LAPORAN PRAKTIKUM STRUKTUR DATA

PEKAN 8: IMPLEMENTASI DAN VISUALISASI ALGORITMA BUBBLE SORT, QUICK SORT, SHELL SORT, DAN MERGE SORT MENGGUNAKAN GUI PADA JAVA

OLEH

Abdullah Al Ramadhani (2411533016)

DOSEN PENGAMPU

DR. Wahyudi, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Sorting adalah metode untuk menyusun elemen-elemen dalam suatu data agar tertata berdasarkan aturan tertentu, seperti dari nilai terkecil ke yang terbesar. Dalam konteks kali ini, digunakan tiga jenis algoritma sorting yang ditampilkan melalui tampilan antarmuka grafis (GUI), yaitu Bubble Sort, Quick Sort, dan Shell Sort. Setiap algoritma memiliki cara kerja masing-masing serta kelebihan dan keterbatasannya.

Yang pertama adalah Bubble Sort, yaitu algoritma paling dasar yang bekerja dengan cara membandingkan pasangan elemen yang bersebelahan, lalu menukarnya jika elemen pertama lebih besar dari yang kedua. Proses ini diulang terus hingga seluruh data terurut. Walau sangat sederhana, algoritma ini memiliki performa yang rendah karena kompleksitas waktunya mencapai $O(n^2)$, sehingga kurang cocok untuk dataset besar. Dalam tampilan GUI, proses pertukaran dan perbandingan divisualisasikan lewat perubahan warna pada elemen yang sedang aktif.

Kemudian Quick Sort, yang merupakan algoritma pengurutan cepat dengan teknik divide and conquer. Prosesnya dimulai dengan memilih sebuah elemen sebagai pivot, lalu membagi array menjadi dua bagian: elemen yang lebih kecil di sisi kiri dan yang lebih besar di sisi kanan. Proses ini dilakukan secara rekursif hingga seluruh array tersusun rapi. Dengan kompleksitas rata-rata O(n log n), Quick Sort dikenal sangat efisien dan sering dipakai dalam berbagai aplikasi nyata. Pada GUI, elemen pivot dan proses partisi divisualisasikan melalui pewarnaan yang membedakan elemen satu dengan lainnya.

Shell Sort adalah versi pengembangan dari Insertion Sort yang menggunakan konsep jarak atau gap antar elemen saat membandingkan. Gap ini secara bertahap dikurangi hingga bernilai satu, lalu algoritma bekerja mirip seperti Insertion Sort, namun dalam kondisi data yang sudah sebagian terurut. Shell Sort lebih cepat dibandingkan Bubble dan Insertion Sort, meski masih kalah dari Quick Sort untuk data dalam skala besar. Visualisasi GUI akan menampilkan proses pengurutan berdasarkan gap, sehingga memudahkan pengguna melihat tahap demi tahap proses penyusunan data.

Terakhir, ada Merge Sort, yang juga menggunakan pendekatan divide and conquer. Cara kerjanya adalah dengan memecah array menjadi bagian-bagian kecil hingga setiap bagian hanya terdiri dari satu elemen. Setelah itu, bagian-bagian kecil ini disatukan kembali dalam kondisi terurut hingga terbentuk array lengkap yang sudah tersusun. Proses ini dilakukan secara bertahap dan rekursif, menjadikan Merge Sort sebagai metode yang stabil dan efisien dalam mengurutkan data.

B. Tujuan

Tujuan dari praktikum ini adalah:

1) Mendalami cara kerja dan logika dari berbagai algoritma pengurutan seperti Bubble Sort, Quick Sort, Shell Sort, serta Merge Sort, baik dari sisi konsep teoritis maupun penerapannya dalam skenario nyata.

2) Menerapkan algoritma-algoritma sorting ke dalam bentuk program visual berbasis antarmuka grafis menggunakan bahasa Java dan pustaka Swing, agar proses pengurutan dapat ditampilkan secara bertahap dan mudah dipahami.

3) Meningkatkan pemahaman terhadap konsep pengurutan data dengan memanfaatkan visualisasi interaktif, yang menampilkan secara jelas proses-proses penting seperti perbandingan, penukaran, dan penggabungan elemen dalam array.

C. Langkah-langkah Pengerjaan

a. BubbleSortGUI

1) Mulailah dengan mendefinisikan package dan mengimpor semua pustaka Java yang diperlukan, terutama yang berkaitan dengan pembuatan antarmuka grafis seperti JFrame, JButton, JLabel, dan komponen GUI lainnya dari library javax.swing serta java.awt.

```
1 package Pekan8;
2
3 import java.awt.BorderLayout;
```

2) Buat kelas utama yang akan bertindak sebagai jendela utama dari aplikasi GUI, dan pastikan kelas ini merupakan turunan dari JFrame agar bisa menampilkan komponen-komponen visual.

```
21 public class BubbleSortGUI extends JFrame {
```

3) Deklarasikan berbagai variabel yang dibutuhkan, seperti array data yang akan disortir, label untuk menampilkan elemen array, input teks dari pengguna, serta komponen lain yang akan digunakan dalam antarmuka.

```
23     private static final long serialVersionUID = 1L;
24     private int[] array;
25     private JLabel[] labelArray;
26     private JButton stepButton, resetButton, setButton;
27     private JTextField inputField;
28     private JPanel panelArray;
29     private JTextArea stepArea;
30     private int i = 1, j;
31     private boolean sorting = false;
32     private int stepCount = 1;
```

4) Tulis method main sebagai titik awal program, di mana objek dari kelas utama (misalnya BubbleSortGUI) akan dibuat dan ditampilkan ke layar.

```
34    public static void main(String[] args) {
35        EventQueue.invokeLater(new Runnable() {
36            public void run() {
37                try {
38                    BubbleSortGUI frame = new BubbleSortGUI();
39                    frame.setVisible(true);
40                } catch (Exception e) {
41                    e.printStackTrace();
42                }
43            }
44        });
45    }
```

5) Buat konstruktor BubbleSortGUI() yang bertugas menyusun antarmuka aplikasi. Di dalamnya, atur properti jendela seperti judul, ukuran, dan operasi ketika ditutup. Tambahkan panel input untuk menerima data dari pengguna, panel visualisasi array, kontrol tombol, serta area untuk menampilkan log langkah. Semua komponen ini kemudian ditambahkan ke frame utama.

```
51    public BubbleSortGUI() {
52        setTitle("Bubble Sort Langkah per Langkah");
53        setSize(750, 400);
54        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
55        setLocationRelativeTo(null);
56        setLayout(new BorderLayout());
57
58        //panel input
59        JPanel inputPanel = new JPanel(new FlowLayout());
60        inputField = new JTextField(30);
61        setButton = new JButton("Set Array");
62        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma):"));
63        inputPanel.add(inputField);
64        inputPanel.add(setButton);
65
66        //panel array visual
67        panelArray = new JPanel();
68        panelArray.setLayout(new FlowLayout());
69
70        //panel kontrol
71        JPanel controlPanel = new JPanel();
72        stepButton = new JButton("Langkah Selanjutnya");
73        resetButton = new JButton("Reset");
74        stepButton.setEnabled(false);
75        controlPanel.add(stepButton);
76        controlPanel.add(resetButton);
```

```
78          //Area text untuk log langkah-langkah
79          stepArea = new JTextArea(8, 60);
80          stepArea.setEditable(false);
81          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
82          JScrollPane scrollPane = new JScrollPane(stepArea);
83
84          //tambahkan panel ke frame
85          add(inputPanel, BorderLayout.NORTH);
86          add(panelArray, BorderLayout.CENTER);
87          add(controlPanel, BorderLayout.SOUTH);
88          add(scrollPane, BorderLayout.EAST);
89
90          //event set array
91          setButton.addActionListener(e -> setArrayFromInput());
92
93          //event langkah selanjutnya
94          stepButton.addActionListener(e -> performStep());
95
96          //event reset
97          resetButton.addActionListener(e -> reset());
```

6) Bangun method setArrayFromInput() yang berfungsi membaca input dari JTextField, memproses string menjadi array integer, dan membuat label visual untuk tiap elemen. Label tersebut lalu ditempatkan dalam panel khusus untuk menampilkan array.

```
101•    private void setArrayFromInput() {
102         String text = inputField.getText().trim();
103         if (text.isEmpty()) return;
104         String[] parts = text.split(",");
105         array = new int[parts.length];
106         try {
107             for (int k = 0; k < parts.length; k++) {
108                 array[k] = Integer.parseInt(parts[k].trim());  }
109         } catch (NumberFormatException e) {
110             JOptionPane.showMessageDialog(this, "Masukkan hanya angka " + "yang dipisahkan koma!", "Error", JOptionPane.ERROR_MESSAGE);
111             return; }
112         i = 0;
113         j = 0;
114         stepCount = 1;
115         sorting = true;
116         stepButton.setEnabled(true);
117         stepArea.setText("");
118         panelArray.removeAll();
119         labelArray = new JLabel[array.length];
120         for (int k = 0; k < array.length; k++) {
121             labelArray[k] = new JLabel(String.valueOf(array[k]));
122             labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
123             labelArray[k].setOpaque(true);
124             labelArray[k].setBackground(Color.WHITE);
125             labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
126             labelArray[k].setPreferredSize(new Dimension(50, 50));
127             labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
128             panelArray.add(labelArray[k]);
129         }
130         panelArray.revalidate();
131         panelArray.repaint();
132     }
```

7) Buat method performStep() yang akan menandai dua elemen yang sedang dibandingkan dalam proses sorting. Jika perlu, elemen-elemen tersebut akan ditukar, dan langkahnya dicatat di area log. Setelahnya, tampilan label diperbarui sesuai isi array terbaru. Jika seluruh proses selesai, tombol sorting akan dinonaktifkan.

```
134•    private void performStep() {
135         if (!sorting || i >= array.length - 1) {
136             sorting = false;
137             stepButton.setEnabled(false);
138             JOptionPane.showMessageDialog(this, "Sorting selesai!");
139             return;
140         }
141         resetHighlights();
142         StringBuilder stepLog = new StringBuilder();
143         labelArray[j].setBackground(Color.CYAN);
144         labelArray[j + 1].setBackground(Color.CYAN);
145         if (array[j] > array[j + 1]) {
146             // Swap
147             int temp = array[j];
148             array[j] = array[j + 1];
149             array[j + 1] = temp;
150             labelArray[j].setBackground(Color.RED);
151             labelArray[j + 1].setBackground(Color.RED);
152             stepLog.append("Langkah ").append(stepCount).append(": Menukar elemen ke-")
153                    .append(j).append(" (").append(array[j + 1]).append(") dengan ke-")
154                    .append(j + 1).append(" (").append(array[j]).append(")\n");
155         } else {
156             stepLog.append("Langkah ").append(stepCount).append(": Tidak ada pertukaran antara ke-")
157                    .append(j).append(" dan ke-").append(j + 1).append("\n");
158         }
159 
160         stepLog.append("Hasil: ").append(arrayToString(array)).append("\n\n");
161         stepArea.append(stepLog.toString());
162         updateLabels();
163         j++;
164         if (j >= array.length - i - 1) {
165             j = 0;
166             i++;
167         }
```

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

        }
    }
```

8) Agar tampilan label tetap mengikuti perubahan data, buat method updateLabels() untuk memperbarui nilai di layar, serta method resetHighlights() untuk mengembalikan warna latar label ke kondisi semula.

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

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

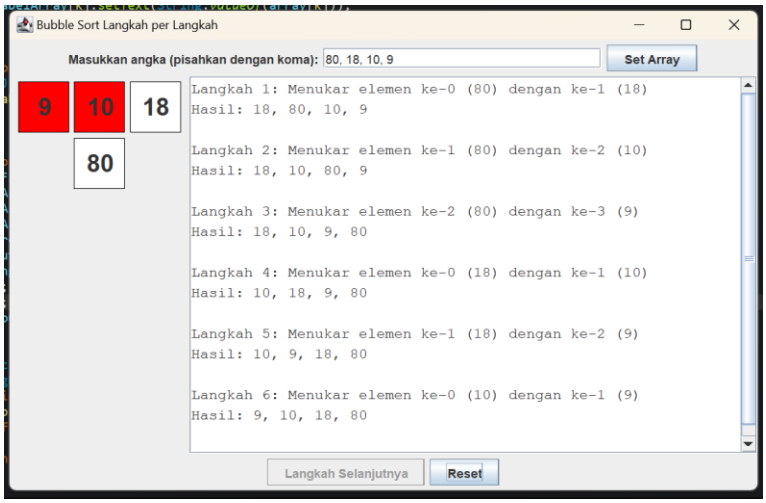
9) Tambahkan method reset() untuk membersihkan seluruh input pengguna, mengosongkan visualisasi array, serta menghapus isi log langkah, sehingga pengguna bisa memulai ulang proses pengurutan.

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

10) Buat method arrayToString() yang berfungsi mengubah isi array menjadi format string agar bisa ditampilkan, misalnya untuk log atau hasil akhir.

```
    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();
    }
}
```

11) Output dari program ini akan menampilkan antarmuka grafis yang memperlihatkan elemen array secara visual dan interaktif, termasuk proses perbandingan dan pertukaran yang terjadi selama sorting.



b. MergeSortGUI

1) Mulailah dengan membuat kelas bernama MergeSortGUI, yang akan menjadi struktur utama dari aplikasi antarmuka grafis. Di dalam kelas ini, deklarasikan semua variabel penting, termasuk untuk menyimpan data array, elemen-elemen antarmuka (seperti tombol, label, dan bidang input), serta variabel status dan indeks yang dibutuhkan untuk mengatur proses Merge Sort langkah demi langkah.

```
1  package Pekan8;
2
3  import javax.swing.*;
6
7  public class MergeSortGUI extends JFrame {
8      private static final long serialVersionUID = 1L;
9
10     private int[] array;
11     private JLabel[] labelArray;
12     private JTextField inputField;
13     private JButton setButton, stepButton, resetButton;
14     private JPanel panelArray;
15     private JTextArea stepArea;
16
17     private Queue<int[]> mergeQueue = new LinkedList<>();
18     private int[] temp;
19     private int left, mid, right, i, j, k;
20     private boolean isMerging = false, copying = false;
21     private int stepCount = 1;
```

2) Buat konstruktor MergeSortGUI() yang bertugas mengatur tampilan dan susunan komponen GUI. Langkah pertama adalah menyiapkan panel untuk memasukkan data berupa angka dan sebuah tombol untuk menetapkan array. Panel berikutnya digunakan untuk menampilkan elemen array dalam bentuk JLabel. Setelah itu, tambahkan panel kontrol yang memuat tombol untuk melanjutkan proses sorting dan tombol reset. Terakhir, tambahkan panel khusus untuk mencatat log proses dan satukan semua panel ke dalam jendela utama.

```
23    public MergeSortGUI() {
24        setTitle("Merge Sort Langkah per Langkah");
25        setSize(900, 500);
26        setDefaultCloseOperation(EXIT_ON_CLOSE);
27        setLocationRelativeTo(null);
28        setLayout(new BorderLayout());
29
30        JPanel inputPanel = new JPanel(new FlowLayout());
31        inputField = new JTextField(30);
32        setButton = new JButton("Set Array");
33        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
34        inputPanel.add(inputField);
35        inputPanel.add(setButton);
36
37        panelArray = new JPanel(new FlowLayout());
38
39        JPanel controlPanel = new JPanel();
40        stepButton = new JButton("Langkah Selanjutnya");
41        resetButton = new JButton("Reset");
42        controlPanel.add(stepButton);
43        controlPanel.add(resetButton);
```

```
45          stepArea = new JTextArea(10, 60);
46          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
47          stepArea.setEditable(false);
48          JScrollPane scrollPane = new JScrollPane(stepArea);
49
50          add(inputPanel, BorderLayout.NORTH);
51          add(panelArray, BorderLayout.CENTER);
52          add(controlPanel, BorderLayout.SOUTH);
53          add(scrollPane, BorderLayout.EAST);
54
55          setButton.addActionListener(e -> setArrayFromInput());
56          stepButton.setEnabled(false);
57          stepButton.addActionListener(e -> performStep());
58          resetButton.addActionListener(e -> reset());
59      }
```

3) Implementasikan method setArrayFromInput(), yang akan mengambil data dari kolom input, memisahkan berdasarkan tanda koma, lalu mengubahnya menjadi array bertipe integer. Di tahap ini juga dibuat label visual untuk masing-masing elemen array dan ditambahkan ke panel tampilan. Selain itu, method ini juga menyiapkan daftar antrian operasi merge yang akan dijalankan.

```
61    private void setArrayFromInput() {
62        String text = inputField.getText().trim();
63        if (text.isEmpty()) return;
64
65        String[] parts = text.split(",");
66        array = new int[parts.length];
67
68        try {
69            for (int k = 0; k < parts.length; k++) {
70                array[k] = Integer.parseInt(parts[k].trim());
71            }
72        } catch (NumberFormatException e) {
73            JOptionPane.showMessageDialog(this, "Masukkan hanya angka dipisahkan koma!", "Error", JOptionPane.ERROR_MESSAGE);
74            return;
75        }
76
77        labelArray = new JLabel[array.length];
78        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]);
        }
        mergeQueue.clear();
        generateMergeSteps(0, array.length - 1);
        stepButton.setEnabled(true);
        stepArea.setText("");
        isMerging = false;
        copying = false;
        stepCount = 1;
        panelArray.revalidate();
        panelArray.repaint();
    }
```

4) Buat method generateMergeSteps(), yang bertugas menyusun langkah-langkah merge sort ke dalam bentuk queue (antrian) sehingga setiap proses penggabungan dapat dilakukan satu per satu secara teratur.

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

5) Agar proses sorting berjalan ketika tombol diklik, buatlah method performStep(). Method ini akan menjalankan satu langkah merge berdasarkan antrian yang telah dibuat sebelumnya.

```
    private void performStep() {
        resetHighlights();

        if (!isMerging && !mergeQueue.isEmpty()) {
            int[] range = mergeQueue.poll();
            left = range[0];
            mid = range[1];
            right = range[2];
            i = left;
            j = mid + 1;
            k = 0;
            temp = new int[right - left + 1];
            isMerging = true;
            copying = false;
            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++];
                    stepArea.append("Langkah " + stepCount++ + ": Salin dari kiri\n");
                } else {
                    temp[k++] = array[j++];
                    stepArea.append("Langkah " + stepCount++ + ": Salin dari kanan\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);
            stepArea.append("Langkah " + stepCount++ + ": Tempelkan ke array utama\n");
            k++;
            return;
        }

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


        if (mergeQueue.isEmpty() && !isMerging) {
            stepArea.append("Selesai.\n");
            stepArea.append("Array akhir: " + Arrays.toString(array) + "\n");
```

```
            // Highlight semua elemen akhir
            for (JLabel label : labelArray) {
                label.setBackground(Color.ORANGE);
            }

            stepButton.setEnabled(false);
            JOptionPane.showMessageDialog(this, "Merge Sort selesai!");
        }
    }
```

6) Seperti pada aplikasi sorting lainnya, buat method resetHighlights() untuk mengembalikan warna latar belakang label array ke warna standar (putih) setelah setiap langkah dijalankan.

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

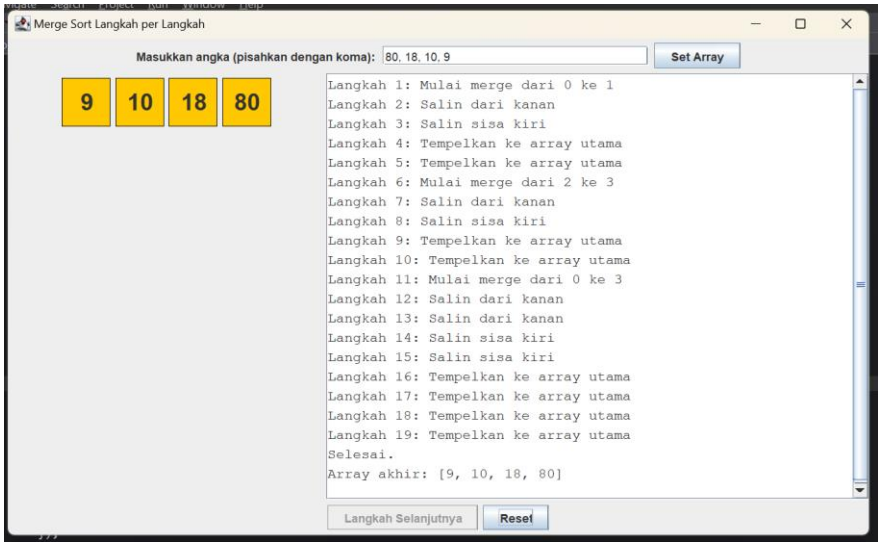
7) Tambahkan method reset(), yang berfungsi untuk menghapus isi input, membersihkan panel array, menghapus log langkah, dan mengembalikan semua status ke kondisi awal agar proses bisa dimulai kembali dari awal.

```
192     private void reset() {
193         inputField.setText("");
194         panelArray.removeAll();
195         panelArray.revalidate();
196         panelArray.repaint();
197         stepArea.setText("");
198         stepButton.setEnabled(false);
199         mergeQueue.clear();
200         isMerging = false;
201         copying = false;
202         stepCount = 1;
203     }
```

8) Buat method main() yang berfungsi menjalankan aplikasi dan menampilkan jendela GUI ke layar.

```
205     public static void main(String[] args) {
206         SwingUtilities.invokeLater(() -> {
207             new MergeSortGUI().setVisible(true);
208         });
209     }
210 }
```

9) Program ini akan menghasilkan antarmuka grafis interaktif, yang memperlihatkan jalannya proses Merge Sort secara bertahap, lengkap dengan visualisasi elemen array yang sedang digabung serta log dari setiap langkah yang dilakukan.



c. QuickSortGUI

1) Awali dengan mendefinisikan kelas khusus yang berfungsi sebagai antarmuka visual untuk menampilkan proses Quick Sort berbasis Swing. Di dalam kelas ini, buat variabel-variabel penting yang akan digunakan untuk menyimpan data array, mengatur tampilan antarmuka, melacak indeks saat ini, status proses, dan keperluan lainnya yang mendukung pelaksanaan Quick Sort secara bertahap.

```
 1 package Pekan8;
 2
 3⊕import java.awt.BorderLayout;
21
22 public class QuickSortGUI extends JFrame {
23
24     private static final long serialVersionUID = 1L;
25
26     private int[] array;
27     private JLabel[] labelArray;
28     private JButton stepButton, resetButton, setButton;
29     private JTextField inputField;
30     private JPanel panelArray;
31     private JTextArea stepArea;
32     private JScrollPane scrollPane;
33
34     private int i = 1, j;
35     private boolean sorting = false;
36     private int stepCount = 1;
37
38     // Tambahan variabel yang dibutuhkan
39     private Stack<int[]> stack = new Stack<>();
40     private int low, high, pivot;
41     private boolean partitioning = false;
42     private boolean isSwapping = false;
```

2) Selanjutnya buat method main(), yang akan menjadi titik awal saat program dijalankan, di mana jendela aplikasi dibuat dan ditampilkan ke pengguna.

```
44●    public static void main(String[] args) {
45●        EventQueue.invokeLater(new Runnable() {
46●            public void run() {
47                try {
48                    QuickSortGUI frame = new QuickSortGUI();
49                    frame.setVisible(true);
50                } catch (Exception e) {
51                    e.printStackTrace();
52                }
53            }
54        });
55    }
```

3) Bangun konstruktor QuickSortGUI() yang akan bertugas menyusun tampilan aplikasi. Di dalamnya, atur properti jendela seperti ukuran, judul, serta pengaturan tata letak. Antarmuka dibagi menjadi beberapa bagian, yakni panel untuk memasukkan angka, panel yang menampilkan elemen array secara visual, panel kontrol untuk tombol-tombol aksi, dan area khusus yang digunakan untuk mencatat log aktivitas sorting.

```
57    public QuickSortGUI() {
58        setTitle("Quick Sort langkah per langkah");
59        setSize(750, 400);
60        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
61        setLocationRelativeTo(null);
62        setLayout(new BorderLayout());
63
64        JPanel inputPanel = new JPanel(new FlowLayout());
65        inputField = new JTextField(30);
66        setButton = new JButton("Set Array");
67        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
68        inputPanel.add(inputField);
69        inputPanel.add(setButton);
70
71        panelArray = new JPanel();
72        panelArray.setLayout(new FlowLayout());
73
74        JPanel controlPanel = new JPanel();
75        stepButton = new JButton("Langkah Selanjutnya");
76        resetButton = new JButton("Reset");
77        stepButton.setEnabled(false);
78        controlPanel.add(stepButton);
79        controlPanel.add(resetButton);
80
81        stepArea = new JTextArea(8, 60);
82        stepArea.setEditable(false);
83        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
84        scrollPane = new JScrollPane(stepArea);
85
86        add(inputPanel, BorderLayout.NORTH);
87        add(panelArray, BorderLayout.CENTER);
88        add(controlPanel, BorderLayout.SOUTH);
89        add(scrollPane, BorderLayout.EAST);
```

```
91          // Event set array
92          setButton.addActionListener(e -> setArrayFromInput());
93
94          // Event langkah selanjutnya
95          stepButton.addActionListener(e -> performStep());
96
97          // Event reset
98          resetButton.addActionListener(e -> reset());
99      }
```

4) Seperti pada program lainnya, buat method setArrayFromInput(), yang membaca data angka yang dimasukkan oleh pengguna, memisahkannya berdasarkan tanda koma, lalu mengonversinya menjadi array bertipe integer. Setiap elemen array akan ditampilkan dalam bentuk JLabel menyerupai kotak kecil di dalam panel array.

```
101    private void setArrayFromInput() {
102        String text = inputField.getText().trim();
103        if (text.isEmpty()) return;
104
105        String[] parts = text.split(",");
106        array = new int[parts.length];
107
108        try {
109            for (int k = 0; k < parts.length; k++) {
110                array[k] = Integer.parseInt(parts[k].trim());
111            }
112        } catch (NumberFormatException e) {
113            JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan koma!",
114                "Error", JOptionPane.ERROR_MESSAGE);
115            return;
116        }
117
118        labelArray = new JLabel[array.length];
119        panelArray.removeAll();
120        for (int k = 0; k < array.length; k++) {
121            labelArray[k] = new JLabel(String.valueOf(array[k]));
122            labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
123            labelArray[k].setOpaque(true);
124            labelArray[k].setBackground(Color.WHITE);
125            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
126            labelArray[k].setPreferredSize(new Dimension(50, 50));
127            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
128            panelArray.add(labelArray[k]);
129        }
```

```
131          stack.clear();
132          stack.push(new int[]{0, array.length - 1});
133          sorting = true;
134          isSwapping = false;
135          partitioning = false;
136          stepCount = 1;
137          stepArea.setText("");
138          stepButton.setEnabled(true);
139
140          panelArray.revalidate();
141          panelArray.repaint();
142      }
```

5) Buat method performStep(), yang menjadi inti dari proses sorting, di mana satu langkah Quick Sort akan dijalankan setiap kali tombol ditekan. Proses ini akan menampilkan pembagian array berdasarkan pivot dan pertukaran elemen jika diperlukan.

```
144    private void performStep() {
145        if (!sorting || (!partitioning && stack.isEmpty())) {
146            sorting = false;
147            stepButton.setEnabled(false);
148            updateLabels();
149            highlightSorted(); // <-- Tambahan penting
150            stepArea.append("Quick Sort selesai.\n");
151            stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n");
152            JOptionPane.showMessageDialog(this, "Quick Sort selesai!");
153            return;
154        }
155        resetHighlights();
156        if (!partitioning) {
157            int[] range = stack.pop();
158            low = range[0];
159            high = range[1];
160            pivot = array[high];
161            i = low - 1;
162            j = low;
163            partitioning = true;
164            stepArea.append("Langkah " + stepCount++ + ": Mulai partition dari index "
165                + low + " hingga " + high + " dengan pivot " + pivot + "\n");
166            highlightPivot(high);
167            return;
168        }
```

```
170        if (j < high) {
171            highlightCompare(j, high);
172            if (array[j] < pivot) {
173                i++;
174                if (i != j) {
175                    swap(i, j);
176                    stepArea.append("Langkah " + stepCount++ + ": Tukar " + array[i] + " dan " + array[j] + "\n");
177                } else {
178                    stepArea.append("Langkah " + stepCount++ + ": " + array[j] + " sudah di posisi yang benar\n");
179                }
180                j++;
181                return;
182            } else {
183                stepArea.append("Langkah " + stepCount++ + ": Lewatkan " + array[j] + " (lebih besar dari pivot)\n");
184                j++;
185                return;
186            }
187        }
188        if (i + 1 != high) {
189            swap(i + 1, high);
190            stepArea.append("Langkah " + stepCount++ + ": Pindahkan pivot ke posisi tengah\n");
191        }
192        updateLabels();
193        int p = i + 1;
194        partitioning = false;
195
196        if (p - 1 > low) stack.push(new int[]{low, p - 1});
197        if (p + 1 < high) stack.push(new int[]{p + 1, high});
198    }
```

6) Agar proses sorting lebih mudah dipahami, tambahkan fungsi visualisasi seperti highlightPivot(int index) untuk memberi tanda warna khusus pada elemen pivot, lalu highlightCompare(int jIndex, int pivotIndex) untuk menandai elemen yang sedang dibandingkan, biasanya dengan warna berbeda (misalnya kuning untuk pivot dan biru untuk yang dibandingkan). Gunakan juga resetHighlights() untuk mengembalikan warna semua elemen ke kondisi semula (putih), serta reset() untuk mengulang proses dari awal dan membersihkan semua status serta tampilan.

```
200     private void highlightPivot(int index) {
201         labelArray[index].setBackground(Color.YELLOW);
202     }
203
204     private void highlightCompare(int jIndex, int pivotIndex) {
205         labelArray[jIndex].setBackground(Color.CYAN);
206         labelArray[pivotIndex].setBackground(Color.YELLOW);
207     }
208
209     private void resetHighlights() {
210         for (JLabel label : labelArray) {
211             label.setBackground(Color.WHITE);
212         }
213     }
214
215     private void swap(int a, int b) {
216         int temp = array[a];
217         array[a] = array[b];
218         array[b] = temp;
219     }
220
221     private void updateLabels() {
222         for (int k = 0; k < array.length; k++) {
223             labelArray[k].setText(String.valueOf(array[k]));
224         }
225     }
```

```
227     private void reset() {
228         inputField.setText("");
229         panelArray.removeAll();
230         panelArray.revalidate();
231         panelArray.repaint();
232         stepArea.setText("");
233         stepButton.setEnabled(false);
234         stack.clear();
235         sorting = false;
236         partitioning = false;
237         stepCount = 1;
238     }
239
240     private void highlightSorted() {
241         for (JLabel label : labelArray) {
242             label.setBackground(Color.GREEN); // warna hijau untuk elemen yang sudah terurut
243         }
244     }
245
246 }
```

7) Jalankan aplikasi dan perhatikan bagaimana program akan menampilkan proses pengurutan secara interaktif, termasuk animasi pembandingan, pivot, pertukaran, serta catatan langkah-langkah yang terjadi di area log, semua divisualisasikan dalam GUI yang ramah pengguna.

d. ShellSortGUI

1) Mulailah dengan membuat kelas utama dan mendeklarasikan sejumlah variabel penting yang akan digunakan untuk mengatur jalannya proses sorting, menyimpan data input dari pengguna dalam bentuk array, serta menyiapkan elemen-elemen visual seperti label dan komponen GUI lainnya untuk keperluan tampilan.

```
1  package Pekan8;
2
3  import java.awt.BorderLayout;
21
22 public class ShellSortGUI extends JFrame {
23     private static final long serialVersionUID = 1L;
24     private int[] array;
25     private JLabel[] labelArray;
26     private JButton stepButton, resetButton, setButton;
27     private JTextField inputField;
28     private JPanel panelArray;
29     private JTextArea stepArea;
30     private int i = 1, j;
31     private boolean sorting = false;
32     private int stepCount = 1;
33     private boolean isSwapping = false;
34     private int gap;
35     private int temp;
```

2) Di dalam konstruktor kelas GUI, atur properti dasar dari jendela utama, seperti judul aplikasi, ukuran jendela, posisi agar muncul di tengah layar, serta layout utama (dalam hal ini BorderLayout). Setelah itu, susun komponen GUI ke dalam panel-panel terpisah seperti panel input, panel untuk menampilkan visualisasi array, dan komponen tambahan lainnya sesuai kebutuhan.

```
37    public ShellSortGUI() {
38        setTitle("Shell Sort Langkah per Langkah");
39        setSize(750, 400);
40        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41        setLocationRelativeTo(null);
42        getContentPane().setLayout(new BorderLayout());
43
44        //panel input
45        JPanel inputPanel = new JPanel(new FlowLayout());
46        inputField = new JTextField(30);
47        setButton = new JButton("Set Array");
48        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma):"));
49        inputPanel.add(inputField);
50        inputPanel.add(setButton);
51
52        //panel array visual
53        panelArray = new JPanel();
54        panelArray.setLayout(new FlowLayout());
55
56        //panel kontrol
57        JPanel controlPanel = new JPanel();
58        stepButton = new JButton("Langkah Selanjutnya");
59        resetButton = new JButton("Reset");
60        stepButton.setEnabled(false);
61        controlPanel.add(stepButton);
62        controlPanel.add(resetButton);
63
64        //Area text untuk log langkah-langkah
65        stepArea = new JTextArea(8, 60);
66        stepArea.setEditable(false);
67        stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
68        JScrollPane scrollPane = new JScrollPane(stepArea);
69
70          //tambahkan panel ke frame
71          getContentPane().add(inputPanel, BorderLayout.NORTH);
72          getContentPane().add(panelArray, BorderLayout.CENTER);
73          getContentPane().add(controlPanel, BorderLayout.SOUTH);
74          getContentPane().add(scrollPane, BorderLayout.EAST);
75
76          //event set array
77          setButton.addActionListener(e -> setArrayFromInput());
78
79          //event langkah selanjutnya
80          stepButton.addActionListener(e -> performStep());
81
82          //event reset
83          resetButton.addActionListener(e -> reset());
84
85      }
86
```

3) Untuk menangani tombol “Set Array”, buat method setArrayFromInput(). Method ini akan membaca data dari input pengguna, memisahkan angka-angka berdasarkan koma, mengubahnya ke dalam bentuk array integer, lalu mengatur nilai awal gap (jarak elemen) sebagai setengah dari panjang array. Label-label visual akan dibuat untuk mewakili elemen array agar bisa divisualisasikan dalam panel tampilan.

```
87    private void setArrayFromInput() {
88        String text = inputField.getText().trim();
89        if (text.isEmpty()) return;
90        String[] parts = text.split(",");
91        array = new int[parts.length];
92        try {
93            for (int k = 0; k < parts.length; k++) {
94                array[k] = Integer.parseInt(parts[k].trim());
95            }
96        } catch (NumberFormatException e) {
97            JOptionPane.showMessageDialog(this, "Masukkan hanya "
98                + "angka yang dipisahkan koma!", "Error", JOptionPane.ERROR_MESSAGE);
99            return;             }
100       gap = array.length / 2;
101       i = gap;
102       sorting = true;
103       stepCount = 1;
104       stepArea.setText("");
105       stepButton.setEnabled(true);
106       panelArray.removeAll();
107       labelArray = new JLabel[array.length];
108       for (int k = 0; k < array.length; k++) {
109           labelArray[k] = new JLabel(String.valueOf(array[k]));
110           labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
111           labelArray[k].setOpaque(true);
112           labelArray[k].setBackground(Color.WHITE);
113           labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
114           labelArray[k].setPreferredSize(new Dimension(50, 50));
115           labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
116           panelArray.add(labelArray[k]);
117       }
118       panelArray.revalidate();
119       panelArray.repaint();
120   }
```

4) Implementasikan method performStep() sebagai pusat logika dari proses pengurutan, yang akan dijalankan setiap kali pengguna menekan tombol “Langkah Berikutnya”. Di sinilah algoritma Shell Sort berjalan langkah demi langkah menggunakan nilai gap yang terus diperkecil.

```
122    private void performStep() {
123        resetHighlights();
124        if (!sorting || gap == 0) {
125            stepArea.append("Shell Sort selesai.\n");
126            stepButton.setEnabled(false);
127            JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
128            stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
129            return;             }
130        if (i < array.length) {
131            if (!isSwapping) {
132                temp = array[i];
133                j = i;
134                isSwapping = true;          }
135            if (j >= gap && array[j - gap] > temp) {
136                array[j] = array[j - gap]; // geser ke kanan
137                labelArray[j].setBackground(Color.GREEN);
138                labelArray[j - gap].setBackground(Color.CYAN);
139                updateLabels();
140                logStep("Geser elemen " + array[j] + " ke kanan");
141                j -= gap;
142            } else {
143                array[j] = temp; // letakkan nilai temp
144                updateLabels();
145                logStep("Tempatkan " + temp + " di posisi " + j);
146                i++;
147                isSwapping = false;         }
148        } else {
149            gap /= 2;
150            i = gap;
151            isSwapping = false;
152            stepArea.append("Langkah " + stepCount++ + ": Kurangi gap menjadi " + gap + "\n\n");
153        }
154    }
```

5) Agar proses lebih informatif, buat method logStep() untuk mencatat setiap aksi atau perubahan yang terjadi ke dalam JTextArea. Gunakan juga method highlight() untuk menyoroti elemen yang sedang dibandingkan atau digeser, serta resetHighlights() untuk menghapus penyorotan dan mengembalikan tampilan label ke warna default.

```
156    private void logStep(String message) {
157        stepArea.append("Langkah " + stepCount + ": " + message + "\n");
158        stepArea.append("Array: " + java.util.Arrays.toString(array) + "\n\n");
159    }
160
161    private void updateLabels() {
162        for (int k = 0; k < array.length; k++) {
163            labelArray[k].setText(String.valueOf(array[k]));
164        }
165    }
166
167    private void resetHighlights() {
168        if (labelArray == null) return;
169        for (JLabel label : labelArray) {
170            label.setBackground(Color.WHITE);
171        }
172    }
```

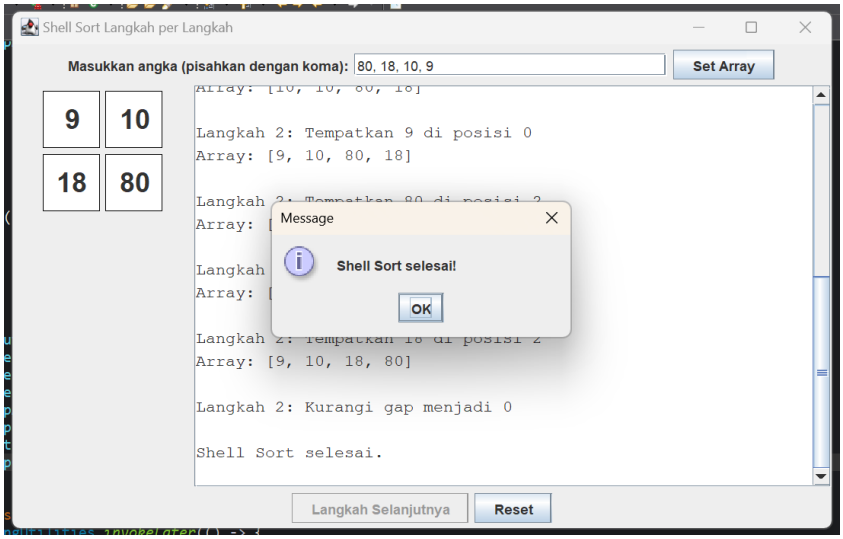
6) Tambahkan method reset() untuk mengatur ulang seluruh tampilan dan data aplikasi, menghapus array yang sedang diproses, membersihkan panel, dan memulai kembali dari kondisi awal.

```
174    private void reset() {
175        inputField.setText("");
176        panelArray.removeAll();
177        panelArray.revalidate();
178        panelArray.repaint();
179        stepArea.setText("");
180        stepButton.setEnabled(false);
181        sorting = false;
182        stepCount = 1;
183    }
```

7) Buat method main() sebagai pintu masuk program, di mana objek dari kelas GUI dibuat dan antarmuka ditampilkan kepada pengguna.

```
185    public static void main(String[] args) {
186        SwingUtilities.invokeLater(() -> {
187            ShellSortGUI gui = new ShellSortGUI();
188            gui.setVisible(true);
189        });
190    }
191
192 }
```

8) Saat program dijalankan, pengguna akan dapat melihat bagaimana algoritma Shell Sort bekerja secara interaktif, dengan visualisasi perubahan array pada setiap langkah serta log informasi yang terus diperbarui.



D. Kesimpulan

Dari hasil implementasi dan simulasi berbagai algoritma pengurutan menggunakan aplikasi berbasis Java Swing, dapat disimpulkan bahwa masing-masing metode—Bubble Sort, Quick Sort, Shell Sort, dan Merge Sort—memiliki ciri khas dan pola kerja tersendiri. Bubble Sort dan Shell Sort menggunakan pendekatan iteratif dengan melakukan perbandingan serta pertukaran antar elemen secara bertahap, sementara Quick Sort dan Merge Sort menerapkan strategi divide and conquer, yaitu membagi data menjadi bagian-bagian kecil sebelum dilakukan proses pengurutan. Penyajian proses secara visual dan bertahap dalam aplikasi ini sangat membantu dalam memperjelas bagaimana algoritma dijalankan, mulai dari tahap perbandingan hingga proses pengurutan selesai.

Pengamatan lebih lanjut menunjukkan bahwa Quick Sort dan Merge Sort unggul dalam hal efisiensi, terutama saat digunakan untuk mengurutkan data dalam jumlah besar. Kedua algoritma ini mampu menyelesaikan pengurutan dengan jumlah langkah yang lebih sedikit dan waktu proses yang lebih singkat jika dibandingkan dengan Bubble Sort maupun Shell Sort. Tampilan grafis interaktif yang dibangun melalui Java Swing membuat pengguna lebih mudah memahami alur kerja setiap algoritma, sehingga aplikasi ini menjadi sarana yang sangat mendukung dalam proses pembelajaran algoritma dan struktur data.

Dengan fleksibilitas dan kemudahan pengembangan yang ditawarkan oleh Java Swing, aplikasi ini berpotensi untuk dikembangkan lebih lanjut agar mendukung algoritma-algoritma lainnya. Visualisasi berbasis GUI semacam ini terbukti efektif dalam memperkuat pemahaman terhadap konsep dasar sorting, serta membuka peluang baru dalam penerapan teknologi untuk keperluan edukasi dan pengembangan perangkat lunak di bidang informatika.