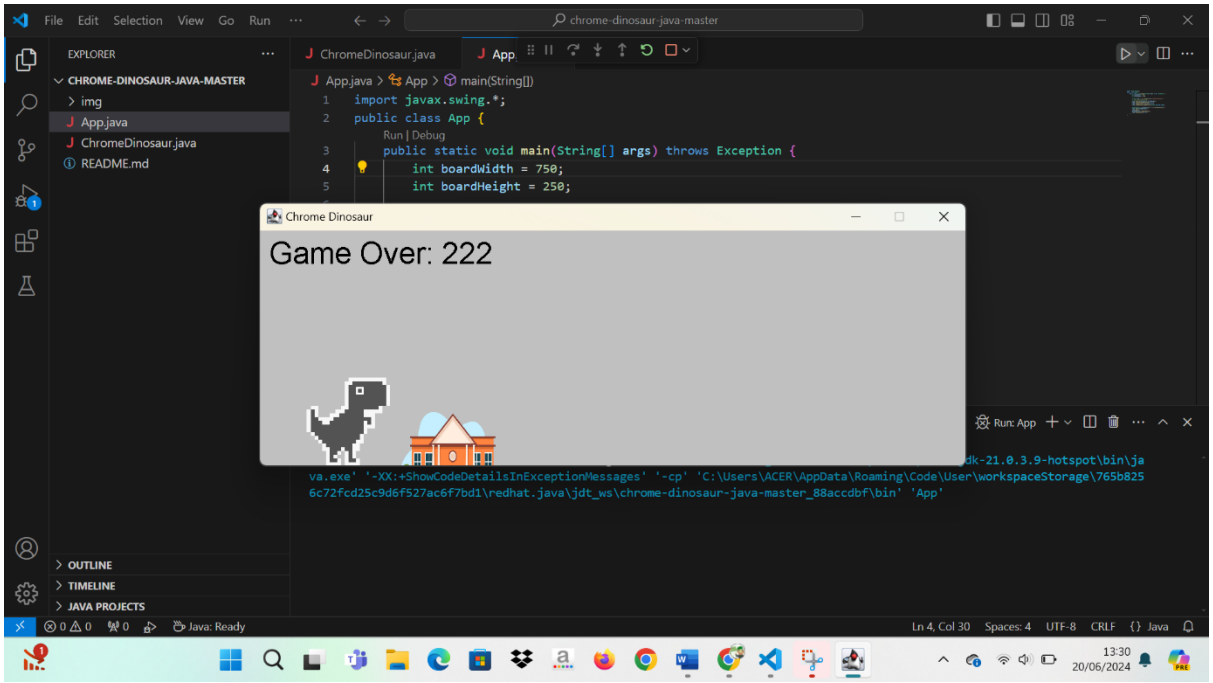
Nama : Muhammad Ansori

Kelas : 2TI03

Nim : 2222105135

PROGRAM APLIKASI GAME CHROME DINOSAURUS

Permainan "Chrome Dinosaur" adalah permainan sederhana yang populer pada browser Google Chrome saat koneksi internet terputus. Konsep dasar permainan ini adalah untuk menghindari rintangan dengan melompati atau merundukinya.



Konsep Permainan:

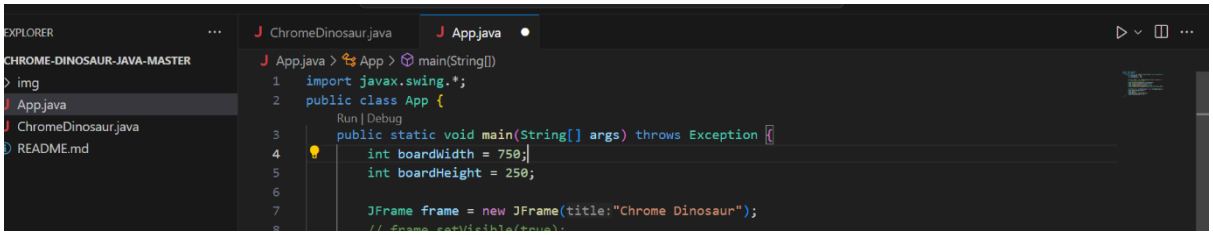
1. **Tampilan Awal**: Permainan dimulai dengan karakter utama, biasanya berupa dinosaurus, berada di posisi awal yang statis di sebelah kiri layar.
2. **Gerakan Karakter**:
    - **Melompat**: Karakter bisa melompat untuk menghindari rintangan. Ini bisa dilakukan dengan menekan tombol space atau tombol panah ke atas.
    - **Merunduk**: Karakter juga bisa merunduk untuk menghindari rintangan yang lebih rendah. Ini biasanya dilakukan dengan menekan tombol panah ke bawah.

3. **Rintangan**: Rintangan muncul dari sebelah kanan layar dan bergerak ke arah karakter. Rintangan ini bisa berupa:
    - **Rintangan Tinggi**: Perlu dilompati dengan melompat.
    - **Rintangan Rendah**: Perlu dihindari dengan merunduk.
    - **Rintangan Ganda**: Kombinasi dari tinggi dan rendah yang membutuhkan respon cepat untuk melompat atau merunduk.
4. **Skor**: Skor pemain bertambah seiring dengan jarak yang ditempuh tanpa menabrak rintangan. Semakin jauh jaraknya, semakin tinggi skornya.
5. **Kesulitan**: Secara bertahap, kecepatan rintangan akan meningkat, sehingga pemain perlu bereaksi lebih cepat dan tepat waktu untuk menghindari tabrakan.
6. **Game Over**: Permainan berakhir ketika karakter menabrak rintangan. Skor tertinggi biasanya ditampilkan untuk pembanding dengan permainan berikutnya.

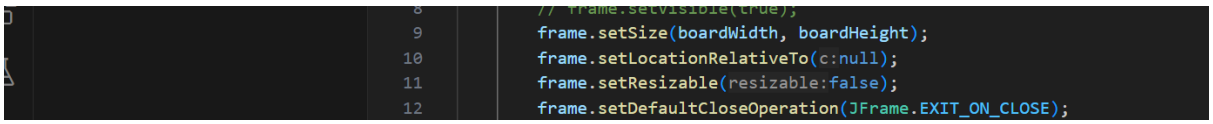
Implementasi dalam Kode:

Dalam implementasi kode yang diberikan sebelumnya:

- Kelas ChromeDinosaur akan mengatur logika permainan seperti menggambar karakter, mengatur perilaku rintangan, mendeteksi input pengguna, dan menghitung skor.
- Kelas App bertindak sebagai pemicu untuk membuat jendela permainan, menambahkan panel permainan, dan menampilkan jendela kepada pengguna.



Imports dan Class Declaration: Kode dimulai dengan mengimpor pustaka javax.swing.*, yang diperlukan untuk GUI berbasis Java. Kelas utama App kemudian dideklarasikan



Pengaturan JFrame:

- frame.setSize(boardWidth, boardHeight);: Mengatur ukuran jendela JFrame menjadi 750x250 piksel.
- frame.setLocationRelativeTo(null);: Mengatur posisi frame di tengah layar.
- frame.setResizable(false);: Mencegah pengguna untuk mengubah ukuran jendela.
- frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);: Mengatur operasi default ketika tombol close di klik, yaitu menutup aplikasi.

```
ChromeDinosaur chromeDinosaur = new ChromeDinosaur();
frame.add(chromeDinosaur);
frame.pack();
```

Menambahkan Komponen:

- ChromeDinosaur chromeDinosaur = new ChromeDinosaur();: Membuat instance dari kelas ChromeDinosaur. ChromeDinosaur adalah kelas yang mendefinisikan panel tempat permainan utama akan ditampilkan.
- frame.add(chromeDinosaur);: Menambahkan panel chromeDinosaur ke dalam frame.
- frame.pack();: Mengatur ukuran frame secara otomatis berdasarkan komponen yang ada di dalamnya. Dalam hal ini, ukuran frame akan disesuaikan dengan ukuran panel chromeDinosaur.

```
17        chromeDinosaur.requestFocus();
18        frame.setVisible(b:true);
19    }
```

Menampilkan Frame:

- chromeDinosaur.requestFocus();: Mengatur fokus pada panel chromeDinosaur, sehingga panel tersebut langsung dapat menerima input pengguna (untuk menggerakkan karakter).
- frame.setVisible(true);: Mengatur agar frame menjadi terlihat (visible) di layar, sehingga aplikasi dapat diakses

```
import javax.swing.*;
public class App {
    public static void main(String[] args) throws Exception {
        int boardWidth = 750;
        int boardHeight = 250;

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

        ChromeDinosaur chromeDinosaur = new ChromeDinosaur();
        frame.add(chromeDinosaur);
        frame.pack();
        chromeDinosaur.requestFocus();
```

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

public class ChromeDinosaur extends JPanel implements ActionListener,
KeyListener {
    int boardWidth = 750;
    int boardHeight = 250;

    //images
    Image dinosaurImg;
    Image dinosaurDeadImg;
    Image dinosaurJumpImg;
    Image cactus1Img;
    Image cactus2Img;
    Image cactus3Img;

    class Block {
        int x;
        int y;
        int width;
        int height;
        Image img;

        Block(int x, int y, int width, int height, Image img) {
            this.x = x;
            this.y = y;
            this.width = width;
            this.height = height;
            this.img = img;
        }
    }

    //dinosaur
    int dinosaurWidth = 88;
    int dinosaurHeight = 94;
    int dinosaurX = 50;
    int dinosaurY = boardHeight - dinosaurHeight;

    Block dinosaur;

    //cactus
    int cactus1Width = 34;
    int cactus2Width = 69;
    int cactus3Width = 102;

    int cactusHeight = 70;
```

```
    int cactusX = 700;
    int cactusY = boardHeight - cactusHeight;
    ArrayList<Block> cactusArray;

    //physics
    int velocityX = -12; //cactus moving left speed
    int velocityY = 0; //dinosaur jump speed
    int gravity = 1;

    boolean gameOver = false;
    int score = 0;

    Timer gameLoop;
    Timer placeCactusTimer;

    public ChromeDinosaur() {
        setPreferredSize(new Dimension(boardWidth, boardHeight));
        setBackground(Color.lightGray);
        setFocusable(true);
        addKeyListener(this);

        dinosaurImg = new ImageIcon(getClass().getResource("./img/dino-
run.gif")).getImage();
        dinosaurDeadImg = new
ImageIcon(getClass().getResource("./img/dino-dead.png")).getImage();
        dinosaurJumpImg = new
ImageIcon(getClass().getResource("./img/dino-jump.png")).getImage();
        cactus1Img = new
ImageIcon(getClass().getResource("./img/cactus1.png")).getImage();
        cactus2Img = new
ImageIcon(getClass().getResource("./img/cactus2.png")).getImage();
        cactus3Img = new
ImageIcon(getClass().getResource("./img/cactus3.png")).getImage();

        //dinosaur
        dinosaur = new Block(dinosaurX, dinosaurY, dinosaurWidth,
dinosaurHeight, dinosaurImg);
        //cactus
        cactusArray = new ArrayList<Block>();

        //game timer
        gameLoop = new Timer(1000/60, this); //1000/60 = 60 frames per
1000ms (1s), update
        gameLoop.start();

        //place cactus timer
        placeCactusTimer = new Timer(1500, new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                placeCactus();
            }
        });
        placeCactusTimer.start();
    }

    void placeCactus() {
        if (gameOver) {
            return;
        }

        double placeCactusChance = Math.random(); //0 - 0.999999
        if (placeCactusChance > .90) { //10% you get cactus3
            Block cactus = new Block(cactusX, cactusY, cactus3Width *
2, cactusHeight * 2, cactus3Img);
            cactusArray.add(cactus);
        }
        else if (placeCactusChance > .70) { //20% you get cactus2
            Block cactus = new Block(cactusX, cactusY, cactus2Width *
2, cactusHeight * 2, cactus2Img);
            cactusArray.add(cactus);
        }
        else if (placeCactusChance > .50) { //20% you get cactus1
            Block cactus = new Block(cactusX, cactusY, cactus1Width *
2, cactusHeight * 2, cactus1Img);
            cactusArray.add(cactus);
        }

        if (cactusArray.size() > 10) {
            cactusArray.remove(0); //remove the first cactus from
ArrayList
        }
    }

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

    public void draw(Graphics g) {
        //dinosaur
        g.drawImage(dinosaur.img, dinosaur.x, dinosaur.y,
dinosaur.width, dinosaur.height, null);

        //cactus
        for (int i = 0; i < cactusArray.size(); i++) {
            Block cactus = cactusArray.get(i);
```

```
            g.drawImage(cactus.img, cactus.x, cactus.y, cactus.width,
cactus.height, null);
        }

        //score
        g.setColor(Color.black);
        g.setFont(new Font("Courier", Font.PLAIN, 32));
        if (gameOver) {
            g.drawString("Game Over: " + String.valueOf(score), 10,
35);
        }
        else {
            g.drawString(String.valueOf(score), 10, 35);
        }
    }

    public void move() {
        //dinosaur
        velocityY += gravity;
        dinosaur.y += velocityY;

        if (dinosaur.y > dinosaurY) { //stop the dinosaur from falling
past the ground
            dinosaur.y = dinosaurY;
            velocityY = 0;
            dinosaur.img = dinosaurImg;
        }

        //cactus
        for (int i = 0; i < cactusArray.size(); i++) {
            Block cactus = cactusArray.get(i);
            cactus.x += velocityX;

            if (collision(dinosaur, cactus)) {
                gameOver = true;
                dinosaur.img = dinosaurDeadImg;
            }
        }

        //score
        score++;
    }

    boolean collision(Block a, Block 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) {
        move();
        repaint();
        if (gameOver) {
            placeCactusTimer.stop();
            gameLoop.stop();
        }
    }

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_SPACE) {
            // System.out.println("JUMP!");
            if (dinosaur.y == dinosaurY) {
                velocityY = -17;
                dinosaur.img = dinosaurJumpImg;
            }

            if (gameOver) {
                //restart game by resetting conditions
                dinosaur.y = dinosaurY;
                dinosaur.img = dinosaurImg;
                velocityY = 0;
                cactusArray.clear();
                score = 0;
                gameOver = false;
                gameLoop.start();
                placeCactusTimer.start();
            }
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {}

    @Override
    public void keyReleased(KeyEvent e) {}

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
```

```
                JFrame frame = new JFrame("Chrome Dinosaur");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setResizable(false);

                ChromeDinosaur chromeDinosaur = new ChromeDinosaur();
                frame.add(chromeDinosaur);
                frame.pack();
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);
            }
        });
    }
}

```