Prodi : Teknik Informatika

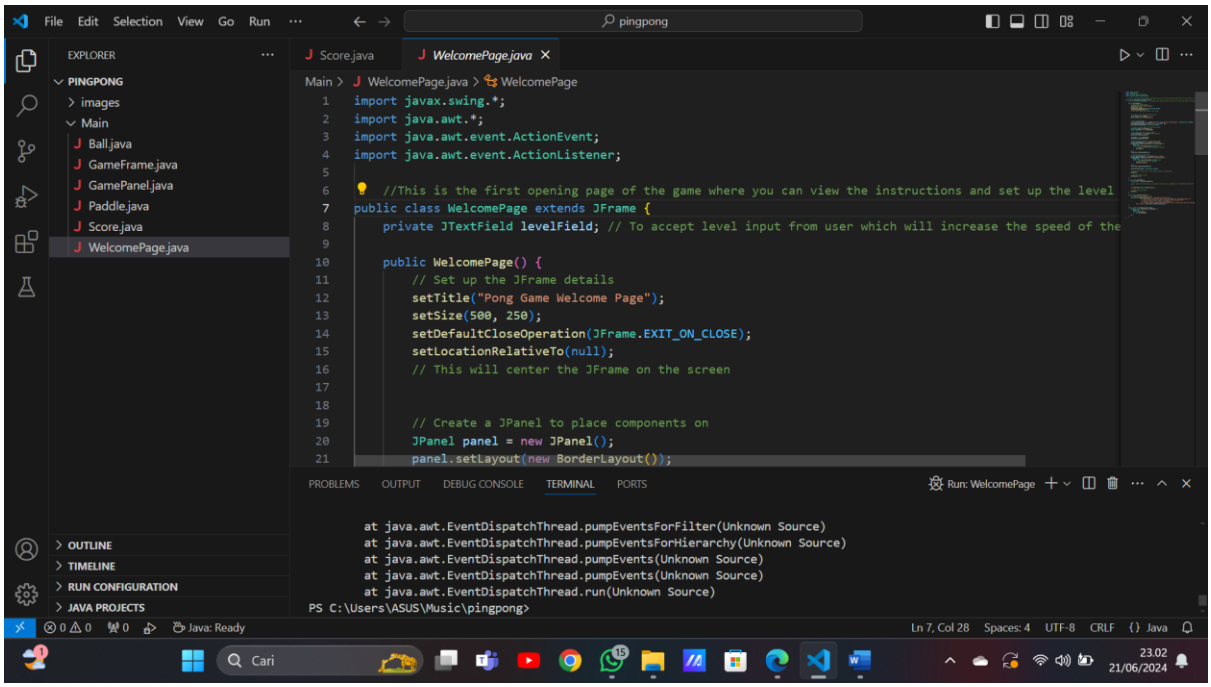
Nama : Muhammad Anggoro Triantomo

Nim : 2222105134

Kelas : 2TI03

Matkul : Pemograman Berorientasi Objek

PINGPONG



1. Ball.java

import java.awt.*;

import java.util.*;

- `import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang berisi kelas-kelas untuk membuat antarmuka pengguna grafis dan menggambar elemen grafis.
- `import java.util.*;`: Mengimpor semua kelas dari paket `java.util`, termasuk kelas `Random` yang digunakan untuk menghasilkan angka acak.

public class Ball extends Rectangle {

Mendefinisikan kelas `Ball` yang merupakan subclass dari `Rectangle`. `Rectangle` adalah kelas dalam `java.awt` yang digunakan untuk merepresentasikan persegi panjang dengan koordinat dan dimensi tertentu.

```
    Random random;
    int vel_x; // velocity with which ball will move horizontally
    int vel_y; // velocity with which ball will move vertically
    int initialSpeed; // speed of the ball in the game which we will take the input
```

- `Random random;`: Instance dari kelas `Random` untuk menghasilkan angka acak.
- `int vel_x;`: Kecepatan horizontal bola.
- `int vel_y;`: Kecepatan vertikal bola.
- `int initialSpeed;`: Kecepatan awal bola yang diberikan sebagai input oleh pengguna.

```
    Ball(int x, int y, int width, int height, int speed) {
        super(x, y, width, height);

        // Initialising the speed of the ball with the level of the game given input by the user
        initialSpeed = speed;
        random = new Random();
        int randomXDirection = random.nextInt(2);
        if(randomXDirection == 0)
            randomXDirection--;
        setXDirection(randomXDirection * initialSpeed);

        int randomYDirection = random.nextInt(2);
        if(randomYDirection == 0)
            randomYDirection--;
        setYDirection(randomYDirection * initialSpeed);
    }
```

- `Ball(int x, int y, int width, int height, int speed)`: Konstruktor untuk kelas `Ball`.
- `super(x, y, width, height);`: Memanggil konstruktor `Rectangle` untuk menginisialisasi posisi `(x, y)` dan ukuran `(width, height)` bola.
- `initialSpeed = speed;`: Menginisialisasi kecepatan awal bola dengan nilai yang diberikan oleh pengguna.
- `random = new Random();`: Membuat instance baru dari `Random`.
- `int randomXDirection = random.nextInt(2);`: Menghasilkan angka acak 0 atau 1 untuk menentukan arah horizontal awal bola.
- `if(randomXDirection == 0) randomXDirection--;`: Jika angka acak adalah 0, ubah menjadi -1 sehingga arah horizontal bisa -1 atau 1.
- `setXDirection(randomXDirection * initialSpeed);`: Mengatur kecepatan horizontal bola berdasarkan arah acak dan kecepatan awal.
- `int randomYDirection = random.nextInt(2);`: Menghasilkan angka acak 0 atau 1 untuk menentukan arah vertikal awal bola.
- `if(randomYDirection == 0) randomYDirection--;`: Jika angka acak adalah 0, ubah menjadi -1 sehingga arah vertikal bisa -1 atau 1.
- `setYDirection(randomYDirection * initialSpeed);`: Mengatur kecepatan vertikal bola berdasarkan arah acak dan kecepatan awal.

```
public void setXDirection(int randomX) {
    vel_x = randomX;
}
```

`public void setXDirection(int randomX)`: Metode untuk mengatur kecepatan horizontal bola (`vel_x`) dengan nilai `randomX`.

```
public void setYDirection(int randomY) {
    vel_y = randomY;
}
```

`public void setYDirection(int randomY)`: Metode untuk mengatur kecepatan vertikal bola (`vel_y`) dengan nilai `randomY`.

```
public void move() {
    x += vel_x;
    y += vel_y;
}
```

`public void move()`: Metode untuk memperbarui posisi bola berdasarkan kecepatannya. Menambahkan `vel_x` ke koordinat `x` dan `vel_y` ke koordinat `y`.

```
    public void draw(Graphics g) {
        g.setColor(Color.black);
        g.fillOval(x, y, height, width);
    }
    public void draw(Graphics g) {
        g.setColor(Color.black);
        g.fillOval(x, y, height, width);
    }
```

- `public void draw(Graphics g)`: Metode untuk menggambar bola pada layar.
- `g.setColor(Color.black);`: Mengatur warna bola menjadi hitam.
- `g.fillOval(x, y, height, width);`: Menggambar bola sebagai oval berwarna hitam pada posisi `(x, y)` dengan ukuran `(width, height)`.

2. GameFrame.java

```
import java.awt.*;
import javax.swing.*;
```

- `import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang menyediakan antarmuka pengguna grafis dan elemen-elemen untuk menggambar.
- `import javax.swing.*;`: Mengimpor semua kelas dari paket `javax.swing`, yang menyediakan kelas-kelas untuk membangun GUI (Graphical User Interface) berbasis Swing.

```
public class GameFrame extends JFrame {
```

Mendefinisikan kelas `GameFrame` yang merupakan subclass dari `JFrame`. `JFrame` adalah kelas dari Swing yang menyediakan sebuah jendela utama untuk aplikasi GUI.

```
    GamePanel panel;
```

Mendeklarasikan variabel `panel` bertipe `GamePanel`. `GamePanel` adalah panel yang akan menampilkan game (harus didefinisikan di tempat lain dalam kode Anda).

```
    GameFrame(int speed) {
        panel = new GamePanel(speed);
        //sending the level of the game argument as the speed of the ball
```

```
        this.add(panel);
        this.setTitle("Pong Game");
        //setting title
        this.setResizable(false);
        this.setBackground(Color.white);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.pack();
        //setting visibility to true to view the GameFrame
        this.setVisible(true);


        this.setLocationRelativeTo(null);
    }
```

- `GameFrame(int speed)`: Konstruktor untuk kelas `GameFrame`. Menerima satu argumen `speed` yang akan digunakan untuk mengatur kecepatan bola dalam game.
- `panel = new GamePanel(speed);`: Membuat instance baru `GamePanel` dengan kecepatan bola (`speed`) yang diberikan sebagai argumen.
- `this.add(panel);`: Menambahkan `panel` ke `GameFrame`. `this` mengacu pada instance saat ini dari `GameFrame`.
- `this.setTitle("Pong Game");`: Mengatur judul jendela `GameFrame` menjadi "Pong Game".
- `this.setResizable(false);`: Mengatur jendela agar tidak dapat diubah ukurannya.
- `this.setBackground(Color.white);`: Mengatur latar belakang jendela menjadi putih.
- `this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`: Mengatur operasi default saat jendela ditutup, yaitu keluar dari aplikasi.
- `this.pack();`: Mengatur ukuran jendela agar pas dengan komponen-komponen yang ada di dalamnya.
- `this.setVisible(true);`: Mengatur visibilitas jendela menjadi `true` sehingga jendela ditampilkan.
- `this.setLocationRelativeTo(null);`: Menempatkan jendela di tengah layar.

3. GamePanel.java

```
import java.awt.*;
```

```
import java.awt.event.*;

import java.util.*;

import javax.swing.*;
```

- `import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang menyediakan antarmuka pengguna grafis dan elemen-elemen untuk menggambar.
- `import java.awt.event.*;`: Mengimpor semua kelas dari paket `java.awt.event`, yang menyediakan event handling.
- `import java.util.*;`: Mengimpor semua kelas dari paket `java.util`, termasuk kelas `Random`.
- `import javax.swing.*;`: Mengimpor semua kelas dari paket `javax.swing`, yang menyediakan kelas-kelas untuk membangun GUI berbasis Swing.

```
public class GamePanel extends JPanel implements Runnable {
```

Mendefinisikan kelas `GamePanel` yang merupakan subclass dari `JPanel` dan mengimplementasikan interface `Runnable`. `JPanel` adalah panel yang dapat menampung komponen GUI lainnya, dan `Runnable` memungkinkan objek `GamePanel` untuk dieksekusi dalam thread terpisah.

```
    static final int WIDTH = 1000;

    static final int height = 555;

    static final Dimension dimen = new Dimension(WIDTH, height);

    static final int BALL_DIAMETER = 20;

    static final int P_WIDTH = 25;

    static final int P_HEIGHT = 100;
```

- `static final int WIDTH = 1000;`: Lebar panel game.
- `static final int height = 555;`: Tinggi panel game.
- `static final Dimension dimen = new Dimension(WIDTH, height);`: Dimensi panel game.
- `static final int BALL_DIAMETER = 20;`: Diameter bola.
- `static final int P_WIDTH = 25;`: Lebar paddle.
- `static final int P_HEIGHT = 100;`: Tinggi paddle.

```
    Thread gameThread;

    Image image;

    Graphics graphics;

    Random random;
```

```
Paddle p1;
Paddle p2;
Ball ball;
Score score;
int speed;
```

- `Thread gameThread;`: Thread untuk menjalankan game loop.
- `Image image;`: Gambar yang digunakan untuk double buffering.
- `Graphics graphics;`: Grafik untuk menggambar pada `image`.
- `Random random;`: Objek `Random` untuk menghasilkan angka acak.
- `Paddle p1;`: Paddle pemain 1.
- `Paddle p2;`: Paddle pemain 2.
- `Ball ball;`: Bola dalam game.
- `Score score;`: Skor game.
- `int speed;`: Kecepatan bola.

```
GamePanel(int speed) {
    this.speed = speed;
    newPaddles();
    newBall(speed);
    score = new Score(WIDTH, height);
    this.setFocusable(true);
    this.addKeyListener(new AL());
    this.setPreferredSize(dimen);

    gameThread = new Thread(this);
    gameThread.start();
}
```

- `GamePanel(int speed)`: Konstruktor untuk kelas `GamePanel`. Menerima satu argumen `speed` yang akan digunakan untuk mengatur kecepatan bola.
- `this.speed = speed;`: Menginisialisasi kecepatan bola.

- `newPaddles();`: Membuat paddle baru.
- `newBall(speed);`: Membuat bola baru dengan kecepatan yang diberikan.
- `score = new Score(WIDTH, height);`: Membuat objek skor.
- `this.setFocusable(true);`: Mengatur panel agar bisa menerima fokus keyboard.
- `this.addKeyListener(new AL());`: Menambahkan `KeyListener` untuk menangani input keyboard.
- `this.setPreferredSize(dimen);`: Mengatur ukuran preferensi panel.
- `gameThread = new Thread(this);`: Membuat thread baru untuk game loop.
- `gameThread.start();`: Memulai thread game.

```
public void newBall(int speed) {
    random = new Random();
    ball = new Ball((WIDTH / 2) - (BALL_DIAMETER / 2), random.nextInt(height - BALL_DIAMETER), BALL_DIAMETER, BALL_DIAMETER, speed);
}
```

- `public void newBall(int speed)`: Metode untuk membuat bola baru.
- `random = new Random();`: Membuat objek `Random` baru.
- `ball = new Ball((WIDTH / 2) - (BALL_DIAMETER / 2), random.nextInt(height - BALL_DIAMETER), BALL_DIAMETER, BALL_DIAMETER, speed);`: Membuat bola baru di tengah layar dengan koordinat `y` acak dan kecepatan yang diberikan.

```
public void newPaddles() {
    p1 = new Paddle(0, (height / 2) - (P_HEIGHT / 2), P_WIDTH, P_HEIGHT, 1);
    p2 = new Paddle(WIDTH - P_WIDTH, (height / 2) - (P_HEIGHT / 2), P_WIDTH, P_HEIGHT, 2);
}
```

- `public void newPaddles()`: Metode untuk membuat paddle baru.
- `p1 = new Paddle(0, (height / 2) - (P_HEIGHT / 2), P_WIDTH, P_HEIGHT, 1);`: Membuat paddle pemain 1 di sisi kiri layar.
- `p2 = new Paddle(WIDTH - P_WIDTH, (height / 2) - (P_HEIGHT / 2), P_WIDTH, P_HEIGHT, 2);`: Membuat paddle pemain 2 di sisi kanan layar.

```
public void paint(Graphics g) {
    image = createImage(getWidth(), getHeight());
```

```
        graphics = image.getGraphics();
        draw(graphics);
        g.drawImage(image, 0, 0, this);
    }
```

- `public void paint(Graphics g)`: Metode untuk menggambar komponen panel.
- `image = createImage(getWidth(), getHeight());`: Membuat gambar untuk double buffering.
- `graphics = image.getGraphics();`: Mendapatkan objek `Graphics` untuk menggambar pada `image`.
- `draw(graphics);`: Memanggil metode `draw` untuk menggambar komponen game pada `graphics`.
- `g.drawImage(image, 0, 0, this);`: Menggambar `image` pada panel.

```
    public void draw(Graphics g) {
        p1.draw(g);
        p2.draw(g);
        ball.draw(g);
        score.draw(g);
        Toolkit.getDefaultToolkit().sync();
    }
```

- `public void draw(Graphics g)`: Metode untuk menggambar semua elemen game.
- `p1.draw(g);`: Menggambar paddle pemain 1.
- `p2.draw(g);`: Menggambar paddle pemain 2.
- `ball.draw(g);`: Menggambar bola.
- `score.draw(g);`: Menggambar skor.
- `Toolkit.getDefaultToolkit().sync();`: Menyinkronkan toolkit untuk animasi yang lebih halus.

```
    public void move() {
        p1.move();
        p2.move();
        ball.move();
    }
```

- `public void move()`: Metode untuk memperbarui posisi semua elemen game.
- `p1.move();`: Memperbarui posisi paddle pemain 1.
- `p2.move();`: Memperbarui posisi paddle pemain 2.
- `ball.move();`: Memperbarui posisi bola.

```
public void checkCollision() {
    if (ball.y <= 0) {
        ball.setYDirection(-ball.vel_y);
    }
    if (ball.y >= height - BALL_DIAMETER) {
        ball.setYDirection(-ball.vel_y);
    }
    if (ball.intersects(p1)) {
        ball.vel_x = Math.abs(ball.vel_x);
        ball.setXDirection(ball.vel_x);
        ball.setYDirection(ball.vel_y);
    }
    if (ball.intersects(p2)) {
        ball.vel_x = Math.abs(ball.vel_x);
        ball.setXDirection(-ball.vel_x);
        ball.setYDirection(ball.vel_y);
    }
    if (p1.y <= 0)
        p1.y = 0;
    if (p1.y >= (height - P_HEIGHT))
        p1.y = height - P_HEIGHT;
```

```
    if (p2.y <= 0)
        p2.y = 0;
    if (p2.y >= (height - P_HEIGHT))
        p2.y = height - P_HEIGHT;
    if (ball.x <= 0) {
        score.player2++;
        newPaddles();
        newBall(speed);
    }
    if (ball.x >= WIDTH - BALL_DIAMETER) {
        score.player1++;
        newPaddles();
        newBall(speed);
    }
}
```

- `public void checkCollision()`: Metode untuk memeriksa dan menangani tabrakan dalam game.
- `if (ball.y <= 0) { ... }`: Jika bola menyentuh tepi atas, pantulkan bola.
- `if (ball.y >= height - BALL_DIAMETER) { ... }`: Jika bola menyentuh tepi bawah, pantulkan bola.
- `if (ball.intersects(p1)) { ... }`: Jika bola menyentuh paddle pemain 1, pantulkan bola.
- `if (ball.intersects(p2)) { ... }`: Jika bola menyentuh paddle pemain 2, pantulkan bola.
- `if (p1.y <= 0) p1.y = 0;`: Jika paddle pemain 1 menyentuh tepi atas, hentikan paddle.
- `if (p1.y >= (height - P_HEIGHT)) p1.y = height - P_HEIGHT;`: Jika paddle pemain 1 menyentuh tepi bawah, hentikan paddle.
- `if (p2.y <= 0) p2.y = 0;`: Jika paddle pemain 2 menyentuh tepi atas, hentikan paddle.
- `if (p2.y >= (height - P_HEIGHT)) p2.y = height - P_HEIGHT;`: Jika paddle pemain 2 menyentuh tepi bawah, hentikan paddle.
- `if (ball.x <= 0) { ... }`: Jika bola keluar dari sisi kiri, tambahkan poin untuk pemain 2, buat paddle dan bola baru.

- `if (ball.x >= WIDTH - BALL_DIAMETER) { ... }`: Jika bola keluar dari sisi kanan, tambahkan poin untuk pemain 1, buat paddle dan bola baru.

```
public void run() {
    long lastTime = System.nanoTime();
    double amountOfTicks = 60.0;
    double ns = 1000000000 / amountOfTicks;
    double delta = 0;
    while (true) {
        long now = System.nanoTime();
        delta += (now - lastTime) / ns;
        lastTime = now;
        if (delta >= 1) {
            move();
            checkCollision();
            repaint();
            delta--;
        }
    }
}
```

- `public void run()`: Metode yang berisi game loop.
- `long lastTime = System.nanoTime();`: Menyimpan waktu saat ini dalam nanodetik.
- `double amountOfTicks = 60.0;`: Menentukan jumlah tick per detik.
- `double ns = 1000000000 / amountOfTicks;`: Menghitung jumlah nanodetik per tick.
- `double delta = 0;`: Menginisialisasi variabel delta.
- `while (true) { ... }`: Loop utama game.
- `long now = System.nanoTime();`: Menyimpan waktu saat ini dalam nanodetik.
- `delta += (now - lastTime) / ns;`: Menghitung delta waktu sejak tick terakhir.
- `lastTime = now;`: Mengupdate waktu terakhir.
- `if (delta >= 1) { ... }`: Jika delta mencapai atau melebihi 1, lakukan operasi berikut:

- `move();`: Memperbarui posisi elemen game.
- `checkCollision();`: Memeriksa tabrakan.
- `repaint();`: Menggambar ulang panel.
- `delta--;`: Mengurangi delta.

```
public class AL extends KeyAdapter {

    public void keyPressed(KeyEvent e) {

        p1.keyPressed(e);

        p2.keyPressed(e);

    }

    public void keyReleased(KeyEvent e) {

        p1.keyReleased(e);

        p2.keyReleased(e);

    }

}
```

• `public class AL extends KeyAdapter { ... }`: Mendefinisikan kelas nested `AL` yang mengextends `KeyAdapter` untuk menangani input keyboard.
• `public void keyPressed(KeyEvent e) { ... }`: Metode untuk menangani event key pressed.

- `p1.keyPressed(e);`: Memanggil metode `keyPressed` pada paddle pemain 1.
- `p2.keyPressed(e);`: Memanggil metode `keyPressed` pada paddle pemain 2.

• `public void keyReleased(KeyEvent e) { ... }`: Metode untuk menangani event key released.

- `p1.keyReleased(e);`: Memanggil metode `keyReleased` pada paddle pemain 1.
- `p2.keyReleased(e);`: Memanggil metode `keyReleased` pada paddle pemain 2.

4. Paddle.java

```
import java.awt.*;

import java.awt.event.*;
```

• `import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang menyediakan antarmuka pengguna grafis dan elemen-elemen untuk menggambar.

- `import java.awt.event.*;`: Mengimpor semua kelas dari paket `java.awt.event`, yang menyediakan event handling.

public class Paddle extends Rectangle {

Mendefinisikan kelas `Paddle` yang merupakan subclass dari `Rectangle`. `Rectangle` adalah kelas dalam paket `java.awt` yang mendefinisikan persegi panjang dengan koordinat, lebar, dan tinggi.

int id;

int yVelocity;

int speed = 10;

int id;

int yVelocity;

int speed = 10;

- `int id;`: ID untuk mengidentifikasi paddle (pemain 1 atau pemain 2).
- `int yVelocity;`: Kecepatan vertikal paddle.

- `int speed = 10;`: Kecepatan pergerakan paddle.

Paddle(int x, int y, int P_Width, int P_Height, int id) {

super(x, y, P_Width, P_Height);

this.id = id;

}

- `Paddle(int x, int y, int P_Width, int P_Height, int id)`: Konstruktor untuk kelas `Paddle`.
- `super(x, y, P_Width, P_Height);`: Memanggil konstruktor dari kelas `Rectangle` untuk menginisialisasi koordinat (x, y), lebar, dan tinggi paddle.

- `this.id = id;`: Menginisialisasi ID paddle.

public void keyPressed(KeyEvent e) {

switch(id) {

case 1:

if(e.getKeyCode() == KeyEvent.VK_W) {

setYDirection(-speed);

```
            }
            if(e.getKeyCode() == KeyEvent.VK_S) {
                setYDirection(speed);
            }
            break;
        case 2:
            if(e.getKeyCode() == KeyEvent.VK_UP) {
                setYDirection(-speed);
            }
            if(e.getKeyCode() == KeyEvent.VK_DOWN) {
                setYDirection(speed);
            }
            break;
    }
}
```

- `public void keyPressed(KeyEvent e)`: Metode untuk menangani event key pressed.
- `switch(id) { ... }`: Memilih tindakan berdasarkan ID paddle.
- `case 1: ...`: Jika ID adalah 1 (paddle pemain 1).
    - `if(e.getKeyCode() == KeyEvent.VK_W) { ... }`: Jika tombol 'W' ditekan, gerakkan paddle ke atas.
    - `if(e.getKeyCode() == KeyEvent.VK_S) { ... }`: Jika tombol 'S' ditekan, gerakkan paddle ke bawah.
- `case 2: ...`: Jika ID adalah 2 (paddle pemain 2).
    - `if(e.getKeyCode() == KeyEvent.VK_UP) { ... }`: Jika tombol 'UP' ditekan, gerakkan paddle ke atas.
    - `if(e.getKeyCode() == KeyEvent.VK_DOWN) { ... }`: Jika tombol 'DOWN' ditekan, gerakkan paddle ke bawah.

```
public void keyReleased(KeyEvent e) {
```

```
        switch(id) {
            case 1:
                if(e.getKeyCode() == KeyEvent.VK_W) {
                    setYDirection(0);
                }
                if(e.getKeyCode() == KeyEvent.VK_S) {
                    setYDirection(0);
                }
                break;
            case 2:
                if(e.getKeyCode() == KeyEvent.VK_UP) {
                    setYDirection(0);
                }
                if(e.getKeyCode() == KeyEvent.VK_DOWN) {
                    setYDirection(0);
                }
                break;
        }
    }
```

- `public void keyReleased(KeyEvent e)`: Metode untuk menangani event key released.
- `switch(id) { ... }`: Memilih tindakan berdasarkan ID paddle.
- `case 1: ...`: Jika ID adalah 1 (paddle pemain 1).
    - `if(e.getKeyCode() == KeyEvent.VK_W) { ... }`: Jika tombol 'W' dilepaskan, hentikan paddle.
    - `if(e.getKeyCode() == KeyEvent.VK_S) { ... }`: Jika tombol 'S' dilepaskan, hentikan paddle.
- `case 2: ...`: Jika ID adalah 2 (paddle pemain 2).

- `if(e.getKeyCode() == KeyEvent.VK_UP) { ... }`: Jika tombol 'UP' dilepaskan, hentikan paddle.
- `if(e.getKeyCode() == KeyEvent.VK_DOWN) { ... }`: Jika tombol 'DOWN' dilepaskan, hentikan paddle.

```
public void setYDirection(int yDirection) {

    yVelocity = yDirection;

}
```

- `public void setYDirection(int yDirection)`: Metode untuk mengatur kecepatan vertikal paddle.

    - `yVelocity = yDirection;`: Menginisialisasi kecepatan vertikal paddle dengan nilai yang diberikan.

```
public void move() {

    y = y + yVelocity;

}
```

- `ublic void move()`: Metode untuk memperbarui posisi paddle berdasarkan kecepatan vertikal.

    - `y = y + yVelocity;`: Mengupdate koordinat `y` paddle dengan menambahkan kecepatan vertikal.

```
public void draw(Graphics g) {

    if(id == 1)

        g.setColor(Color.blue);

    else

        g.setColor(Color.red);

    g.fillRect(x, y, width, height);

}
```

- `public void draw(Graphics g)`: Metode untuk menggambar paddle.
- `if(id == 1) g.setColor(Color.blue);`: Jika ID adalah 1, set warna paddle menjadi biru.
- `else g.setColor(Color.red);`: Jika ID bukan 1, set warna paddle menjadi merah.

- `g.fillRect(x, y, width, height);`: Menggambar persegi panjang dengan koordinat, lebar, dan tinggi paddle.

5. Score.java

```
import java.awt.*;
```

`import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang menyediakan antarmuka pengguna grafis dan elemen-elemen untuk menggambar.

```
public class Score extends Rectangle {
```

`public class Score extends Rectangle`: Mendefinisikan kelas `Score` yang merupakan subclass dari `Rectangle`. `Rectangle` adalah kelas dalam paket `java.awt` yang mendefinisikan persegi panjang dengan koordinat, lebar, dan tinggi.

```
    static int GAME_WIDTH;

    static int GAME_HEIGHT;

    int player1;

    int player2;
```

- `static int GAME_WIDTH;`: Lebar permainan yang bersifat statis.
- `static int GAME_HEIGHT;`: Tinggi permainan yang bersifat statis.
- `int player1;`: Skor untuk pemain 1.
- `int player2;`: Skor untuk pemain 2.

```
    Score(int GAME_WIDTH, int GAME_HEIGHT) {

        Score.GAME_WIDTH = GAME_WIDTH;

        Score.GAME_HEIGHT = GAME_HEIGHT;

    }
```

- `Score(int GAME_WIDTH, int GAME_HEIGHT)`: Konstruktor untuk kelas `Score`.
- `Score.GAME_WIDTH = GAME_WIDTH;`: Menginisialisasi lebar permainan.
- `Score.GAME_HEIGHT = GAME_HEIGHT;`: Menginisialisasi tinggi permainan.

```
    public void draw(Graphics g) {

        g.setColor(Color.black);
```

```
        g.setFont(new Font("Consolas", Font.ROMAN_BASELINE, 60));


        g.drawLine(GAME_WIDTH / 2, 0, GAME_WIDTH / 2, GAME_HEIGHT);


        g.drawString(String.valueOf(player1 / 10) + String.valueOf(player1 % 10),
    (GAME_WIDTH / 2) - 85, 50);

        g.drawString(String.valueOf(player2 / 10) + String.valueOf(player2 % 10),
    (GAME_WIDTH / 2) + 20, 50);

      }
```

- `public void draw(Graphics g)`: Metode untuk menggambar skor pemain pada layar.
- `g.setColor(Color.black);`: Mengatur warna grafik menjadi hitam.
- `g.setFont(new Font("Consolas", Font.ROMAN_BASELINE, 60));`: Mengatur font untuk teks yang akan digambar, dengan font "Consolas", gaya `Font.ROMAN_BASELINE`, dan ukuran 60.
- `g.drawLine(GAME_WIDTH / 2, 0, GAME_WIDTH / 2, GAME_HEIGHT);`: Menggambar garis vertikal di tengah layar untuk memisahkan area kedua pemain.
- `g.drawString(String.valueOf(player1 / 10) + String.valueOf(player1 % 10), (GAME_WIDTH / 2) - 85, 50);`: Menggambar skor pemain 1 di sebelah kiri garis tengah. Skor ditampilkan dengan dua digit.

- `g.drawString(String.valueOf(player2 / 10) + String.valueOf(player2 % 10), (GAME_WIDTH / 2) + 20, 50);`: Menggambar skor pemain 2 di sebelah kanan garis tengah. Skor ditampilkan dengan dua digit.

6. WelcomePage.java

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;
```

- `import javax.swing.*;`: Mengimpor semua kelas dari paket `javax.swing` yang menyediakan elemen antarmuka pengguna.
- `import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt` untuk grafis dan elemen GUI lainnya.
- `import java.awt.event.ActionEvent;`: Mengimpor kelas `ActionEvent` untuk menangani peristiwa aksi.

- `import java.awt.event.ActionListener;`: Mengimpor antarmuka `ActionListener` untuk mendengarkan peristiwa aksi.

public class WelcomePage extends JFrame {

    private JTextField levelField; // To accept level input from user which will increase the speed of the ball

- `public class WelcomePage extends JFrame`: Mendefinisikan kelas `WelcomePage` yang merupakan subclass dari `JFrame`.

- `private JTextField levelField;`: Deklarasi variabel `levelField` untuk menerima input level dari pengguna yang akan meningkatkan kecepatan bola.

public WelcomePage() {

    // Set up the JFrame details

    setTitle("Pong Game Welcome Page");

    setSize(500, 250);

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setLocationRelativeTo(null);

    // This will center the JFrame on the screen

- `setTitle("Pong Game Welcome Page");`: Mengatur judul jendela.
- `setSize(500, 250);`: Mengatur ukuran jendela.
- `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`: Mengatur tindakan saat jendela ditutup.

- `setLocationRelativeTo(null);`: Memposisikan jendela di tengah layar.

    // Create a JPanel to place components on

    JPanel panel = new JPanel();

    panel.setLayout(new BorderLayout());

- `JPanel panel = new JPanel();`: Membuat panel untuk menempatkan komponen.

- `panel.setLayout(new BorderLayout());`: Mengatur layout panel menjadi `BorderLayout`.

    // Add welcome message

JLabel welcomeMessage = new JLabel("Welcome to Pong Game by ProjectGurukul", SwingConstants.CENTER);

welcomeMessage.setFont(new Font("Serif", Font.BOLD, 16));

panel.add(welcomeMessage, BorderLayout.NORTH);

- `JLabel welcomeMessage = new JLabel(...)`: Membuat label selamat datang.
- `welcomeMessage.setFont(new Font("Serif", Font.BOLD, 16));`: Mengatur font label.

- `panel.add(welcomeMessage, BorderLayout.NORTH);`: Menambahkan label ke panel di bagian utara.

// Create a central panel for input and buttons

GridLayout gd=new GridLayout(2,2);

JPanel centerPanel = new JPanel(gd);

- `GridLayout gd = new GridLayout(2, 2);`: Membuat layout grid dengan 2 baris dan 2 kolom.

- `JPanel centerPanel = new JPanel(gd);`: Membuat panel dengan layout grid.

// Add label and field for level input

JLabel levelLabel = new JLabel("Enter Level:");

centerPanel.add(levelLabel);

levelField = new JTextField();

centerPanel.add(levelField);

- `JLabel levelLabel = new JLabel("Enter Level:");`: Membuat label untuk input level.
- `centerPanel.add(levelLabel);`: Menambahkan label ke panel tengah.
- `levelField = new JTextField();`: Membuat field teks untuk input level.

- `centerPanel.add(levelField);`: Menambahkan field teks ke panel tengah.

// Adding a play now button to the welcome page

JButton playNowButton = new JButton("Play Now");

playNowButton.addActionListener(new ActionListener() {

```
    @Override
    public void actionPerformed(ActionEvent e) {
        // Action to perform when Play Now is clicked
        startGame();
    }
});
centerPanel.add(playNowButton);
```

- `JButton playNowButton = new JButton("Play Now");`: Membuat tombol "Play Now".
- `playNowButton.addActionListener(new ActionListener() {...});`: Menambahkan pendengar aksi untuk tombol.
- `centerPanel.add(playNowButton);`: Menambahkan tombol ke panel tengah.

```
// Adding a How To Play button
JButton howToPlayButton = new JButton("How To Play");
howToPlayButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Show instructions for the game
        showInstructions();
    }
});
centerPanel.add(howToPlayButton);
```

- `JButton howToPlayButton = new JButton("How To Play");`: Membuat tombol "How To Play".
- `howToPlayButton.addActionListener(new ActionListener() {...});`: Menambahkan pendengar aksi untuk tombol.
- `centerPanel.add(howToPlayButton);`: Menambahkan tombol ke panel tengah.

```
panel.add(centerPanel, BorderLayout.CENTER);
```

`panel.add(centerPanel, BorderLayout.CENTER);`: Menambahkan panel tengah ke panel utama di bagian tengah.

```
    // Adding the created panel to the Jframe created in welcome page
    add(panel);

    // Make the JFrame visible
    setVisible(true);
}
```

- `add(panel);`: Menambahkan panel utama ke jendela `JFrame`.
- `setVisible(true);`: Membuat jendela terlihat.

```
private void startGame() {
    String level = levelField.getText();

    /* After taking the level input, we are going to send this as a parameter to the GameFrame constructor */

    new GameFrame(Integer.parseInt(level));

    // Close the welcome page
    dispose();
}
```

- `private void startGame()`: Metode untuk memulai permainan.
- `String level = levelField.getText();`: Mendapatkan input level dari field teks.
- `new GameFrame(Integer.parseInt(level));`: Membuat objek `GameFrame` baru dengan parameter level.
- `dispose();`: Menutup halaman selamat datang.

```
private void showInstructions() {
```

```
    // Displaying Instructions to play
    JOptionPane.showMessageDialog(this, "Pong Game Instructions:\n" +
                "1. Use the Pg up and Pg down keys to move the paddle for the player-right.\n" +
                "2. Use the W and S keys to control the paddle for the player-left."+
                "3. Prevent the ball from touching the ground.\n" +
                "4. The player gains a point when the opponent fails to bounce ball back with the paddle.",
            "How To Play", JOptionPane.INFORMATION_MESSAGE);
}
```

- `private void showInstructions()`: Metode untuk menampilkan instruksi permainan.
- `JOptionPane.showMessageDialog(...)`: Menampilkan dialog pesan dengan instruksi permainan.

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new WelcomePage();
        }
    });
}
```

- `public static void main(String[] args)`: Metode utama untuk menjalankan aplikasi.
- `SwingUtilities.invokeLater(new Runnable() {...});`: Menjalankan kode dalam thread event-dispatching Swing.
- `new WelcomePage();`: Membuat objek `WelcomePage` baru untuk menampilkan halaman selamat datang.