LAPORAN PRATIKUM

"PEKAN 8"

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

DOSEN PENGAMPU:

Wahyudi, Dr. S.T. M.T.

DISUSUN OLEH:

Karimah Irsyadiyah (2411533018)

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

T.A 2024/2025

Daftar Pustaka

BAB I
PENDAHULUAN

1.1 Latar Belakang

Struktur data merupakan fondasi penting dalam dunia pemrograman, khususnya dalam pengolahan dan pengurutan data. Salah satu bentuk implementasinya adalah penggunaan algoritma pengurutan (sorting), seperti *Bubble Sort, Merge Sort, Quick Sort* dan *Shell Sort*. Dalam praktikum ini, kita menggabungkan pemahaman algoritma dengan visualisasi menggunakan antarmuka grafis berbasis *Java Swing*. Visualisasi membantu mahasiswa memahami langkah-langkah sorting secara nyata dan interaktif.

1.2 Tujuan

- Memahami cara kerja algoritma Bubble Sort, Merge Sort, Quick Sort dan Shell Sort.
- Mengimplementasikan algoritma tersebut dalam bentuk antarmuka grafis (GUI).
- Menampilkan proses sorting langkah demi langkah secara visual untuk memperkuat pemahaman logika program.

1.3 Alat dan Bahan

- Perangkat Keras: Laptop/PC
- Perangkat komputer dengan IDE (contoh: IntelliJ IDEA / Eclipse).
- Java Development Kit (JDK).

BAB II
PEMBAHASAN

2.1 Langkah-langkah Praktikum dan Pembahasan Program

A. Class : BubbleSortGUI.java

a) Tujuan Program:

Aplikasi GUI Java untuk melakukan **algoritma BubbleSort** secara **langkah per langkah**.

b) Struktur dan Penjelasan Komponen Program:

1. package pekan8;
    - Menandakan file ini berada dalam package bernama pekan8, sesuai struktur proyek modular Java.
2. import java.awt. dan import javax.swing.;**
    - Mengimpor pustaka GUI Java AWT dan Swing.
        - **AWT** untuk layout dan warna.
        - **Swing** untuk elemen GUI seperti JFrame, JLabel, JTextField, JButton, JTextArea, dll.
3. public class BubbleSortGUI extends JFrame
    - Membuat kelas GUI utama.
    - Turunan dari JFrame, artinya BubbleSortGUI adalah sebuah jendela GUI utama.
4. Deklarasi Variabel:
    - int[] array: Array angka yang akan disorting.
    - JLabel[] labelArray: Label visual untuk setiap elemen array.
    - JButton stepButton, resetButton, setButton: Tombol aksi.
    - JTextField inputField: Field input untuk array.
    - JPanel panelArray: Panel menampilkan elemen array dalam bentuk label.
    - JTextArea stepArea: Area teks untuk mencatat langkah-langkah sorting.
    - int i, j: Indeks untuk iterasi algoritma Bubble Sort.
    - boolean sorting: Menandai apakah proses sorting sedang berjalan.
    - int stepCount: Menyimpan jumlah langkah yang telah dilakukan.

5. Method main()
   - Menjalankan GUI dengan EventQueue.invokeLater(), agar GUI berjalan pada thread yang aman sesuai best practice Swing.
6. Constructor BubbleSortGUI()

   Menyiapkan dan menyusun elemen-elemen GUI:
   - Panel Input (Utara):
     - Input angka dari user (inputField) dan tombol Set Array.
   - Panel Array (Tengah):
     - Menampilkan visual elemen array sebagai label-label kotak.
   - Panel Kontrol (Selatan):
     - Tombol aksi Langkah Selanjutnya dan Reset.
   - Panel Log (Timur):
     - Area teks (JTextArea) dengan JScrollPane untuk mencatat log setiap langkah sorting.

   Event Listener:
   - setButton: Mengubah input string menjadi array integer, lalu tampilkan visual label.
   - stepButton: Melakukan satu langkah Bubble Sort.
   - resetButton: Mengembalikan GUI ke kondisi awal.
7. Fungsi-Fungsi Utama:

   setArrayFromInput()
   - Membaca input string angka, parsing menjadi array integer.
   - Inisialisasi variabel, membuat label per elemen array.
   - Menyusun label ke panelArray.

   performStep()
   - Melakukan satu langkah Bubble Sort:
     - Bandingkan dua elemen bertetangga (array[j] dan array[j+1]).
     - Tukar jika perlu.
     - Tandai elemen yang dibandingkan dengan **warna Cyan**, dan yang ditukar dengan **merah**.
     - Log setiap langkah ke stepArea.
   - Proses iteratif dengan variabel i dan j untuk memanage perbandingan dan iterasi luar-dalam Bubble Sort.

   updateLabels()

- Memperbarui isi label sesuai data terbaru di array.

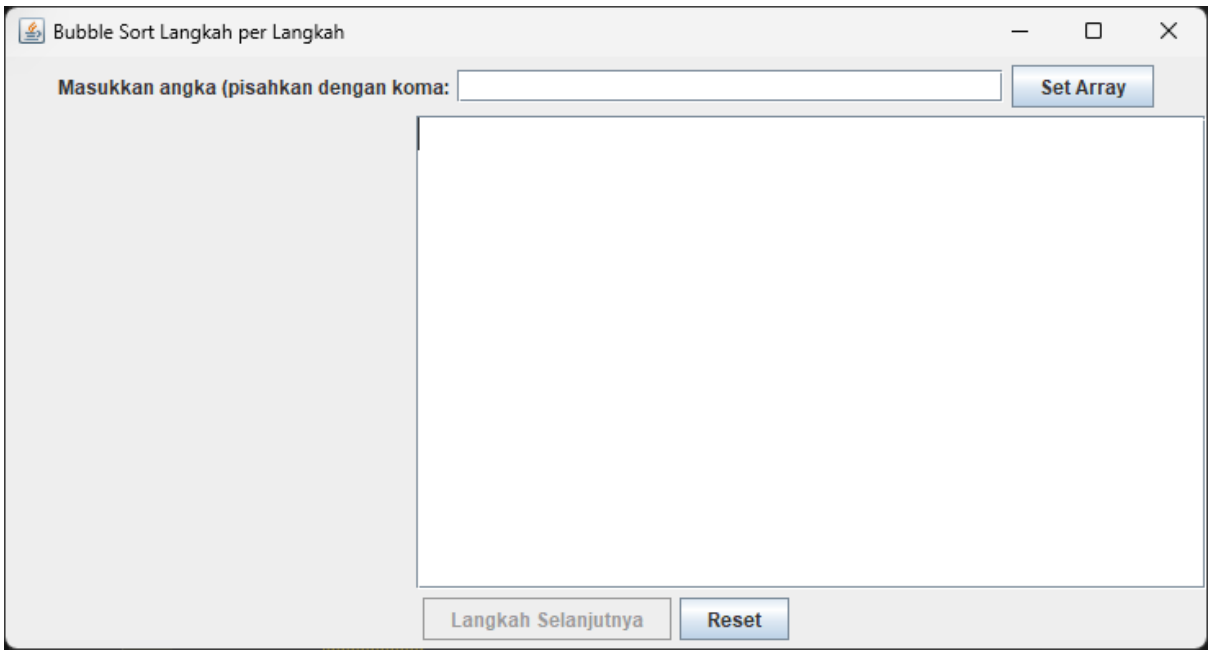
  resetHighlights()
- Mengembalikan warna label ke putih.

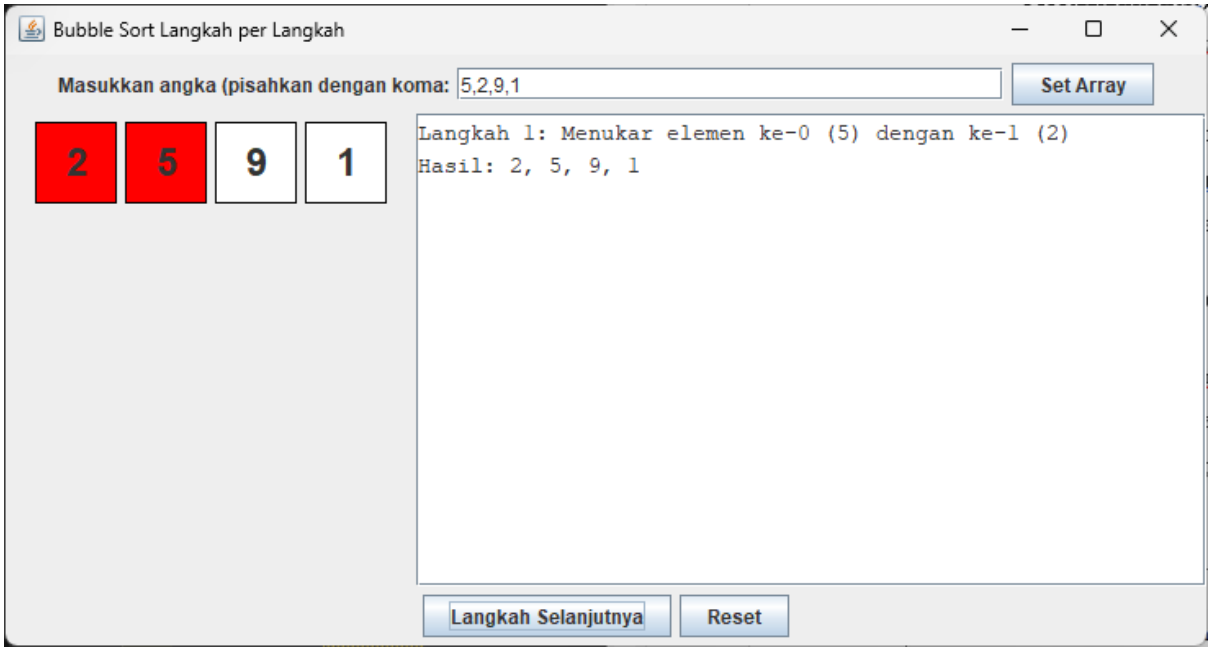
  reset()
- Menghapus data dan reset tampilan GUI ke kondisi awal.

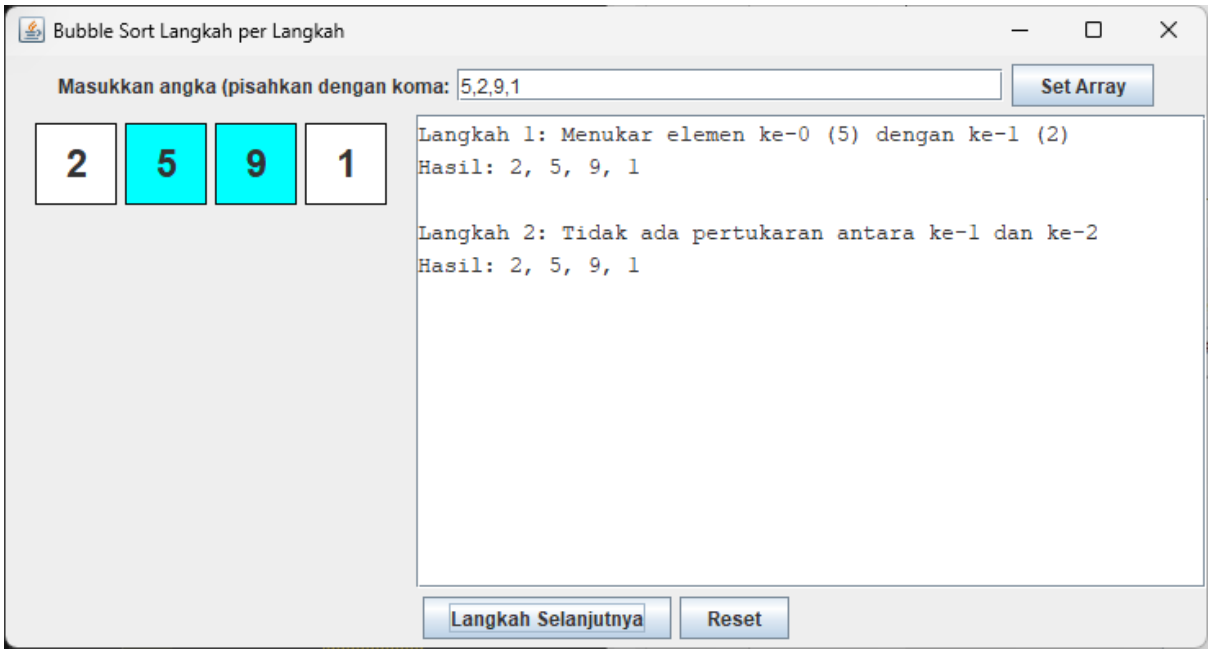
  arrayToString(int[] arr)
- Mengubah isi array menjadi string yang dipisahkan koma.

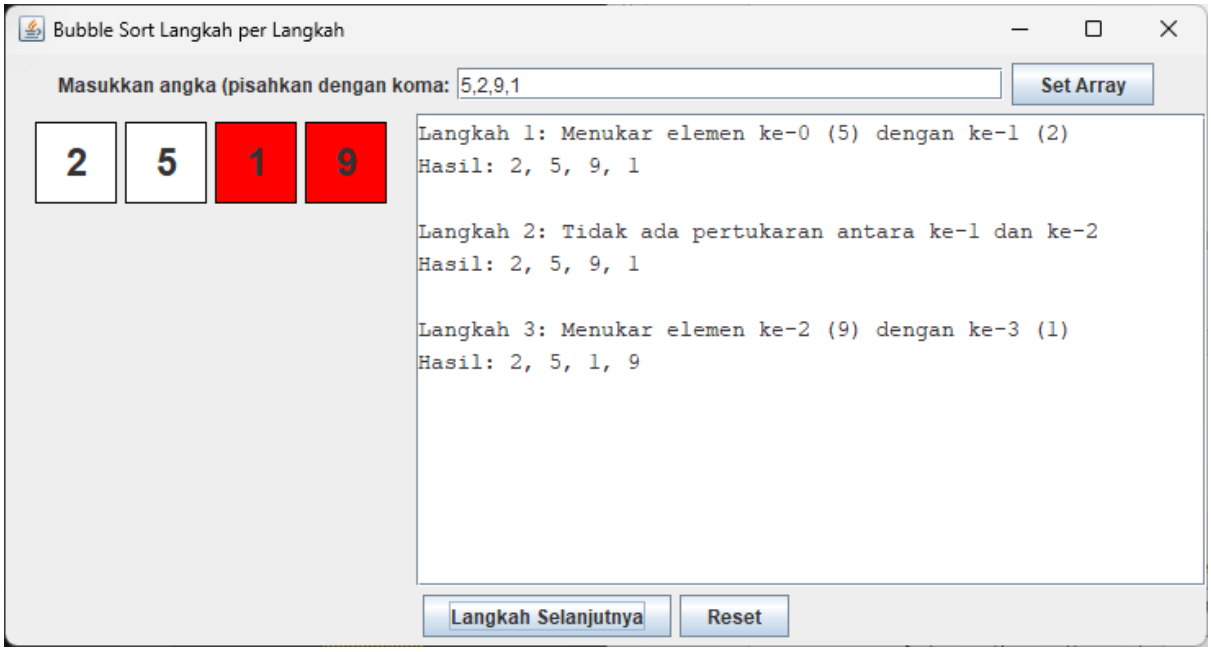
c) Fitur Utama Program:

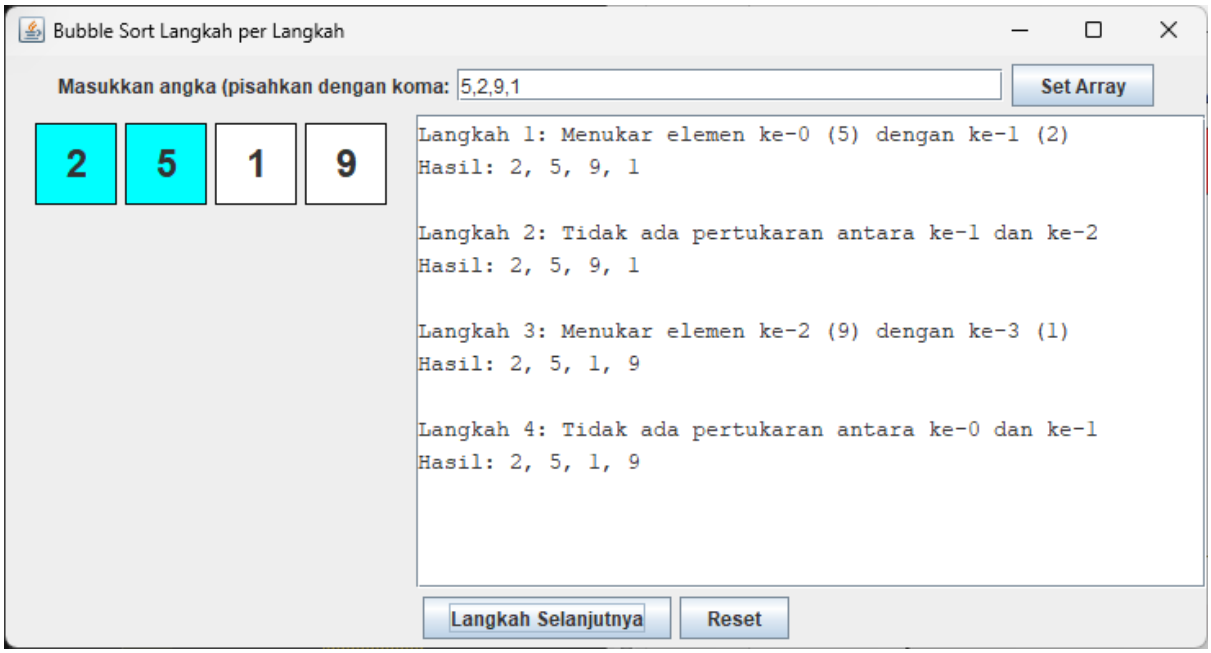
- Pengguna dapat memasukkan angka seperti 5,2,9,1.
- Dapat menjalankan langkah sorting satu demi satu (step-by-step).
- Menampilkan log proses sorting secara real time.
- Tampilan elemen array menggunakan label visual yang berubah warna.

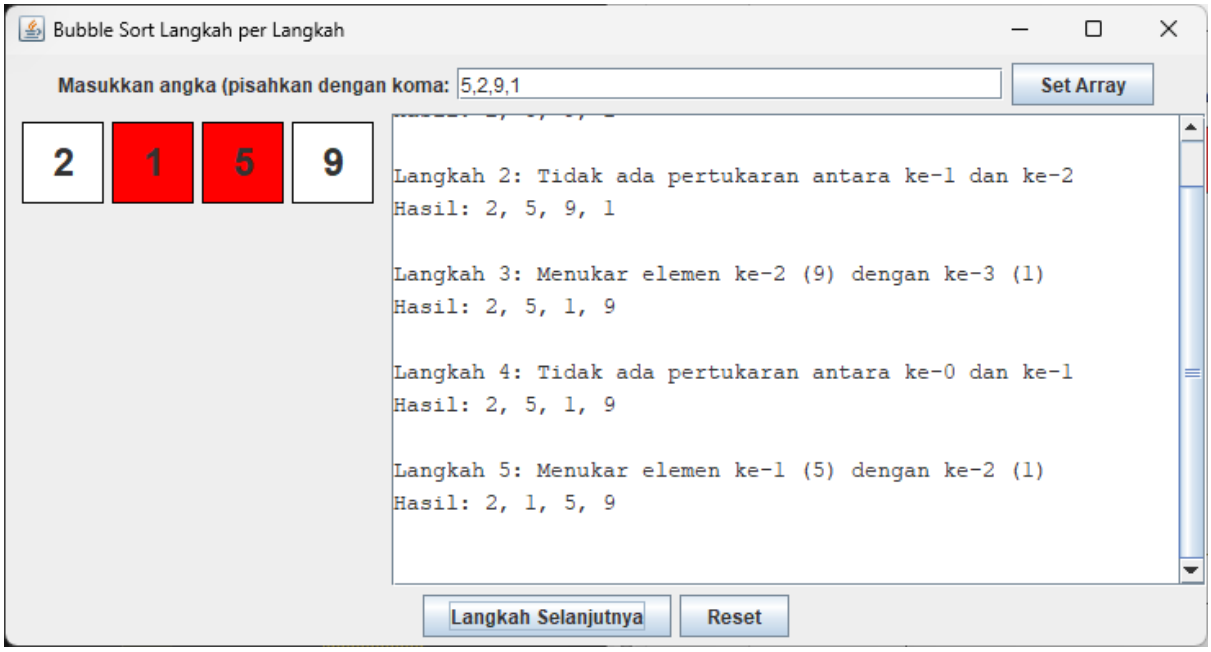


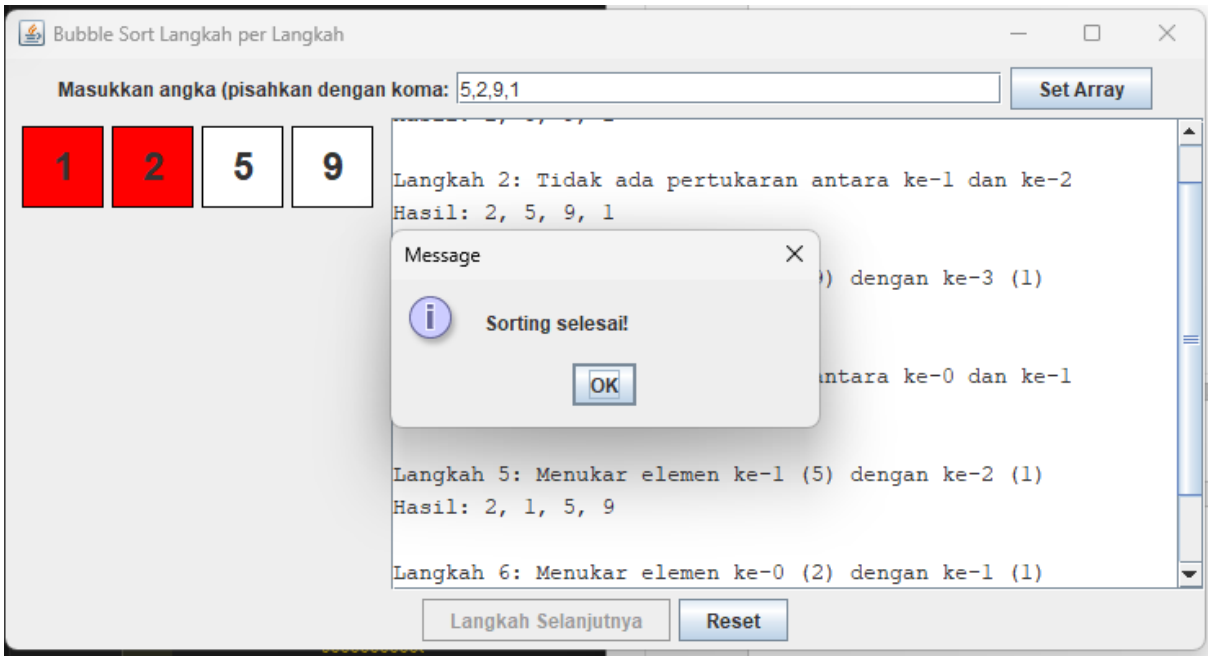

Bubble Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma:
5,2,9,1
Set Array
2
5
9
1
Langkah 1: Menukar elemen ke-0 (5) dengan ke-1 (2)
Hasil: 2, 5, 9, 1
Langkah Selanjutnya
Reset


Bubble Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma:
5,2,9,1
Set Array
2
5
9
1
Langkah 1: Menukar elemen ke-0 (5) dengan ke-1 (2)
Hasil: 2, 5, 9, 1
Langkah 2: Tidak ada pertukaran antara ke-1 dan ke-2
Hasil: 2, 5, 9, 1
Langkah Selanjutnya
Reset


Bubble Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma: 5,2,9,1
Set Array
2
5
1
9
Langkah 1: Menukar elemen ke-0 (5) dengan ke-1 (2)
Hasil: 2, 5, 9, 1
Langkah 2: Tidak ada pertukaran antara ke-1 dan ke-2
Hasil: 2, 5, 9, 1
Langkah 3: Menukar elemen ke-2 (9) dengan ke-3 (1)
Hasil: 2, 5, 1, 9
Langkah Selanjutnya
Reset


Bubble Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma: 5,2,9,1
Set Array
2
5
1
9
Langkah 1: Menukar elemen ke-0 (5) dengan ke-1 (2)
Hasil: 2, 5, 9, 1
Langkah 2: Tidak ada pertukaran antara ke-1 dan ke-2
Hasil: 2, 5, 9, 1
Langkah 3: Menukar elemen ke-2 (9) dengan ke-3 (1)
Hasil: 2, 5, 1, 9
Langkah 4: Tidak ada pertukaran antara ke-0 dan ke-1
Hasil: 2, 5, 1, 9
Langkah Selanjutnya
Reset






```
package pekan8;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.*;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import pekan8.BubbleSortGUI;

public class BubbleSortGUI extends JFrame {


    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;

    private int i = 1, j;
    private boolean sorting = false;
    private int stepCount = 1;
    private int minIndex;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    BubbleSortGUI frame = new BubbleSortGUI();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public BubbleSortGUI() {
        setTitle("Bubble Sort Langkah per Langkah");
        setSize(750, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
```

```
        // Panel Input
        JPanel inputPanel = new JPanel(new FlowLayout());
        inputField = new JTextField(30);
        setButton = new JButton("Set Array");
        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma:"));
        inputPanel.add(inputField);
        inputPanel.add(setButton);

        // Panel Array visual
        panelArray = new JPanel();
        panelArray.setLayout(new FlowLayout());

        // Panel Kontrol
        JPanel controlPanel = new JPanel();
        stepButton = new JButton("Langkah Selanjutnya");
        resetButton = new JButton("Reset");
        stepButton.setEnabled(false);
        controlPanel.add(stepButton);
        controlPanel.add(resetButton);

        // Area Teks untuk log langkah-langkah
        stepArea = new JTextArea(8, 60);
        stepArea.setEditable(false);
        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(stepArea);

        // Tambahkan Panel ke Frame
        add(inputPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(scrollPane, BorderLayout.EAST);

        // Event Set Array
        setButton.addActionListener(e -> setArrayFromInput());

//      // Event Langkah Selanjutnya
        stepButton.addActionListener(e -> performStep());
//
//      // Event Reset
//      resetButton.addActionListener(e -> reset());


    }
    private void setArrayFromInput() {
        String text = inputField.getText().trim();
        if (text.isEmpty()) return;
        String[] parts = text.split(",");
```

```
        array = new int[parts.length];
        try {
            for (int k = 0; k < parts.length; k++) {
                array[k] = Integer.parseInt(parts[k].trim());
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Masukkan hanya angka "
                    + "yang dipisahkan koma!", "Error",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        i = 0;
        j = 0;
        stepCount = 1;
        sorting = true;
        stepButton.setEnabled(true);
        stepArea.setText("");
        panelArray.removeAll();
        labelArray = new JLabel[array.length];
        for (int k = 0; k < array.length; k++) {
            labelArray[k] = new JLabel(String.valueOf(array[k]));
            labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
            labelArray[k].setOpaque(true);
            labelArray[k].setBackground(Color.WHITE);
            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLA
CK));
            labelArray[k].setPreferredSize(new Dimension(50, 50));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
            panelArray.add(labelArray[k]);
        }

        panelArray.revalidate();
        panelArray.repaint();


    }
    private void performStep() {
        if (!sorting || i >= array.length - 1) {
            sorting = false;
            stepButton.setEnabled(false);
            JOptionPane.showMessageDialog(this, "Sorting selesai!");
            return;
        }

        resetHighlights();
        StringBuilder stepLog = new StringBuilder();
        labelArray[j].setBackground(Color.CYAN);
```

```
        labelArray[j + 1].setBackground(Color.CYAN);

        if (array[j] > array[j + 1]) {
            // Swap
            int temp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = temp;
            labelArray[j].setBackground(Color.RED);
            labelArray[j + 1].setBackground(Color.RED);
            stepLog.append("Langkah ").append(stepCount).append(": Menukar
elemen ke-")
                    .append(j).append(" (").append(array[j + 1]).append(")
dengan ke-")
                    .append(j + 1).append("
(").append(array[j]).append(")\n");
        } else {
            stepLog.append("Langkah ").append(stepCount).append(": Tidak ada
pertukaran antara ke-")
                    .append(j).append(" dan ke-").append(j + 1).append("\n");
        }

        stepLog.append("Hasil:
").append(arrayToString(array)).append("\n\n");
        stepArea.append(stepLog.toString());
        updateLabels();
        j++;

        if (j >= array.length - i - 1) {
            j = 0;
            i++;
        }

        stepCount++;
        if (i >= array.length - 1) {
            sorting = false;
            stepButton.setEnabled(false);
            JOptionPane.showMessageDialog(this, "Sorting selesai!");
        }
    }
    private void updateLabels() {
        for (int k = 0; k < array.length; k++) {
            labelArray[k].setText(String.valueOf(array[k]));
        }
    }

    private void resetHighlights() {
        for (JLabel label : labelArray) {
            label.setBackground(Color.WHITE);
```

```
        }
    }

    private void reset() {
        inputField.setText("");
        panelArray.removeAll();
        panelArray.revalidate();
        panelArray.repaint();
        stepArea.setText("");
        stepButton.setEnabled(false);
        sorting = false;
        i = 0;
        j = 0;
        stepCount = 1;
    }

    private String arrayToString(int[] arr) {
        StringBuilder sb = new StringBuilder();
        for (int k = 0; k < arr.length; k++) {
            sb.append(arr[k]);
            if (k < arr.length - 1) sb.append(", ");
        }
        return sb.toString();
    }



}
```

B. Class : MergeSortGUI.java

a) Tujuan Program:

Aplikasi GUI Java untuk melakukan algoritma **Merge Sort** secara langkah per langkah.

b) Struktur dan Penjelasan Komponen Program:

1. package pekan8;
    - Menandakan file berada dalam folder atau package pekan8.
2. import statements
    - Mengimpor kelas-kelas dari pustaka AWT dan Swing untuk membangun antarmuka grafis.
    - Queue, LinkedList, dan Arrays digunakan untuk struktur data dan manipulasi array.
3. public class MergeSortGUI extends JFrame
    - Kelas utama GUI yang merupakan turunan dari JFrame (jendela GUI utama).
4. Deklarasi Variabel:
    - int[] array: Menyimpan data angka yang akan disorting oleh algoritma Merge Sort.
    - JLabel labelArray: Array label yang merepresentasikan elemen-elemen array secara visual di antarmuka pengguna (GUI).
    - JButton stepButton, resetButton, setButton: Tombol interaktif yang digunakan untuk melakukan aksi:
    - stepButton: Menjalankan satu langkah Merge Sort.
    - resetButton: Mengembalikan GUI ke kondisi awal.
    - setButton: Membaca input dari user dan menginisialisasi array.
    - JTextField inputField: Kolom input bagi pengguna untuk memasukkan data array dalam format angka yang dipisahkan dengan koma.
    - JPanel panelArray: Panel utama yang menampilkan visualisasi array (label-label angka).
    - JTextArea stepArea: Area teks yang digunakan untuk menampilkan log atau catatan langkah-langkah sorting secara real-time.
    - Queue<int[]> mergeQueue: Antrian (queue) yang menyimpan urutan operasi merge dalam bentuk pasangan indeks [left, mid, right] untuk mengatur proses penggabungan sub-array dalam Merge Sort.
    - int left, mid, right, i, j, k: Variabel-variabel indeks untuk mengelola batas dan iterasi selama proses merge:

- left, mid, right: Batas kiri, tengah, dan kanan dari sub-array.
- i, j: Penunjuk elemen pada bagian kiri dan kanan yang sedang dibandingkan.
- k: Indeks penyalinan ke array sementara temp.
- int[] temp: Array sementara yang digunakan dalam proses penggabungan dua sub-array (merge) sebelum hasil disalin kembali ke array utama.
- boolean isMerging, copying: Flag (penanda status):
- isMerging: Menandai apakah sedang dalam proses merge aktif.
- copying: Menandai apakah sedang dalam tahap penyalinan dari array sementara ke array utama.
- int stepCount: Menyimpan jumlah total langkah yang telah dilakukan selama proses sorting untuk ditampilkan di log.

5. Method main()
   - Memulai program dengan EventQueue.invokeLater() untuk menjalankan GUI secara thread-safe.
6. Konstruktor MergeSortGUI()

   Menyusun elemen GUI:

   - **Panel Input (Utara):**
     - inputField dan tombol Set Array.
   - **Panel Array (Tengah):**
     - Visualisasi elemen array (JLabel).
   - **Panel Kontrol (Selatan):**
     - Tombol Langkah Selanjutnya dan Reset.
   - **Panel Log (Timur):**
     - stepArea untuk menampilkan log langkah-langkah secara detail.
7. Fungsi-Fungsi Utama:

   setArrayFromInput()

   - Membaca input user, mengubah menjadi int[], menampilkan label, dan memulai urutan merge:
     - Menyiapkan mergeQueue berisi potongan (sub-array) yang perlu di-merge berdasarkan rekursi.

     generateMergeSteps(int l, int r)

   - Membentuk antrian langkah-langkah merge (bottom-up).
   - Menyimpan triplet [left, mid, right] ke dalam mergeQueue.

performStep()

- Inti dari simulasi Merge Sort satu langkah per klik:
  - **Tahap 1 (ambil range):** Ambil rentang dari mergeQueue.
  - **Tahap 2 (merge):** Bandingkan dan salin elemen dari dua sub-array ke temp[].
  - **Tahap 3 (copy back):** Salin isi temp[] ke array asli.
  - Menampilkan semua proses secara real-time di stepArea.

logStep(String message)

- Menuliskan informasi langkah ke dalam stepArea.

updateLabels()

- Memperbarui isi JLabel sesuai isi array.

resetHighlights(), highlightRange(), highlightCompare(), highlightCopy()

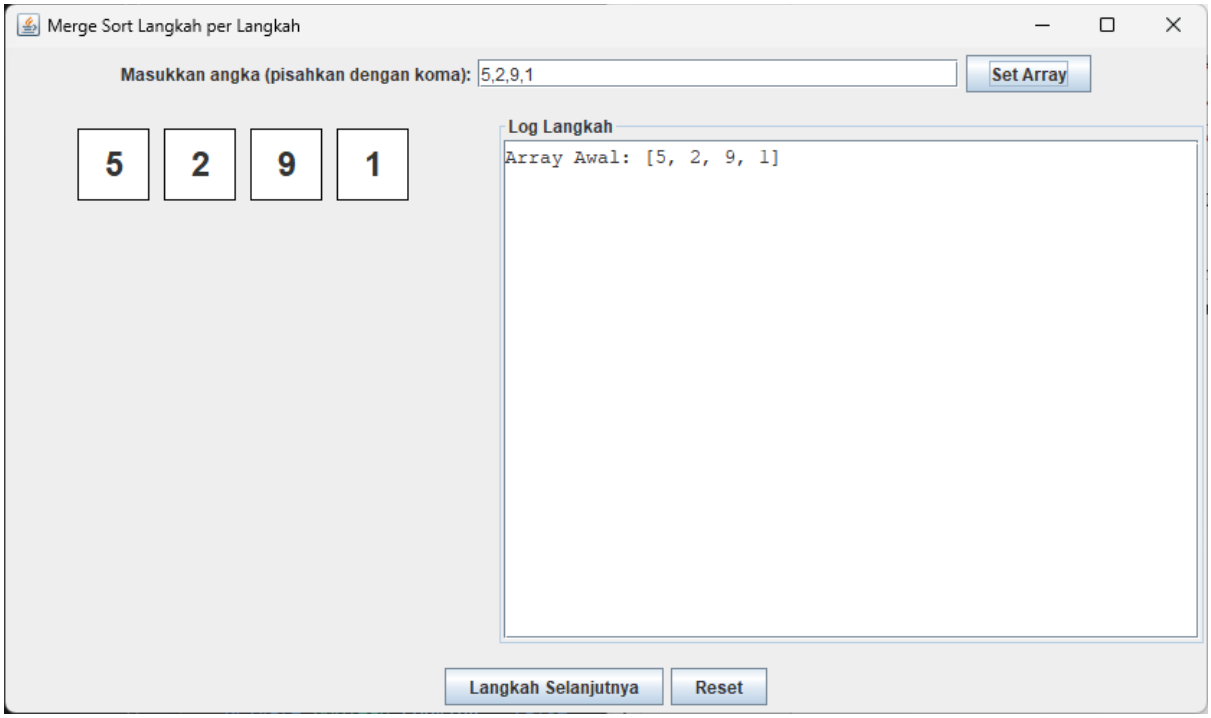
- Mengelola warna label untuk menunjukkan status (dibandingkan, disalin, selesai).

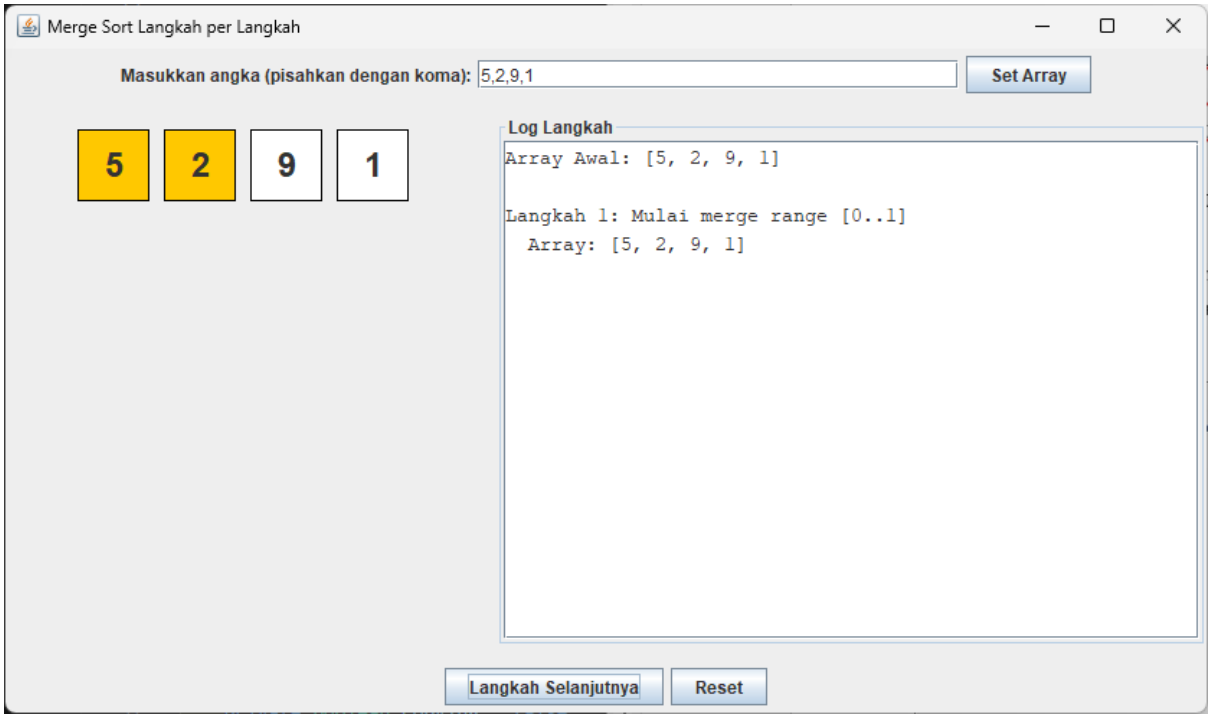
reset()

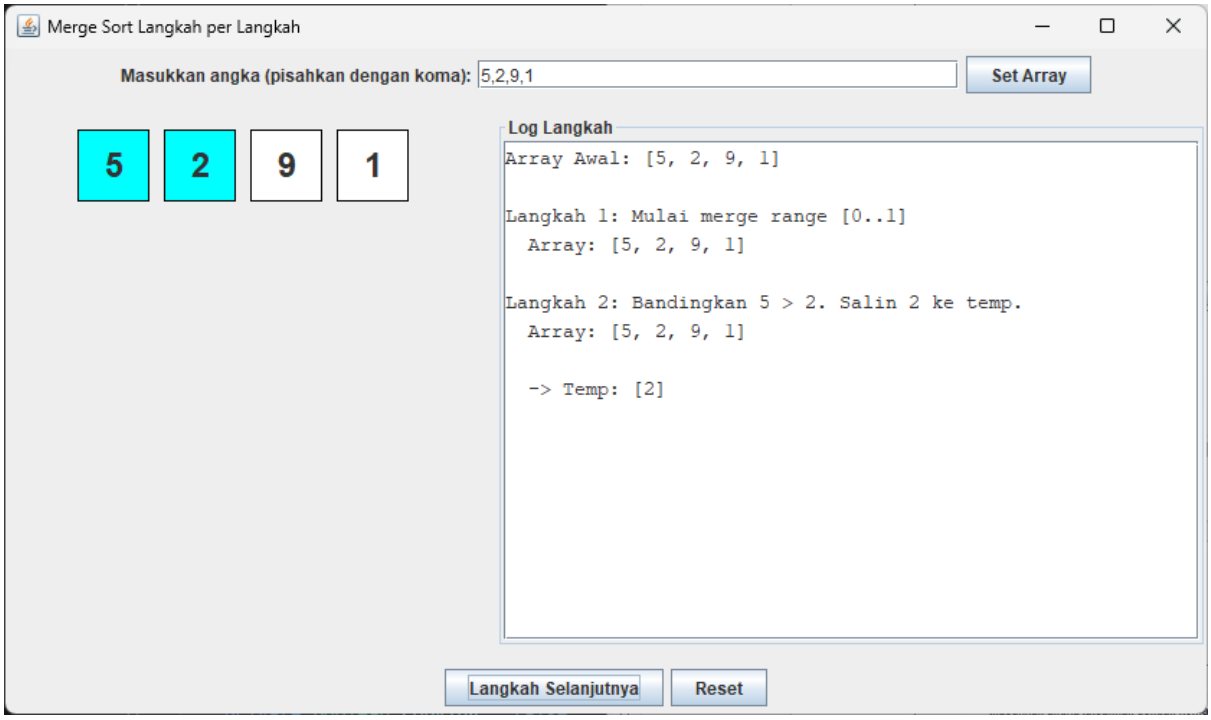
- Mengembalikan seluruh GUI ke kondisi awal.

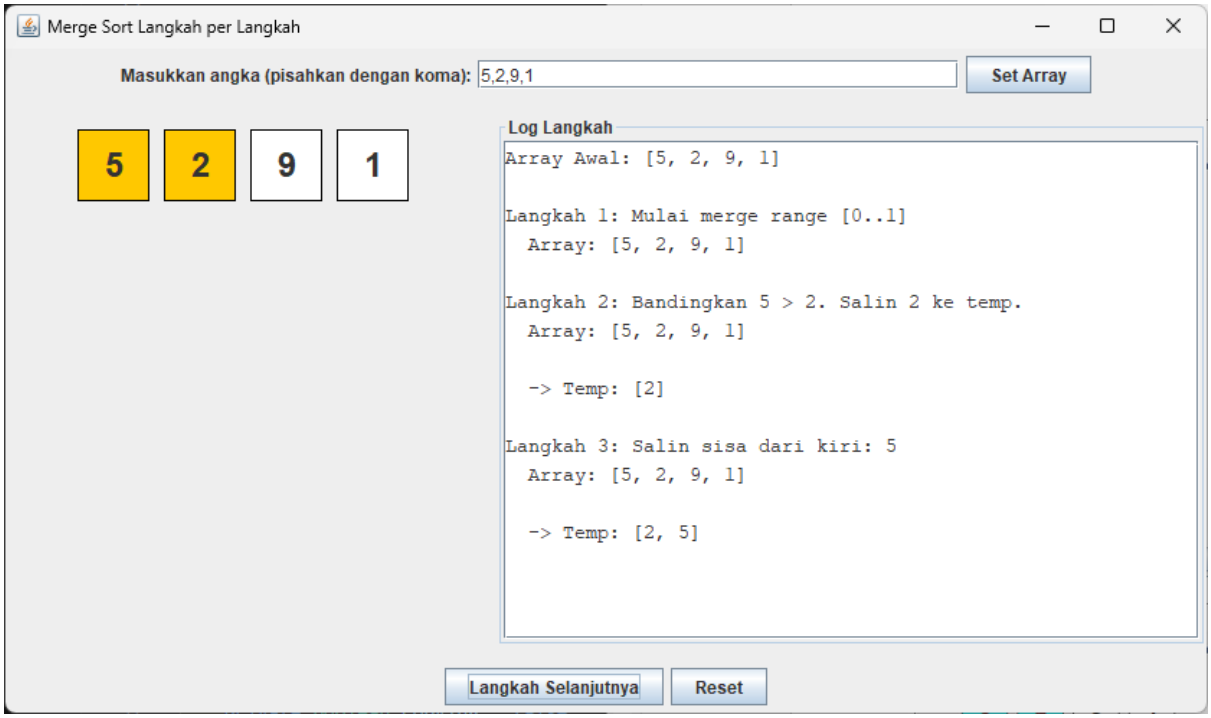
c) Fitur Utama Program:

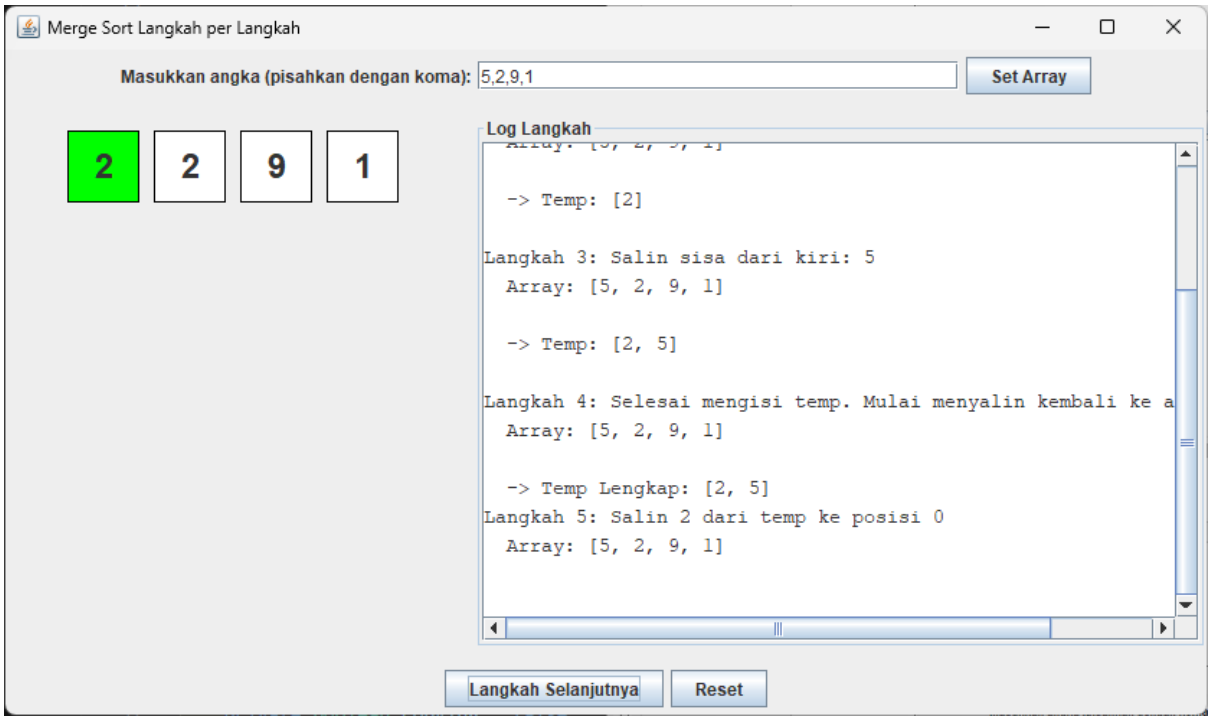
- Pengguna dapat memasukkan angka seperti 5,2,9,1.
- Dapat menjalankan langkah sorting satu demi satu (step-by-step).
- Menampilkan log proses sorting di sisi kanan aplikasi secara real time.

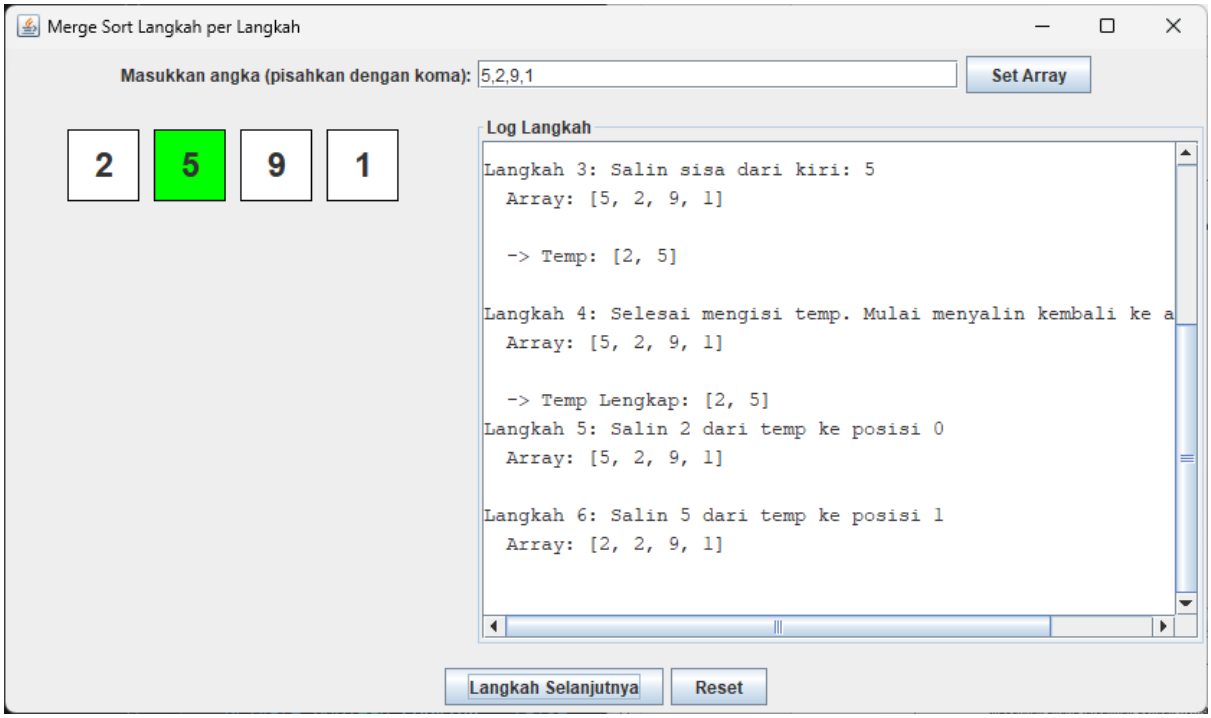


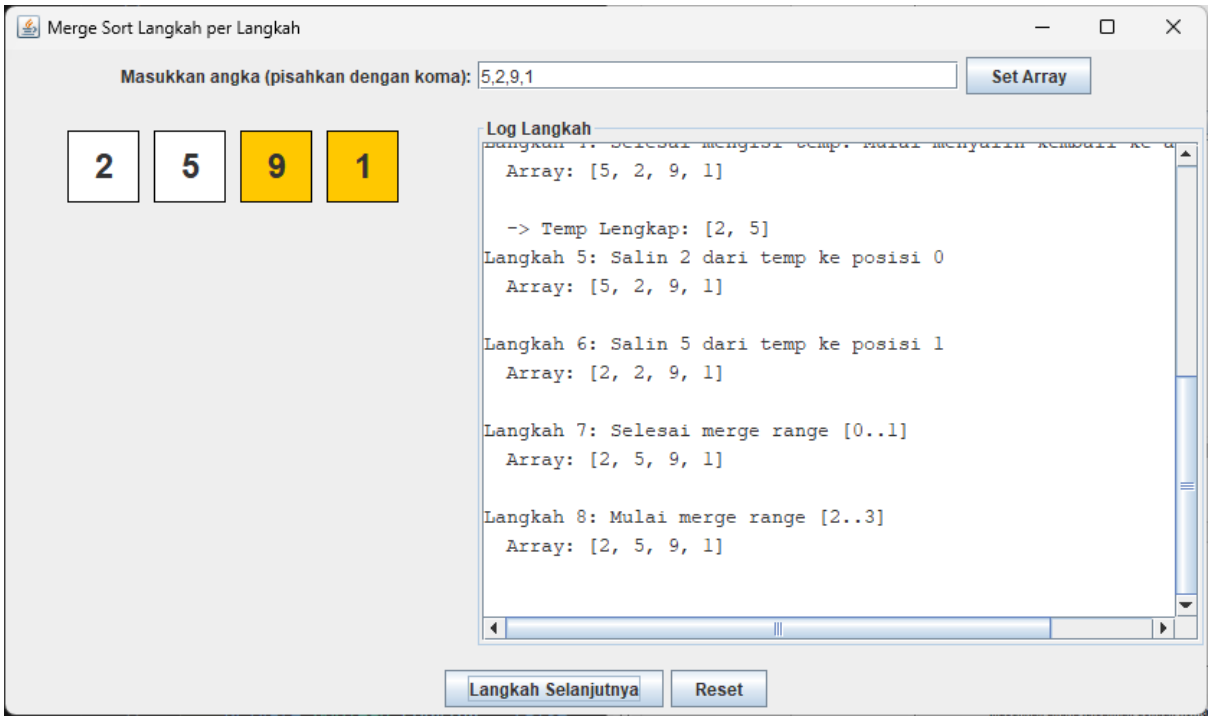

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
Langkah 1: Mulai merge range [0..1]
Array: [5, 2, 9, 1]
Langkah Selanjutnya
Reset

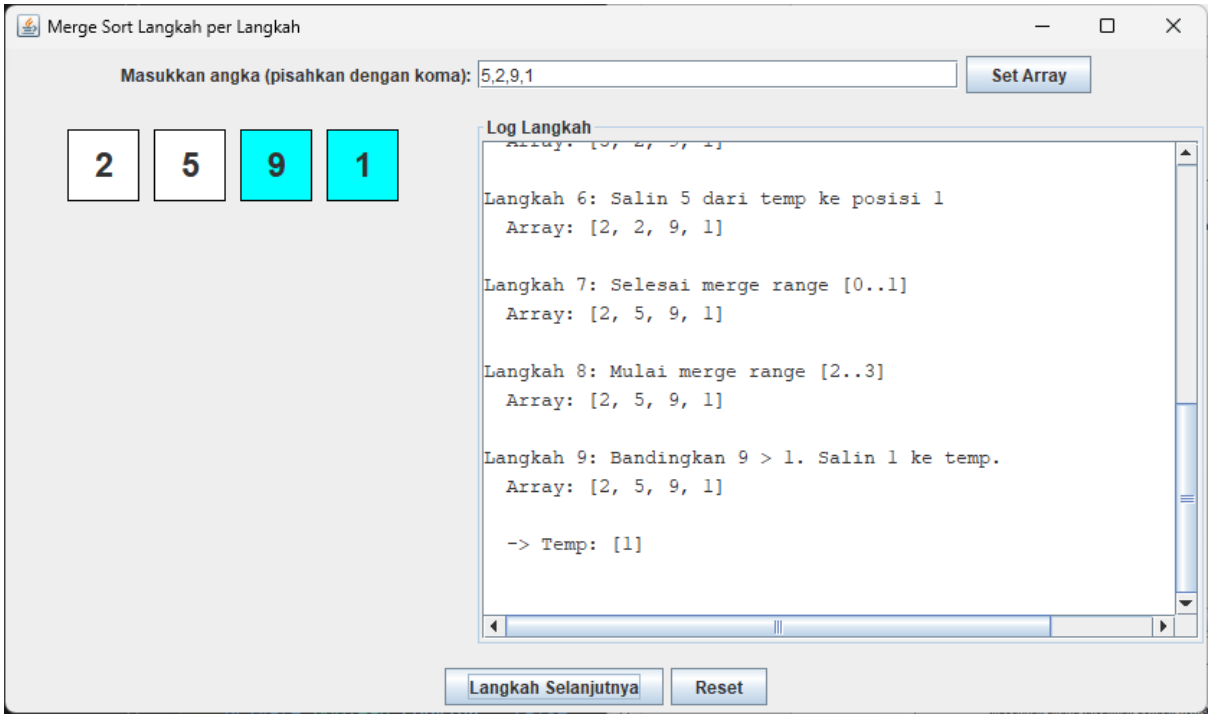

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
Langkah 1: Mulai merge range [0..1]
Array: [5, 2, 9, 1]
Langkah 2: Bandingkan 5 > 2. Salin 2 ke temp.
Array: [5, 2, 9, 1]
-> Temp: [2]
Langkah Selanjutnya
Reset

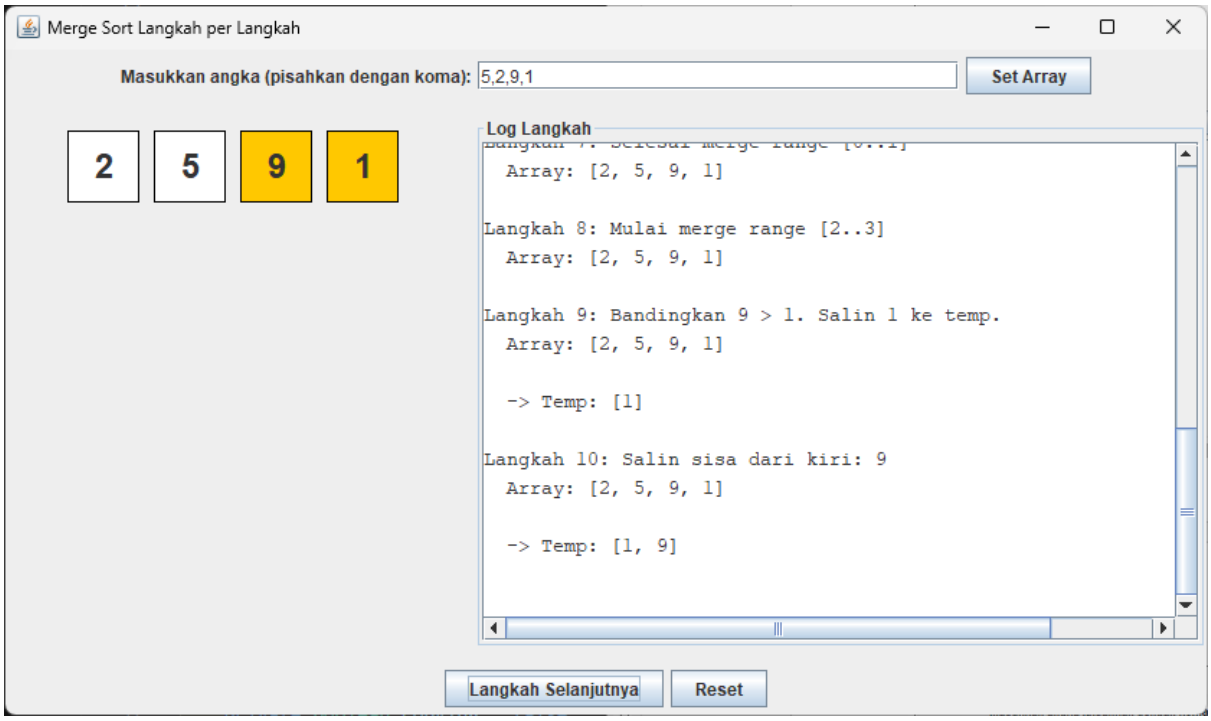

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
Langkah 1: Mulai merge range [0..1]
Array: [5, 2, 9, 1]
Langkah 2: Bandingkan 5 > 2. Salin 2 ke temp.
Array: [5, 2, 9, 1]
-> Temp: [2]
Langkah 3: Salin sisa dari kiri: 5
Array: [5, 2, 9, 1]
-> Temp: [2, 5]
Langkah Selanjutnya
Reset

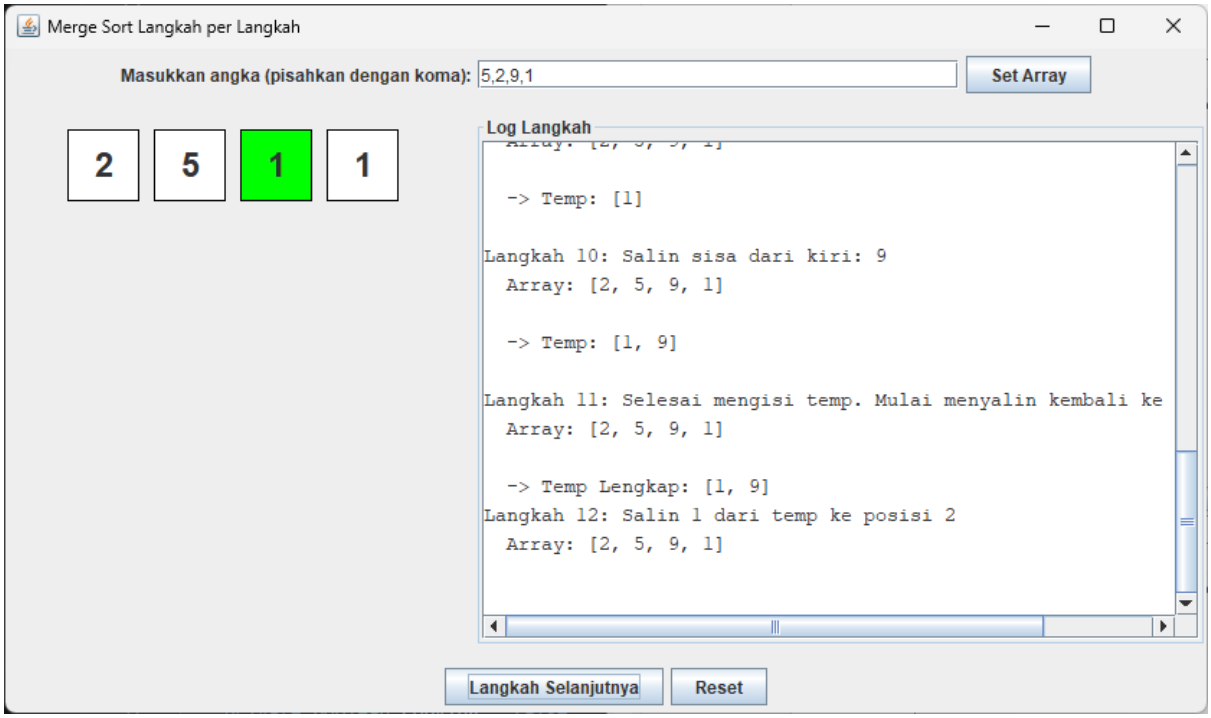

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
2
9
1
Log Langkah
-> Temp: [2]
Langkah 3: Salin sisa dari kiri: 5
Array: [5, 2, 9, 1]
-> Temp: [2, 5]
Langkah 4: Selesai mengisi temp. Mulai menyalin kembali ke a
Array: [5, 2, 9, 1]
-> Temp Lengkap: [2, 5]
Langkah 5: Salin 2 dari temp ke posisi 0
Array: [5, 2, 9, 1]
Langkah Selanjutnya
Reset

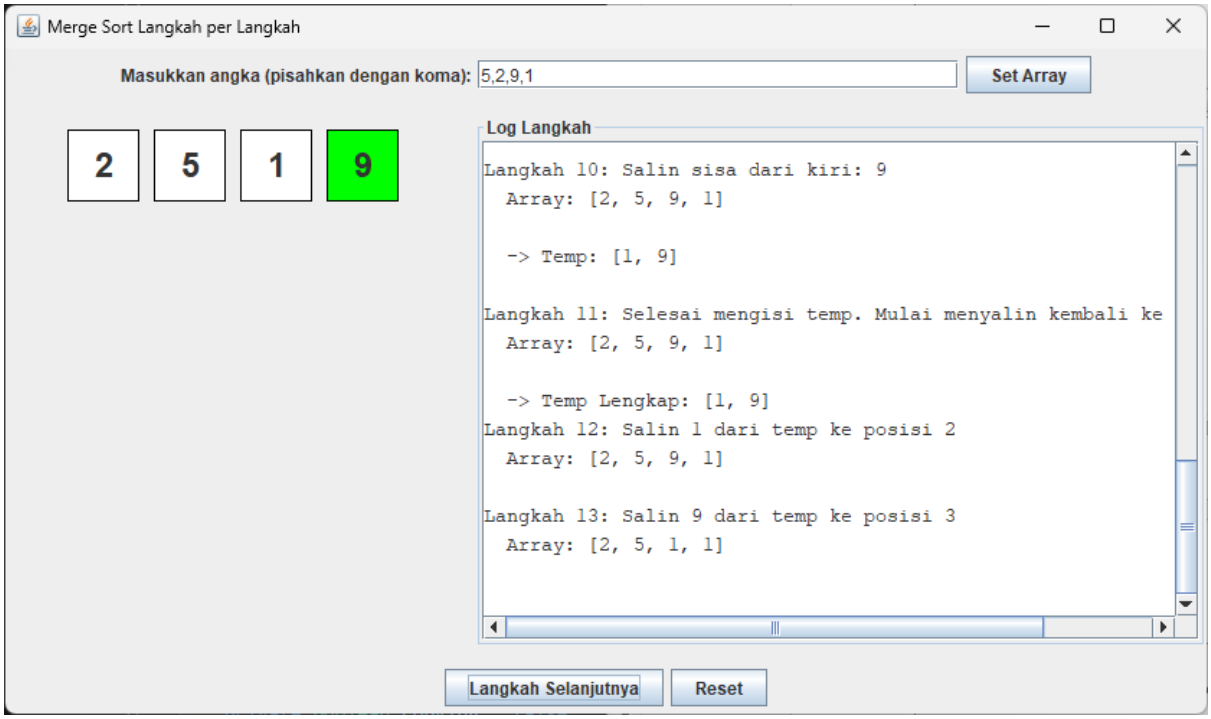

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
9
1
Log Langkah
Langkah 3: Salin sisa dari kiri: 5
Array: [5, 2, 9, 1]
-> Temp: [2, 5]
Langkah 4: Selesai mengisi temp. Mulai menyalin kembali ke a
Array: [5, 2, 9, 1]
-> Temp Lengkap: [2, 5]
Langkah 5: Salin 2 dari temp ke posisi 0
Array: [5, 2, 9, 1]
Langkah 6: Salin 5 dari temp ke posisi 1
Array: [2, 2, 9, 1]
Langkah Selanjutnya
Reset

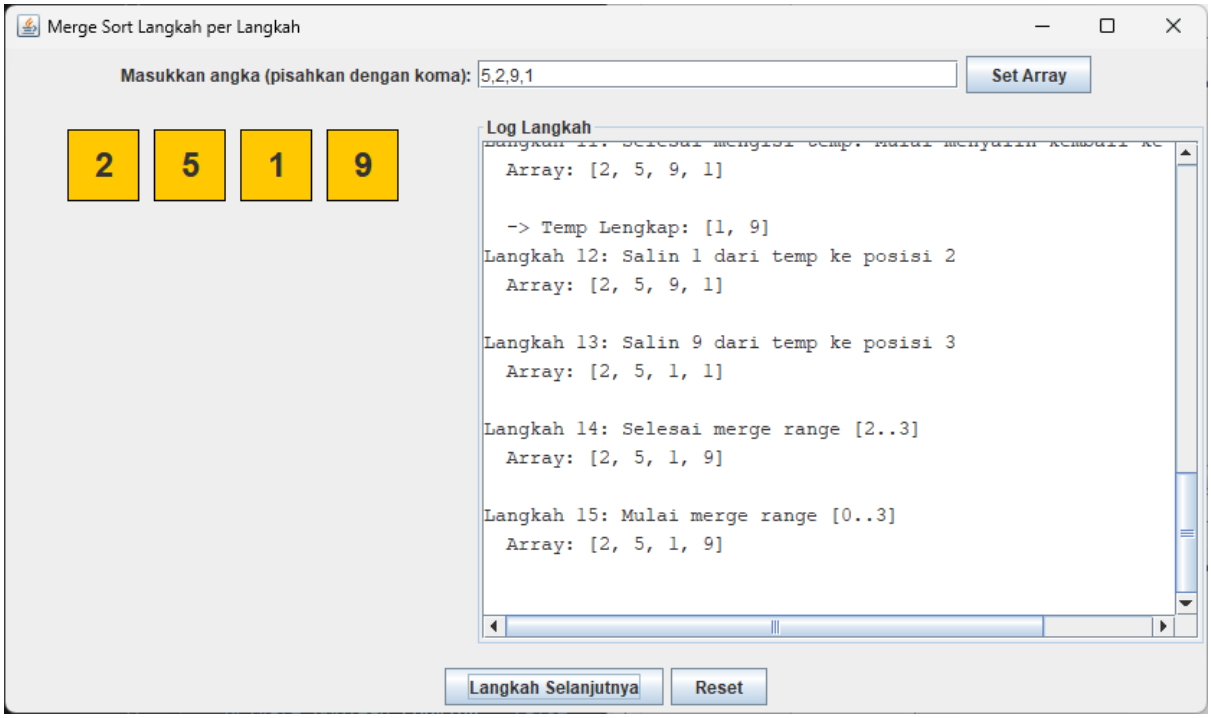

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
9
1
Log Langkah
Array: [5, 2, 9, 1]
-> Temp Lengkap: [2, 5]
Langkah 5: Salin 2 dari temp ke posisi 0
Array: [5, 2, 9, 1]
Langkah 6: Salin 5 dari temp ke posisi 1
Array: [2, 2, 9, 1]
Langkah 7: Selesai merge range [0..1]
Array: [2, 5, 9, 1]
Langkah 8: Mulai merge range [2..3]
Array: [2, 5, 9, 1]
Langkah Selanjutnya
Reset

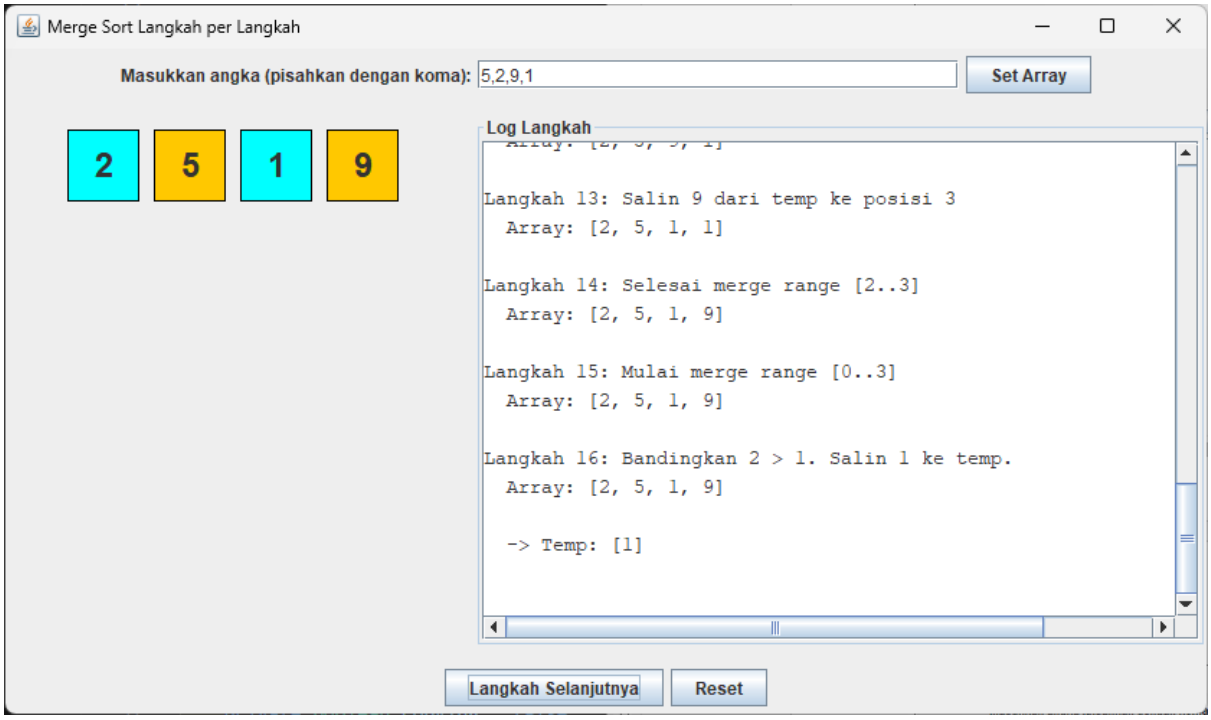

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
9
1
Log Langkah
Langkah 6: Salin 5 dari temp ke posisi 1
Array: [2, 2, 9, 1]
Langkah 7: Selesai merge range [0..1]
Array: [2, 5, 9, 1]
Langkah 8: Mulai merge range [2..3]
Array: [2, 5, 9, 1]
Langkah 9: Bandingkan 9 > 1. Salin 1 ke temp.
Array: [2, 5, 9, 1]
-> Temp: [1]
Langkah Selanjutnya
Reset

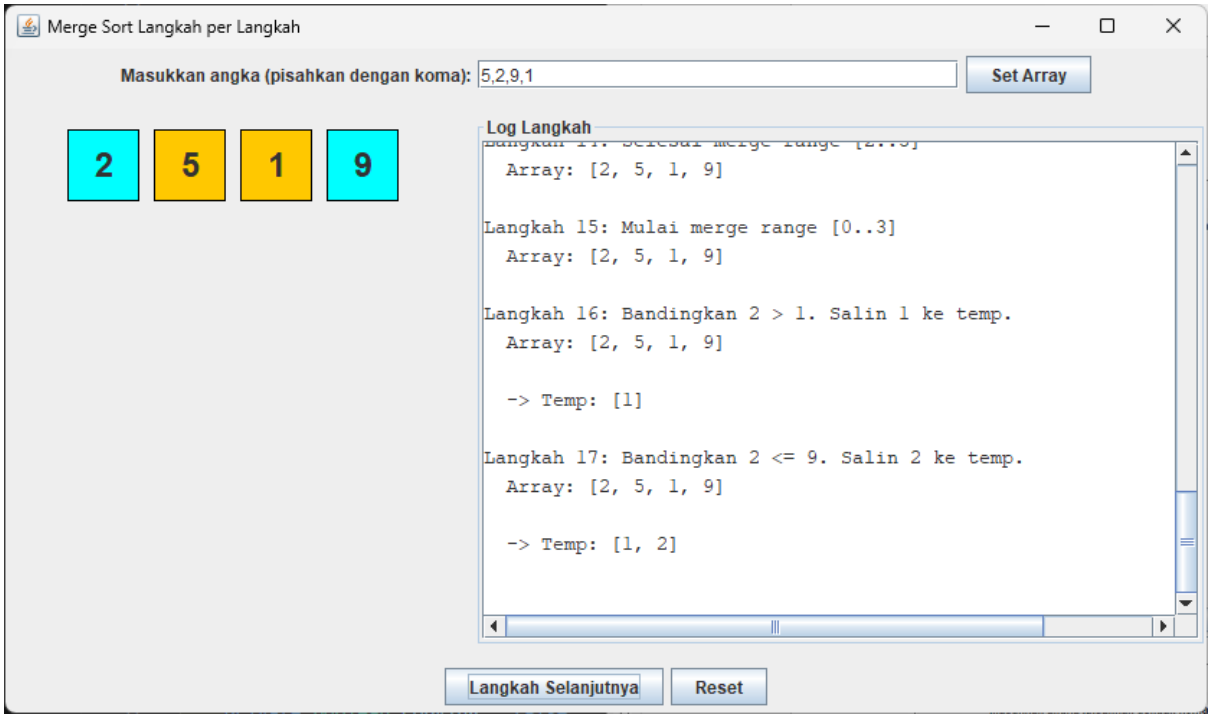

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
9
1
Log Langkah
Array: [2, 5, 9, 1]
Langkah 8: Mulai merge range [2..3]
Array: [2, 5, 9, 1]
Langkah 9: Bandingkan 9 > 1. Salin 1 ke temp.
Array: [2, 5, 9, 1]
-> Temp: [1]
Langkah 10: Salin sisa dari kiri: 9
Array: [2, 5, 9, 1]
-> Temp: [1, 9]
Langkah Selanjutnya
Reset

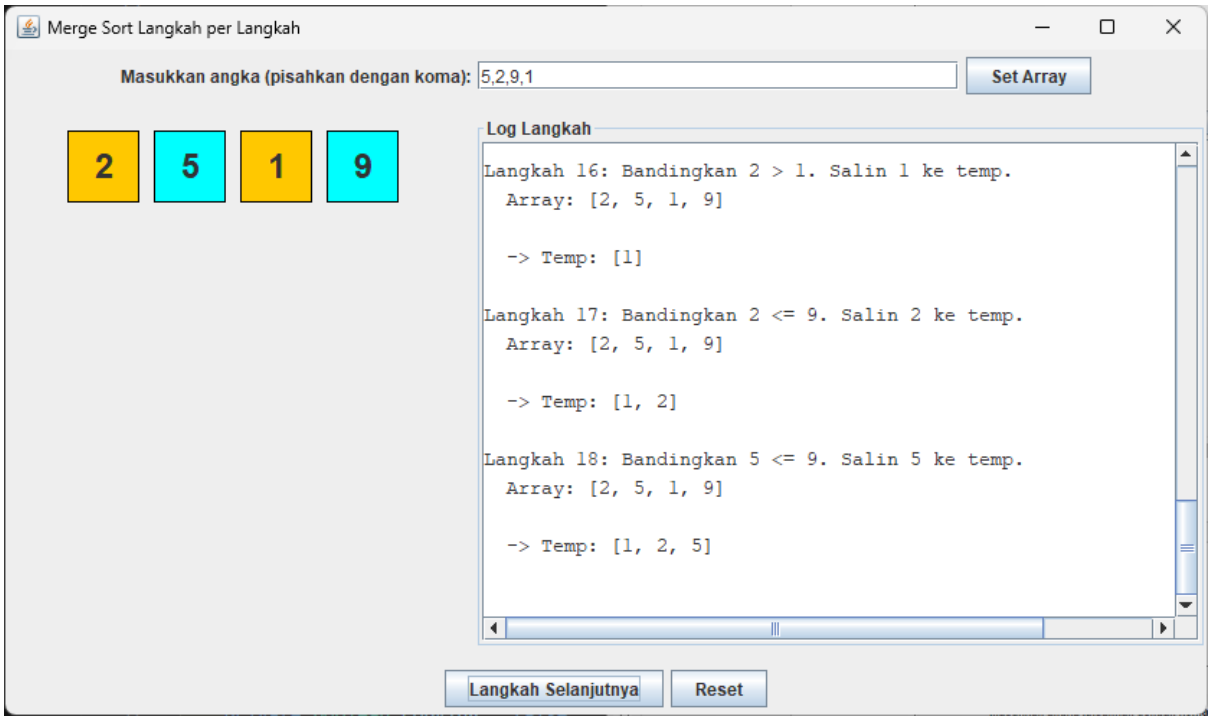

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
1
Log Langkah
-> Temp: [1]
Langkah 10: Salin sisa dari kiri: 9
Array: [2, 5, 9, 1]
-> Temp: [1, 9]
Langkah 11: Selesai mengisi temp. Mulai menyalin kembali ke
Array: [2, 5, 9, 1]
-> Temp Lengkap: [1, 9]
Langkah 12: Salin 1 dari temp ke posisi 2
Array: [2, 5, 9, 1]
Langkah Selanjutnya
Reset

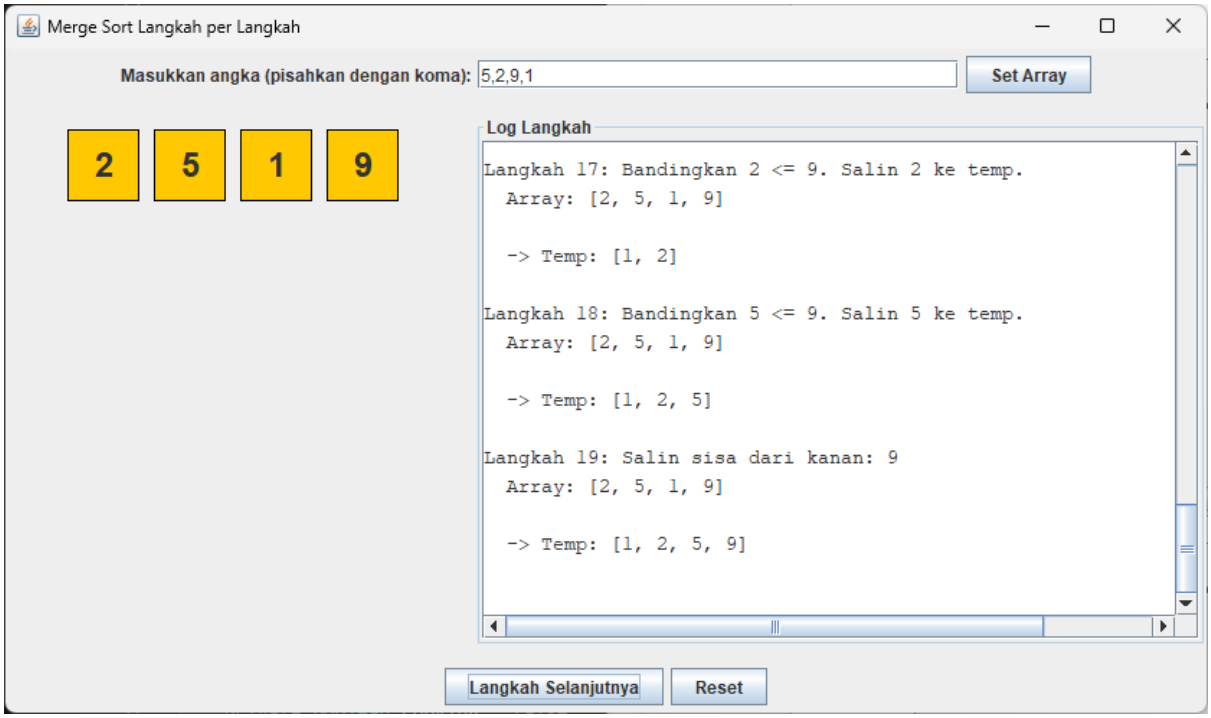

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Langkah 10: Salin sisa dari kiri: 9
Array: [2, 5, 9, 1]
-> Temp: [1, 9]
Langkah 11: Selesai mengisi temp. Mulai menyalin kembali ke
Array: [2, 5, 9, 1]
-> Temp Lengkap: [1, 9]
Langkah 12: Salin 1 dari temp ke posisi 2
Array: [2, 5, 9, 1]
Langkah 13: Salin 9 dari temp ke posisi 3
Array: [2, 5, 1, 1]
Langkah Selanjutnya
Reset

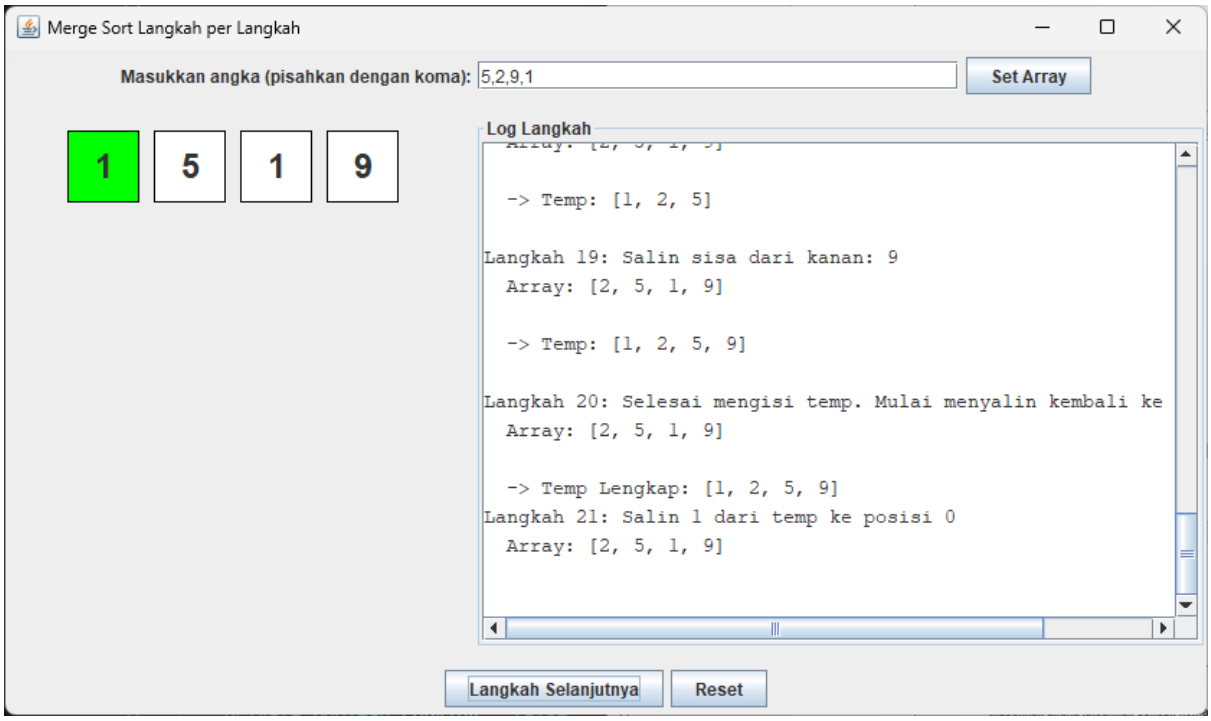

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Array: [2, 5, 9, 1]
-> Temp Lengkap: [1, 9]
Langkah 12: Salin 1 dari temp ke posisi 2
Array: [2, 5, 9, 1]
Langkah 13: Salin 9 dari temp ke posisi 3
Array: [2, 5, 1, 1]
Langkah 14: Selesai merge range [2..3]
Array: [2, 5, 1, 9]
Langkah 15: Mulai merge range [0..3]
Array: [2, 5, 1, 9]
Langkah Selanjutnya
Reset

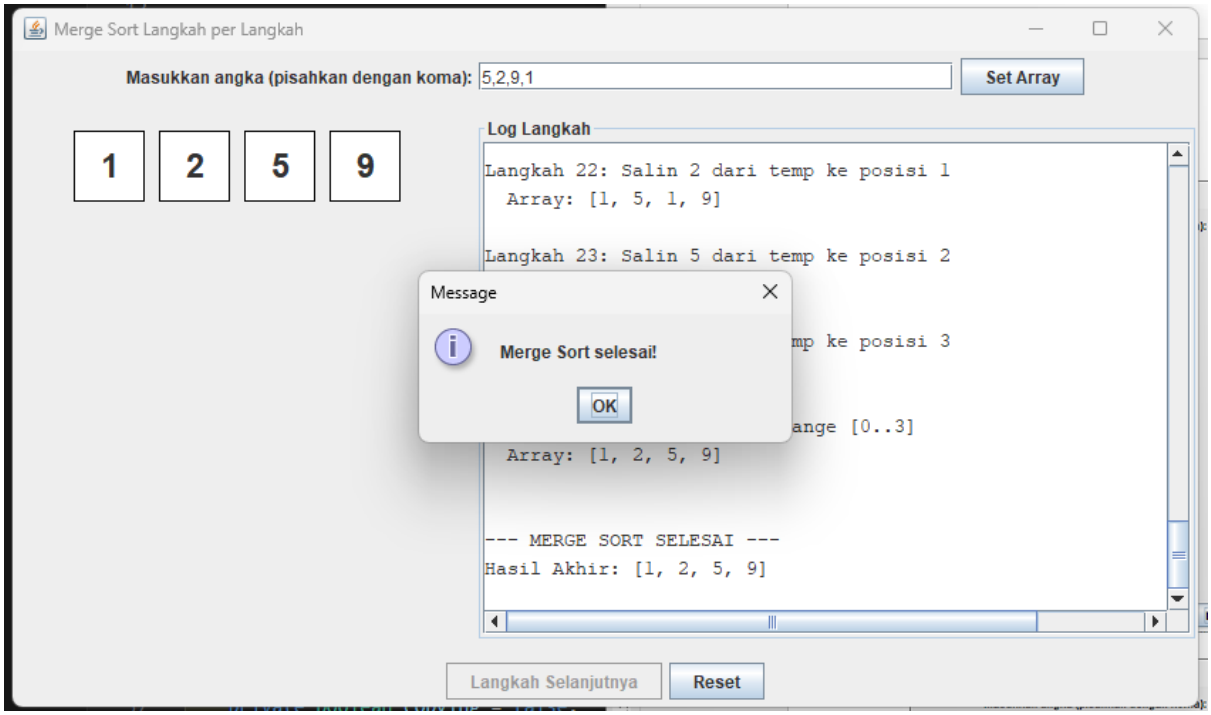

Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Langkah 13: Salin 9 dari temp ke posisi 3
Array: [2, 5, 1, 1]
Langkah 14: Selesai merge range [2..3]
Array: [2, 5, 1, 9]
Langkah 15: Mulai merge range [0..3]
Array: [2, 5, 1, 9]
Langkah 16: Bandingkan 2 > 1. Salin 1 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1]
Langkah Selanjutnya
Reset


Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Array: [2, 5, 1, 9]
Langkah 15: Mulai merge range [0..3]
Array: [2, 5, 1, 9]
Langkah 16: Bandingkan 2 > 1. Salin 1 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1]
Langkah 17: Bandingkan 2 <= 9. Salin 2 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1, 2]
Langkah Selanjutnya
Reset


Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Langkah 16: Bandingkan 2 > 1. Salin 1 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1]
Langkah 17: Bandingkan 2 <= 9. Salin 2 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1, 2]
Langkah 18: Bandingkan 5 <= 9. Salin 5 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1, 2, 5]
Langkah Selanjutnya
Reset


Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
2
5
1
9
Log Langkah
Langkah 17: Bandingkan 2 <= 9. Salin 2 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1, 2]
Langkah 18: Bandingkan 5 <= 9. Salin 5 ke temp.
Array: [2, 5, 1, 9]
-> Temp: [1, 2, 5]
Langkah 19: Salin sisa dari kanan: 9
Array: [2, 5, 1, 9]
-> Temp: [1, 2, 5, 9]
Langkah Selanjutnya
Reset


Merge Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
1
5
1
9
Log Langkah
-> Temp: [1, 2, 5]
Langkah 19: Salin sisa dari kanan: 9
Array: [2, 5, 1, 9]
-> Temp: [1, 2, 5, 9]
Langkah 20: Selesai mengisi temp. Mulai menyalin kembali ke
Array: [2, 5, 1, 9]
-> Temp Lengkap: [1, 2, 5, 9]
Langkah 21: Salin 1 dari temp ke posisi 0
Array: [2, 5, 1, 9]
Langkah Selanjutnya
Reset




```
package pekan8;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.FlowLayout;
import java.awt.Font;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Queue;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class MergeSortGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
```

```
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;

    private int stepCount = 1;
    private Queue<int[]> mergeQueue;
    private boolean isMerging = false;
    private boolean copying = false;

    private int left, mid, right;
    private int i, j, k;
    private int[] temp;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                MergeSortGUI frame = new MergeSortGUI();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public MergeSortGUI() {
        setTitle("Merge Sort Langkah per Langkah");
        setSize(850, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        mergeQueue = new LinkedList<>();

        JPanel topPanel = new JPanel(new BorderLayout());
        JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        inputField = new JTextField(30);
        setButton = new JButton("Set Array");
        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan
koma):"));
        inputPanel.add(inputField);
        inputPanel.add(setButton);
        topPanel.add(inputPanel, BorderLayout.CENTER);

        panelArray = new JPanel();
        panelArray.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JPanel controlPanel = new JPanel();
        stepButton = new JButton("Langkah Selanjutnya");
```

```
        resetButton = new JButton("Reset");
        stepButton.setEnabled(false);
        controlPanel.add(stepButton);
        controlPanel.add(resetButton);

        JPanel logPanel = new JPanel(new BorderLayout());
        logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));
        stepArea = new JTextArea(8, 60);
        stepArea.setEditable(false);
        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(stepArea);
        logPanel.add(scrollPane, BorderLayout.CENTER);

        add(topPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(logPanel, BorderLayout.EAST);

        setButton.addActionListener(e -> setArrayFromInput());
        stepButton.addActionListener(e -> performStep());
        resetButton.addActionListener(e -> reset());
    }

    private void performStep() {
        resetHighlights();

        if (!isMerging && !mergeQueue.isEmpty()) {
            int[] range = mergeQueue.poll();
            left = range[0];
            mid = range[1];
            right = range[2];

            temp = new int[right - left + 1];
            i = left;
            j = mid + 1;
            k = 0;
            copying = false;
            isMerging = true;

            logStep("Mulai merge range [" + left + ".." + right + "]");
            highlightRange(left, right);
            return;
        }

        if (isMerging && !copying) {
            highlightRange(left, right);

            if (i <= mid && j <= right) {
```

```
                highlightCompare(i, j);
                if (array[i] <= array[j]) {
                    logStep("Bandingkan " + array[i] + " <= " + array[j] +
". Salin " + array[i] + " ke temp.");
                    temp[k++] = array[i++];
                } else {
                    logStep("Bandingkan " + array[i] + " > " + array[j] + ".
Salin " + array[j] + " ke temp.");
                    temp[k++] = array[j++];
                }
                stepArea.append("  -> Temp: " +
Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
                return;
            }

            if (i <= mid) {
                logStep("Salin sisa dari kiri: " + array[i]);
                temp[k++] = array[i++];
                stepArea.append("  -> Temp: " +
Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
                return;
            }

            if (j <= right) {
                logStep("Salin sisa dari kanan: " + array[j]);
                temp[k++] = array[j++];
                stepArea.append("  -> Temp: " +
Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
                return;
            }

            copying = true;
            k = 0;
            logStep("Selesai mengisi temp. Mulai menyalin kembali ke array
utama.");
            stepArea.append("  -> Temp Lengkap: " + Arrays.toString(temp) +
"\n");
            return;
        }

        if (isMerging && copying) {
            if (k < temp.length) {
                logStep("Salin " + temp[k] + " dari temp ke posisi " + (left
+ k));
                array[left + k] = temp[k];
                updateLabels();
                highlightCopy(left + k);
                k++;
```

```
                return;
            } else {
                isMerging = false;
                copying = false;
                logStep("Selesai merge range [" + left + ".." + right +
"]");

                if (mergeQueue.isEmpty() && !isMerging) {
                    stepArea.append("\n--- MERGE SORT SELESAI ---\n");
                    stepArea.append("Hasil Akhir: " + Arrays.toString(array)
+ "\n");
                    stepButton.setEnabled(false);
                    JOptionPane.showMessageDialog(this, "Merge Sort
selesai!");
                    for (JLabel label : labelArray)
                        label.setBackground(Color.LIGHT_GRAY);
                }
            }
        }
    }

    private void setArrayFromInput() {
        String text = inputField.getText().trim();
        if (text.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String[] parts = text.split(",");
        array = new int[parts.length];
        try {
            for (int i = 0; i < parts.length; i++) {
                array[i] = Integer.parseInt(parts[i].trim());
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Masukkan hanya angka!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        stepCount = 1;
        isMerging = false;
        copying = false;
        if (mergeQueue != null) {
            mergeQueue.clear();
        }

        panelArray.removeAll();
```

```
        stepArea.setText("");

        labelArray = new JLabel[array.length];
        for (int i = 0; i < array.length; i++) {
            labelArray[i] = new JLabel(String.valueOf(array[i]));
            labelArray[i].setFont(new Font("Arial", Font.BOLD, 24));
            labelArray[i].setOpaque(true);
            labelArray[i].setBackground(Color.WHITE);
            labelArray[i].setBorder(BorderFactory.createLineBorder(Color.BLA
CK));
            labelArray[i].setPreferredSize(new Dimension(50, 50));
            labelArray[i].setHorizontalAlignment(SwingConstants.CENTER);
            panelArray.add(labelArray[i]);
        }

        generateMergeSteps(0, array.length - 1);

        stepButton.setEnabled(true);
        stepArea.setText("Array Awal: " + Arrays.toString(array) + "\n\n");

        panelArray.revalidate();
        panelArray.repaint();
    }

    private void generateMergeSteps(int l, int r) {
        if (l < r) {
            int m = l + (r - l) / 2;
            generateMergeSteps(l, m);
            generateMergeSteps(m + 1, r);
            mergeQueue.add(new int[] { l, m, r });
        }
    }

    private void updateLabels() {
        for (int i = 0; i < array.length; i++) {
            labelArray[i].setText(String.valueOf(array[i]));
        }
    }

    private void resetHighlights() {
        if (labelArray == null) return;
        for (JLabel label : labelArray) {
            label.setBackground(Color.WHITE);
        }
    }

    private void highlightRange(int l, int r) {
        for (int i = l; i <= r; i++) {
```

```
            labelArray[i].setBackground(Color.ORANGE);
        }
    }

    private void highlightCompare(int i, int j) {
        labelArray[i].setBackground(Color.CYAN);
        labelArray[j].setBackground(Color.CYAN);
    }

    private void highlightCopy(int i) {
        labelArray[i].setBackground(Color.GREEN);
    }

    private void reset() {
        if (inputField != null) inputField.setText("");
        if (panelArray != null) {
            panelArray.removeAll();
            panelArray.revalidate();
            panelArray.repaint();
        }
        if (stepArea != null) stepArea.setText("");
        if (stepButton != null) stepButton.setEnabled(false);
        if (mergeQueue != null) mergeQueue.clear();
        isMerging = false;
        copying = false;
        stepCount = 1;
        array = null;
        labelArray = null;
    }

    private void logStep(String message) {
        stepArea.append("Langkah " + stepCount + ": " + message + "\n");
        if (array != null) {
            stepArea.append("  Array: " + Arrays.toString(array) + "\n\n");
        }
        stepArea.setCaretPosition(stepArea.getDocument().getLength());
        stepCount++;
    }
}
```

C. Class : QuickSortGUI.java

a) Tujuan Program:

Aplikasi GUI Java untuk melakukan algoritma **Quick Sort** secara langkah per langkah.

b) Struktur dan Penjelasan Komponen Program:

1. package pekan8;
    - Menandakan file berada dalam folder atau package pekan8.
2. import statements
    - Mengimpor kelas-kelas dari pustaka AWT dan Swing untuk membangun antarmuka grafis.
    - Queue, LinkedList, dan Arrays digunakan untuk struktur data dan manipulasi array.
3. public class QuickSortGUI extends JFrame
    - Membuat class utama untuk GUI, merupakan turunan dari JFrame sebagai jendela utama antarmuka.
4. Deklarasi Variabel:
    - int[] array: Menyimpan data integer yang akan disorting menggunakan Quick Sort.
    - JLabel[] labelArray: Label-label GUI untuk menampilkan elemen-elemen array secara visual.
    - JButton stepButton, resetButton, setButton: Tiga tombol aksi:
    - stepButton: Jalankan satu langkah Quick Sort.
    - resetButton: Reset GUI ke kondisi awal.
    - setButton: Membaca input array dari user.
    - JTextField inputField: Kolom input untuk memasukkan angka (dipisahkan koma).
    - JPanel panelArray: Panel GUI tempat elemen array divisualisasikan dengan label.
    - JTextArea stepArea: Area teks untuk mencatat setiap langkah proses sorting.
    - int i, j: Indeks yang digunakan selama proses partisi.
    - boolean sorting: Menandai apakah proses sorting sedang aktif.
    - int stepCount: Menghitung dan menampilkan jumlah langkah Quick Sort yang sedang berlangsung.
    - Stack<int[]> stack: Stack menyimpan pasangan indeks low dan high untuk setiap proses partisi yang belum selesai.
    - int low, high, pivot: Batas-batas partisi dan nilai pivot yang sedang diproses.

- boolean partitioning: Status apakah sedang berada dalam tahap partisi dari Quick Sort.

5. Method main()
   - Memulai program dengan EventQueue.invokeLater() untuk menjalankan GUI secara thread-safe.
6. Konstruktor MergeSortGUI()

   Menyusun elemen GUI:

   - **Panel Input (Utara):**
     - inputField dan tombol Set Array.
   - **Panel Array (Tengah):**
     - Visualisasi elemen array (JLabel).
   - **Panel Kontrol (Selatan):**
     - Tombol Langkah Selanjutnya dan Reset.
   - **Panel Log (Timur):**
     - stepArea untuk menampilkan log langkah-langkah secara detail.
7. Fungsi-Fungsi Utama:
   - performStep():

     Melakukan satu langkah partisi Quick Sort:
     - Memilih pivot.Menukar elemen yang lebih kecil ke kiri pivot.
     - Menempatkan pivot ke posisi final.
     - Menambahkan sub-array baru ke stack untuk partisi berikutnya.
   - setArrayFromInput():

     Membaca input string, parsing ke array integer, dan membangun tampilan visual array.
   - updateLabels():

     Memperbarui isi label untuk mencerminkan isi array terbaru setelah pertukaran.
   - resetHighlights(boolean markAsSorted):

     Mengatur ulang warna label. Jika markAsSorted = true, semua label diberi warna abu-abu.
   - highlightPivot(int index), highlightCompare(int jIndex, int pivotIndex):

     Memberi warna khusus untuk pivot dan elemen yang sedang dibandingkan.
   - swap(int a, int b):

     Menukar dua elemen dalam array.

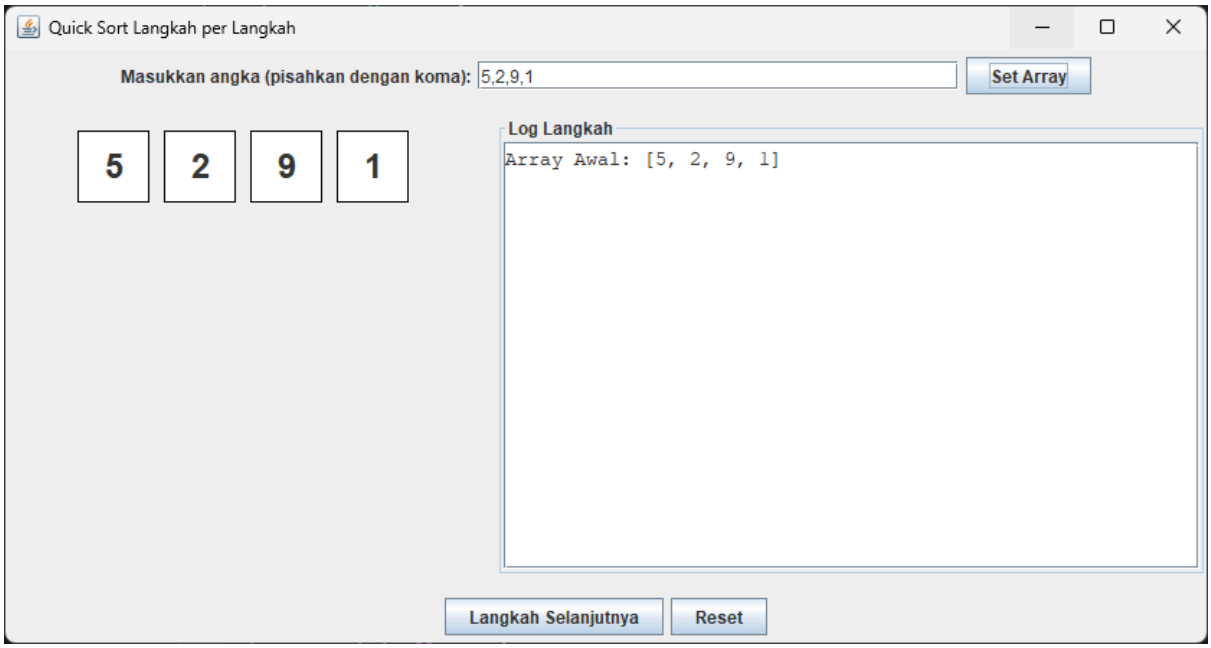
- reset():

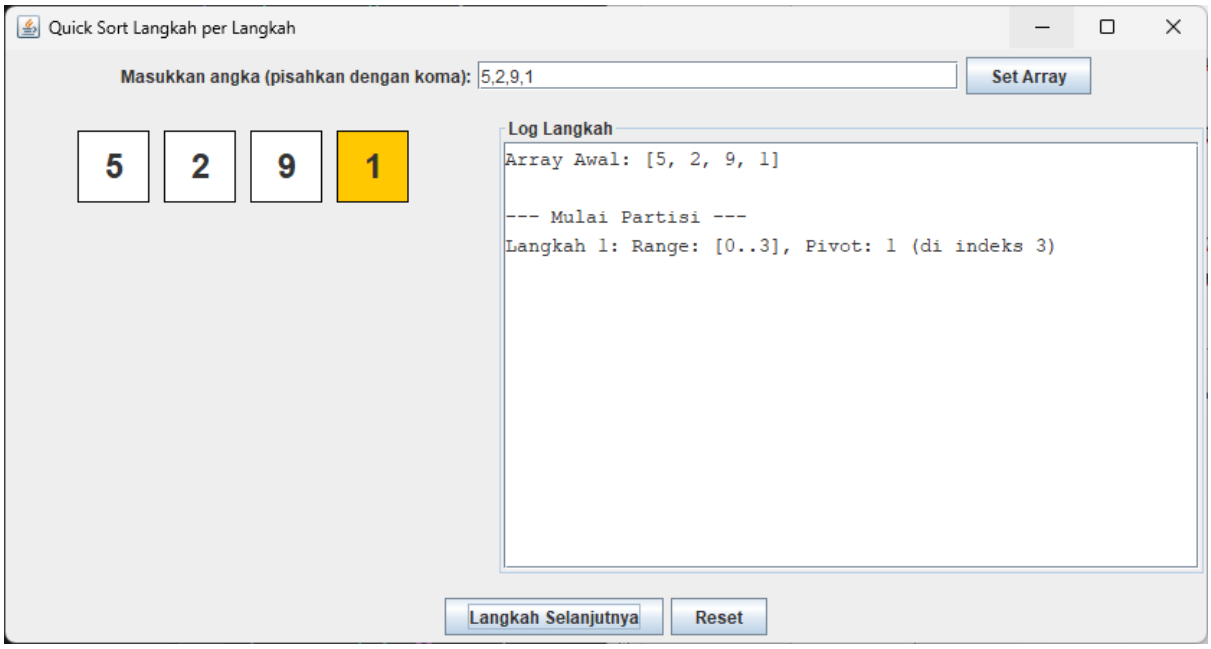
  Mengatur ulang seluruh tampilan GUI dan log.
- logStep(String message):

  Menambahkan log langkah ke stepArea.

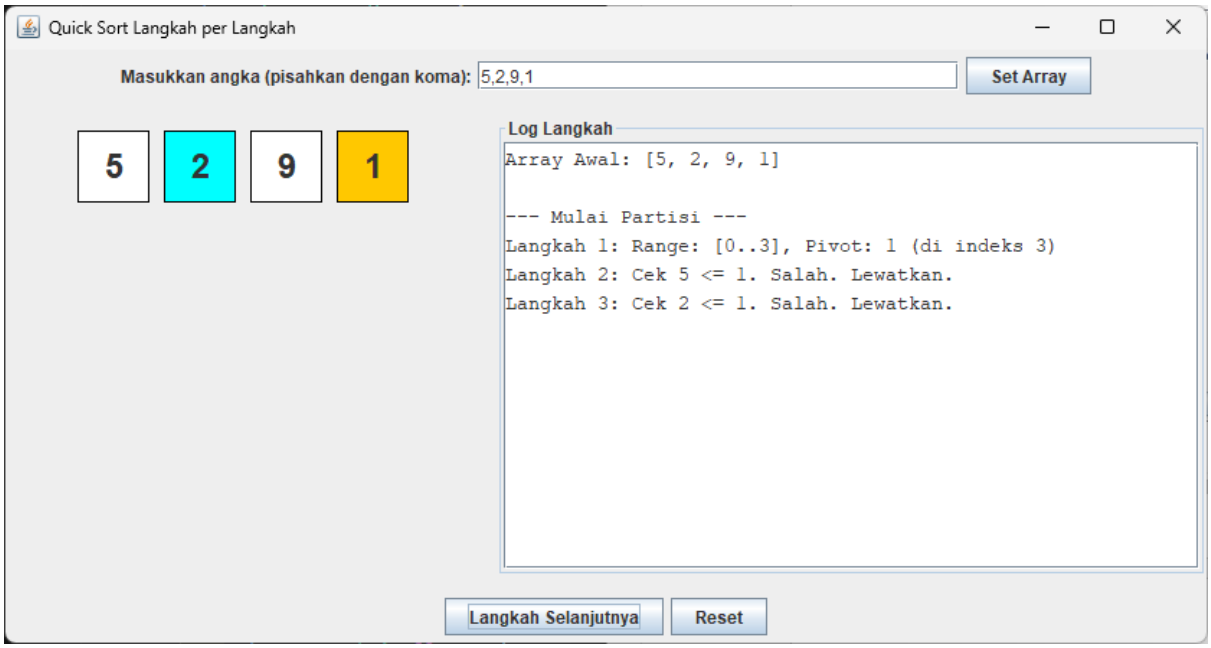
c) Fitur Utama Program:

- Pengguna dapat memasukkan angka seperti 5,2,9,1.
- Dapat menjalankan langkah sorting satu demi satu (step-by-step).
- Menampilkan log proses sorting di sisi kanan aplikasi secara real time.

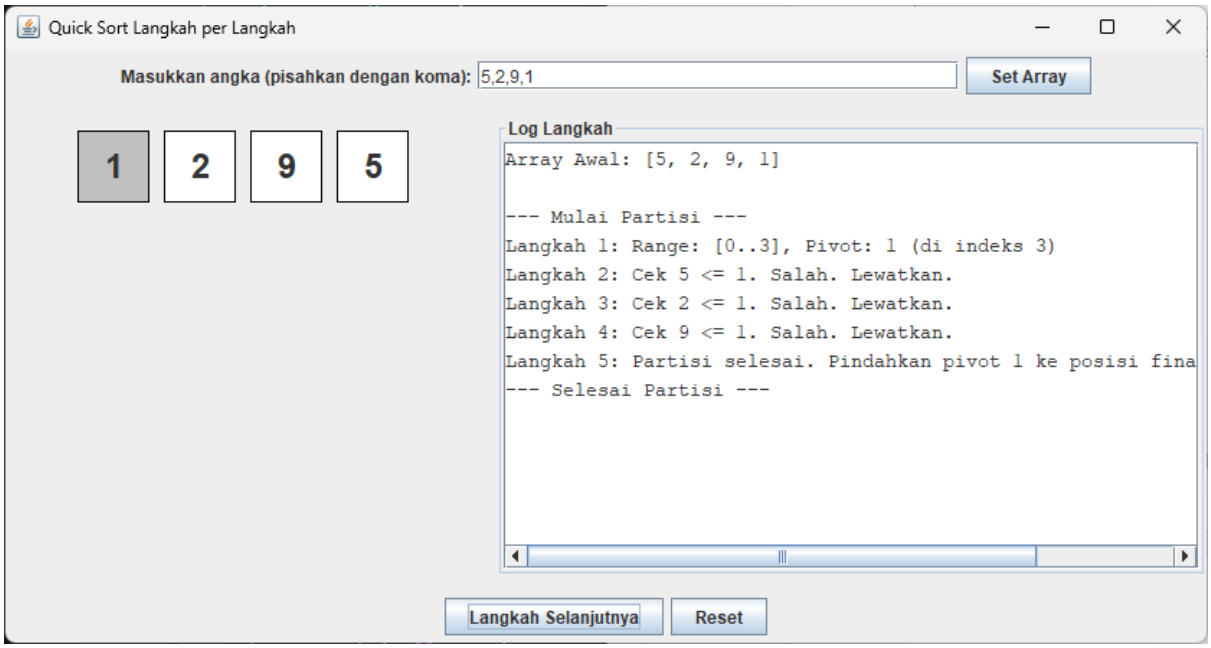


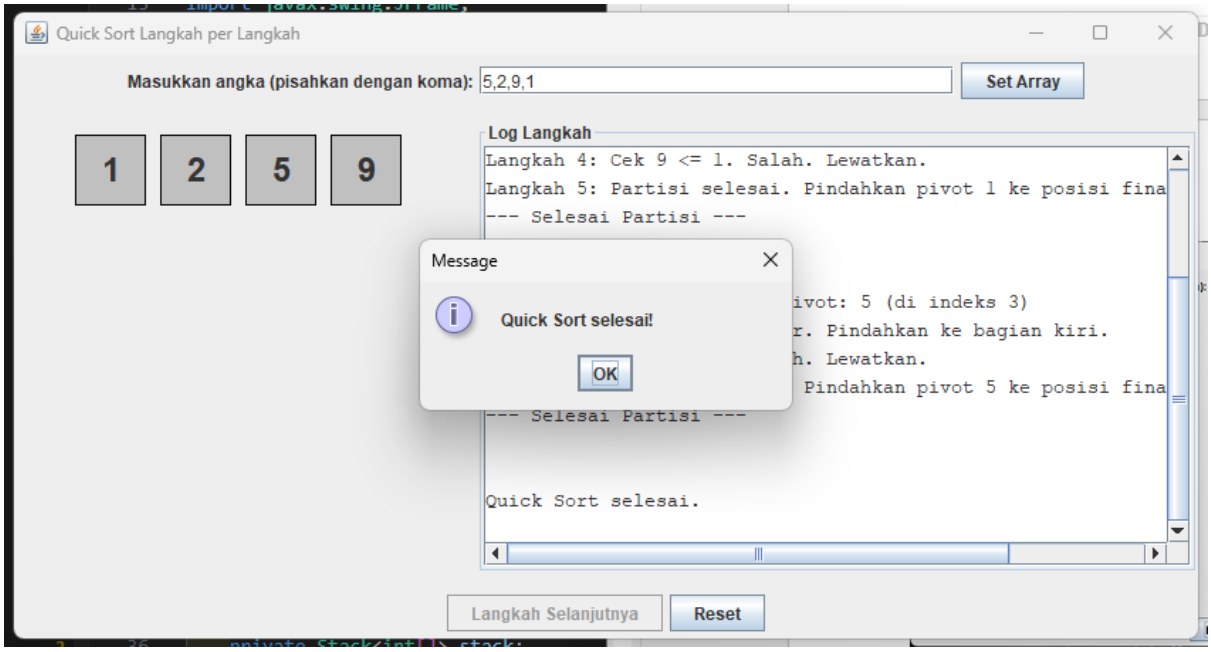



Quick Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
--- Mulai Partisi ---
Langkah 1: Range: [0..3], Pivot: 1 (di indeks 3)
Langkah 2: Cek 5 <= 1. Salah. Lewatkan.
Langkah Selanjutnya
Reset


Quick Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
--- Mulai Partisi ---
Langkah 1: Range: [0..3], Pivot: 1 (di indeks 3)
Langkah 2: Cek 5 <= 1. Salah. Lewatkan.
Langkah 3: Cek 2 <= 1. Salah. Lewatkan.
Langkah Selanjutnya
Reset


Quick Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
--- Mulai Partisi ---
Langkah 1: Range: [0..3], Pivot: 1 (di indeks 3)
Langkah 2: Cek 5 <= 1. Salah. Lewatkan.
Langkah 3: Cek 2 <= 1. Salah. Lewatkan.
Langkah 4: Cek 9 <= 1. Salah. Lewatkan.
Langkah Selanjutnya
Reset


Quick Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
1
2
9
5
Log Langkah
Array Awal: [5, 2, 9, 1]
--- Mulai Partisi ---
Langkah 1: Range: [0..3], Pivot: 1 (di indeks 3)
Langkah 2: Cek 5 <= 1. Salah. Lewatkan.
Langkah 3: Cek 2 <= 1. Salah. Lewatkan.
Langkah 4: Cek 9 <= 1. Salah. Lewatkan.
Langkah 5: Partisi selesai. Pindahkan pivot 1 ke posisi fina
--- Selesai Partisi ---
Langkah Selanjutnya
Reset




```
package pekan8;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.FlowLayout;
import java.awt.Font;
import java.util.Stack;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class QuickSortGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;
```

```
    private int i = 0, j = 0;
    private boolean sorting = false;
    private int stepCount = 1;

    private Stack<int[]> stack;
    private int low, high, pivot;
    private boolean partitioning = false;

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    QuickSortGUI frame = new QuickSortGUI();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public QuickSortGUI() {
        setTitle("Quick Sort Langkah per Langkah");
        setSize(850, 450);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));

        stack = new Stack<>();

        JPanel topPanel = new JPanel(new BorderLayout());
        JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        inputField = new JTextField(30);
        setButton = new JButton("Set Array");
        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan
koma):"));
        inputPanel.add(inputField);
        inputPanel.add(setButton);
        topPanel.add(inputPanel, BorderLayout.CENTER);

        panelArray = new JPanel();
        panelArray.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JPanel controlPanel = new JPanel();
        stepButton = new JButton("Langkah Selanjutnya");
        resetButton = new JButton("Reset");
        stepButton.setEnabled(false);
```

```
        controlPanel.add(stepButton);
        controlPanel.add(resetButton);

        JPanel logPanel = new JPanel(new BorderLayout());
        logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));

        stepArea = new JTextArea(8, 60);
        stepArea.setEditable(false);
        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(stepArea);
        logPanel.add(scrollPane, BorderLayout.CENTER);


        add(topPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(logPanel, BorderLayout.EAST);

        setButton.addActionListener(e -> setArrayFromInput());
        stepButton.addActionListener(e -> performStep());
        resetButton.addActionListener(e -> reset());
    }

    private void performStep() {

        if (stack.isEmpty() && !partitioning) {
            if(sorting){
                sorting = false;
                stepButton.setEnabled(false);
                resetHighlights(true); // Warnai semua jadi abu-abu
                stepArea.append("\nQuick Sort selesai.\n");
                JOptionPane.showMessageDialog(this, "Quick Sort selesai!");
            }
            return;
        }

        resetHighlights(false);

        if (!partitioning) {
            int[] range = stack.pop();
            low = range[0];
            high = range[1];

            if(low >= high) {
                performStep();
                return;
            }

            pivot = array[high];
```

```
            i = low - 1;
            j = low;
            partitioning = true;
            stepArea.append("--- Mulai Partisi ---\n");
            logStep("Range: [" + low + ".." + high + "], Pivot: " + pivot +
" (di indeks " + high +")");
            highlightPivot(high);
            return;
        }

        if (j < high) {
            highlightCompare(j, high);
            if (array[j] <= pivot) {
                i++;
                logStep("Cek " + array[j] + " <= " + pivot + ". Benar.
Pindahkan ke bagian kiri.");
                if (i != j) {
                    logStep(" -> Tukar arr[" + i + "]=" + array[i] + "
dengan arr[" + j + "]=" + array[j]);
                    swap(i, j);
                    updateLabels();
                }
            } else {
                logStep("Cek " + array[j] + " <= " + pivot + ". Salah.
Lewatkan.");
            }
            j++;
            return;
        }

        int pivotFinalIndex = i + 1;
        logStep("Partisi selesai. Pindahkan pivot " + pivot + " ke posisi
finalnya di indeks " + pivotFinalIndex);
        swap(pivotFinalIndex, high);
        updateLabels();

        partitioning = false;

        labelArray[pivotFinalIndex].setBackground(Color.LIGHT_GRAY);

        stepArea.append("--- Selesai Partisi ---\n\n");

        stack.push(new int[] {low, pivotFinalIndex - 1});
        stack.push(new int[] {pivotFinalIndex + 1, high});
    }

    private void highlightPivot(int index) {
        labelArray[index].setBackground(Color.ORANGE);
```

```
    }

    private void highlightCompare(int jIndex, int pivotIndex) {
        labelArray[jIndex].setBackground(Color.CYAN);
        if(labelArray[pivotIndex].getBackground() != Color.LIGHT_GRAY) {
            labelArray[pivotIndex].setBackground(Color.ORANGE);
        }
    }

    private void setArrayFromInput() {
        String text = inputField.getText().trim();
        if (text.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String[] parts = text.split(",");
        if (parts.length < 2) {
            JOptionPane.showMessageDialog(this, "Masukkan setidaknya dua
angka!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        array = new int[parts.length];

        try {
            for (int k = 0; k < parts.length; k++) {
                array[k] = Integer.parseInt(parts[k].trim());
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang
dipisahkan dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        labelArray = new JLabel[array.length];
        panelArray.removeAll();

        for (int k = 0; k < array.length; k++) {
            labelArray[k] = new JLabel(String.valueOf(array[k]));
            labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLA
CK));
            labelArray[k].setPreferredSize(new Dimension(50, 50));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
            labelArray[k].setOpaque(true);
            labelArray[k].setBackground(Color.WHITE);
            panelArray.add(labelArray[k]);
        }
```

```
        stack.clear();
        stack.push(new int[] {0, array.length - 1});
        sorting = true;
        partitioning = false;
        stepCount = 1;
        stepArea.setText("Array Awal: " + java.util.Arrays.toString(array) +
"\n\n");
        stepButton.setEnabled(true);

        panelArray.revalidate();
        panelArray.repaint();
    }

    private void updateLabels() {
        for (int k = 0; k < array.length; k++) {
            labelArray[k].setText(String.valueOf(array[k]));
        }
    }

    private void resetHighlights(boolean markAsSorted) {
        if (labelArray == null) return;
        for (JLabel label : labelArray) {
            if (markAsSorted || label.getBackground() == Color.LIGHT_GRAY) {
                label.setBackground(Color.LIGHT_GRAY);
            } else {
                label.setBackground(Color.WHITE);
            }
        }
    }

    private void swap(int a, int b) {
        int temp = array[a];
        array[a] = array[b];
        array[b] = temp;
    }

    private void reset() {
        inputField.setText("");
        panelArray.removeAll();
        panelArray.revalidate();
        panelArray.repaint();
        stepArea.setText("");
        stepButton.setEnabled(false);
        if(stack != null) stack.clear();
        sorting = false;
        partitioning = false;
        stepCount = 1;
        array = null;
```

```
        labelArray = null;
    }

    private void logStep(String message) {
        stepArea.append("Langkah " + stepCount + ": " + message + "\n");
        stepArea.setCaretPosition(stepArea.getDocument().getLength());
        stepCount++;
    }
}
```

D. Class : ShellSortGUI.java

a) Tujuan Program:

Aplikasi GUI Java untuk melakukan algoritma **Shell Sort** secara langkah per langkah.

b) Struktur dan Penjelasan Komponen Program:

1. package pekan8;

   Menandakan bahwa file ini berada dalam package (folder) bernama pekan8.

2. import java.awt.*, javax.swing.*, java.util.Arrays
   - Mengimpor pustaka-pustaka Java yang dibutuhkan:
   - AWT & Swing untuk pembuatan GUI.
   - Arrays untuk mencetak isi array ke dalam log.

3. public class ShellSortGUI extends JFrame

   Mendefinisikan kelas utama ShellSortGUI yang merupakan turunan dari JFrame, komponen utama GUI.

4. Deklarasi Variabel:
   - int[] array: Menyimpan data integer yang akan diurutkan menggunakan Shell Sort.
   - JLabel[] labelArray: Visualisasi elemen array dalam bentuk label di GUI.
   - JButton stepButton, resetButton, setButton:
     - stepButton: Menjalankan satu langkah Shell Sort.
     - resetButton: Reset tampilan GUI.
     - setButton: Membaca input dari inputField dan memulai proses sorting.
   - JTextField inputField: Kolom input angka yang dimasukkan oleh pengguna.
   - JPanel panelArray: Panel untuk menampilkan array dalam bentuk label visual.
   - JTextArea stepArea: Area teks untuk menampilkan log langkah demi langkah sorting.

5. Variabel Pendukung Sorting:
    - int gap: Jarak antar elemen dalam setiap pass Shell Sort.
    - int i, j: Indeks iterasi; i untuk elemen saat ini, j untuk proses geser.
    - int temp: Nilai elemen yang akan disisipkan.
    - boolean sorting: Menandakan proses sorting sedang berlangsung.
    - boolean isSwapping: Menunjukkan apakah dalam proses menggeser elemen untuk penyisipan.
    - int stepCount: Mencatat jumlah langkah sorting untuk penomoran log.

6. Method main()

Memulai GUI dengan menjalankan ShellSortGUI pada thread yang aman untuk GUI (Event Dispatch Thread) menggunakan EventQueue.invokeLater().

7. Constructor ShellSortGUI()

- Mengatur tata letak GUI:
    - Top panel: Input array + tombol "Set Array".
    - Center panel: Label array yang divisualisasikan.
    - Control panel (bawah): Tombol "Langkah Selanjutnya" dan "Reset".
    - Log panel (kanan): JTextArea untuk log langkah.
- Menambahkan event listener untuk setiap tombol.

8. Fungsi-fungsi Utama:

- performStep():

  Menjalankan satu langkah sorting berdasarkan logika Shell Sort:
    - Fase penggeseran (if isSwapping true): geser elemen jika perlu.
    - Fase awal elemen baru atau perubahan gap.
    - Menentukan posisi penyisipan dan memberi warna visualisasi (kuning, cyan, merah).
    - Mengurangi gap hingga selesai.
- setArrayFromInput():

  Membaca input dari user, menginisialisasi array, menghitung gap, dan menyiapkan GUI untuk mulai sorting.
- updateLabels():

  Memperbarui isi label sesuai perubahan nilai array.
- resetHighlights():

  Menghapus warna pada label, mengembalikan ke warna default (putih).

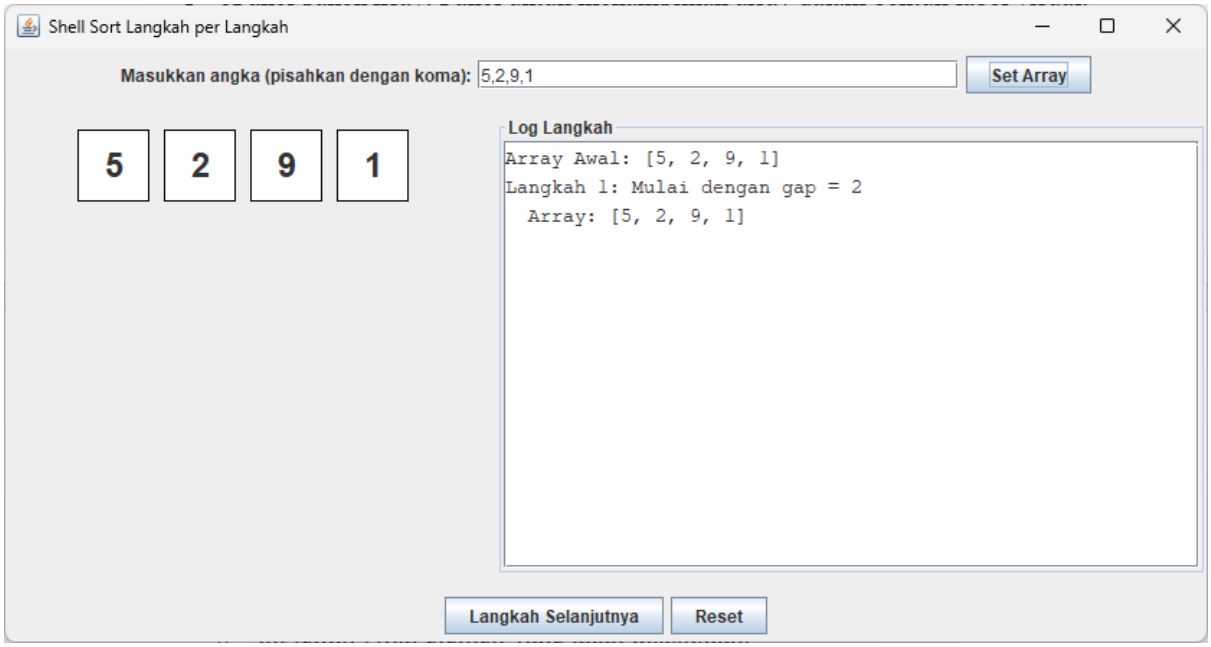
- logStep(String message):

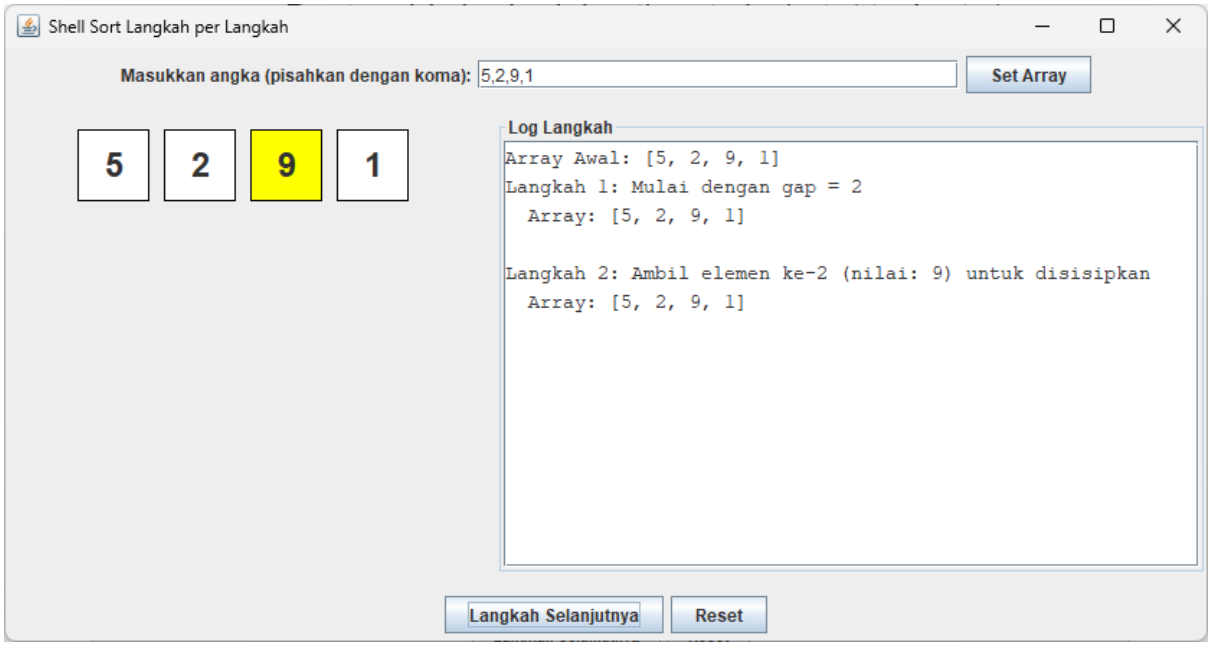
  Menambahkan langkah log ke stepArea dan mencetak isi array setiap langkah.
- reset():

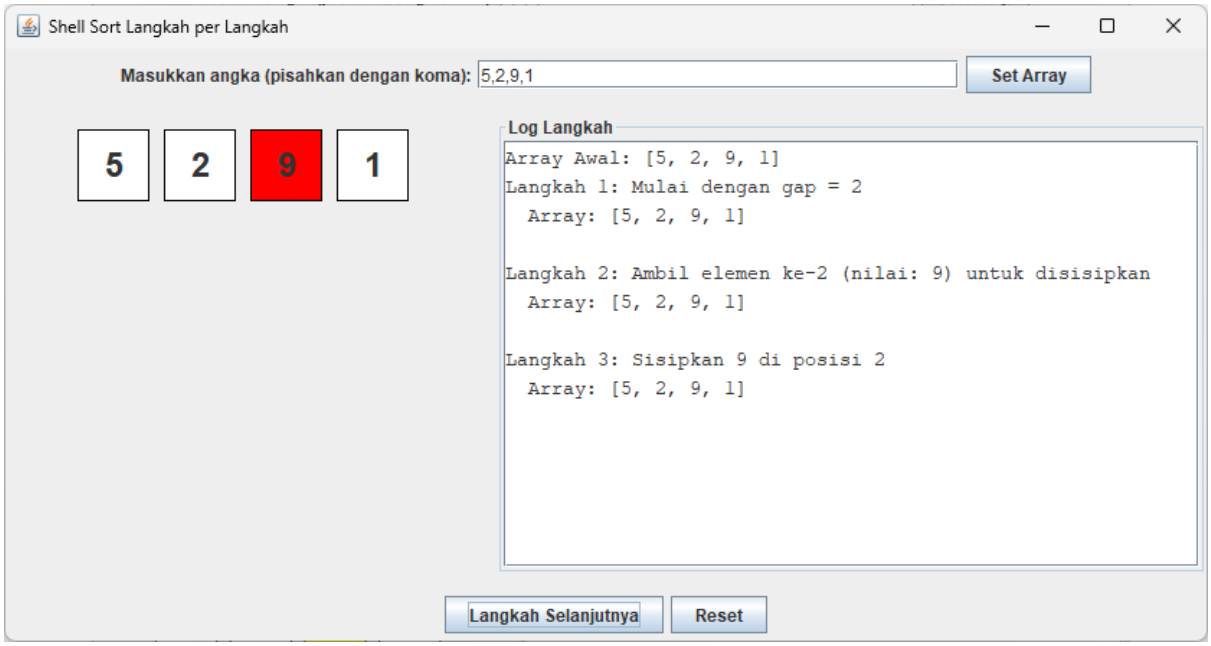
  Menghapus semua data dan mengembalikan GUI ke kondisi awal.

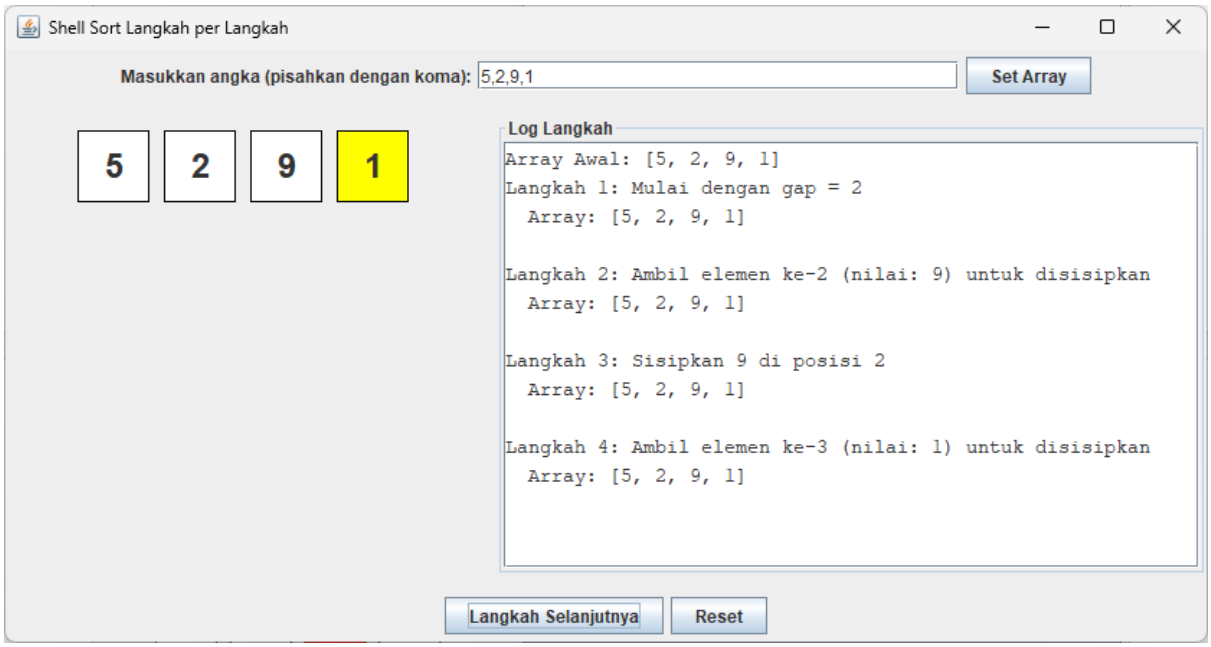
c) Fitur Utama Program:

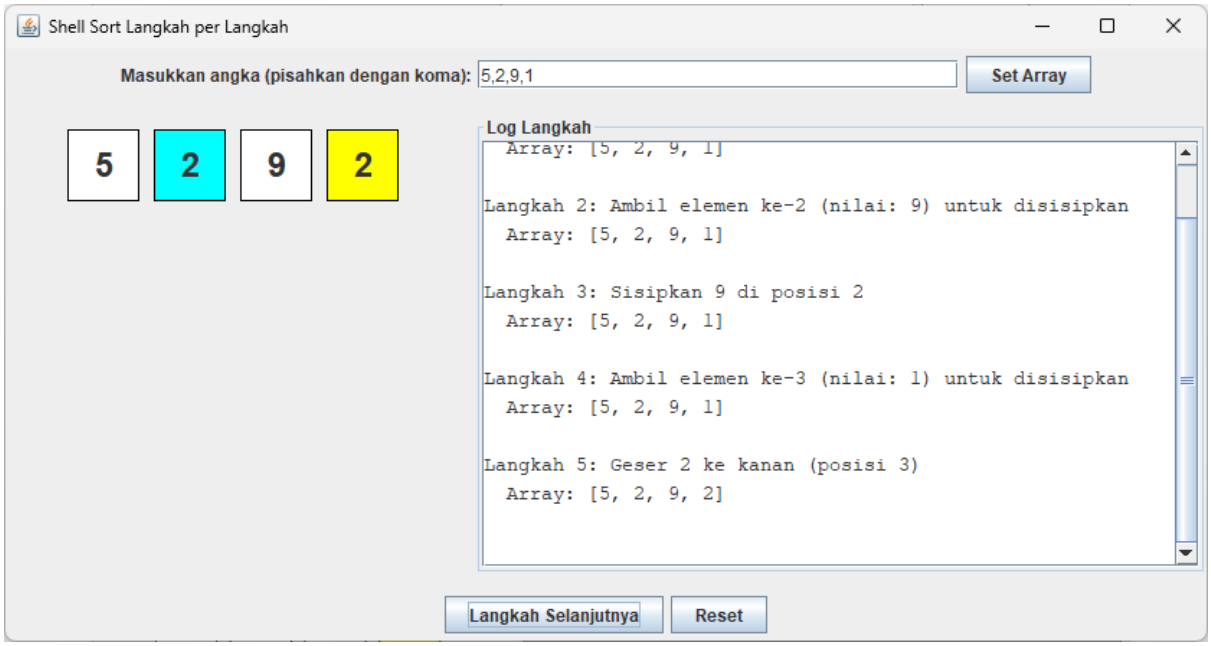
- Pengguna dapat memasukkan angka seperti 5,2,9,1.
- Dapat menjalankan langkah sorting satu demi satu (step-by-step).
- Menampilkan log proses sorting di sisi kanan aplikasi secara real time.

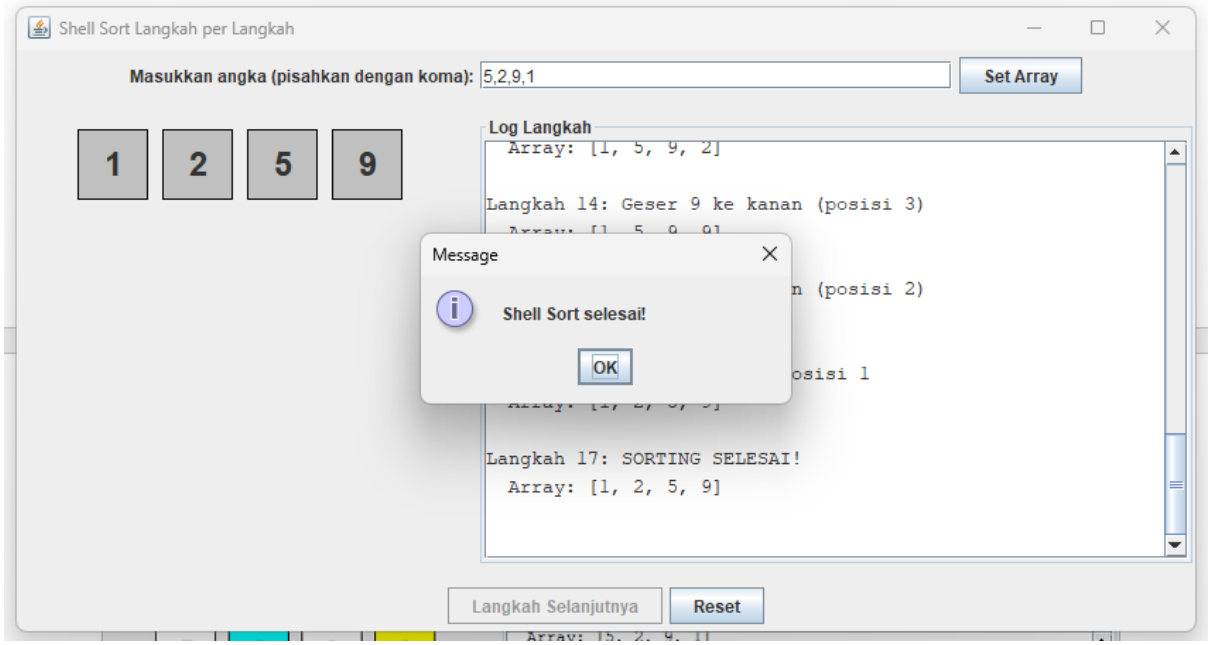





Shell Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
Langkah 1: Mulai dengan gap = 2
Array: [5, 2, 9, 1]
Langkah 2: Ambil elemen ke-2 (nilai: 9) untuk disisipkan
Array: [5, 2, 9, 1]
Langkah 3: Sisipkan 9 di posisi 2
Array: [5, 2, 9, 1]
Langkah Selanjutnya
Reset


Shell Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
5,2,9,1
Set Array
5
2
9
1
Log Langkah
Array Awal: [5, 2, 9, 1]
Langkah 1: Mulai dengan gap = 2
Array: [5, 2, 9, 1]
Langkah 2: Ambil elemen ke-2 (nilai: 9) untuk disisipkan
Array: [5, 2, 9, 1]
Langkah 3: Sisipkan 9 di posisi 2
Array: [5, 2, 9, 1]
Langkah 4: Ambil elemen ke-3 (nilai: 1) untuk disisipkan
Array: [5, 2, 9, 1]
Langkah Selanjutnya
Reset






```
package pekan8;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.FlowLayout;
import java.awt.Font;
import java.util.Arrays;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
```

```
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

public class ShellSortGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;

    // Variabel state untuk visualisasi langkah per langkah
    private int gap;
    private int i; // Indeks luar (elemen yang akan disisipkan)
    private int j; // Indeks dalam (posisi untuk penyisipan)
    private int temp; // Nilai elemen yang sedang diproses
    private boolean sorting = false;
    private boolean isSwapping = false; // Flag penanda sedang dalam fase
geser/sisip
    private int stepCount = 1;

    public ShellSortGUI() {
        setTitle("Shell Sort Langkah per Langkah");
        setSize(850, 450);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // FIX: Menghapus karakter spasi aneh
        setLayout(new BorderLayout(10, 10));

        JPanel topPanel = new JPanel(new BorderLayout());
        JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        inputField = new JTextField(30);
        setButton = new JButton("Set Array");
        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan
koma):"));
        inputPanel.add(inputField);
        inputPanel.add(setButton);
        topPanel.add(inputPanel, BorderLayout.CENTER);

        panelArray = new JPanel();
        panelArray.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
```

```
        JPanel controlPanel = new JPanel();
        stepButton = new JButton("Langkah Selanjutnya");
        resetButton = new JButton("Reset");
        stepButton.setEnabled(false);
        controlPanel.add(stepButton);
        controlPanel.add(resetButton);

        JPanel logPanel = new JPanel(new BorderLayout());
        logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));
        // FIX: Menghapus karakter spasi aneh
        stepArea = new JTextArea(8, 60);
        stepArea.setEditable(false);
        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
        JScrollPane scrollPane = new JScrollPane(stepArea);
        logPanel.add(scrollPane, BorderLayout.CENTER);

        add(topPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(logPanel, BorderLayout.EAST);

        setButton.addActionListener(e -> setArrayFromInput());
        stepButton.addActionListener(e -> performStep());
        resetButton.addActionListener(e -> reset());
    }

    private void performStep() {
        if (!sorting) return;

        // FIX: Logika reset highlight dipindahkan agar tidak menghapus
highlight langkah sebelumnya terlalu cepat
        resetHighlights();

        // FASE 1: Sedang dalam proses menggeser dan mencari posisi sisip
        if (isSwapping) {
            // Cek apakah masih bisa geser ke kiri
            if (j >= gap && array[j - gap] > temp) {
                // Visualisasi perbandingan
                labelArray[j - gap].setBackground(Color.CYAN); // Elemen
pembanding
                labelArray[j].setBackground(Color.YELLOW);     // Posisi
saat ini

                // Lakukan pergeseran
                array[j] = array[j - gap];
                logStep("Geser " + array[j - gap] + " ke kanan (posisi " + j
+ ")");
```

```
                updateLabels();

                j -= gap; // Mundur sejauh gap untuk perbandingan
selanjutnya
            } else {
                // Posisi sudah ditemukan, sisipkan elemen
                array[j] = temp;
                logStep("Sisipkan " + temp + " di posisi " + j);
                labelArray[j].setBackground(Color.RED); // Tandai posisi
penyisipan
                updateLabels();

                // Siap untuk elemen berikutnya di iterasi i
                i++;
                isSwapping = false; // Keluar dari fase penyisipan
            }
        }
        // FASE 2: Memulai proses untuk elemen baru (i) atau mengganti gap
        else {
            // Jika iterasi i untuk gap saat ini sudah selesai
            if (i >= array.length) {
                gap /= 2; // Perkecil gap
                if (gap > 0) {
                    logStep("PASS SELESAI. Ganti gap menjadi " + gap);
                    i = gap; // Mulai iterasi i dari gap yang baru
                } else {
                    // Sorting selesai total
                    logStep("SORTING SELESAI!");
                    sorting = false;
                    stepButton.setEnabled(false);
                    for (JLabel label : labelArray) {
                        label.setBackground(Color.LIGHT_GRAY);
                    }
                    JOptionPane.showMessageDialog(this, "Shell Sort
selesai!");
                    return;
                }
            }

            // Memulai fase penyisipan untuk elemen di posisi i
            temp = array[i];
            j = i;
            logStep("Ambil elemen ke-" + i + " (nilai: " + temp + ") untuk
disisipkan");
            labelArray[i].setBackground(Color.YELLOW); // Tandai elemen yang
akan disisipkan
            isSwapping = true;
        }
```

```
    }

    private void logStep(String message) {
        stepArea.append("Langkah " + stepCount + ": " + message + "\n");
        stepArea.append("  Array: " + Arrays.toString(array) + "\n\n");
        stepArea.setCaretPosition(stepArea.getDocument().getLength());
        stepCount++;
    }

    private void setArrayFromInput() {
        String text = inputField.getText().trim();
        if (text.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!",
"Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String[] parts = text.split(",");
        array = new int[parts.length];

        try {
            for (int k = 0; k < parts.length; k++) {
                array[k] = Integer.parseInt(parts[k].trim());
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang
dipisahkan dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        // Reset state
        gap = array.length / 2;
        i = gap;
        j = i;
        stepCount = 1;
        isSwapping = false;

        stepArea.setText("Array Awal: " + Arrays.toString(array) + "\n");
        logStep("Mulai dengan gap = " + gap);

        sorting = true;
        stepButton.setEnabled(true);
        panelArray.removeAll();
        labelArray = new JLabel[array.length];

        for (int k = 0; k < array.length; k++) {
            labelArray[k] = new JLabel(String.valueOf(array[k]));
            labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
```

```
            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLA
CK));
            labelArray[k].setPreferredSize(new Dimension(50, 50));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
            // FIX: Menghapus karakter spasi aneh
            labelArray[k].setOpaque(true);
            labelArray[k].setBackground(Color.WHITE);
            panelArray.add(labelArray[k]);
        }
        panelArray.revalidate();
        panelArray.repaint();
    }

    private void updateLabels() {
        for (int k = 0; k < array.length; k++) {
            labelArray[k].setText(String.valueOf(array[k]));
        }
    }

    private void resetHighlights() {
        if (labelArray == null) return;
        for (JLabel label : labelArray) {
            label.setBackground(Color.WHITE);
        }
    }

    private void reset() {
        inputField.setText("");
        panelArray.removeAll();
        panelArray.revalidate();
        panelArray.repaint();
        stepArea.setText("");
        stepButton.setEnabled(false);
        sorting = false;
        array = null;
        labelArray = null;
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            ShellSortGUI gui = new ShellSortGUI();
            gui.setVisible(true);
        });
    }
}
```

BAB III
PENUTUP

3.1 Kesimpulan

Dari praktikum pekan 7 ini, dapat disimpulkan bahwa:

- Algoritma *Insertion Sort* dan *Selection Sort* dapat dipahami dengan lebih baik melalui visualisasi langkah demi langkah.
- Pemrograman GUI dengan Java Swing memungkinkan representasi visual yang interaktif dan edukatif.
- *Insertion Sort* lebih efisien untuk array kecil yang hampir terurut, sementara *Selection Sort* bekerja secara konsisten meskipun tidak seefisien untuk data besar.
- Pembuatan GUI seperti ini dapat memperdalam pemahaman terhadap konsep dasar algoritma dan struktur data.