me to figure the code out when I came across bugs, moreover, it also showed me how I can debug them, despite the fact that I had a hard time searching for which one is similar to the problem I have at hand, I could not bear the thought of not having both of those websites helping me at all.

All in all, there were some ups and downs while completing this project, but after countless nights of not sleeping and working on the code day in and day out, I managed to get it done before the due date with a few days left to spare for me to check on minor details, that is the benefit of starting earlier on the project if I could say so, I mentioned in my previous report that I wanted to start earlier on projects and I did just that so I am definitely proud of myself for making it a reality. For my last remarks, making projects can be time-consuming and stressful a lot of the time but with a lot of determination and persistence I feel like everyone will be able to do it and the feeling of finishing it and seeing it run smoothly beats the exhaustion and all I felt was complete and utter joy.

Project Link

Pac-Man OOP Final Project GitHub Link

References

https://www.fontsc.com/ ➜ Pacman Fonts

https://app.creately.com/ ➜ UML Diagram

https://filmora.wondershare.com/ ➜ Video Demo Editor

https://stackoverflow.com/ ➜ Help for errors

https://giphy.com/ ➜ GIFS

https://www.resizepixel.com/ ➜ Resizing Pixels of the GIFS

https://www.freeiconspng.com/ ➜ Image for Pacman Watermark

https://www.canva.com/ ➜ Intro and Outro of the Video Demo

https://www.youtube.com/ ➜ Background Music for the Video Demo

https://www.y2mate.com/ ➜ Youtube Video to MP4 Convertor

https://signaturely.com/ ➜ Online Signature Maker to Sign the Declaration of Originality