LAPORAN PRAKTIKUM 8

STRUKTUR DATA

Oleh:
Ibrahim Mousa Dhani
2411532010

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

Mata Kuliah : Struktur Data

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG

A. Pendahuluan

Melanjutkan dari praktikum yang sebelumnya, pada praktikum ini kita akan membahas dan mempelajari 4 contoh algoritma pengurutan yaitu Bubble, Shell, Quick, Merge Sort menggunakan GUI (Graphical User Interface).

Bubble Sort: Mengurutkan data dengan membandingkan dan menukar elemen bersebelahan secara berulang hingga seluruh data terurut. Sederhana, tapi lambat untuk data besar. Algoritma ini disebut "bubble" karena nilai terbesar akan "menggelembung" ke akhir array setelah setiap iterasi. Bubble Sort memiliki performa yang buruk pada data berukuran besar karena kompleksitas waktunya adalah $O(n^2)$.

Shell Sort: Versi lebih cepat dari insertion sort yang mengurutkan data dengan jarak tertentu (gap), lalu mengecilkan jaraknya sampai urutan akhir tercapai. Kompleksitas waktu Shell Sort tergantung pada pilihan gap sequence, namun umumnya lebih efisien daripada bubble dan insertion sort.

Quick Sort: Mengurutkan dengan memilih satu elemen sebagai pivot, lalu membagi data menjadi dua bagian (lebih kecil dan lebih besar dari pivot), dan mengurutkannya secara rekursif. Quick sort sangat efisien dengan rata-rata kompleksitas waktu O(n log n), tetapi performanya bisa menurun menjadi $O(n^2)$ jika pemilihan pivot buruk, seperti selalu memilih elemen pertama pada array yang sudah terurut.

Merge Sort: Memecah data menjadi bagian-bagian kecil, mengurutkannya, lalu menggabungkannya kembali secara berurutan. Efisien dan stabil untuk data besar. erge Sort memiliki kompleksitas waktu O(n log n) di semua kasus, sehingga sangat stabil dan efisien untuk data besar, meskipun membutuhkan ruang tambahan untuk proses merge.

B. Tujuan Praktikum

1. Mengetahui dan memahami konsep dasar Bubble, Shell, Quick dan Merge Sort
2. Mampu memahami proses algoritma Bubble, Shell, Quick dan Merge Sort
3. Mengimplementasikan dasar Bubble, Shell, Quick dan Merge Sort
   dalam bahasa pemrograman Java, menggunakan GUI (Graphical User Interface

C. Langkah Langkah

a. Shell Sort

1. Sama seperti Insertion dan selection sort pada praktikum sebelumnya. Pertama buat package baru dan beri nama pekan8, karna kita akan menggunakan GUI kita buat JFrame terlebih dahulu , caranya klik kanan pada bagian package klik new, lalu others, pilih WindowBuilder, pilih swing, dan pilih JFrame lalu tuliskan nama kelas nya yaitu ShellSortGUI.
2. Import Library yang dibutuhkan

```
package pekan8;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;

import java.awt.FlowLayout;
import java.awt.Font;
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 javax.swing.SwingUtilities;
```

3. Deklarasikan semua komponen GUI dan variabel agar bisa diakses oleh semua metode di dalam kelas tersebut

```
public class ShellSortGUI extends JFrame {
    private boolean isSwapping = false;
    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 gap;
    private int temp;
```

4. Buat Constructor dan Desain GUI, pertama atur pproperti Frame Utama: beri judul, ukuran, dan operasi saat ditutup.

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

5. Buat panel-panel yaitu Inisialisasi JPanel untuk setiap bagian: input, visualisasi array, kontrol, dan log. Buat komponen yaitu inisialisasi semua komponen seperti JTextField, JButton, dan JTextArea

```
// 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);
```

6. Masukkan komponen ke dalam panel yang sesuai. Kemudian, masukkan panel-panel tersebut ke dalam JFrame utama menggunakan BorderLayout.

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

7. Tambahkan addActionListener pada setiap tombol yang sudah dibuat tadi yang nanti akan terhubung dengan metode yang akan dibuat setelah ini, bisa disable terlebih dahulu agar tidak terjadi error sampai semua metode nya selesai dibuat

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

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

// Event Reset
resetButton.addActionListener(e -> reset());
```

8. Lalu buat metode setArrayFromInput yang berfungsi untuk membaca input angka dari pengguna yang dimasukkan ke dalam JTextField, kemudian memproses input tersebut menjadi array integer, dan menampilkannya secara visual di antarmuka pengguna. Input dipisahkan menggunakan koma, lalu dikonversi menjadi elemen array; jika ditemukan karakter non-angka, akan muncul pesan kesalahan melalui dialog JOptionPane. Setelah array berhasil dibuat, program mengatur nilai awal untuk proses Shell Sort seperti gap, i, dan stepCount, serta mengaktifkan tombol “Langkah Selanjutnya” agar pengguna dapat memulai proses sortir. Panel visual panelArray dikosongkan lalu diisi ulang dengan label-label JLabel yang mewakili tiap angka dalam array, lengkap dengan styling seperti ukuran, warna, dan border agar tampil menarik dan mudah dibaca. Akhirnya, panel diperbarui menggunakan revalidate() dan repaint() untuk memastikan semua perubahan tampil secara langsung di layar.

```
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 dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    gap = array.length / 2;
    i = gap;
    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();
```

9. Lalu buat metode performStep yang berfungsi untuk menjalankan satu langkah dari algoritma Shell Sort setiap kali tombol "Langkah Selanjutnya" diklik. Langkah kerjanya yaitu jika gap == 0, berarti sorting selesai.Jika belum selesai: Ambil nilai array[i] dan simpan ke temp. Bandingkan dengan elemen di posisi j - gap. Jika lebih besar, geser elemen ke kanan. Jika tidak, tempatkan temp di posisi yang sesuai. Ulangi proses sampai semua elemen disortir. Setelah i melewati akhir array, kurangi nilai gap dan ulangi.

```
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;
            }

            if (j >= gap && array[j - gap] > temp) {
                array[j] = array[j - gap]; // geser ke kanan
                labelArray[j].setBackground(Color.GREEN);
                labelArray[j - gap].setBackground(Color.CYAN);
                updateLabels();
                logStep("Geser elemen " + array[j] + " ke kanan");
                j -= gap;
                return;
            } else {
                array[j] = temp; // letakkan nilai temp
                updateLabels();
                logStep("Tempatkan " + temp + " di posisi " + j);
                i++;
                isSwapping = false;
            }
        } else {
            gap /= 2;
            i = gap;
            isSwapping = false;
            stepArea.append("Langkah " + stepCount++ + ": Kurangi gap menjadi " + gap + "\n\n");
```

10. Buat metode updateLabels yang berfungsi untuk memabantu menyinkronkan tampilan, setelah terjadi pertukaran nilai di dalam array, metode ini akan memperbarui teks pada setiap JLabel agar sesuai.

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

11. Lalu buat metode logStep yang berfungsi untuk mencatat dan menampilkan setiap langkah proses sorting ke dalam area teks (stepArea) yang berada di sebelah kanan tampilan GUI. Metode ini menerima satu parameter berupa String yang berisi deskripsi aktivitas yang sedang dilakukan

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

12. Buat metode resetHighlight berfungsi untuk mengembalikan warna latar belakang (background) semua elemen array yang divisualisasikan ke warna putih, setelah sebelumnya mungkin diberi warna khusus saat proses sorting berlangsung

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

13. Selanjutnya buat metode reset yang berfungsi untuk mengembalikan/mereset angka dan semua operasi yang telah dilakukan seperti semula

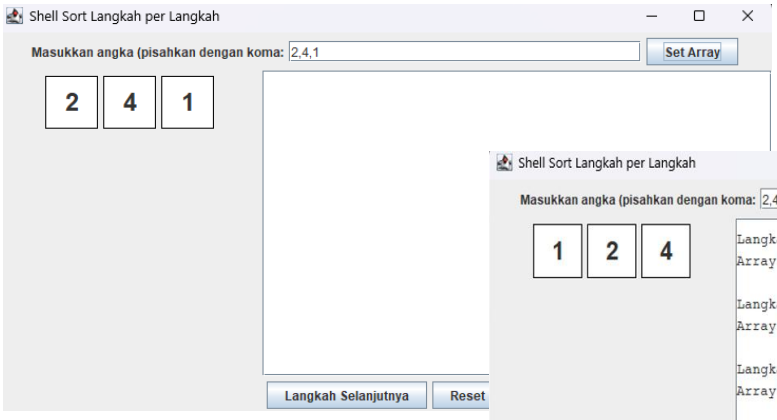
```
private void reset() {
    inputField.setText("");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText("");
    stepButton.setEnabled(false);
    sorting = false;
    i = 1;
    stepCount = 1;
```

14. Buat metode main, yang berfungsi untuk membuat dan menampilkan jendela aplikasi, metode main ini otomatis muncul saat membuat class dengan JFrame menggunakan WindowBuilder saat awal tadi

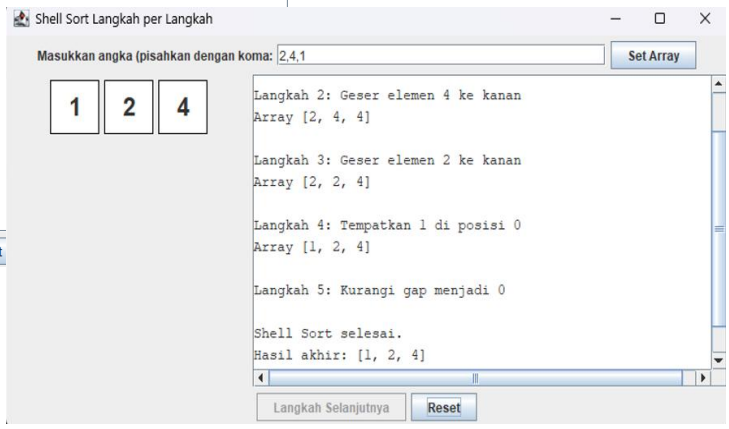
```
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        ShellSortGUI gui = new ShellSortGUI();
        gui.setVisible(true);
    });
}
```

15. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut :

Disini saya menginputkan angka 2, 4, 1, lalu klik Set Array dan akan muncul gambar di sebelah kiri layer



lalu klik tombol Langkah Selanjutnya dan akan muncul



penjelasan pada log, , lakukan proses tersebut berulang kali hingga muncul pop up Sorting telah sel

b. Bubble Sort

1. Sama seperti yang sebelumnya, buat pada package pekan7 class baru, karna kita akan menggunakan GUI kita buat JFrame terlebih dahulu , caranya klik kanan pada bagian package klik new, lalu others, pilih WindowBuilder, pilih swing, dan pilih JFrame lalu tuliskan nama kelas nya yaitu BubbleSortGUI.
2. Selanjutnya import library yang dibutuhkan

```
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 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;
```

3. Deklarasikan semua komponen GUI dan variabel agar bisa diakses oleh semua metode di dalam kelas tersebut

```
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;
```

4. Selanjutnya kita akan mendesain antarmuka user (GUI). Dengan cara buat Constructor dan Desain GUI, pertama atur properti Frame Utama: beri judul, ukuran, dan operasi saat ditutup

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

5. Buat panel-panel yaitu Inisialisasi JPanel untuk setiap bagian: input, visualisasi array, kontrol, dan log. Buat komponen yaitu inisialisasi semua komponen seperti JTextField, JButton, dan JTextArea

```
//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");
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);
```

6. Masukkan komponen ke dalam panel yang sesuai. Kemudian, masukkan panel-panel tersebut ke dalam JFrame utama menggunakan BorderLayout.

```
//Tambahkan panel ke frame
add(inputPanel, BorderLayout.NORTH);
add(panelArray, BorderLayout.CENTER);
add(controlPanel, BorderLayout.SOUTH);
add(scrollPane, BorderLayout.EAST);
```

7. Tambahkan addActionListener pada setiap tombol yang sudah dibuat tadi yang nanti akan terhubung dengan metode yang akan dibuat setelah ini, bisa disable terlebih dahulu agar tidak terjadi error sampai semua metode

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

//Event langkah selanjutnya
stepButton.addActionListener(e -> performStep());

//event reset
resetButton.addActionListener(e -> reset());
```

8. Buat metode setArrayFromInput yang berfungsi untuk membaca teks dari inputField, memecah angka berdasarkan koma, dan menyimpannya dalam array int[]. Buat label untuk setiap elemen array dan tampilkan secara horizontal. Reset kondisi seperti i, j, dan stepCount, serta aktifkan tombol “Langkah Selanjutnya”.

```
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 dengan 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();
```

9. Lalu buat fungsi performStep yang berfungsi menjalankan satu langkah dari algoritma Bubble Sort. Bandingkan elemen j dan j+1, tukar jika perlu, lalu tampilkan log langkah di stepArea. Gunakan warna CYAN untuk menandai elemen yang sedang dibandingkan dan RED jika terjadi penukaran. Setelah mencapai akhir satu iterasi, naikkan i dan reset j ke 0. Ketika i >= array.length - 1, sorting selesai dan tombol dinonaktifkan.

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

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

        if (array[j] > array[j + 1]) {
            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!");
```

10. Buat metode updateLabels yang berfungsi untuk memabantu menyinkronkan tampilan, setelah terjadi pertukaran nilai di dalam array, metode ini akan memperbarui teks pada setiap JLabel agar sesuai.

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

11. Buat metode resetHighlight berfungsi untuk mengembalikan warna latar belakang (background) semua elemen array yang divisualisasikan ke warna putih, setelah sebelumnya mungkin diberi warna khusus saat proses sorting berlangsung

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

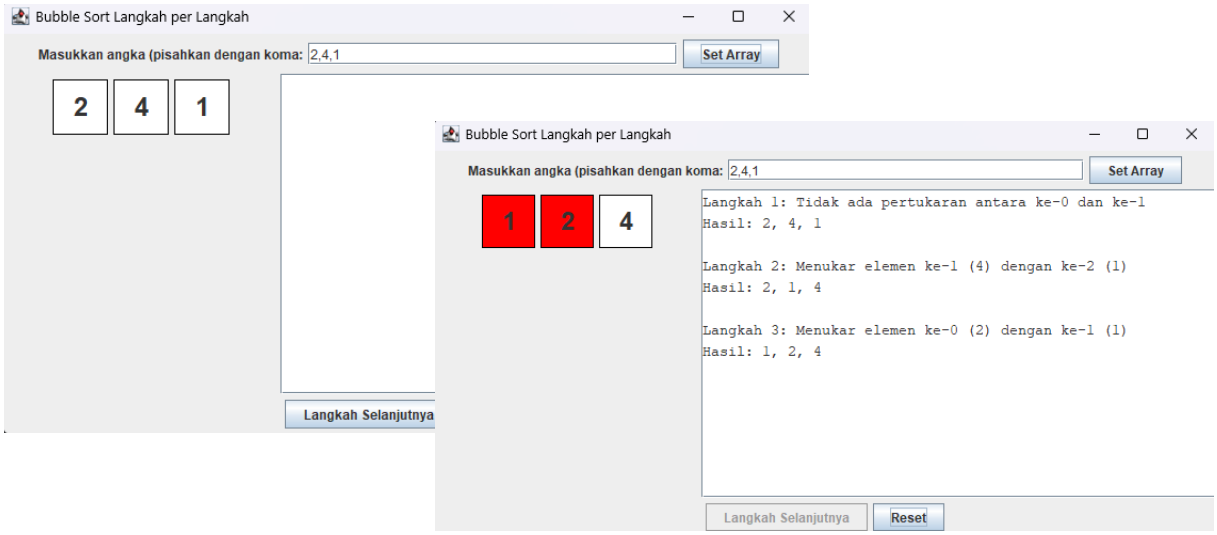
12. Selanjutnya buat metode reset yang berfungsi untuk mengembalikan/mereset angka dan semua operasi yang telah dilakukan seperti semula

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

13. Terakhir metode arrayToString ini merupakan metode pembantu yang mengubah array integer menjadi String agar mudah dibaca dan ditampilkan di log

```
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();
}
```

14. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut :
    Disini saya menginputkan angka 2, 4, 1, lalu klik Set Array dan akan muncul gambar di sebelah kiri layer. Lalu klik tombol Langkah Selanjutnya dan akan muncul penjelasan pada log, lakukan proses tersebut berulang kali hingga muncul pop up Sorting telah selesai



c. QuickSort

1. Sama seperti yang sebelumnya, buat pada package pekan7 class baru, karna kita akan menggunakan GUI kita buat JFrame terlebih dahulu , caranya klik kanan pada bagian package klik new, lalu others, pilih WindowBuilder, pilih swing, dan pilih JFrame lalu tuliskan nama kelas nya yaitu SeelectionSortGUI.
2. Selanjutnya import library yang dibutuhkan

```
import java.util.Stack;
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 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;
```

3. Deklarasikan semua komponen GUI dan variabel agar bisa diakses oleh semua metode di dalam kelas tersebut

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

4. Selanjutnya kita akan mendesain antarmuka user (GUI). Dengan cara buat Constructor dan Desain GUI, pertama atur properti Frame Utama: beri judul, ukuran, dan operasi saat ditutup

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

5. Buat panel-panel yaitu Inisialisasi JPanel untuk setiap bagian: input, visualisasi array, kontrol, dan log. Buat komponen yaitu inisialisasi semua komponen seperti JTextField, JButton, dan JTextArea.

```
//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");
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);
```

6. Masukkan komponen ke dalam panel yang sesuai. Kemudian, masukkan panel-panel tersebut ke dalam JFrame utama menggunakan BorderLayout.

```
//Tambahkan panel ke frame
add(inputPanel, BorderLayout.NORTH);
add(panelArray, BorderLayout.CENTER);
add(controlPanel, BorderLayout.SOUTH);
add(scrollPane, BorderLayout.EAST);
```

7. Tambahkan addActionListener pada setiap tombol yang sudah dibuat tadi yang nanti akan terhubung dengan metode yang akan dibuat setelah ini, bisa disable terlebih dahulu agar tidak terjadi error sampai semua metode nya selesai dibuat

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

//Event langkah selanjutnya
stepButton.addActionListener(e -> performStep());

//event reset
resetButton.addActionListener(e -> reset());
```

8. Buat metode setArrayFromInput yang berfungsi untuk membaca teks dari inputField, memecah angka berdasarkan koma, dan menyimpannya dalam array int[]. Buat label untuk setiap elemen array dan tampilkan secara horizontal. Push indeks awal ke stack ([0, array.length - 1]) untuk memulai Quick Sort. Reset kondisi stepCount, serta aktifkan tombol “Langkah Selanjutnya”.

```
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" +
            " dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    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]);
    }
    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();
}
```

9. Lalu buat metode performStep yang berfungsi untuk menjalankan proses Quick Sort satu langkah demi satu langkah. Pertama, metode ini memeriksa apakah proses penyortiran telah selesai. Jika belum, dan tidak sedang dalam proses pembagian (partitioning), maka akan mengambil sepasang indeks (low dan high) dari stack, menentukan elemen pivot, dan memulai proses partition. Selama partitioning, elemen dibandingkan satu per satu dengan pivot; jika elemen lebih kecil atau sama dengan pivot, maka dilakukan penukaran posisi. Setelah semua elemen dibandingkan, pivot dipindahkan ke posisi yang tepat, lalu subarray kiri dan kanan dari pivot ditambahkan ke stack jika masih perlu diurutkan. Metode ini memungkinkan visualisasi langkah demi langkah dari algoritma Quick Sort.

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

    resetHighlights();
    if (!partitioning) {
        int[] range = stack.pop();
        low = range[0];
        high = range[1];
        pivot = array[high];
        i = low - 1;
        j = low;
        partitioning = true;
        stepArea.append("Langkah " + stepCount++ + ": Mulai partition dari index " +
            low + " ke " + high + " dengan pivot " + pivot + "\n");
        highlightPivot(high);
        return;
    }

    if (j < high) {
        highlightCompare(j, high);
        if (array[j] <= pivot) {
            i++;
            if (i != j) {
                swap(i, j);
                stepArea.append("Langkah " + stepCount++ + ": Tukar " + array[j] + " dan " + array[i] + "\n");
            } else {
                stepArea.append("Langkah " + stepCount++ + ": " + array[j] + " sudah di posisi yang benar\n");
            }
            updateLabels();
        } else {
            stepArea.append("Langkah " + stepCount++ + ": Lewatkan " + array[j] + " (lebih besar dari pivot)\n");
        }
        j++;
        return;
    }

    // Final swap
    if (i + 1 != high) {
        swap(i + 1, high);
        stepArea.append("Langkah " + stepCount++ + ": Pindahkan pivot ke posisi tengah\n");
    }
    updateLabels();
    int p = i + 1;
    partitioning = false;
    // Push subproblems
    if (p - 1 > low) {stack.push(new int[] { low,p - 1 });
    if (p + 1 < high) {stack.push(new int[] { p + 1, high });
```

10. Buat metode highlightPivot yang berfungsi untuk menandai elemen pivot pada tampilan array dengan memberi warna kuning pada label elemen di indeks yang diberikan. Ini memudahkan pengguna mengenali elemen yang saat itu dijadikan pivot dalam proses Quick Sort.

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

11. Lalu buat metode highlightCompare yang berfungsi untuk menandai dua elemen yang sedang dibandingkan. Label pada indeks jIndex diwarnai **biru muda (cyan)** sebagai elemen yang sedang diperiksa, dan indeks pivotIndex tetap diwarnai **kuning** sebagai penanda pivot.

```
private void highlightCompare(int jIndex, int pivotIndex) {
    labelArray[jIndex].setBackground(Color.CYAN);
    labelArray[pivotIndex].setBackground(Color.YELLOW);
```

12. Buat metode resetHighlights berfungsi untuk mengembalikan warna latar belakang (background) semua elemen array yang divisualisasikan ke warna putih, setelah sebelumnya mungkin diberi warna khusus saat proses sorting berlangsung

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

13. Buat metode swap yang digunakan untuk menukar dua elemen dalam array pada indeks a dan b. Ini adalah langkah penting dalam algoritma Quick Sort ketika elemen yang lebih kecil dari pivot dipindahkan ke posisi yang benar.

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

14. Buat metode updateLabels yang berfungsi untuk memabantu menyinkronkan tampilan, setelah terjadi pertukaran nilai di dalam array, metode ini akan memperbarui teks pada setiap JLabel agar sesuai.

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

15. Selanjutnya buat metode reset yang berfungsi untuk mengembalikan/mereset angka dan semua operasi yang telah dilakukan seperti semula

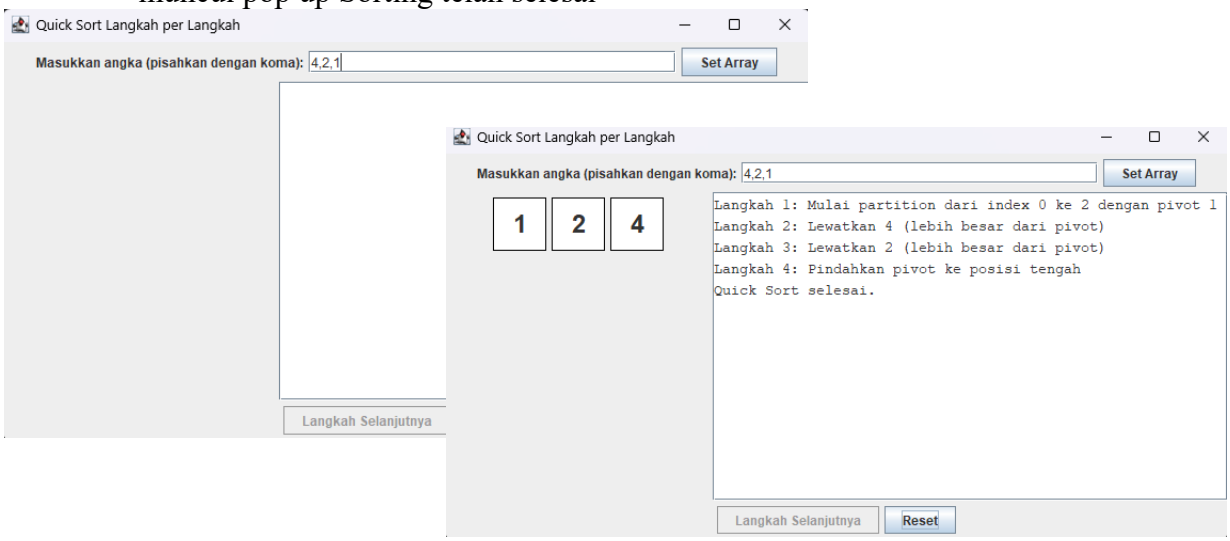
```
private void reset() {
    inputField.setText("");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText("");
    stepButton.setEnabled(false);
    stack.clear();
    sorting = false;
    partitioning = false;
    stepCount = 1;
```

16. Terakhir buat metode main yang berfungsi untuk membuat dan menampilkan jendela aplikasi, metode main ini otomatis muncul saat membuat class dengan JFrame menggunakan WindowBuilder saat awal tadi

```
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();
```

17. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut :

    Disini saya menginputkan angka 4, 2, 1 lalu klik Set Array dan akan muncul gambar di sebelah kiri layer. . Lalu klik tombol Langkah Selanjutnya dan akan muncul penjelasan pada log, lakukan proses tersebut berulang kali hingga muncul pop up Sorting telah selesai



d. MergeSort

1. Sama seperti yang sebelumnya, buat pada package pekan7 class baru, karna kita akan menggunakan GUI kita buat JFrame terlebih dahulu , caranya klik kanan pada bagian package klik new, lalu others, pilih WindowBuilder, pilih swing, dan pilih JFrame lalu tuliskan nama kelas nya yaitu MergeSortGUI.
2. Selanjutnya import library yang dibutuhkan

```
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.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;
```

3. Deklarasikan semua komponen GUI dan variabel agar bisa diakses oleh semua metode di dalam kelas tersebut

```
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 boolean isMerging = false;
    private Queue<int[]> mergeQueue = new LinkedList<>();
    private int left, mid, right, i, j, k;
    private int[] temp;
    private boolean copying = false;
```

4. Selanjutnya kita akan mendesain antarmuka user (GUI). Dengan cara buat Constructor dan Desain GUI, pertama atur properti Frame Utama: beri judul, ukuran, dan operasi saat ditutup

```
public MergeSortGUI() {
    setTitle("Merge Sort Langkah per Langkah");
    setSize(800, 450);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(5, 5));
```

5. Buat panel-panel yaitu Inisialisasi JPanel untuk setiap bagian: input, visualisasi array, kontrol, dan log. Buat komponen yaitu inisialisasi semua komponen seperti JTextField, JButton, dan JTextArea.

```
//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");
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);
```

6. Masukkan komponen ke dalam panel yang sesuai. Kemudian, masukkan panel-panel tersebut ke dalam JFrame utama menggunakan BorderLayout.

```
//Tambahkan panel ke frame
add(inputPanel, BorderLayout.NORTH);
add(panelArray, BorderLayout.CENTER);
add(controlPanel, BorderLayout.SOUTH);
add(scrollPane, BorderLayout.EAST);
```

7. Tambahkan addActionListener pada setiap tombol yang sudah dibuat tadi yang nanti akan terhubung dengan metode yang akan dibuat setelah ini, bisa disable terlebih dahulu agar tidak terjadi error sampai semua metode

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

//Event langkah selanjutnya
stepButton.addActionListener(e -> performStep());

//event reset
resetButton.addActionListener(e -> reset());
```

8. Lalu buat metode setArrayFromInput yang berfungsi untuk mengambil data input dari pengguna berupa angka-angka yang dipisahkan koma, kemudian mengonversinya menjadi array integer. Setelah data berhasil diparsing, setiap elemen array ditampilkan ke dalam panel sebagai JLabel dengan desain visual (ukuran, warna, dan border). Metode ini juga mempersiapkan proses merge dengan memanggil generateMergesteps, yang membuat antrian langkah-langkah merge sort. Selain itu, tombol “Langkah Selanjutnya” diaktifkan, area teks dihapus, dan beberapa variabel status diatur ulang agar proses visualisasi siap dijalankan dari awal.

```java
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;
    String[] parts = text.split(",");
    array = new int[parts.length];
    try {
        for (int idx = 0; idx < parts.length; idx++) {
            array[idx] = Integer.parseInt(parts[idx].trim());
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    panelArray.removeAll();
    labelArray = new JLabel[array.length];
    for (int idx = 0; idx < array.length; idx++) {
        labelArray[idx] = new JLabel(String.valueOf(array[idx]));
        labelArray[idx].setFont(new Font("Arial", Font.BOLD, 24));
        labelArray[idx].setOpaque(true);
        labelArray[idx].setBackground(Color.WHITE);
        labelArray[idx].setBorder(BorderFactory.createLineBorder(Color.BLACK));
        labelArray[idx].setPreferredSize(new Dimension(50, 50));
        labelArray[idx].setHorizontalAlignment(SwingConstants.CENTER);
        panelArray.add(labelArray[idx]);
    }
    mergeQueue.clear();
    generateMergesteps(0, array.length - 1);
    stepButton.setEnabled(true);
    stepArea.setText("");
    stepCount = 1;
    isMerging = false;
    copying = false;
    panelArray.revalidate();
    panelArray.repaint();
```

9. Buat metode performStep bertugas menjalankan satu langkah dalam algoritma merge sort setiap kali tombol ditekan. Langkah-langkahnya mencakup: mengambil rentang indeks dari antrian merge, membandingkan elemen dari dua subarray, menyalin elemen ke array sementara (temp), lalu menyalin hasilnya kembali ke array utama sambil memperbarui tampilan label. Proses ini dibagi menjadi dua fase utama yaitu **pembandingan dan penyalinan**. Selama proses berlangsung, warna label diubah untuk menandai elemen yang sedang dibandingkan atau disalin. Jika semua langkah selesai, tombol dinonaktifkan dan pesan “Selesai” ditampilkan.

```java
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;
        stepArea.append("Langkah " + stepCount++ +
                        ": Mulai merge dari " + left + " ke " + right + "\n");
        return;
    }

    if (isMerging && !copying) {
        if (i <= mid && j <= right) {
            labelArray[i].setBackground(Color.CYAN);
            labelArray[j].setBackground(Color.CYAN);
            if (array[i] <= array[j]) {
                temp[k++] = array[i++];
            } else {
                temp[k++] = array[j++];
            }
            stepArea.append("Langkah " + stepCount++ + ": Bandingkan dan salin elemen\n");
            return;
        } else if (i <= mid) {
            temp[k++] = array[i++];
            stepArea.append("Langkah " + stepCount++ + ": Salin sisa kiri\n");
            return;
        } else if (j <= right) {
            temp[k++] = array[j++];
            stepArea.append("Langkah " + stepCount++ + ": Salin sisa kanan\n");
            return;
        } else {
            copying = true;
            k = 0;
            return;
        }
    }
    if (copying && k < temp.length) {
        array[left + k] = temp[k];
        labelArray[left + k].setText(String.valueOf(temp[k]));
        labelArray[left + k].setBackground(Color.GREEN);
        k++;
        stepArea.append("Langkah " + stepCount++ + ": Tempelkan ke array utama\n");
        return;
    }

    if (copying && k == temp.length) {
        isMerging = false;
        copying = false;
    }

    if (mergeQueue.isEmpty() && !isMerging) {
        stepArea.append("Selesai.\n");
        stepButton.setEnabled(false);
        JOptionPane.showMessageDialog(this, "Merge Sort selesai!");
    }
```

10. Buat metode generateMergesteps dalam program MergeSortGUI berfungsi untuk menentukan dan menyusun urutan langkah-langkah merge sort yang akan dijalankan secara bertahap. Mempersiapkan semua langkah merge berdasarkan pembagian array sehingga algoritma merge sort dapat divisualisasikan secara bertahap.

```
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});
```

11. Buat metode resetHighlights berfungsi untuk mengembalikan warna latar belakang (background) semua elemen array yang divisualisasikan ke warna putih, setelah sebelumnya mungkin diberi warna khusus saat proses sorting berlangsung

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

12. Selanjutnya buat metode reset yang berfungsi untuk mengembalikan/mereset angka dan semua operasi yang telah dilakukan seperti semula

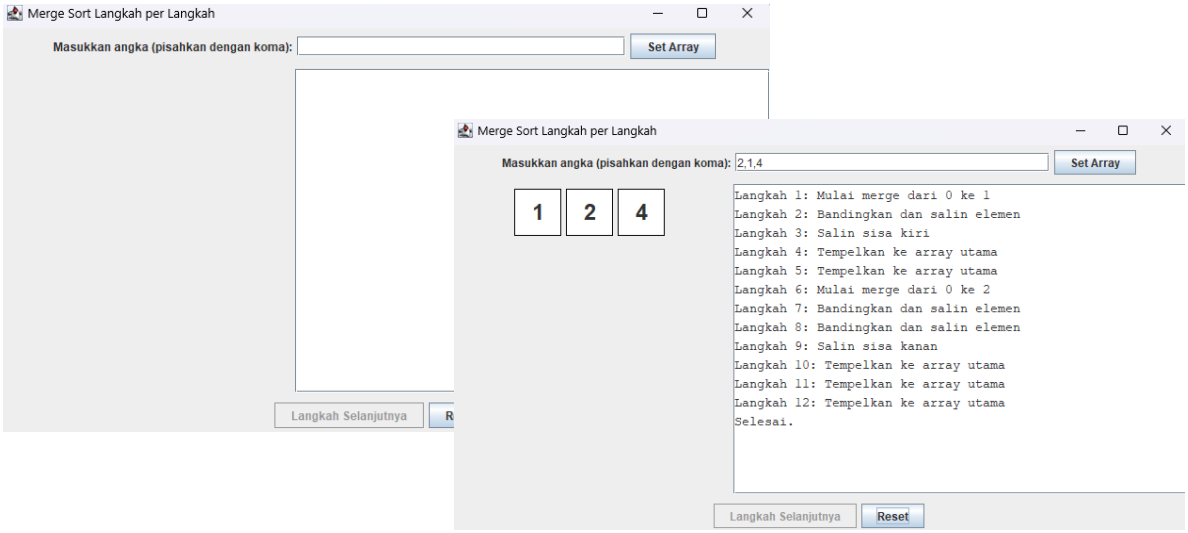
```
private void reset() {
    inputField.setText("");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText("");
    stepButton.setEnabled(false);
    stack.clear();
    sorting = false;
    partitioning = false;
    stepCount = 1;
```

13. Terakhir buat metode main yang berfungsi untuk membuat dan menampilkan jendela aplikasi, metode main ini otomatis muncul saat membuat class dengan JFrame menggunakan WindowBuilder saat awal tadi

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

14. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut :

Disini saya menginputkan angka 2, 4, 1 lalu klik Set Array dan akan muncul gambar di sebelah kiri layer. . Lalu klik tombol Langkah Selanjutnya dan akan muncul penjelasan pada log, lakukan proses tersebut berulang kali hingga muncul pop up Sorting telah selesai



D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan pada praktikum ini kita lebih dapat memahami dan mengetahui konsep dasar dan algoritma/cara kerja Bubble, Shell, Quick, dan Merge Sort menggunakan GUI (Graphical User Interface) untuk membuat program yang interaktif bagi user dan menampilkan/memvisualisasikan algoritma nya secara langkah per langkah.