# Laporan Praktikum

# Struktur Data


**Disusun Oleh :**

**SASKIA ALIFAH (2411531002)**

**Dosen Pengampu : Dr. Wahyudi, S.T, M.T**

**Departemen Informatika**

**Fakultas Teknologi Informasi**

**Universitas Andalas**

**Tahun 2025**

## A. TUJUAN PRAKTIKUM

1. **Memahami prinsip kerja dari berbagai algoritma pengurutan**: Merge Sort, Quick Sort, Bubble Sort, dan Shell Sort, serta membandingkan keefektifan dan efisiensinya.
2. **Mengimplementasikan masing-masing algoritma sorting ke dalam program Java berbasis GUI** (Graphical User Interface) menggunakan pustaka Swing untuk memvisualisasikan proses sorting secara langkah demi langkah.
3. **Meningkatkan pemahaman konsep algoritma sorting melalui pendekatan interaktif**, dengan menampilkan setiap langkah proses pengurutan secara visual agar lebih mudah dipahami.

## B. PENDAHULUAN

Sorting atau pengurutan data merupakan salah satu operasi dasar yang sangat penting dalam bidang struktur data dan algoritma. Proses sorting bertujuan untuk menyusun elemen-elemen dalam suatu struktur data, seperti array atau list, berdasarkan urutan tertentu—biasanya menaik (ascending) atau menurun (descending). Dengan data yang terurut, proses pencarian dan analisis data dapat dilakukan dengan lebih efisien dan optimal. Oleh karena itu, pemahaman tentang berbagai algoritma sorting menjadi krusial dalam pengembangan aplikasi dan sistem berbasis data.

Terdapat berbagai macam algoritma pengurutan yang dikembangkan, masing-masing dengan karakteristik, kompleksitas, dan strategi yang berbeda. Dalam praktikum ini, empat algoritma utama yang dipelajari adalah Bubble Sort, Shell Sort, Merge Sort, dan Quick Sort. Bubble Sort dikenal sebagai algoritma sederhana yang bekerja dengan cara menukar elemen berdekatan, sedangkan Shell Sort mengembangkan konsep dari Insertion Sort dengan penggunaan gap untuk meningkatkan efisiensi. Di sisi lain, Merge Sort dan Quick Sort termasuk ke dalam algoritma divide and conquer yang membagi data menjadi bagian-bagian kecil sebelum mengurutkannya secara rekursif.

Melalui praktikum ini, setiap algoritma tidak hanya diimplementasikan dalam kode, tetapi juga divisualisasikan melalui antarmuka grafis berbasis Java Swing. Pendekatan visual ini memudahkan mahasiswa untuk memahami proses pengurutan langkah demi langkah, sehingga meningkatkan pemahaman konseptual terhadap cara kerja masing-masing algoritma. Dengan demikian, praktikum ini diharapkan dapat memberikan pemahaman yang mendalam, baik secara logika pemrograman maupun struktur data, khususnya dalam hal efisiensi dan penerapan algoritma sorting.

## C. METODE PRAKTIKUM

### 1. Kelas BubbleSortGUI

#### a) Deskripsi Kelas

Kelas BubbleSortGUI adalah **program Java GUI** (Graphical User Interface) berbasis JFrame untuk **memvisualisasikan algoritma Bubble Sort secara langkah per langkah**. Program ini memungkinkan pengguna memasukkan array angka, lalu menjalankan proses sorting satu per satu langkah sambil melihat perubahan visual dan log di layar.

#### b) Penjelasan Fungsi dan Cara Kerja

Program ini adalah aplikasi Java berbasis GUI yang menampilkan proses *Bubble Sort* secara visual dan bertahap. Program ini dirancang agar pengguna dapat melihat setiap langkah pertukaran data dengan tampilan antarmuka yang intuitif. Dengan memanfaatkan komponen Swing, data akan divisualisasikan dalam bentuk label, dan setiap proses sorting dicatat dalam area teks. Berikut ini adalah penjelasan mendetail untuk setiap bagian utama dalam program:

i. **Method main()**

Method main() adalah titik awal eksekusi program. Di dalamnya, GUI dijalankan melalui EventQueue.invokeLater() agar berjalan di *event-dispatch thread*, yang merupakan praktik standar dalam pemrograman Swing. Ini bertujuan untuk memastikan tampilan GUI aman dan responsif. Objek dari kelas utama (BubbleSortGUI) dibuat dan ditampilkan melalui setVisible(true), seperti terlihat pada sintaks berikut:

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

ii. **Konstruktor BubbleSortGUI()**

Konstruktor ini bertanggung jawab untuk menyusun seluruh tampilan antarmuka pengguna. Pertama, judul jendela ditentukan dan layout utama diatur menggunakan BorderLayout. Kemudian, berbagai panel ditambahkan: inputPanel untuk menerima input angka melalui JTextField dan tombol Set Array, panelArray untuk menampilkan elemen-elemen array sebagai JLabel, serta controlPanel yang berisi tombol "Langkah Selanjutnya" dan "Reset". Selain itu, area teks (stepArea) ditambahkan di sisi kanan untuk mencatat log

langkah-langkah sorting. Beberapa ActionListener juga didefinisikan untuk masing-masing tombol, seperti terlihat di bagian akhir konstruktor:

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

iii. **Method setArrayFromInput()**

Method ini digunakan untuk mengambil input dari inputField, memisahkan string berdasarkan koma, dan mengonversinya menjadi array integer. Setelah input berhasil diproses, elemen array divisualisasikan sebagai label di dalam panelArray. Jika terjadi kesalahan input, seperti karakter non-angka, maka akan ditampilkan pesan kesalahan melalui JOptionPane. Berikut salah satu bagian penting dari method ini:

```
String[] parts = text.split(",");
array = new int[parts.length];
for (int k = 0; k < parts.length; k++) {
   array[k] = Integer.parseInt(parts[k].trim());
}
```

Setelah array terbentuk, tampilan diperbarui dan tombol “Langkah Selanjutnya” diaktifkan untuk memulai proses sorting.

iv. **Method performStep()**

Method ini menjalankan satu langkah dari algoritma Bubble Sort. Saat tombol “Langkah Selanjutnya” ditekan, dua elemen array dibandingkan. Jika perlu ditukar, maka dilakukan pertukaran dan warna label diubah untuk menunjukkan proses tersebut. Proses ini juga dicatat dalam stepArea. Setelah menyelesaikan satu iterasi dalam array, indeks dinaikkan untuk melanjutkan ke langkah berikutnya. Jika sorting telah selesai, tombol dimatikan dan ditampilkan pesan:

```
if (i == array.length - 1) {
   sorting = false;
   stepButton.setEnabled(false);
   JOptionPane.showMessageDialog(this, "Sorting selesai!");
}
```

v. **Method updateLabels()**

Method ini bertugas memperbarui isi dari label-label yang menampilkan array, sehingga visualisasi selalu sinkron dengan data yang ada. Setiap label akan menampilkan angka terbaru sesuai isi array:

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

vi. **Method resetHighlights()**

Method ini mengembalikan warna latar dari setiap JLabel menjadi putih. Ini diperlukan agar proses visualisasi tetap jelas setiap langkahnya, tanpa campuran warna dari langkah sebelumnya:

```
labelArray[k].setOpaque(true);
labelArray[k].setBackground(Color.WHITE);
```

vii. **Method reset()**

Method ini digunakan untuk mengembalikan aplikasi ke kondisi awal. Input dihapus, tampilan array direset, tombol "Langkah Selanjutnya" dimatikan, dan variabel kontrol seperti i, j, dan status sorting diatur ulang. Hal ini memungkinkan pengguna untuk memulai ulang proses sorting dengan data baru.

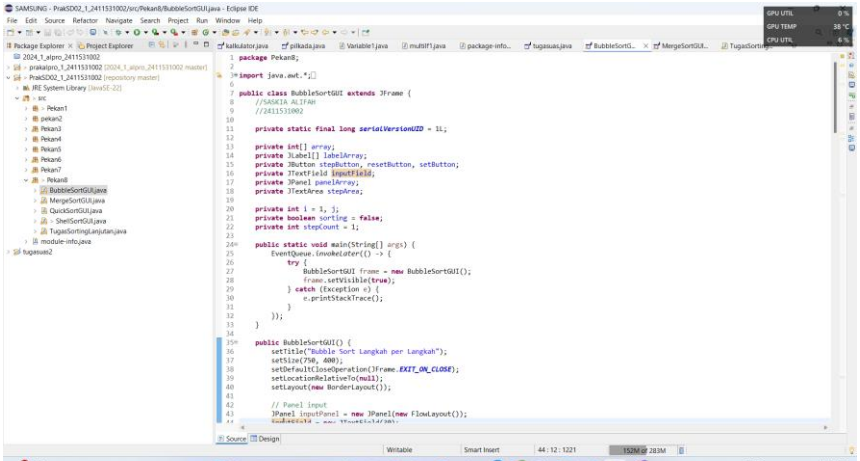
```
inputField.setText("");
panelArray.removeAll();
stepArea.setText("");
stepButton.setEnabled(false);
sorting = false;
```

viii. **Method arrayToString(int[] arr)**

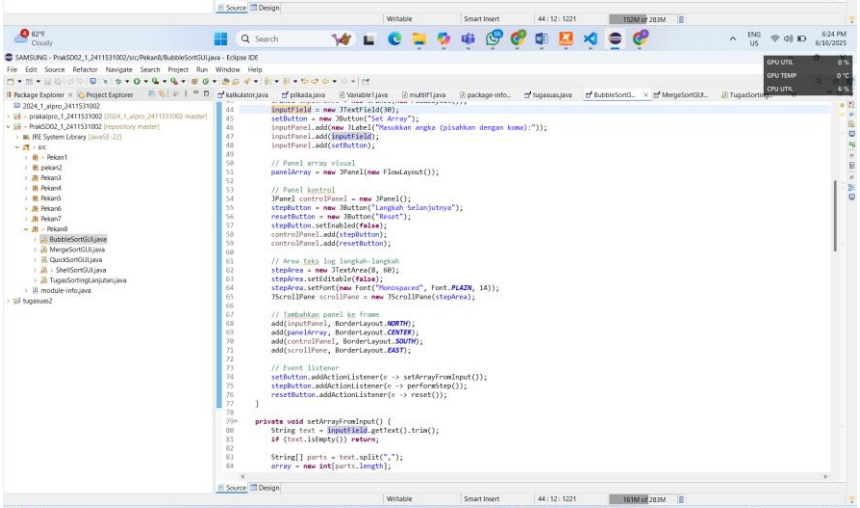
Method ini berfungsi untuk mengubah array integer menjadi representasi string yang mudah dibaca, dipisahkan dengan koma. Hasil dari method ini digunakan untuk menampilkan status array dalam stepArea setelah setiap langkah sorting:

```
for (int k = 0; k < arr.length; k++) {
   sb.append(arr[k]);
   if (k < arr.length - 1) sb.append(", ");
}
```

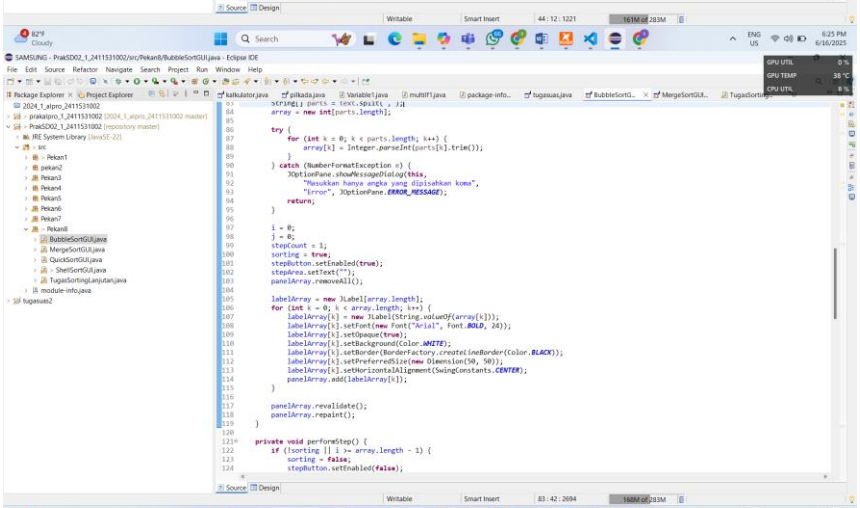
**c) Syntax**



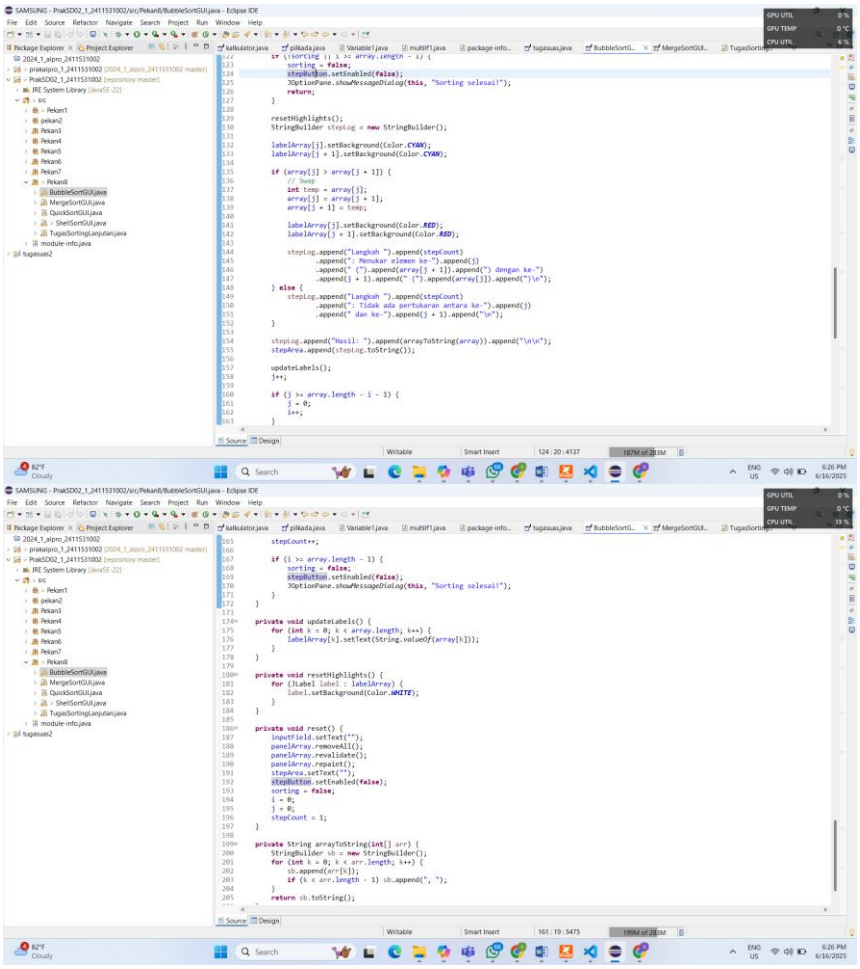
```java
package Pekan8;

import java.awt.*;

public class BubbleSortGUI extends JFrame {
    //SASEIA ALIFAH
    //2411531002

    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;

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

    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());
```



```java
        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(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 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 listener
        setButton.addActionListener(e -> setArrayFromInput());
        stepButton.addActionListener(e -> performStep());
        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];
```



```java
        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.BLACK));
            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);
```



## d) Diagram Alir

 **Mulai**

 **i = 1** (Inisialisasi indeks untuk elemen kedua dalam array)

 **Selama i < panjang array (n):**

a. Simpan elemen array pada posisi i ke dalam variabel key
b. Set j = i - 1
c. **Selama** j >= 0 **dan** array[j] > key:
- Geser array[j] ke array[j+1]
- Kurangi nilai j

d. Sisipkan key ke posisi j + 1
e. Tambahkan nilai i (lanjut ke elemen berikutnya)

 **Selesai** (jika seluruh elemen telah diproses)

## e) Error Handling

Program ini telah dirancang dengan beberapa penanganan kesalahan (*error handling*) untuk mencegah kegagalan atau bug saat dijalankan:

- **Input Tidak Valid**
  Jika pengguna memasukkan karakter non-numerik (misalnya huruf, simbol) atau format tidak sesuai (misalnya titik koma, spasi ganda), maka program akan memunculkan dialog error menggunakan JOptionPane.showMessageDialog(). Hal ini mencegah program melakukan parsing terhadap input yang salah.
- **Tombol Salah Urutan**
  Tombol "Langkah Selanjutnya" (stepButton) awalnya dinonaktifkan dan baru akan aktif setelah pengguna menekan "Set Array". Hal ini mencegah pengguna menjalankan proses sorting sebelum array tersedia.
- **ArrayIndexOutOfBoundsException**
  Potensi kesalahan karena indeks melampaui batas array dihindari dengan memastikan bahwa proses berhenti saat indeks i mencapai panjang array. Pada titik ini, tombol akan dinonaktifkan dan pesan bahwa proses telah selesai akan ditampilkan.

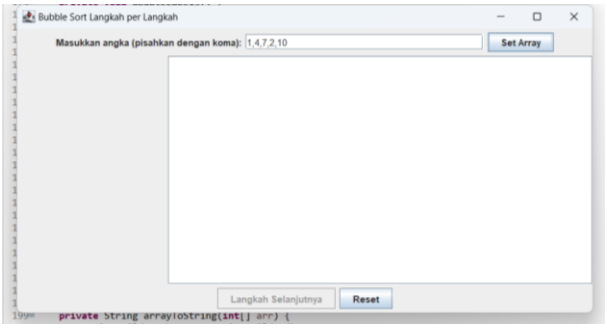
## f) Alur Sintaks Program

Berikut ini adalah urutan logika program dari awal dijalankan hingga proses selesai:

1. Program dijalankan, jendela GUI ditampilkan melalui konstruktor kelas utama (BubbleSortGUI atau InsertionSortGUI).
2. Pengguna memasukkan angka-angka ke dalam inputField, dipisahkan dengan koma.
3. Pengguna menekan tombol **"Set Array"**, yang akan mengubah input string menjadi array integer dan menampilkannya sebagai label di panel utama.
4. Pengguna menekan tombol **"Langkah Selanjutnya"** berulang kali. Setiap tekan akan menjalankan satu langkah dari algoritma Insertion Sort.
5. Proses penyisipan elemen terjadi satu per satu, dan setiap perubahan array dicatat dalam stepArea untuk keperluan visualisasi.
6. Setelah seluruh elemen selesai diproses, dialog pop-up ditampilkan untuk memberitahu bahwa sorting telah selesai.
7. Pengguna dapat menekan tombol **"Reset"** untuk menghapus input sebelumnya dan memulai kembali proses sorting dengan data baru.

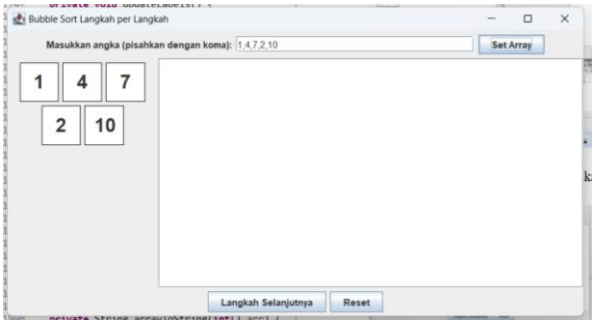
## g) Output

i. **Masukkan angka.**
masukkan angka yang ingin di urutkan 1,4,7,2,10.



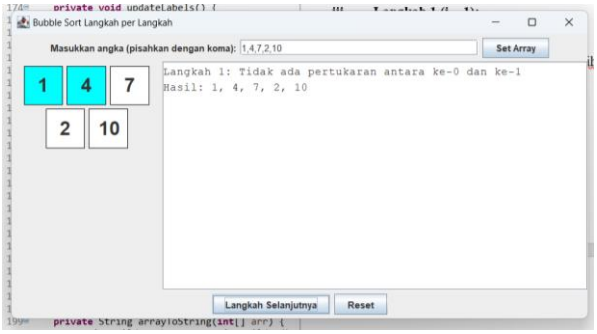

**ii. Tekan “Set Array”.**

Setelah ditekan set array, maka dikiri akan muncul kotak kotak angka seperti ini.

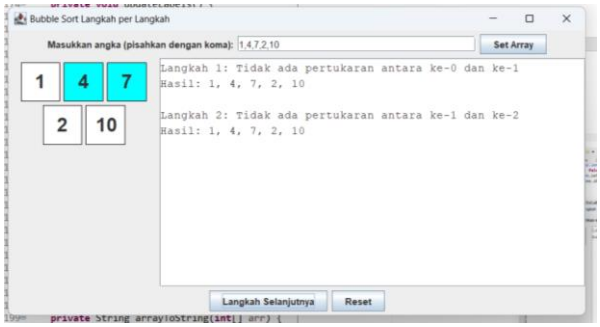



**iii. Pass 1**

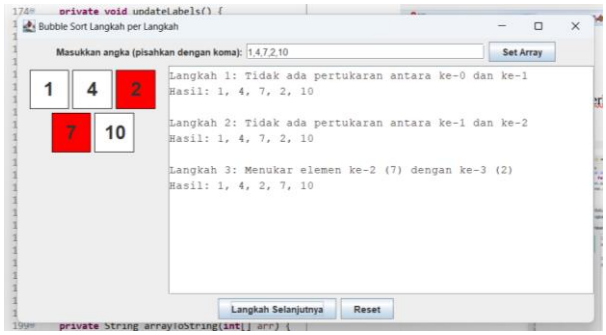
- **Langkah 1:** Bandingkan indeks 0 dan 1 → 1 vs 4 → tidak ditukar
  ➤ Array: [1, 4, 7, 2, 10]



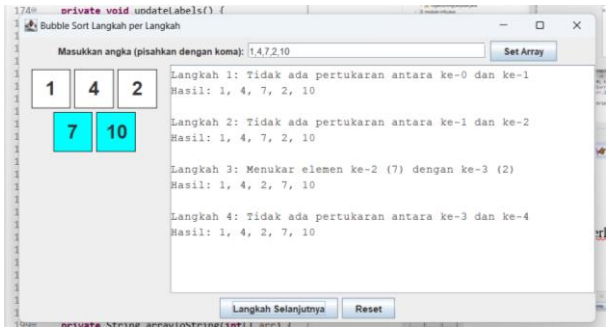

- **Langkah 2:** Bandingkan indeks 1 dan 2 → 4 vs 7 → tidak ditukar
  ➤ Array: [1, 4, 7, 2, 10]




- **Langkah 3:** Bandingkan indeks 2 dan 3 → 7 vs 2 → tukar
  ➤ Array: [1, 4, 2, 7, 10]



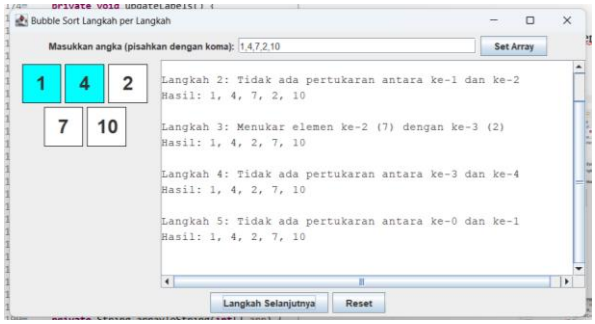

- **Langkah 4:** Bandingkan indeks 3 dan 4 → 7 vs 10 → tidak ditukar
  ➤ Array: [1, 4, 2, 7, 10]



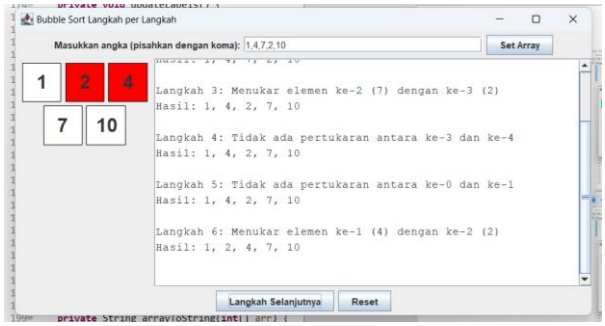

Elemen terbesar 10 sudah berada di posisi akhir.

iv. Pass 2

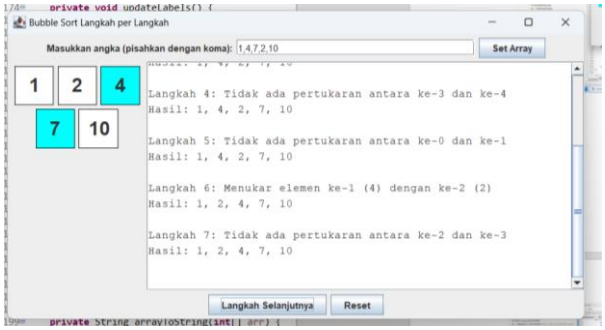
- **Langkah 5:** Bandingkan indeks 0 dan 1 → 1 vs 4 → tidak ditukar
  ➤ Array: [1, 4, 2, 7, 10]




- **Langkah 6:** Bandingkan indeks 1 dan 2 → 4 vs 2 → tukar
  ➤ Array: [1, 2, 4, 7, 10]

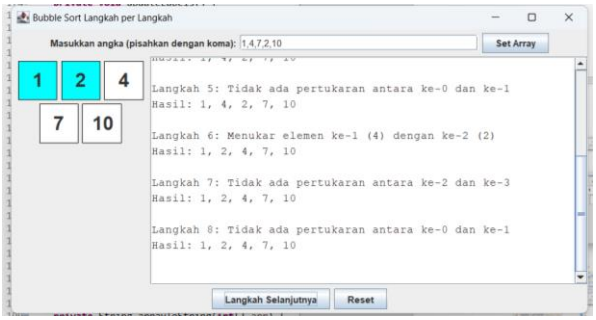



- **Langkah 7:** Bandingkan indeks 2 dan 3 → 4 vs 7 → tidak ditukar
  ➤ Array: [1, 2, 4, 7, 10]

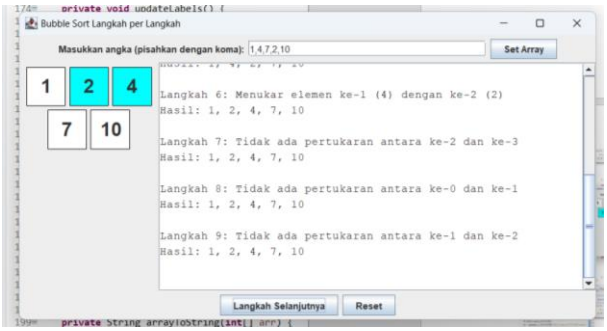



v. Pass 3

- **Langkah 8:** Bandingkan indeks 0 dan 1 → 1 vs 2 → tidak ditukar
  ➤ Array: [1, 2, 4, 7, 10]



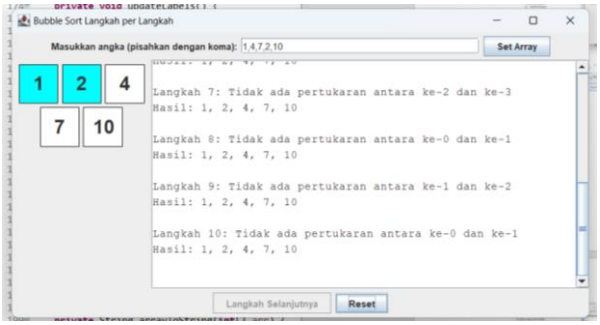

- **Langkah 9:** Bandingkan indeks 1 dan 2 → 2 vs 4 → tidak ditukar
  ➤ Array: [1, 2, 4, 7, 10]




Elemen terbesar ketiga 4 sudah benar.

vi. Pass 4

- **Langkah 10:** Bandingkan indeks 0 dan 1 → 1 vs 2 → tidak ditukar
  ➤ Array: [1, 2, 4, 7, 10]

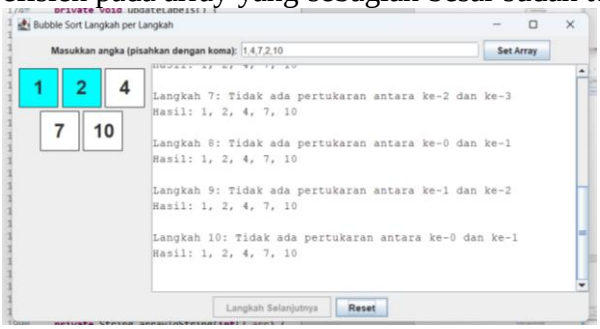



Semua elemen sudah terurut.

**vii. Kesimpulan output**

Setelah 4 langkah, array berhasil diurutkan secara ascending. Elemen yang benar-benar disisipkan ulang hanyalah angka 2, karena berada di posisi yang salah. Sisanya sudah dalam posisi tepat sejak awal, sehingga tidak banyak geseran yang terjadi. Inilah yang membuat Insertion Sort sangat

efisien pada array yang sebagian besar sudah terurut.




## 2. KelasMergeSortGUI

### a) Deskripsi Kelas

MergeSortGUI adalah kelas yang menampilkan visualisasi algoritma *Merge Sort* secara bertahap menggunakan GUI berbasis Java Swing. Kelas ini memungkinkan pengguna untuk:

- Memasukkan data array dalam format teks.
- Menyusun array secara bertahap menggunakan tombol “Langkah Selanjutnya”.
- Melihat setiap perubahan array dan proses *merge* melalui panel visual dan area log.

Proses sorting dilakukan dengan menyimpan langkah-langkah *merge* dalam antrian (queue) dan mengeksekusinya satu per satu saat tombol ditekan, sehingga memudahkan pemahaman proses rekursif merge sort secara interaktif.

### b) Penjelasan Fungsi dan Cara Kerja

*i. Package dan Import*

Program ini berada dalam package *Pekan8*, yang menunjukkan bahwa file ini merupakan bagian dari modul atau folder Pekan8. Package digunakan untuk mengelompokkan file Java agar lebih terstruktur.
Import javax.swing.* digunakan untuk membuat antarmuka grafis seperti JFrame, JLabel, JButton, JTextField, JTextArea, dan JScrollPane. Import java.awt.* digunakan untuk mengatur tata letak dan elemen GUI dasar.
Sedangkan java.util.LinkedList dan java.util.Queue digunakan untuk menyimpan dan menampilkan langkah-langkah proses Merge Sort secara bertahap menggunakan struktur antrian (queue).

```
package Pekan8;

import javax.swing.*;
import java.awt.*;
import java.util.LinkedList;
import java.util.Queue;
```

*ii. Deklarasi Kelas dan Variabel Atribut*

Kelas MergeSortGUI merupakan turunan dari JFrame, sehingga mewarisi sifat-sifat dari jendela aplikasi GUI Java Swing. Variabel array menyimpan nilai-nilai bilangan yang dimasukkan oleh pengguna. Variabel temp digunakan untuk menyimpan hasil penggabungan (merge) saat proses berlangsung.
labelArray adalah array dari JLabel yang digunakan untuk menampilkan array secara visual. steps merupakan antrian (Queue) yang menyimpan hasil-hasil setiap langkah merge yang akan ditampilkan satu per satu.
Komponen GUI seperti stepButton, resetButton, dan setButton digunakan sebagai kontrol untuk memulai, melanjutkan, dan mengulang proses sorting. Panel panelArray digunakan untuk menampilkan array dalam bentuk visual, sedangkan stepArea mencatat log atau riwayat langkah-langkah merge yang telah dilakukan.

```
public class MergeSortGUI extends JFrame {
   private int[] array;
   private int[] temp;
   private JLabel[] labelArray;
   private Queue<int[]> steps;
   private JTextArea stepArea;
   private JPanel panelArray;
   private JButton stepButton, resetButton, setButton;
   private JTextField inputField;
}
```

*iii. Method main()*

Method main() merupakan titik awal eksekusi program. Untuk memastikan GUI berjalan secara aman, kode GUI dijalankan di dalam EventQueue.invokeLater() yang memastikan bahwa GUI diproses di *Event Dispatch Thread*, yaitu thread khusus untuk menangani komponen Swing.

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

*iv. Konstruktor MergeSortGUI()*

Konstruktor MergeSortGUI() bertugas membangun tampilan antarmuka pengguna. Program menetapkan judul jendela, ukuran, lokasi tengah, dan tata

letak menggunakan BorderLayout. Panel input berada di atas, berisi inputField untuk data dan tombol "Set Array". Panel tengah menampilkan array dalam bentuk label.
Panel bawah berisi tombol kontrol "Langkah Selanjutnya" dan "Reset". Di sebelah kanan, terdapat stepArea yang menampilkan log dari proses merge, dibungkus dalam JScrollPane.
Setiap tombol dihubungkan dengan aksi tertentu, seperti memulai array, melangkah ke proses selanjutnya, dan mereset tampilan.

```
public MergeSortGUI() {
    setTitle("Merge Sort Langkah per Langkah");
    setSize(800, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    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);

    panelArray = new JPanel(new FlowLayout());

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

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

    add(inputPanel, BorderLayout.NORTH);
    add(panelArray, BorderLayout.CENTER);
    add(controlPanel, BorderLayout.SOUTH);
    add(scrollPane, BorderLayout.EAST);

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

*v. Method setArrayFromInput()*

Method ini mengambil input teks dari inputField, memisahkannya berdasarkan koma, lalu mengonversinya menjadi array integer. Array ini disiapkan untuk proses Merge Sort. Setiap elemen array ditampilkan sebagai JLabel di panelArray. Kemudian, method mergeSort() akan dipanggil untuk menyusun semua langkah sorting ke dalam queue steps. Tombol langkah selanjutnya diaktifkan.

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;

    try {
        String[] parts = text.split(",");
        array = new int[parts.length];
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim());
        }

        temp = new int[array.length];
        labelArray = new JLabel[array.length];
        steps = new LinkedList<>();
        stepArea.setText("");

        panelArray.removeAll();
        for (int k = 0; k < array.length; k++) {
            labelArray[k] = new JLabel(String.valueOf(array[k]));
            labelArray[k].setPreferredSize(new Dimension(40, 40));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);

labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
            panelArray.add(labelArray[k]);
        }

        mergeSort(0, array.length - 1);

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

        stepButton.setEnabled(true);

    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang
dipisahkan dengan koma.", "Input Tidak Valid",
JOptionPane.ERROR_MESSAGE);
    }
}
```

*vi. Method mergeSort(int left, int right)*

Method ini menjalankan algoritma Merge Sort secara rekursif. Array dibagi menjadi dua bagian hingga masing-masing berisi satu elemen. Kemudian, bagian-bagian tersebut digabung kembali dengan bantuan method merge() dan hasilnya disimpan ke dalam queue steps.

```
private void mergeSort(int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(left, mid);
        mergeSort(mid + 1, right);
        merge(left, mid, right);
    }
}
```

*vii. Method merge(int left, int mid, int right)*

Method ini melakukan penggabungan dua subarray terurut. Data hasil merge disalin ke temp, lalu ke array utama. Setiap hasil penggabungan disalin dan dimasukkan ke dalam steps untuk ditampilkan kemudian.

```
private void merge(int left, int mid, int right) {
    int i = left, j = mid + 1, k = left;

    while (i <= mid && j <= right) {
        if (array[i] <= array[j]) {
            temp[k++] = array[i++];
        } else {
            temp[k++] = array[j++];
        }
    }

    while (i <= mid) {
        temp[k++] = array[i++];
    }

    while (j <= right) {
        temp[k++] = array[j++];
    }

    for (int m = left; m <= right; m++) {
        array[m] = temp[m];
    }

    int[] snapshot = array.clone();
    steps.add(snapshot);
}
```

*viii. Method performStep()*

Method ini mengeksekusi satu langkah merge yang telah disiapkan di dalam queue. Data array yang telah diurutkan sebagian akan ditampilkan di panelArray, dan log-nya ditambahkan ke stepArea. Jika queue kosong, artinya proses sudah selesai.

```
private void performStep() {
    if (!steps.isEmpty()) {
        int[] current = steps.poll();
        for (int i = 0; i < current.length; i++) {
            labelArray[i].setText(String.valueOf(current[i]));
        }
        stepArea.append("Langkah: " + arrayToString(current) + "\n");

        if (steps.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Sorting selesai!");
            stepButton.setEnabled(false);
        }
    }
}
```

*ix. Method reset()*

Method ini digunakan untuk mengatur ulang semua komponen GUI dan variabel logika. Input teks, array visual, dan log langkah akan dikosongkan.

```
java
CopyEdit
private void reset() {
    inputField.setText("");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText("");
    stepButton.setEnabled(false);
}
```

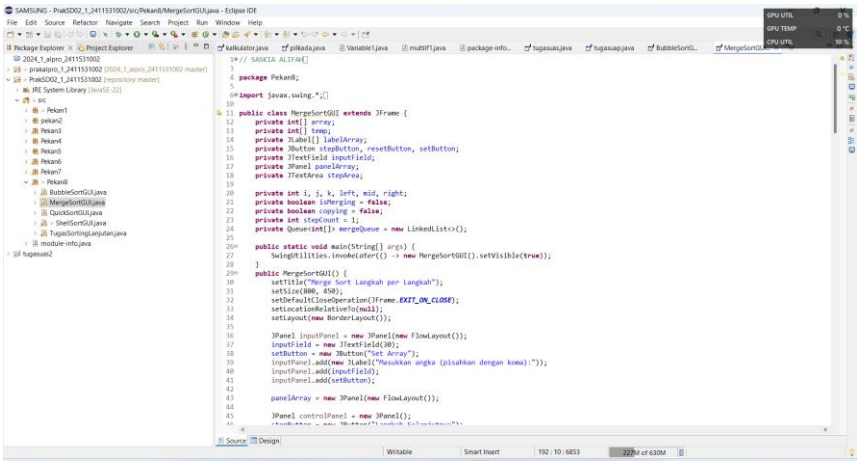
*x. Method arrayToString(int[] arr)*

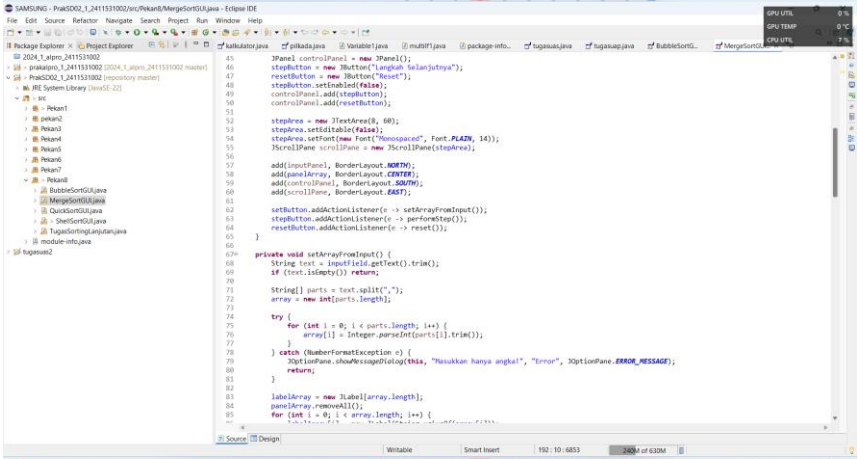
Method ini mengubah array integer menjadi string dengan elemen dipisahkan koma, yang akan digunakan sebagai log di area teks.

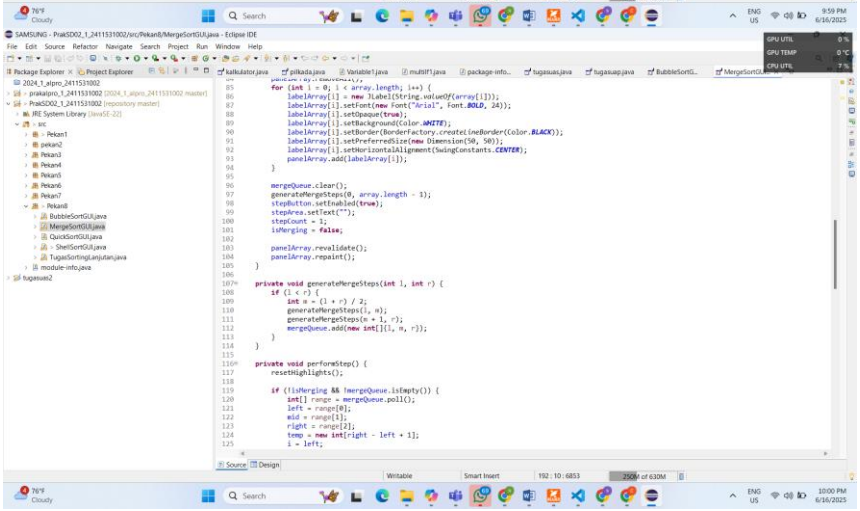
```
private String arrayToString(int[] arr) {
    StringBuilder sb = new StringBuilder();
    for (int k = 0; k < arr.length; k++) {
        sb.append(arr[k]);
        if (k < arr.length - 1) sb.append(", ");
```

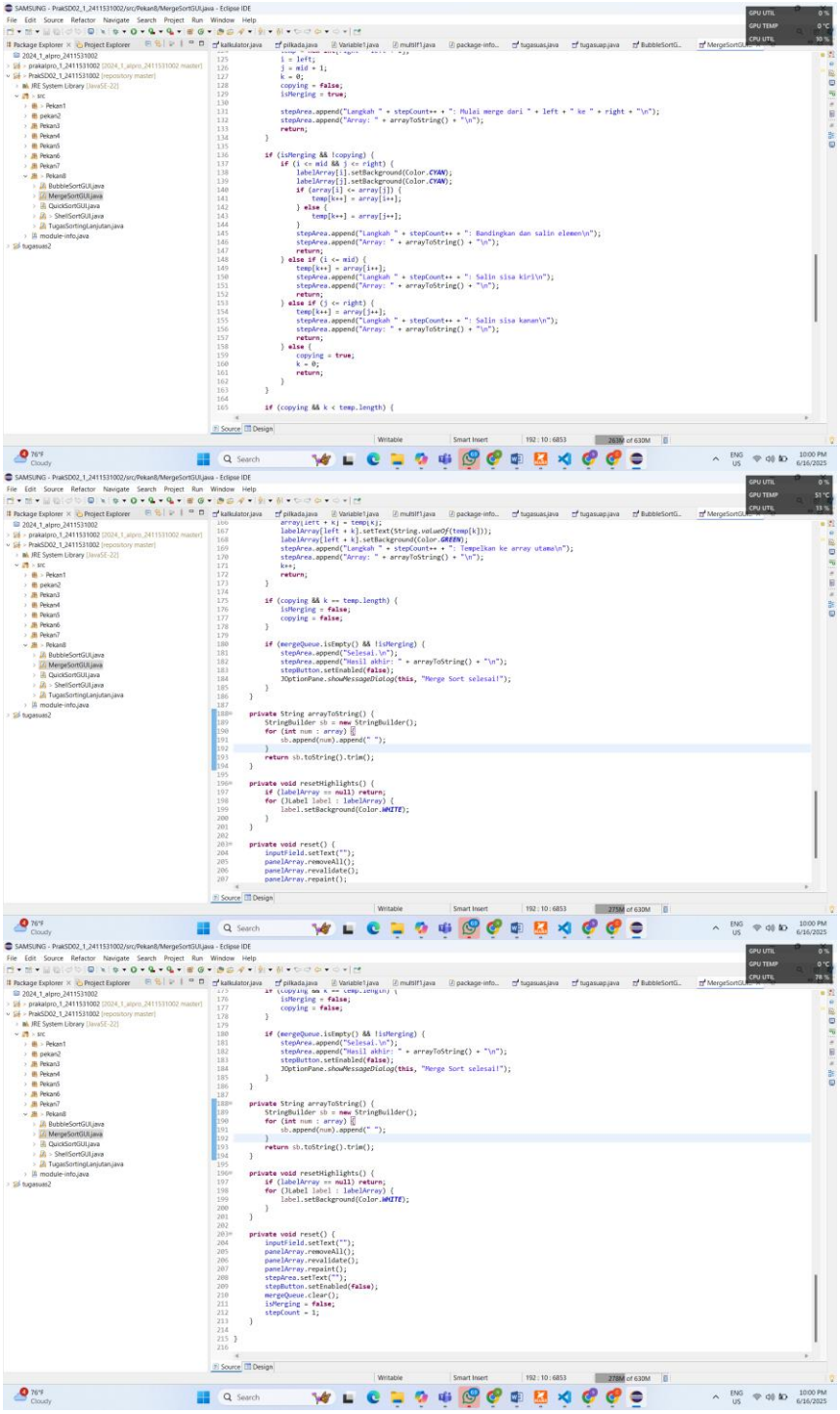
```
        }
        return sb.toString();
    }
```

**c) Syntax:**









## d) Diargram alur

- Mulai
- Jika array memiliki lebih dari satu elemen:
    - Bagi array menjadi dua bagian (kiri dan kanan)
    - Rekursif: Panggil merge sort untuk bagian kiri
    - Rekursif: Panggil merge sort untuk bagian kanan
    - Gabungkan dua bagian tersebut secara terurut (merge)
- Jika hanya satu elemen, kembalikan (basis rekursi)
- Selesai

## e) Error Handling

- Validasi input dilakukan dengan blok try-catch untuk menangani NumberFormatException saat parsing input ke integer.
- Jika input kosong atau mengandung karakter bukan angka, akan muncul **dialog peringatan (JOptionPane)** kepada pengguna.
- Tombol "Langkah Selanjutnya" hanya aktif jika array telah diset.
- Sorting tidak akan berjalan jika array tidak valid, dan proses bisa di-reset kapan saja dengan tombol “Reset”.

## f) Alur Sintaks Program

 GUI ditampilkan menggunakan JFrame dan layout diatur.

 Pengguna memasukkan input angka yang ingin diurutkan, dipisahkan dengan koma.

 Saat tombol **"Set Array"** ditekan:

- Array dikonversi dari string ke integer.
- Visualisasi array ditampilkan dalam bentuk label.

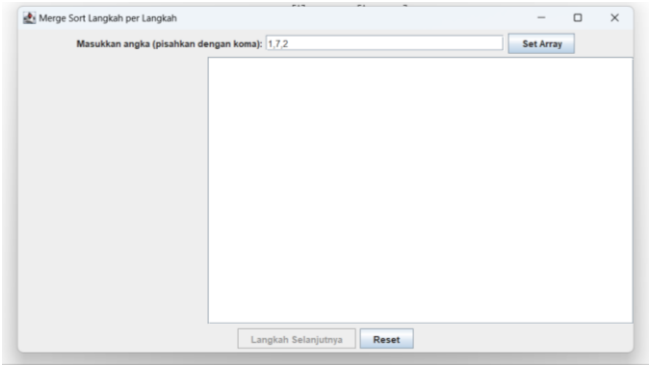
 Tombol **"Langkah Selanjutnya"** digunakan untuk menjalankan Merge Sort langkah demi langkah (menggunakan queue).

 Setelah penggabungan selesai, muncul **notifikasi** bahwa proses sorting selesai.

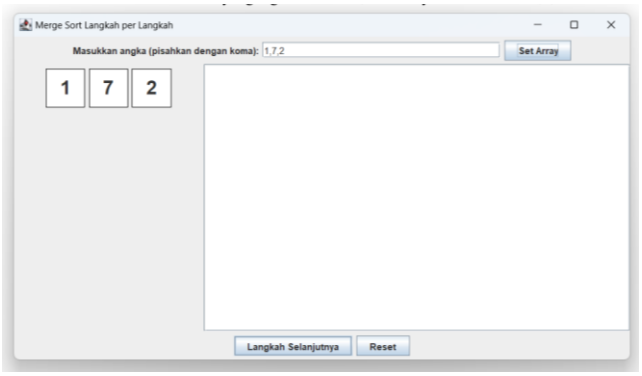
 Tombol **"Reset"** menghapus semua hasil dan memulai ulang proses.

## g) Output Program

i. **Masukkan angka**
masukkanlah yang ingin di urutkan, disini saya memasukkan 1,7,2.




**ii. Pencet set array**
maka akan muncul tampilan seperti kotak kotak di sebelah kiri.




### iii. Langkah Pengerjaan

Langkah 1: Mulai merge dari 0 ke 1

Array: 1 7 2

Langkah 2: Bandingkan dan salin elemen

Array: 1 7 2

Langkah 3: Salin sisa kanan

Array: 1 7 2

Langkah 4: Tempelkan ke array utama

Array: 1 7 2

Langkah 5: Tempelkan ke array utama

Array: 1 7 2

Langkah 6: Mulai merge dari 0 ke 2

Array: 1 7 2

Langkah 7: Bandingkan dan salin elemen

Array: 1 7 2

Langkah 8: Bandingkan dan salin elemen

Array: 1 7 2

Langkah 9: Salin sisa kiri

Array: 1 7 2

Langkah 10: Tempelkan ke array utama

Array: 1 7 2

Langkah 11: Tempelkan ke array utama

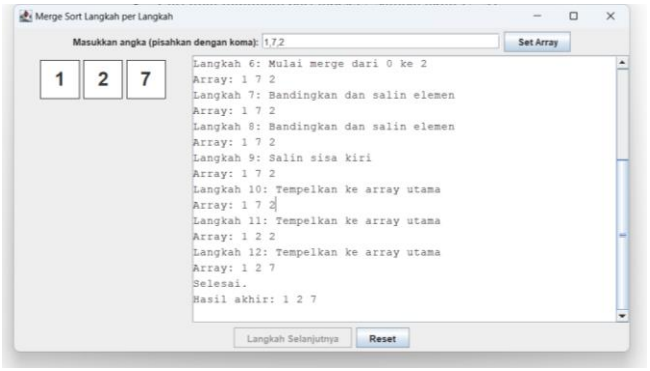
Array: 1 2 2

Langkah 12: Tempelkan ke array utama

Array: 1 2 7

Selesai.

Hasil akhir: 1 2 7




## 3. Kelas QuickSortGUI

### a) Deskripsi Kelas

QuickSortGUI adalah kelas utama untuk memvisualisasikan proses algoritma Quick Sort secara interaktif dalam aplikasi berbasis GUI. Kelas ini mewarisi JFrame, memungkinkan pembuatan jendela utama aplikasi. Antarmuka pengguna mencakup field input untuk array, panel visualisasi array, tombol kontrol ("Set Array", "Langkah Selanjutnya", "Reset"), serta area teks untuk menampilkan log tiap langkah sorting. Sorting dilakukan secara bertahap setiap klik "Langkah Selanjutnya", memudahkan pemahaman logika Quick Sort yang kompleks.

### b) Penjelasan Fungsi dan Cara Kerja

*i. Package dan Import*

Program ini berada dalam package Pekan8, yang menandakan bahwa file ini merupakan bagian dari folder atau modul minggu ke-8. Package mempermudah organisasi kode dalam proyek Java. Import java.awt.* digunakan untuk komponen GUI dasar seperti BorderLayout dan FlowLayout, sementara javax.swing.* menyediakan komponen GUI tingkat lanjut seperti JFrame, JLabel, JButton, dan JTextArea. Class Queue dan LinkedList dari java.util.* digunakan untuk

menyimpan dan mengelola langkah-langkah Quick Sort secara berurutan menggunakan struktur data antrian (queue).

```
package Pekan8;

import javax.swing.*;
import java.awt.*;
import java.util.LinkedList;
import java.util.Queue;
```

*ii. Deklarasi Kelas dan Variabel Atribut*

Kelas QuickSortGUI merupakan subclass dari JFrame, yang artinya kelas ini mewarisi kemampuan jendela aplikasi GUI Java. Variabel array menyimpan data yang akan diurutkan. labelArray adalah array label untuk menampilkan visualisasi elemen array. Komponen GUI lainnya mencakup setButton untuk menyimpan input array, stepButton untuk menjalankan langkah Quick Sort, resetButton untuk mengatur ulang program, inputField untuk menerima masukan array, serta panelArray dan stepArea untuk visualisasi array dan log proses sorting. Queue<int[]> steps menyimpan daftar langkah pemanggilan Quick Sort (berupa pasangan indeks kiri-kanan) yang akan dieksekusi satu per satu.

```
public class QuickSortGUI extends JFrame {
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;

    private Queue<int[]> steps = new LinkedList<>();
    private int stepCount = 1;
}
```

*iii. Method main()*

Method main() adalah titik awal program Java. Untuk memastikan bahwa semua komponen GUI dibuat dan dijalankan di thread yang sesuai, kode GUI ditempatkan dalam EventQueue.invokeLater(). Hal ini mencegah error pada tampilan dan memastikan GUI merespon secara konsisten.

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

*iv. Konstruktor QuickSortGUI()*

Konstruktor QuickSortGUI() digunakan untuk membangun antarmuka pengguna. Jendela disusun dengan BorderLayout. Di bagian atas (NORTH) terdapat panel input berisi inputField dan tombol setButton. Panel tengah (CENTER) digunakan untuk menampilkan elemen array dengan panelArray. Di bawah (SOUTH), terdapat stepButton dan resetButton. Sebelah kanan (EAST) menampilkan stepArea dalam JScrollPane untuk mencatat langkah Quick Sort. Setiap tombol diberi aksi khusus menggunakan ActionListener.

```
public QuickSortGUI() {
    setTitle("Quick Sort Langkah per Langkah");
    setSize(800, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout());

    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);

    panelArray = new JPanel(new FlowLayout());

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

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

    add(inputPanel, BorderLayout.NORTH);
    add(panelArray, BorderLayout.CENTER);
    add(controlPanel, BorderLayout.SOUTH);
    add(scrollPane, BorderLayout.EAST);

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

*v. Method setArrayFromInput()*

Method ini memproses input dari inputField, mengubah string yang dipisahkan koma menjadi array integer. Jika format input tidak sesuai, akan muncul pesan kesalahan. Array divisualisasikan ke dalam labelArray, lalu langkah awal Quick Sort (rentang indeks array) dimasukkan ke dalam steps. Tombol stepButton diaktifkan sebagai sinyal bahwa pengguna bisa mulai melangkah.

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;

    try {
        String[] parts = text.split(",");
        array = new int[parts.length];
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim());
        }

        stepCount = 1;
        steps.clear();
        steps.add(new int[]{0, array.length - 1});
        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].setPreferredSize(new Dimension(40, 40));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);

labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
            panelArray.add(labelArray[k]);
        }
        panelArray.revalidate();
        panelArray.repaint();

    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang
dipisahkan dengan koma.", "Input Tidak Valid",
JOptionPane.ERROR_MESSAGE);
    }
}
```

*vi. Method performStep()*

Method ini menjalankan satu langkah dari algoritma Quick Sort. Rentang indeks diambil dari Queue. Jika low < high, maka dilakukan partisi menggunakan metode partition(). Indeks partisi akan menentukan dua langkah baru yang dimasukkan

kembali ke Queue. Proses ini dicatat dalam stepArea dan ditampilkan secara visual. Jika langkah-langkah sudah habis, maka sorting dianggap selesai.

```
private void performStep() {
    if (steps.isEmpty()) {
        stepButton.setEnabled(false);
        JOptionPane.showMessageDialog(this, "Sorting selesai!");
        return;
    }

    int[] range = steps.poll();
    int low = range[0];
    int high = range[1];

    if (low < high) {
        int pivotIndex = partition(low, high);
        steps.add(new int[]{low, pivotIndex - 1});
        steps.add(new int[]{pivotIndex + 1, high});

        stepArea.append("Langkah " + stepCount + ": Partisi dari indeks " + low + "
ke " + high + "\n");
        stepArea.append("Pivot di indeks " + pivotIndex + ", Hasil: " +
arrayToString(array) + "\n\n");
        updateLabels();
        stepCount++;
    }
}
```

*vii. Method partition(int low, int high)*

Method ini menjalankan logika partisi Quick Sort dengan menggunakan elemen terakhir sebagai pivot. Elemen yang lebih kecil dari pivot dipindahkan ke kiri. Setelah selesai, pivot ditempatkan pada posisi akhir yang benar. Indeks akhir pivot dikembalikan agar dapat digunakan dalam langkah selanjutnya.

```
private int partition(int low, int high) {
    int pivot = array[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            i++;
            int temp = array[i];
            array[i] = array[j];
            array[j] = temp;
        }
    }
    int temp = array[i + 1];
    array[i + 1] = array[high];
    array[high] = temp;
    return i + 1;
}
```

*viii. Method reset()*

Method ini mengatur ulang semua elemen GUI dan logika program. Input dan hasil langkah-langkah dibersihkan, tampilan array dihapus, serta tombol langkah dinonaktifkan. Program siap untuk menerima input baru.

```
private void reset() {
   inputField.setText("");
   panelArray.removeAll();
   panelArray.revalidate();
   panelArray.repaint();
   stepArea.setText("");
   stepButton.setEnabled(false);
   steps.clear();
   stepCount = 1;
}
```

*ix. Method updateLabels()*

Method ini memperbarui tampilan label array untuk mencerminkan perubahan nilai setelah langkah partisi dilakukan. Setiap elemen labelArray akan menampilkan nilai terbaru dari array.

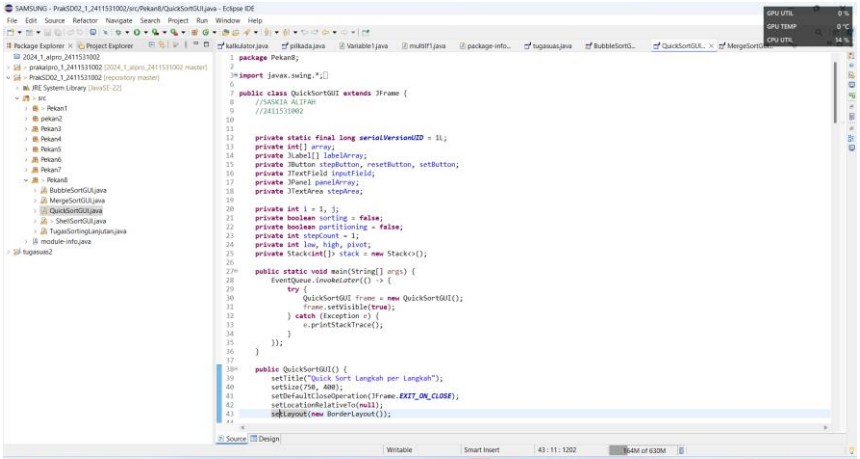
```
private void updateLabels() {
   for (int k = 0; k < array.length; k++) {
      labelArray[k].setText(String.valueOf(array[k]));
   }
}
```

*x. Method arrayToString(int[] arr)*

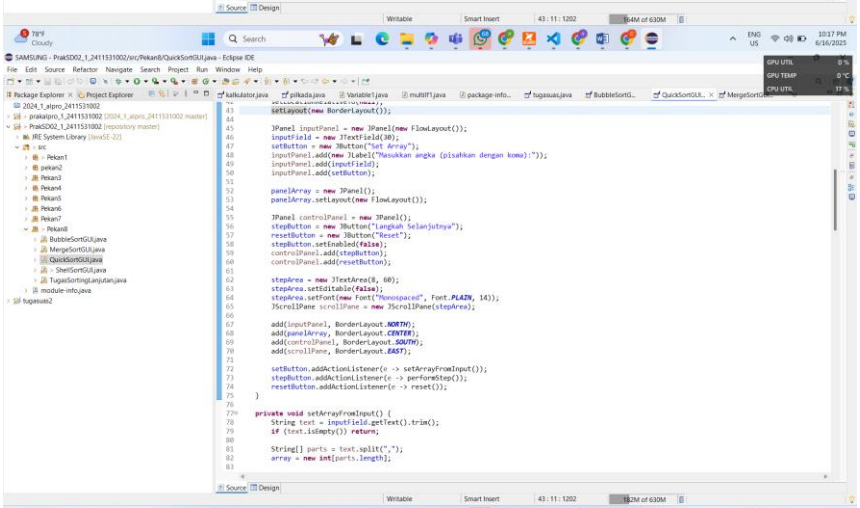
Method ini mengubah array integer menjadi string yang dipisahkan koma, sehingga bisa ditampilkan di stepArea.

```
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();
}
```

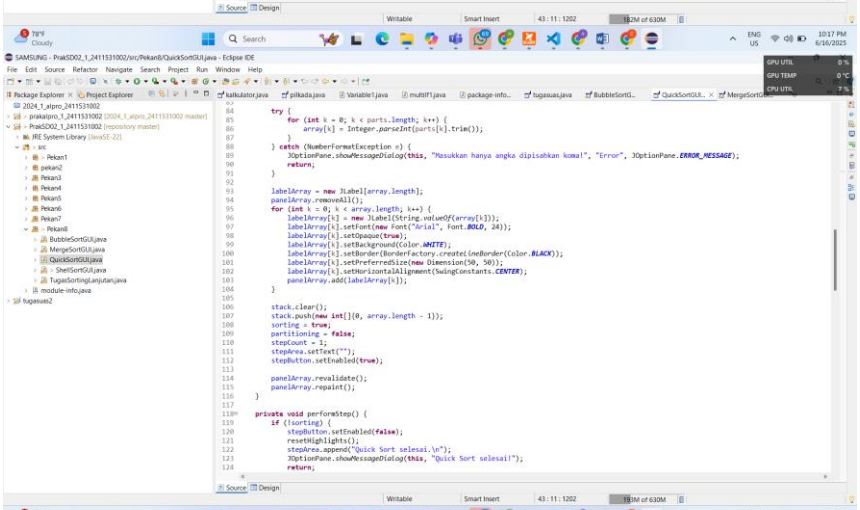
**c) Syntax:**



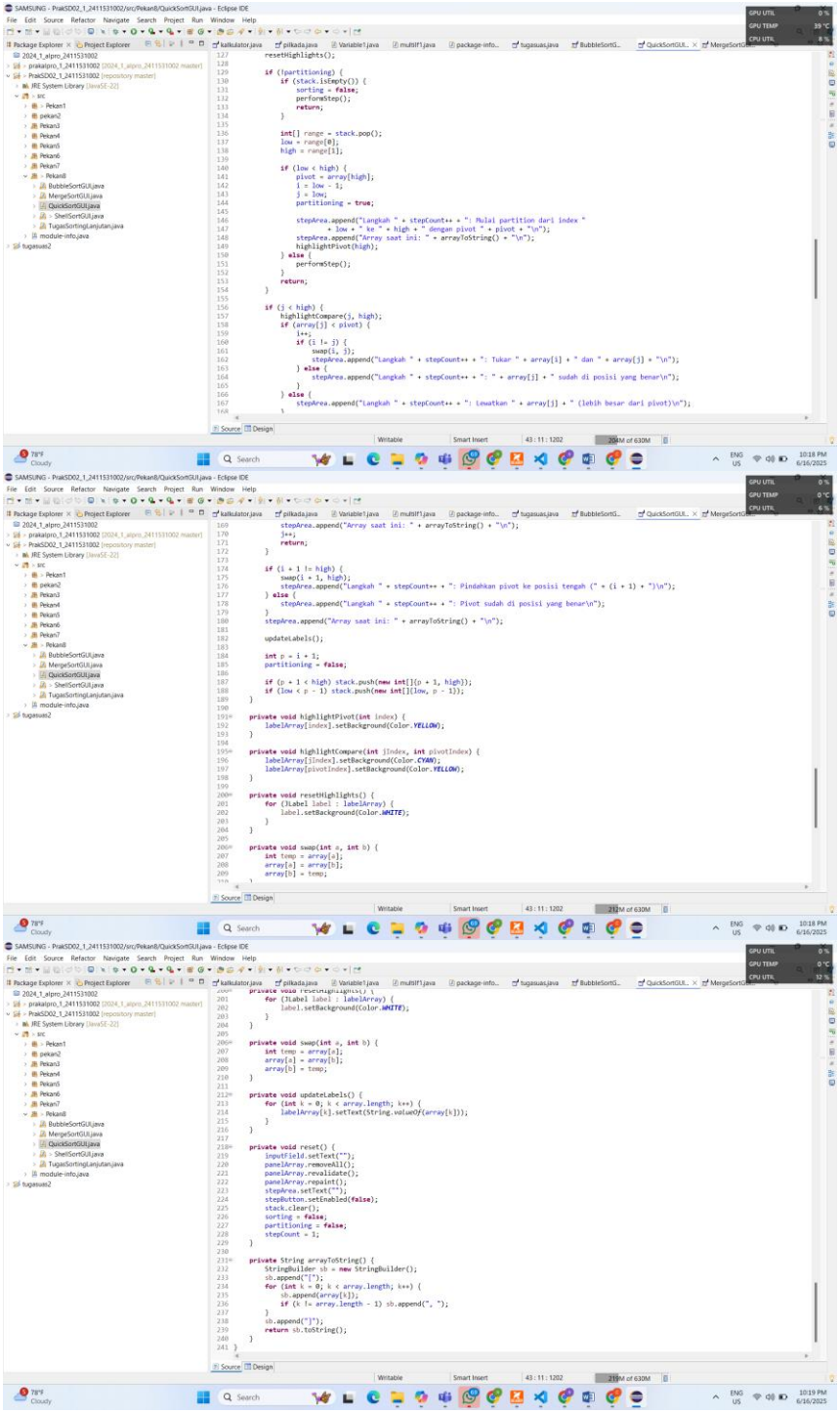
```
package Pekan8;

import javax.swing.*;

public class QuickSortGUI extends JFrame {
    //5456IA ALIFAH
    //2411531002


    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 boolean partitioning = false;
    private int stepCount = 1;
    private int low, high, pivot;
    private Stack<int[]> stack = new Stack<>();

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

    public QuickSortGUI() {
        setTitle("Quick Sort Langkah per Langkah");
        setSize(750, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
```



```
        setLayout(new BorderLayout());

        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);

        panelArray = new JPanel();
        panelArray.setLayout(new FlowLayout());

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

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

        add(inputPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(scrollPane, BorderLayout.EAST);

        setButton.addActionListener(e -> setArrayFromInput());
        stepButton.addActionListener(e -> performStep());
        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 dipisahkan 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].setOpaque(true);
            labelArray[k].setBackground(Color.WHITE);
            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
            labelArray[k].setPreferredSize(new Dimension(50, 50));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
            panelArray.add(labelArray[k]);
        }

        stack.clear();
        stack.push(new int[]{0, array.length - 1});
        sorting = true;
        partitioning = false;
        stepCount = 1;
        stepArea.setText("");
        stepButton.setEnabled(true);

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

    private void performStep() {
        if (!sorting) {
            stepButton.setEnabled(false);
            resetHighlights();
            stepArea.append("Quick Sort selesai.\n");
            JOptionPane.showMessageDialog(this, "Quick Sort selesai!");
            return;
```



### d) Diargram alur

- Mulai
- Masukkan input array
- Push [0, n-1] ke stack
- Selama stack tidak kosong:

- Pop rentang [low, high]
- Pilih pivot (array[high])
- Partisi array dan tempatkan pivot di posisi benar
- Push sub-array kiri dan kanan ke stack

 Selesai jika stack kosong

**e) Error Handling**

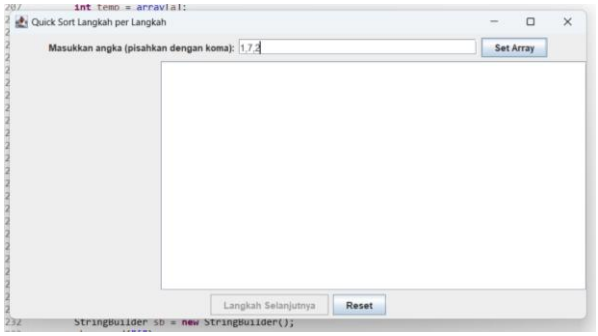
- Validasi input angka menggunakan try-catch.
- Stack kosong → sorting selesai.
- Tidak bisa klik tombol saat belum input array.
- JOptionPane muncul saat input salah atau sorting selesai.

**f) Alur Sintaks Program**

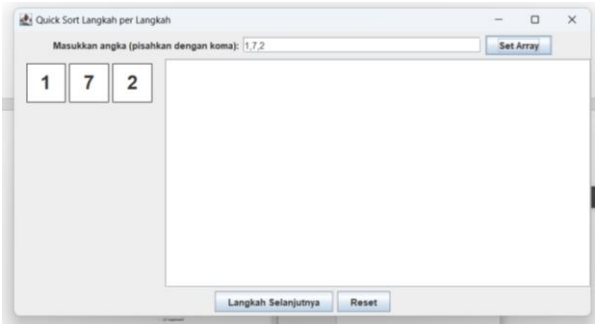
- GUI muncul.
- Pengguna input array → klik "Set Array".
- Visualisasi array tampil.
- Klik "Langkah Selanjutnya" untuk proses sorting satu demi satu.
- Setiap langkah ditampilkan di area log.
- Jika selesai, akan muncul pesan pop-up.
- Klik "Reset" untuk mengulang.

**g) Output Program**

i. **Masukkan angka**
masukkanlah yang ingin di urutkan, disini saya memasukkan 1,7,2.




**ii. Pencet set array**
maka akan muncul tampilan seperti kotak kotak di sebelah kiri.




### iii. Langkah 1

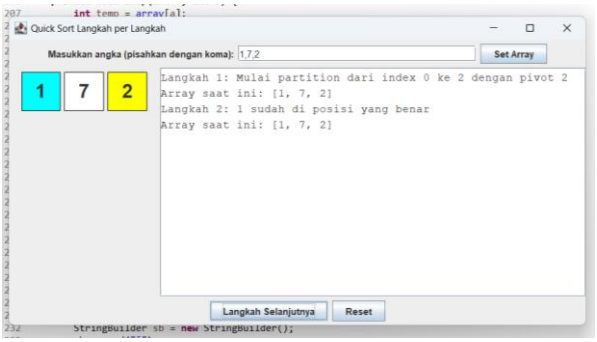
**Pivot = 2** (elemen terakhir)
Partition dimulai dari indeks 0 ke 2.
Quick Sort memilih elemen terakhir sebagai pivot. Pada tahap ini, pivot yang dipilih adalah 2. Proses akan membandingkan setiap elemen dari kiri ke kanan dengan pivot untuk menentukan posisinya.

**Array saat ini:** [1, 7, 2]


### iv. Langkah 2

**Bandingkan 1 dengan pivot 2 → 1 < 2**, jadi tetap di tempat.
Elemen 1 lebih kecil dari pivot, sehingga dibiarkan di tempatnya. Pointer indeks partisi (biasanya disebut i) naik satu untuk menandai posisi pertukaran berikutnya.

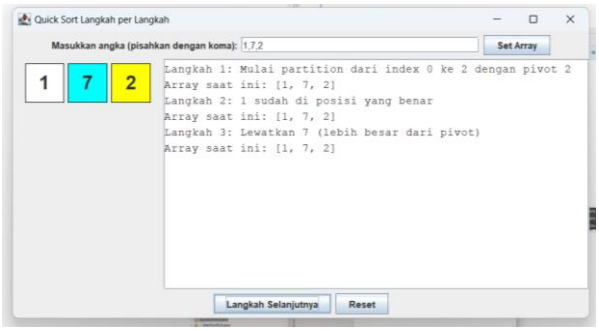
**Array saat ini:** [1, 7, 2]




### v. Langkah 3

**Bandingkan 7 dengan pivot 2 → 7 > 2**, jadi dilewati.
Karena 7 lebih besar dari pivot, tidak dilakukan pertukaran. Pointer tidak bergerak, dan lanjut ke tahap akhir partisi.

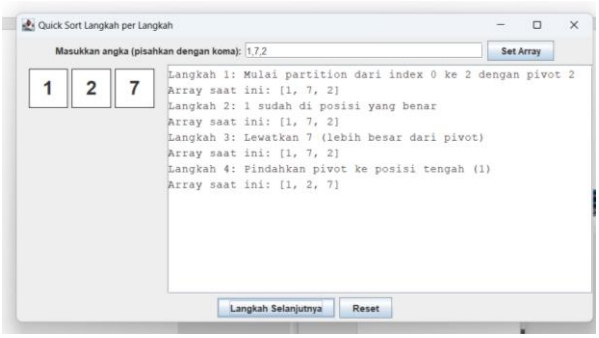
**Array saat ini:** [1, 7, 2]




### vi. Langkah 4

**Tukar pivot (2) dengan elemen di posisi indeks 1 (7)**
Array menjadi: [1, 2, 7]
Pivot 2 ditukar ke posisi yang seharusnya — yaitu ke indeks tempat seharusnya elemen lebih besar dari 2 dimulai. Maka 2 dan 7 ditukar. Sekarang elemen sebelum pivot sudah lebih kecil, dan sesudahnya lebih besar. Quick Sort selesai karena bagian kiri [1] dan kanan [7] hanya satu elemen.

**Array saat ini:** [1, 2, 7]




### vii. Kesimpulan Output:

Quick Sort berhasil mengurutkan array dengan memilih elemen terakhir sebagai pivot (yaitu angka 2). Melalui proses partisi, elemen yang lebih kecil dari pivot dibiarkan di kiri, sedangkan yang lebih besar tetap di kanan. Setelah pivot ditempatkan di posisi yang benar, algoritma selesai karena setiap sub-array sisanya hanya berisi satu elemen. Proses ini menunjukkan efisiensi Quick Sort dalam menyortir array kecil dengan sedikit langkah.


## 4. Kelas ShellSortGUI

### a) Deskripsi Kelas

ShellSortGUI adalah kelas utama untuk visualisasi algoritma Shell Sort dalam bentuk aplikasi GUI. Kelas ini merupakan turunan dari JFrame, yang berarti dapat membuat jendela aplikasi grafis. Komponen utamanya terdiri dari input field untuk menerima array angka dari pengguna, panel visual yang menampilkan array dalam bentuk label, tombol kontrol seperti Set Array, Langkah Selanjutnya, dan Reset, serta area teks untuk mencatat dan menampilkan log setiap langkah dalam proses sorting. Proses Shell Sort dijalankan secara bertahap, di mana setiap kali tombol "Langkah Selanjutnya" ditekan, satu langkah sorting dieksekusi. Hal ini memungkinkan pengguna untuk memahami cara kerja algoritma Shell Sort secara interaktif dan visual, terutama konsep gap-based sorting yang menjadi ciri khas algoritma ini.

### b) Penjelasan Fungsi dan Cara Kerja

*i. Package dan Import*

Program ini berada dalam package Pekan8, yang menunjukkan bahwa file ini merupakan bagian dari modul atau folder Pekan8. Package dalam Java membantu pengelompokan kelas agar lebih terstruktur.
Import java.awt.* digunakan untuk komponen GUI dasar seperti layout dan warna. javax.swing.* digunakan untuk komponen GUI tingkat lanjut seperti JFrame, JLabel, JButton, JTextField, dan JScrollPane. Program ini juga mengimpor event listener ActionListener untuk menangani aksi tombol.

```
package Pekan8;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
```

*ii. Deklarasi Kelas dan Variabel Atribut*

Kelas ShellSortGUI merupakan turunan dari JFrame, yang berarti kelas ini mewarisi semua kemampuan dari jendela antarmuka Swing.
Terdapat beberapa atribut penting:

- array menyimpan nilai-nilai angka dari input.
- labelArray adalah array dari JLabel yang merepresentasikan setiap elemen array secara visual.
- stepButton, resetButton, dan setButton adalah tombol utama GUI.
- inputField digunakan untuk menerima input angka dari pengguna.
- panelArray menampilkan elemen array dalam bentuk label.
- stepArea mencatat setiap langkah Shell Sort.
- Variabel tambahan seperti gap, i, j, stepCount, sorting, dan temp digunakan untuk mengatur logika langkah demi langkah algoritma Shell Sort.

```
public class ShellSortGUI extends JFrame {
   private int[] array;
   private JLabel[] labelArray;
   private JButton stepButton, resetButton, setButton;
   private JTextField inputField;
   private JPanel panelArray;
   private JTextArea stepArea;

   private int gap, i, j, temp, stepCount = 1;
   private boolean sorting = false, isSwapping = false;
}
```

*iii. Method main()*

Method main() merupakan titik awal eksekusi program Java. Untuk menjaga kestabilan GUI, program dijalankan dalam SwingUtilities.invokeLater(), yang memastikan eksekusi berlangsung di Event Dispatch Thread (EDT), thread khusus untuk komponen GUI Swing.

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

*iv. Konstruktor ShellSortGUI()*

Konstruktor ini bertugas membangun dan menyusun seluruh tampilan GUI.

- inputPanel di bagian atas berisi JTextField dan tombol Set Array.

- panelArray di tengah digunakan untuk menampilkan array sebagai label.
- controlPanel di bawah menampilkan tombol "Langkah Selanjutnya" dan "Reset".
- stepArea di sebelah kanan dalam JScrollPane berfungsi untuk mencatat setiap langkah Shell Sort.
  Event listener ditambahkan ke setiap tombol untuk menghubungkan antarmuka dengan logika pemrosesan.

```
public ShellSortGUI() {
   ...
   setButton.addActionListener(e -> setArrayFromInput());
   stepButton.addActionListener(e -> performStep());
   resetButton.addActionListener(e -> reset());
}
```

*v. Method setArrayFromInput()*

Method ini bertugas membaca input dari pengguna melalui inputField. Input berupa angka yang dipisahkan koma akan diubah menjadi array integer. Jika terjadi kesalahan format, maka akan muncul pesan peringatan.
Setelah array berhasil dibuat:

- Panel visual (panelArray) diperbarui dengan label-label baru.
- gap diinisialisasi sebagai panjang array / 2 sebagai ciri khas Shell Sort.
- Variabel kontrol i, j, dan stepCount juga diinisialisasi untuk memulai proses sorting.

```
private void setArrayFromInput() {
   ...
   gap = array.length / 2;
   i = gap;
   j = i;
   sorting = true;
   stepCount = 1;
   ...
}
```

*vi. Method performStep()*

Method ini menjalankan **satu langkah algoritma Shell Sort**.

- Selama sorting aktif dan gap > 0, program akan membandingkan elemen-elemen yang terpisah sejauh gap.
- Jika ditemukan elemen yang lebih besar dari temp, elemen tersebut digeser ke kanan.
- Jika tidak, temp ditempatkan pada posisi yang benar.
- Setelah satu bagian selesai (i >= array.length), nilai gap dibagi dua dan proses dimulai kembali.

- Saat gap menjadi 0, sorting dianggap selesai dan tombol langkah akan dinonaktifkan.

```
private void performStep() {
    ...
    if (!sorting || gap == 0) {
        stepArea.append("Shell Sort selesai.\n");
        stepButton.setEnabled(false);
        return;
    }
    ...
}
```

*vii. Method reset()*

Method ini digunakan untuk mengatur ulang program ke kondisi awal. Semua input, array visual, dan log langkah akan dibersihkan. Tombol "Langkah Selanjutnya" akan dinonaktifkan. Variabel kontrol juga dikembalikan ke nilai awal.

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

*viii. Method logStep(String message)*

Method ini digunakan untuk mencatat langkah yang sedang dijalankan ke dalam stepArea, lengkap dengan kondisi array saat ini. Setiap pemanggilan method ini akan mencetak satu langkah dan array terkini.

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

*ix. Method updateLabels()*

Method ini memperbarui tampilan visual dari labelArray agar mencerminkan isi terkini dari array. Ini dilakukan setiap kali terjadi pertukaran atau geseran elemen dalam array.

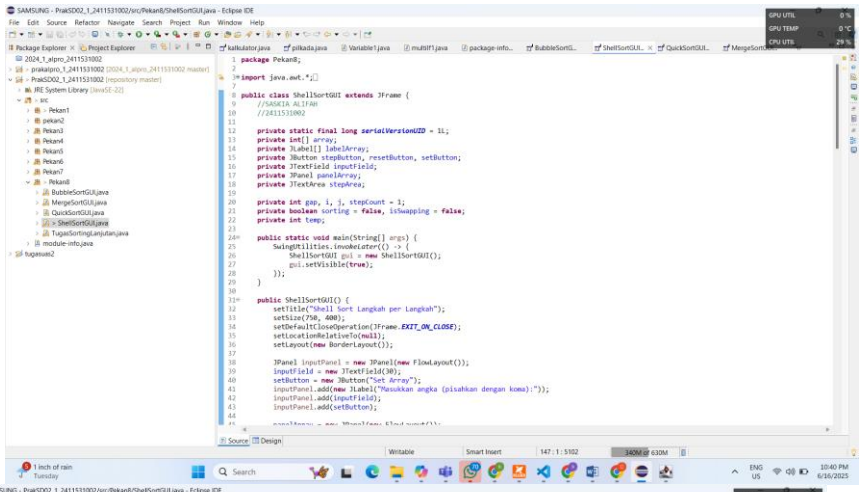
```
private void updateLabels() {
    for (int k = 0; k < array.length; k++) {
        labelArray[k].setText(String.valueOf(array[k]));
    }
}
```

*x. Method resetHighlights()*

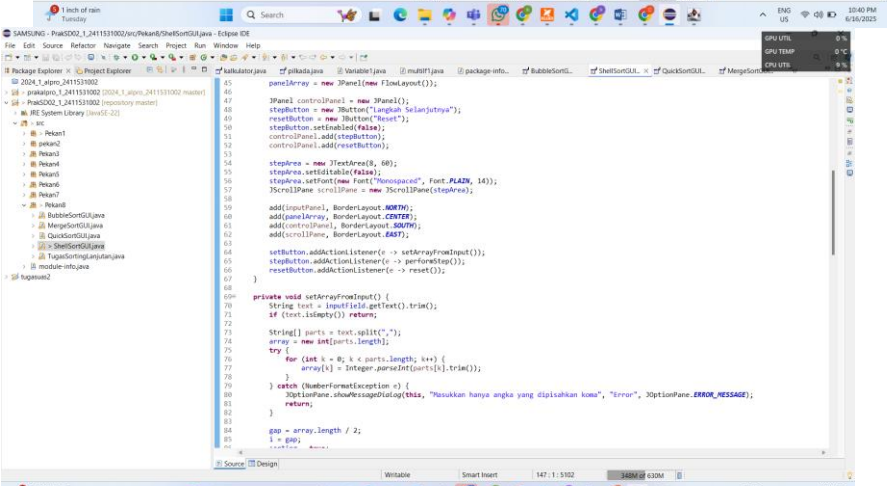
Method ini digunakan untuk menghapus pewarnaan latar belakang pada label array, agar setiap langkah baru dimulai dengan tampilan yang bersih tanpa warna sisa langkah sebelumnya.

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

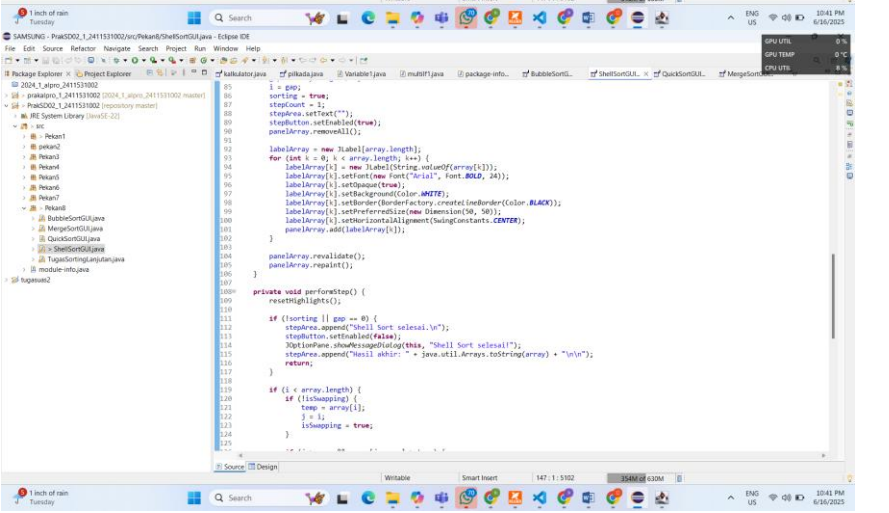
**c)Syntax:**



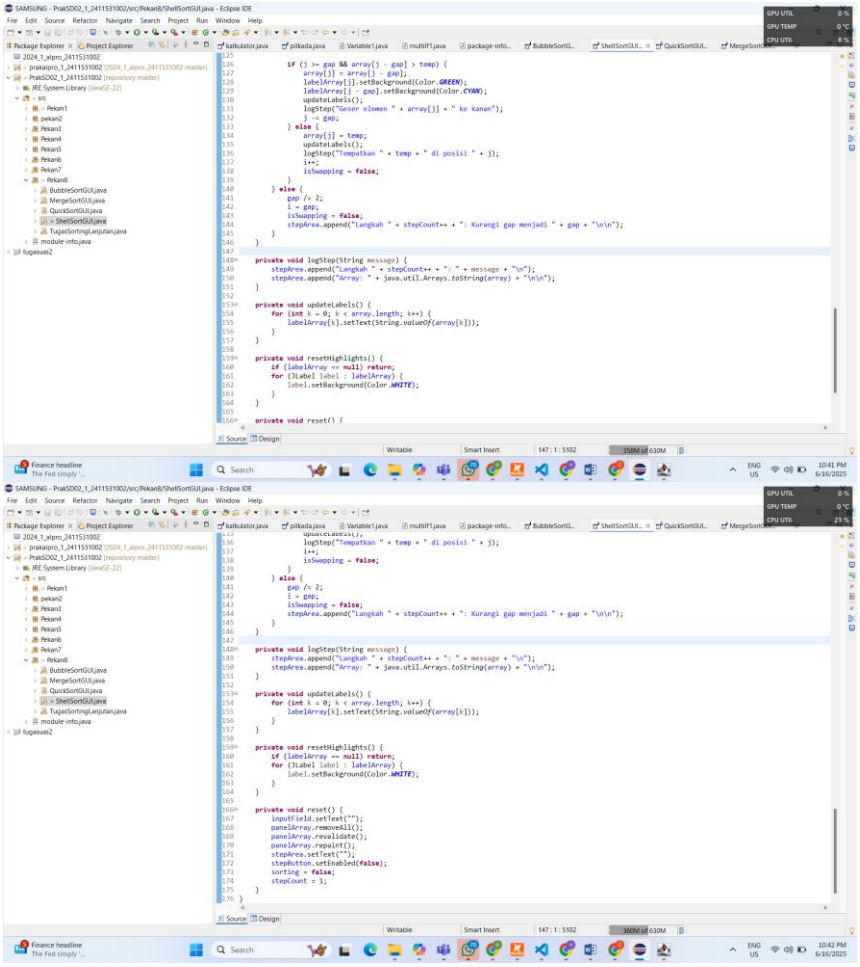
```java
package Pekan8;

import java.awt.*;

public class ShellSortGUI extends JFrame {
    //SASKIA ALIFAH
    //2411531002

    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 gap, i, j, stepCount = 1;
    private boolean sorting = false, isSwapping = false;
    private int temp;

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

    public ShellSortGUI() {
        setTitle("Shell Sort Langkah per Langkah");
        setSize(750, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());

        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);
```



```java
        panelArray = new JPanel(new FlowLayout());

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

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

        add(inputPanel, BorderLayout.NORTH);
        add(panelArray, BorderLayout.CENTER);
        add(controlPanel, BorderLayout.SOUTH);
        add(scrollPane, BorderLayout.EAST);

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

        gap = array.length / 2;
        i = gap;
```



```java
        i = gap;
        sorting = true;
        stepCount = 1;
        stepArea.setText("");
        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].setOpaque(true);
            labelArray[k].setBackground(Color.WHITE);
            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
            labelArray[k].setPreferredSize(new Dimension(50, 50));
            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
            panelArray.add(labelArray[k]);
        }

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

    private void performStep() {
        resetHighlights();

        if (!sorting || gap == 0) {
            stepArea.append("Shell Sort selesai.\n");
            stepButton.setEnabled(false);
            JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
            stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
            return;
        }

        if (i < array.length) {
            if (!isSwapping) {
                temp = array[i];
                j = i;
                isSwapping = true;
            }
```



## d) Diargram alur

- Mulai
- Inisialisasi gap = n/2
- Selama gap > 0:
    - Untuk i dari gap hingga n-1:
        - simpan nilai array[i] sebagai temp
        - j = i
        - selama j >= gap dan array[j - gap] > temp:
            - array[j] = array[j - gap]
            - j = j - gap
        - array[j] = temp
    - gap dibagi 2 (gap = gap / 2)
- Selesai

## e) Error Handling

- **Validasi input** dilakukan menggunakan blok try-catch untuk menangani kesalahan saat parsing angka, seperti NumberFormatException.
- **Tombol Langkah Selanjutnya** hanya diaktifkan jika input valid dan proses sorting dimulai.
- Jika proses telah selesai atau input kosong, tombol akan **dinonaktifkan** untuk mencegah error.

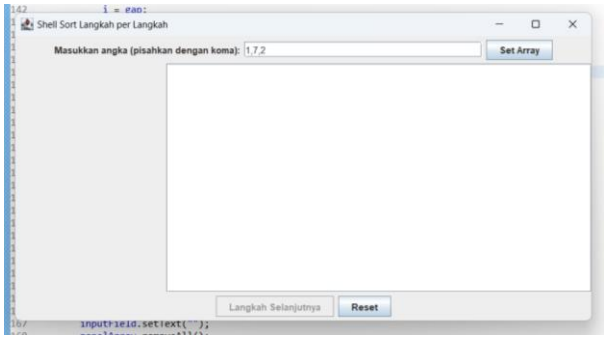
- Menggunakan **JOptionPane** untuk memberi pesan kesalahan (misalnya jika input tidak valid) atau notifikasi saat proses sorting telah selesai.

## f) Alur Sintaks Program

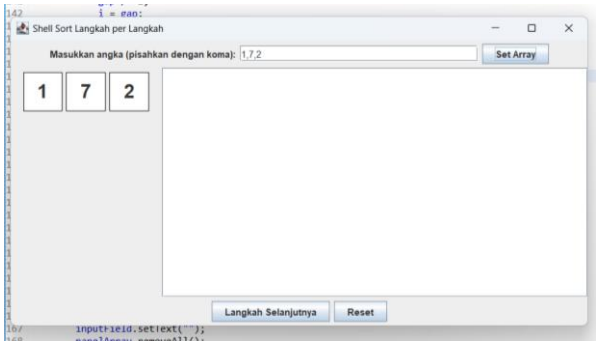
1. GUI diinisialisasi dan jendela utama ditampilkan.
2. Pengguna memasukkan array bilangan bulat dipisahkan koma, lalu menekan tombol "Set Array".
3. Visualisasi array akan muncul dalam bentuk **kotak-kotak label**.
4. Pengguna menekan tombol "Langkah Selanjutnya" untuk menjalankan satu langkah dari proses **Shell Sort**.
5. Proses sorting dilakukan berdasarkan nilai **gap**, dan setiap langkah dicatat di area teks (log).
6. Setelah gap menjadi nol dan array tersortir, akan muncul dialog notifikasi bahwa proses selesai.
7. Pengguna dapat menekan tombol "Reset" untuk mengosongkan input, array, dan log, lalu memulai ulang proses dari awal.

## g) Output Program

i. **Masukkan angka**
masukkanlah yang ingin di urutkan, disini saya memasukkan 1,7,2.



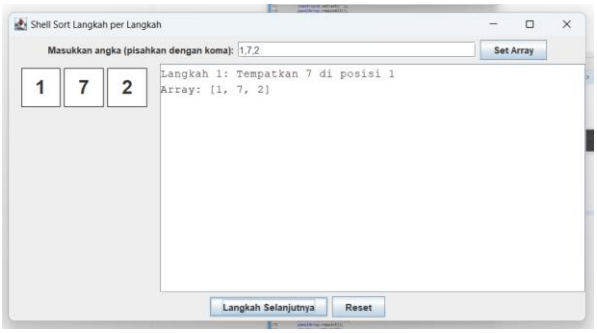

ii. **Pencet set array**
maka akan muncul tampilan seperti kotak kotak di sebelah kiri.




iii. **Langkah 1: Tempatkan 7 di posisi 1**

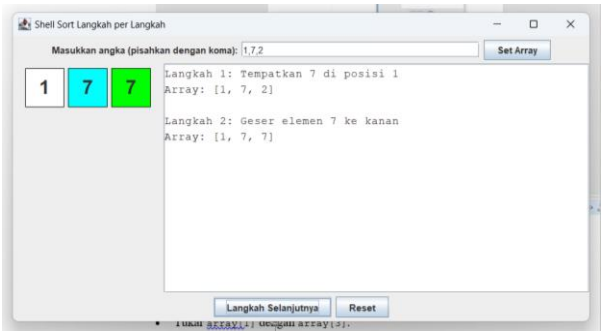
Pada tahap awal, gap = 1 (karena panjang array kecil). Shell Sort bekerja seperti Insertion Sort ketika gap = 1.

- Mulai dari indeks ke-1 (nilai = 7), dibandingkan dengan elemen sebelumnya (1).
- Karena 7 > 1, maka **tidak perlu digeser**.
- **7 tetap di posisinya**.
- Array saat ini: [1, 7, 2]



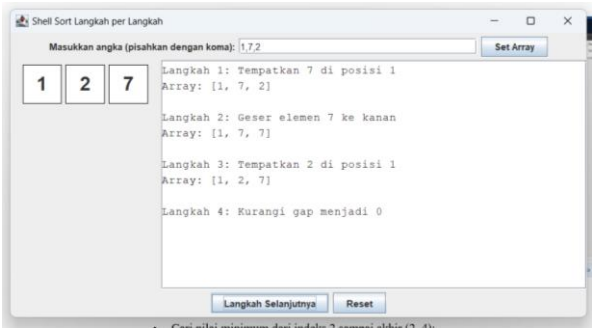

**iv. Langkah 2: Geser elemen 7 ke kanan**

- Sekarang indeks ke-2 (nilai = 2) yang sedang dievaluasi.
- Dibandingkan dengan elemen sebelumnya yaitu 7 di indeks ke-1.
- Karena **2 < 7**, maka **7 digeser ke kanan** untuk memberi ruang bagi 2.
- Array sementara menjadi: [1, 7, 7]



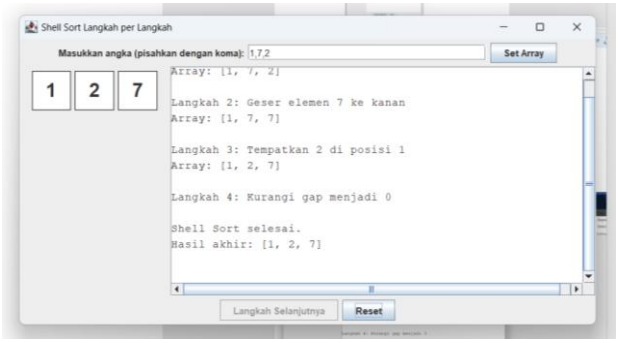

**v.** Langkah 3: Tempatkan 2 di posisi 1

- Setelah 7 digeser, periksa elemen sebelum 7 yaitu 1 di indeks 0.
- Karena **2 > 1**, maka tidak perlu menggeser lagi.
- 2 ditempatkan di indeks ke-1 (tempat kosong hasil geseran sebelumnya).
- Array setelah langkah ini menjadi: [1, 2, 7]



vi. Langkah 4: Kurangi gap menjadi 0

- Setelah seluruh elemen dievaluasi dengan gap = 1, gap dibagi dua → gap = 0.
- Ini menandakan proses sorting telah selesai.



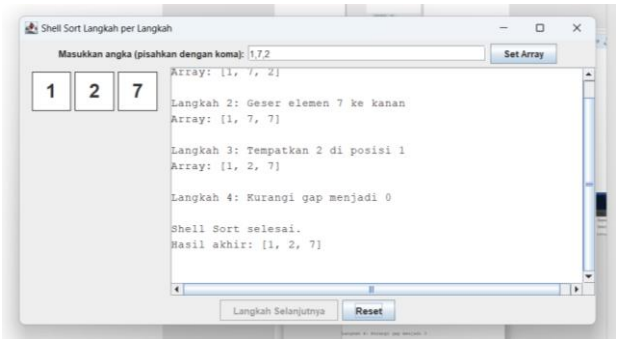

vii. **Kesimpulan Output:**

Pada algoritma **Shell Sort**, elemen-elemen diurutkan dengan cara membandingkan dan menukar elemen yang terpisah oleh jarak tertentu (*gap*), yang secara bertahap dikurangi hingga menjadi 1. Saat *gap* menjadi 1, proses serupa dengan **Insertion Sort** dilakukan sebagai langkah akhir penyempurnaan.

Berdasarkan input [1, 7, 2], proses Shell Sort berlangsung sebagai berikut:

- Elemen **7** dibandingkan dengan elemen sebelumnya **1**, tidak ada perubahan karena sudah benar.
- Elemen **2** dibandingkan dengan **7**, karena lebih kecil, maka **7 digeser ke kanan**.
- Kemudian **2** ditempatkan di posisi yang benar setelah dibandingkan dengan **1**.
- Hasil akhirnya menjadi **[1, 2, 7]**.

Shell Sort secara efisien menyusun data dengan **mengeliminasi elemen-elemen besar sejak awal**, meminimalisir jumlah perpindahan pada tahap akhir.



## D. KESIMPULAN

Praktikum ini menunjukkan bahwa setiap algoritma sorting memiliki cara kerja dan efisiensi yang berbeda. Bubble Sort sederhana namun lambat, Shell Sort lebih cepat dengan teknik pengelompokan, sedangkan Merge dan Quick Sort unggul dalam kecepatan dengan pendekatan divide and conquer.

Dengan visualisasi GUI, proses sorting dapat diamati secara interaktif, sehingga memudahkan pemahaman logika di balik algoritma. Praktikum ini membantu mahasiswa memahami konsep sorting tidak hanya dari sisi teori, tapi juga penerapan dalam bentuk program.