UAS PEMROGRAMAN BERORIENTASI OBJEK
MOHAMMAD CAKRA ADISURYA
2222105125
2TI03

FMEMBUAT GAME FLAPPY BIRD

1. Kelas `App`

Kelas ini berfungsi sebagai entry point untuk aplikasi. Berikut adalah detailnya:

```
import javax.swing.*;

public class App {
    public static void main(String[] args) throws Exception {
        int boardWidth = 360;
        int boardHeight = 640;

        JFrame frame = new JFrame("Flappy Bird");
        // frame.setVisible(true);
        frame.setSize(boardWidth, boardHeight);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        FlappyBird flappyBird = new FlappyBird();
        frame.add(flappyBird);
        frame.pack();
        flappyBird.requestFocus();
        frame.setVisible(true);
    }
}
```

Penjelasan:

- **JFrame**: Membuat jendela aplikasi dengan judul "Flappy Bird".
- **Ukuran Jendela**: `frame.setSize(boardWidth, boardHeight)` mengatur ukuran jendela.
- **Posisi Jendela**: `frame.setLocationRelativeTo(null)` memposisikan jendela di tengah layar.
- **Non-Resizable**: `frame.setResizable(false)` mencegah pengguna mengubah ukuran jendela.
- **Default Close Operation**: `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)` mengatur aplikasi agar keluar saat jendela ditutup.
- **Menambahkan Panel**: `frame.add(flappyBird)` menambahkan instance `FlappyBird` ke dalam frame.
- **Menampilkan Jendela**: `frame.setVisible(true)` menampilkan jendela.

2. Kelas `FlappyBird`

Kelas ini mengatur logika permainan, rendering grafis, dan interaksi pengguna. Kelas ini memperluas `JPanel` dan mengimplementasikan `ActionListener` serta `KeyListener`.

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.*;

public class FlappyBird extends JPanel implements ActionListener, KeyListener
{
    int boardWidth = 360;
    int boardHeight = 640;

    //images
    Image backgroundImg;
    Image birdImg;
    Image topPipeImg;
    Image bottomPipeImg;

    //bird class
    int birdX = boardWidth/8;
    int birdY = boardWidth/2;
    int birdWidth = 34;
    int birdHeight = 24;

    class Bird {
        int x = birdX;
        int y = birdY;
        int width = birdWidth;
        int height = birdHeight;
        Image img;

        Bird(Image img) {
            this.img = img;
        }
    }

    //pipe class
    int pipeX = boardWidth;
    int pipeY = 0;
    int pipeWidth = 64;  //scaled by 1/6
    int pipeHeight = 512;

    class Pipe {
```

```
        int x = pipeX;
        int y = pipeY;
        int width = pipeWidth;
        int height = pipeHeight;
        Image img;
        boolean passed = false;

        Pipe(Image img) {
            this.img = img;
        }
    }

    //game logic
    Bird bird;
    int velocityX = -4; //move pipes to the left speed (simulates bird moving
right)
    int velocityY = 0; //move bird up/down speed.
    int gravity = 1;

    ArrayList<Pipe> pipes;
    Random random = new Random();

    Timer gameLoop;
    Timer placePipeTimer;
    boolean gameOver = false;
    double score = 0;

    FlappyBird() {
        setPreferredSize(new Dimension(boardWidth, boardHeight));
        // setBackground(Color.blue);
        setFocusable(true);
        addKeyListener(this);

        //load images
        backgroundImg = new
ImageIcon(getClass().getResource("./flappybirdbg.png")).getImage();
        birdImg = new
ImageIcon(getClass().getResource("./flappybird.png")).getImage();
        topPipeImg = new
ImageIcon(getClass().getResource("./toppipe.png")).getImage();
        bottomPipeImg = new
ImageIcon(getClass().getResource("./bottompipe.png")).getImage();

        //bird
        bird = new Bird(birdImg);
        pipes = new ArrayList<Pipe>();

        //place pipes timer
```

```
        placePipeTimer = new Timer(1500, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
              // Code to be executed
              placePipes();
            }
        });
        placePipeTimer.start();

        //game timer
        gameLoop = new Timer(1000/60, this); //how long it takes to start
timer, milliseconds gone between frames
        gameLoop.start();
    }

    void placePipes() {
        //(0-1) * pipeHeight/2.
        // 0 -> -128 (pipeHeight/4)
        // 1 -> -128 - 256 (pipeHeight/4 - pipeHeight/2) = -3/4 pipeHeight
        int randomPipeY = (int) (pipeY - pipeHeight/4 -
Math.random()*(pipeHeight/2));
        int openingSpace = boardHeight/4;

        Pipe topPipe = new Pipe(topPipeImg);
        topPipe.y = randomPipeY;
        pipes.add(topPipe);

        Pipe bottomPipe = new Pipe(bottomPipeImg);
        bottomPipe.y = topPipe.y  + pipeHeight + openingSpace;
        pipes.add(bottomPipe);
    }


    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        draw(g);
    }

    public void draw(Graphics g) {
        //background
        g.drawImage(backgroundImg, 0, 0, this.boardWidth, this.boardHeight,
null);

        //bird
        g.drawImage(birdImg, bird.x, bird.y, bird.width, bird.height, null);

        //pipes
        for (int i = 0; i < pipes.size(); i++) {
```

```
            Pipe pipe = pipes.get(i);
            g.drawImage(pipe.img, pipe.x, pipe.y, pipe.width, pipe.height,
null);
        }

        //score
        g.setColor(Color.white);

        g.setFont(new Font("Arial", Font.PLAIN, 32));
        if (gameOver) {
            g.drawString("Game Over: " + String.valueOf((int) score), 10, 35);
        }
        else {
            g.drawString(String.valueOf((int) score), 10, 35);
        }

    }

    public void move() {
        //bird
        velocityY += gravity;
        bird.y += velocityY;
        bird.y = Math.max(bird.y, 0); //apply gravity to current bird.y, limit
the bird.y to top of the canvas

        //pipes
        for (int i = 0; i < pipes.size(); i++) {
            Pipe pipe = pipes.get(i);
            pipe.x += velocityX;

            if (!pipe.passed && bird.x > pipe.x + pipe.width) {
                score += 0.5; //0.5 because there are 2 pipes! so 0.5*2 = 1, 1
for each set of pipes
                pipe.passed = true;
            }

            if (collision(bird, pipe)) {
                gameOver = true;
            }
        }

        if (bird.y > boardHeight) {
            gameOver = true;
        }
    }

    boolean collision(Bird a, Pipe b) {
```

```
        return a.x < b.x + b.width &&   //a's top left corner doesn't reach
b's top right corner
               a.x + a.width > b.x &&   //a's top right corner passes b's top
left corner
               a.y < b.y + b.height &&  //a's top left corner doesn't reach
b's bottom left corner
               a.y + a.height > b.y;    //a's bottom left corner passes b's
top left corner
    }

    @Override
    public void actionPerformed(ActionEvent e) { //called every x milliseconds
by gameLoop timer
        move();
        repaint();
        if (gameOver) {
            placePipeTimer.stop();
            gameLoop.stop();
        }
    }

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_SPACE) {
            // System.out.println("JUMP!");
            velocityY = -9;

            if (gameOver) {
                //restart game by resetting conditions
                bird.y = birdY;
                velocityY = 0;
                pipes.clear();
                gameOver = false;
                score = 0;
                gameLoop.start();
                placePipeTimer.start();
            }
        }
    }

    //not needed
    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyReleased(KeyEvent e) {}
}
```

Penjelasan:

- **Ukuran Board**: `boardWidth` dan `boardHeight` mengatur ukuran area permainan.
- **Image Loading**: `ImageIcon` digunakan untuk memuat gambar latar belakang, burung, dan pipa.
- **Bird Class**: Kelas `Bird` digunakan untuk merepresentasikan burung dengan koordinat (x, y), ukuran, dan gambar.
- **Pipe Class**: Kelas `Pipe` digunakan untuk merepresentasikan pipa dengan koordinat (x, y), ukuran, gambar, dan status apakah sudah dilewati burung.
- **Kecepatan dan Gravitasi**: `velocityX`, `velocityY`, dan `gravity` mengatur gerakan burung dan pipa.
- **ArrayList untuk Pipa**: `pipes` digunakan untuk menyimpan daftar pipa.
- **Random**: Objek `random` digunakan untuk menghasilkan koordinat pipa secara acak.
- **Timer**: `gameLoop` dan `placePipeTimer` digunakan untuk mengatur loop permainan dan penempatan pipa secara periodik.
- • **Koordinat Acak**: `randomPipeY` menentukan posisi Y pipa atas secara acak.
- • **Opening Space**: `openingSpace` mengatur jarak antara pipa atas dan bawah.
- • **Menambahkan Pipa**: Pipa atas dan bawah ditambahkan ke daftar `pipes`.
- • **Latar Belakang**: `g.drawImage` menggambar latar belakang.
- • **Burung**: `g.drawImage` menggambar burung.
- • **Pipa**: Loop menggambar semua pipa dalam daftar `pipes`.
- • **Skor**: Menggambar skor pada layar.
- • **Gravitasi**: `velocityY` ditambahkan dengan `gravity` untuk mensimulasikan gravitasi.
- • **Gerakan Burung**: `bird.y` di-update berdasarkan `velocityY`.
- • **Gerakan Pipa**: Pipa bergerak ke kiri berdasarkan `velocityX`.
- • **Skor**: Skor bertambah saat burung melewati pipa.
- • **Deteksi Tabrakan**: Jika burung bertabrakan dengan pipa atau jatuh ke bawah, `gameOver` diatur ke `true`.
- **Logika Tabrakan**: Memeriksa apakah ada bagian dari burung yang bertabrakan dengan pipa.
- • **Gerakan dan Render**: Memanggil `move` untuk meng-update posisi dan `repaint` untuk menggambar ulang.
- • **Menghentikan Timer**: Jika permainan selesai (`gameOver`), timer untuk pipa dan loop permainan dihentikan.
- • **Lompat**: Jika tombol `SPACE` ditekan, `velocityY` diatur ke -9 untuk membuat burung melompat.
- • **Restart Game**: Jika permainan selesai, permainan di-reset dengan mengatur ulang posisi burung, menghapus pipa, dan mengatur skor ke 0.

Kesimpulan

Kode ini adalah implementasi dari game "Flappy Bird" menggunakan Java dan Swing. Kelas `App` mengatur jendela aplikasi, sementara kelas `FlappyBird` mengatur logika permainan, rendering grafis, dan interaksi pengguna. Game ini menggunakan timer untuk menggerakkan objek dan memperbarui tampilan setiap frame, serta mendeteksi tabrakan dan mengatur input keyboard untuk mengontrol burung.