LAPORAN PRAKTIKUM STRUKTUR DATA

IMPLEMENTASI ALGORITMA SORTING BUBBLE, SHELL, QUICK DAN MERGESORT PADA PEMROGRAMAN JAVA

Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

ASISTEN PRAKTIKUM : RAHMAT DWIRIZKI OLDERS

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Praktikum ini dilakukan untuk mengurutkan (sorting) data yang merupakan salah satu proses fundamental untuk mengorganisasi data secara terstruktur dan efisien. Sorting memudahkan dalam proses pencarian, analisis dan tampilan data. Membuat GUI dengan algoritma bubble, shell, quick dan mergesort yang dapat menginputkan angka lalu program mengikuti (flow) dari penerapan algoritma sorting pada logic program sehingga dapat melihat visualisasi sorting secara bertahap.

1. Bubble Sort

   Merupakan algoritma sorting paling dasar yang bekerja dengan membandingkan pasangan elemen secara berurutan, lalu menukarnya jika berada dalam urutan yang salah. Proses ini diulang hingga tidak ada lagi pasangan elemen yang perlu ditukar. Nilai yang paling besar (atau kecil) "menggelembung" ke posisi akhir (atau awal) array seperti gelembung air.

   Namun, dari sisi efisiensi, Bubble Sort termasuk lambat karena memiliki kompleksitas waktu rata-rata dan terburuk $O(n^2)$. Artinya, semakin banyak data, maka waktu yang dibutuhkan akan meningkat secara eksponensial. Untuk dataset besar, algoritma ini sangat tidak disarankan karena performanya buruk dibandingkan metode lainnya.

2. Shell Sort

   Merupakan algoritma sorting penyempurnaan dari Insertion Sort yang mengatasi kelemahan saat elemen yang harus disisipkan berada jauh dari posisi akhir. Shell Sort membagi array menjadi beberapa sub-array berdasarkan gap (jarak antar elemen), kemudian dilakukan insertion sort pada masing-masing sub-array. Gap ini akan mengecil secara bertahap hingga akhirnya dilakukan sorting biasa saat gap bernilai 1. Teknik ini membantu mempercepat proses sorting dengan memindahkan elemen-elemen ke posisi yang lebih dekat dengan urutan akhirnya sejak awal. Meskipun tidak secepat Quick Sort atau Merge Sort dalam kasus besar, Shell Sort memiliki performa lebih baik dari Bubble atau Insertion Sort dan tidak membutuhkan memori tambahan.

3. Quick Sort

   Merupakan algoritma sorting yang sangat efisien dan banyak digunakan dalam praktik pemrograman. Quick Sort menggunakan pendekatan *divide and conquer* dengan memilih satu elemen sebagai pivot, lalu membagi array ke dalam dua bagian: elemen yang lebih kecil dari pivot di kiri, dan yang lebih besar di kanan. Kedua bagian ini kemudian disorting secara rekursif. Kelebihan utama Quick Sort adalah kecepatan dan efisiensinya, terutama untuk dataset besar. Dalam kasus rata-rata, kompleksitas waktunya adalah $O(n \log n)$, namun dalam kasus terburuk (misalnya jika pivot selalu elemen terkecil atau terbesar), bisa mencapai $O(n^2)$. Namun, dengan teknik pemilihan pivot yang baik (seperti median-of-three), performa Quick Sort tetap stabil dan cepat.

4. Merge Sort

   Merupakan algoritma sorting yang menggunakan pendekatan divide and conquer, dengan memecah array menjadi dua bagian hingga ukuran satu elemen, kemudian digabungkan kembali (merge) dalam urutan yang benar. Proses penggabungan dilakukan dengan membandingkan elemen dari masing-masing bagian dan memasukkannya ke array baru. Keunggulan Merge Sort adalah

stabilitas dan konsistensinya: kompleksitas waktunya selalu O(n log n) di semua kasus, baik terbaik, rata-rata, maupun terburuk. Namun, berbeda dengan Quick Sort, Merge Sort membutuhkan ruang tambahan (memori) sebesar O(n) untuk menyimpan hasil penggabungan, sehingga kurang efisien dalam konteks penggunaan memori.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. Memahami dan mengaplikasikan algoritma Sorting (bubble, shell, quick dan mergeSort) dalam program java untuk mengurutkan data.
2. Membuat GUI dengan implementasi sorting sort.

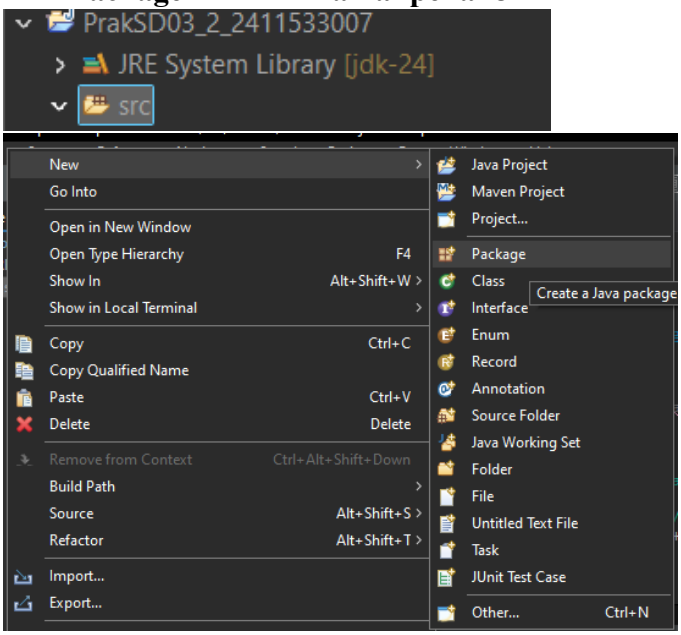
C. Langkah kerja praktikum

a. Alat dan Bahan

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integreted Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)

b. Package pekan 8

1. Buat new package**. Buka java project** yang telah dibuat sebelumnya, lalu **klik kanan** pada folder '**src**', setelahnya akan muncul list option, pilih '**new**', lalu '**Package**' dan beri **nama** '**pekan8**'.



c. Program BubbleSortGUI

1. Klik kanan pada package pekan8, new, other, windowBuilder lalu Jframe dan beri nama file GUI "**BubbleSortGUI"**.
2. Import package java util unutk menipulasi array dan struktur data, java awt untuk komponen GUI dasar berupa layout, dan java swing untuk membuat komponen GUI seperti Jbutton dan Jlabel.

```
1 package pekan8;
2 import java.util.*;
3 import java.awt.*;
4 import javax.swing.*;
```

3. Deklarasi class "BubbleSortGUI" dengan memnggunakna JFFrame (jendela utama aplikasi Swing).

```
6 public class BubbleSortGUI extends JFrame {
```

4. Membuat Variabel Global, array unutk menyimpan angka yang akan diurutkan. Label Array yang menampilkan secara visual. StepButton,reset dan setButton sebagai tombol GUI. InputField tempat input angka adri user. panelArray unutk menampilkan angka dalam bentuk label. stepArea unutk mencetak langkah-langkah sorting. Variabel i dan j variabel iterasi dalam insertion sort. Sorting boolean untuk memeriksa sorting sedang berjalan. StepCount sebagai nomor langkah sorting.

```
7       private static final long serialVersionUID = 1L;
8       private int[] array;
9       private JLabel[] labelArray;
10      private JButton stepButton, resetButton, setButton;
11      private JTextField inputField;
12      private JPanel panelArray;
13      private JTextArea stepArea;
14
15      private int i = 1, j;
16      private boolean sorting = false;
17      private int stepCount = 1;
```

5. **Main method**. Menjalankan GUI. EventQueue unutk menjalakan kode GUI, menjalakannya di thread yang benar. new InsertionSortGUI(i) membuat object baru dari kelas InsertionSortGUI turunan Jframe. frame.setVisible(true); untuk menampilkan GUI ke display.

```
22      public static void main(String[] args) {
23          EventQueue.invokeLater(new Runnable() {
24              public void run() {
25                  try {
26                      BubbleSortGUI frame = new BubbleSortGUI();
27                      frame.setVisible(true);
28                  } catch (Exception e) {
29                      e.printStackTrace();
30                  }
31              }
32          });
33      }
```

6. Konstruktor GUI, menentukan model/tampilan dari GUInya berupa ukuran, layout dan semua komponen GUI.

```
38●     public BubbleSortGUI() {
39          setTitle("Bubbl Sort Langkah per Langkah");
40          setSize(750, 400);
41          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
42          setLocationRelativeTo(null);
43          setLayout(new BorderLayout());
```

Panel input, terdiri dari inputField unutk tempat user mengetik angka dan setButton tombol yang memproses input dna menampilkan array.

```
45          //Panel input
46          JPanel inputPanel = new JPanel (new FlowLayout());
47          inputField = new JTextField(30);
48          setButton = new JButton("Set Array");
49          inputPanel.add(new JLabel("Masukan angka (pisahkan dengan koma):"));
50          inputPanel.add(inputField);
51          inputPanel.add(setButton);
```

Panel array visual digunakna unutk menampilkan elemen array sebagai label(Jlabel) secara horizontal.

```
53          //Panel array visual
54          panelArray = new JPanel();
55          panelArray.setLayout(new FlowLayout());
```

Panel tombol kontrol terdiri dari tombol step dan reset button.

```
57          //Panel kontrol
58          JPanel controlPanel = new JPanel();
59          stepButton = new JButton("Langkah selanjutnya");
60          resetButton = new JButton ("Reset");
61          stepButton.setEnabled(false);
62          controlPanel.add(stepButton);
63          controlPanel.add(resetButton);
```

Area teks yang menampilkan log langkah-langkah sorting dari tiap penyeleseian array.

```
65          //Area teks unutk log langkah-langkah
66          stepArea = new JTextArea(8, 60);
67          stepArea.setEditable(false);
68          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
69          JScrollPane scrollPane = new JScrollPane(stepArea);
```

Menambahkan panel ke frame, fungsi add() unutk menempatkan panel ke dalam frame menggunakan layout manager “Border layout”, yang membagi frame menjadi 5 bagain utama yaitu north,south,east, west dan center.

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

Menambahkan event listener ke tombol, yaitu event handler ke setButton, yang ketika ditekan akan menjalankan method.

```
77          //Event Set Array
78          setButton.addActionListener(e -> setArrayFromInput());
79
80          //Event Langkah Selanjutnya
81          stepButton.addActionListener(e -> performStep());
82
83          //Event reset
84          resetButton.addActionListener(e -> reset());
85      }
```

7. **Method setArrayFromInput()** berfungsi mengambil input array dari user, memprosesnya, dan menampilkannya secara visual. Modifier "private" artinya method ini hanya bisa dipanggil di dalam kelas ini. "void" mathod tidak mengembalikan nilai.

```
87          //Function setArray
88          private void setArrayFromInput() {
```

Mengambil input teks input field, '.trim()' berfugnsi menghapus spasi di awal/akhir. Jika input kosong maka method berhenti.

```
89              String text = inputField.getText().trim();
90              if (text.isEmpty()) return;
```

Memisahkan input menjadi array string. 'split(",")' berfugnsi memisahkan input berdasarkan koma. 'array = new int[parts.length];' Membuat array integer sebanyak jumlah angka yang diinput user.

```
91              String[] parts = text.split(",");
92              array = new int[parts.length];
```

Try{} and catch {} berfungsi unutk menangani kesalahn/eror saat program berjalan, agar program tidak langsung berhenti/crash melainkan menampilkna pesan eror ke user.

Konversi String ke Integer dari array "parts" menggunakan for loop yang blok kode nya akan mengubah tiap elemen hasil looping menjadi integer 'Integer.parseInt()'. 'trim()' menghapus spasi jika ada.

'catch (NumberFormatException e)' jika eror terjadi, maka program tidak akan langusng crash, melainkan tampil popup peringatan ke user menggunakan 'JOptionPane.showMessageDialog', lalu menghentikan ekskusi method dengan 'return;' agar tidak lanjut ke bagian lain.

Block try-catch memastikan input tidak valid yang menyebabkan program eror, dengan memberitahu tahu user dengan popup message, proses parsing hanya dilakukan jika semua input valid. Membuat program lebih user-frienly dan dapat menghandle eror dengan baik.

```
93          try {
94              for (int k = 0; k < parts.length; k++) {
95                  array[k] = Integer.parseInt(parts[k].trim()); }
96          } catch (NumberFormatException e) {
97              JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan "
98              + "dengan koma!", "Eror", JOptionPane.ERROR_MESSAGE);
99              return;
100         }
```

Inisialisasi variabel I dan j sebagai index perulangan bubble sort. stepCount berfungsi menghitung banyaknya proses sorting dilakukan. Sorting boolean true sebagai tanda bahwa proses sorting berjalan.

```
100         i = 0;
101         j = 0;
102         stepCount = 1;
103         sorting = true;
```

Mengaktifkan button yaitu tombol unutk proses sorting satu per satu setelah input berhasil. StepArea dikosongkan agar siap menampilkan log sorting baru.

```
104             stepButton.setEnabled (true);
105             stepArea.setText("");
```

Reset tampilan panel array. Menghapus semau komponen lama dari panelArray, siapkan array Jlabel untuk menampilkan elemen array ke user.

```
106             panelArray.removeAll();
107             labelArray = new JLabel [array.length];
```

Tampilkan elemen array ke GUI, loop setiap elemen array, buat Jlabel berisi angka array.

```
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].setBorder(BorderFactory.createLineBorder(Color.BLACK));
112             labelArray[k].setPreferredSize(new Dimension(50,50));
113             labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
114             panelArray.add(labelArray[k]);
115         }
```

Refresh tampilan panel. revalidate(), Memberitahu layout manager untuk memperbarui posisi/ukuran komponen. repaint(), Memaksa komponen untuk di-redraw ulang, agar perubahan terlihat.

```
115             }
116             panelArray.revalidate();
117             panelArray.repaint();
118         }
```

Penjelasan method setArrayFromInput() : Mengambil input angka dari user dalam bentuk string, mengonversinya ke array integer (parsing), Menampilkan array tersebut sebagai deretan kotak angka di tampilan GUI, mengatur ulang kondisi agar siap untuk sorting visual secara bertahap. Sebagai langkah awal sebleum proses sorting dilakukan.

8. **Method performStep(),** modifier private artinya hanya dapat diakses dalam class ini. ‘Void’ artinya method ini tidak mengembalikan nilai.

```
122     private void performStep() {
```

Memeriksa kondisi awal apakah sorting masih berlangsung. For loop mengecek apakah indeks ‘i’ masih dalam batas array dan proses sorting masih aktif. ‘i’ merupakan elemen arrray yang sedang dimasukan ke posisi yg tepat.

```
123             if (i < array.length && sorting) {
```

Mengambil nilia kunci(key). Key adalah nilai yang ingin dimasukkan ke posisi yang benar di bagian array yang sudah terurut. j adalah indeks untuk membandingkan key dengan elemen sebelumnya.

```
123             if (i < array.length && sorting) {
124                 int key = array[i];
125                 j = i - 1;
```

Logging langkah ke JtextArea. stepLog digunakan untuk membuat teks penjelasan tiap langkah sorting. Mencatat langkah ke berapa, dan elemen mana yang sedang dimasukkan (key)

```
127             StringBuilder stepLog = new StringBuilder();
128             stepLog.append("Langkah ").append(stepCount)
129             .append(": Memasukkan ").append(key).append("\n");
```

Proses Sorting (Insertion Sort). Loop ini **menggeser elemen** yang lebih besar dari key ke kanan untuk memberi ruang bagi key. Loop ini **menggeser elemen** yang lebih besar dari key ke kanan untuk memberi ruang bagi key.

```
131                 while(j >= 0 && array[j] > key) {
132                     array [j + 1] = array[j];
133                     j--;
134                 }
135                 array[j + 1] = key;
```

Memperbarui tampilan JLabel array di panel. Menambahkan hasil perubahan array ke stepArea agar user bisa melihat hasil tiap langkah.

```
135             updateLabels();
136             stepLog.append("Hasil: ").append(arrayToString(array)).append("\n\n");
137             stepArea.append(stepLog.toString());
```

Naikkan Indeks dan Langkah.

```
139                 i++;
140                 stepCount++;
```

Akhiri Jika Sorting Selesai. Jika ‘i’ sudah mencapai akhir array, proses sorting selesei, stepButton dimatikan, menampilkan popup bahwa sorting selesei.

```
142             if (i == array.length) {
143                 sorting = false;
144                 stepButton.setEnabled(false);
145                 JOptionPane.showMessageDialog(this, "Sorting selesei!");
146             }
147         }
148     }
```

9. **Method performStep(),** berfungsi melakukan satu langkah sorting menggunakan algoritma Bubble Sort, lalu menampilkan hasil perbandingan dan/atau pertukaran elemen dalam array secara visual di layar (GUI).
Pengecekan status sorting, jika proses sorting tidak aktif (!sorting) atau interai sudah mencapi akhir array (artinya sudah selesei), maka sorting boolean di set false, stepButton dinonaktifkan, Menampilkan pesan sorting telah selesei lalu keluar dari fungsi(return).

```
124  private void performStep() {
125      if (!sorting || i >= array.length - 1) {
126          sorting = false;
127          stepButton.setEnabled(false);
128          JOptionPane.showMessageDialog(this, "Sorting selesei!");
129          return;
130      }
```

Memanggil method resetHightLights(), unutk membersihkan warna dari label array sebelumnya

```
131         resetHighLights();
```

Inisialisasi log langkah, menyiapkan log teks untuk ditampilkan di stepArea.

```
132         StringBuilder stepLog = new StringBuilder();
```

Highlight Elemen yang Dibandingkan, memberi warna cyan pada dua elemen array yang sedang diabndingkan (j dan j +1).

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

Bandingkan dan tukar jika perlu, jika nilai di index j > dari j + 1, maka tukar nilai (swap).

```
135         if (array[j] > array[j + 1]) {
136             //swap
137             int temp = array[j];
138             array[j] = array [j + 1];
139             array [j + 1] = temp;
```

Setelah ditukar, kedua elemen yang ditukar diberi warna merah untuk menandai adanya perubahan.

```
140             labelArray[j].setBackground(Color.RED);
141             labelArray[j + 1].setBackground(Color.RED);
```

Menampilkan log pesan bahwa pertukaran terjadi dengan menyebutkan index dan nilainya.

```
142         stepLog.append("Langkah ").append(array[j + 1]).append(": Menukar elemen ke-")
143         .append(j).append(" (").append(array[j + 1]).append(") dengan ke -")
144         .append(j + 1).append(" (").append(array[j]).append(") \n");
```

Jika tidak ada pertukaran, maka catat dalam log bahwa nilai sudah dalam urutan yang benar.

```
145         } else {
146             stepLog.append("Langkah ").append(stepCount)
147             .append(": Tidak ada pertukaran antara ke-")
148             .append(j).append(" dan ke -").append(j + 1).append("\n");
149         }
```

Tambahkan hasil dan perbarui tampilan

```
150         stepLog.append("Hasil: ").append(arrayToString(array))
151         .append("\n\n");stepArea.append(stepLog.toString());
152         updateLabels();
```

Update index j dan I, index j dinaikan satu unutk perbandingan berikutnya. Jika j sudah mencapai batsas loop (karena setiap iterasi i terbesar sudah di ujung), maka j di-reset ke 0 dan i dinaikkan.

```
153             j++;
154             if (j >= array.length - i - 1) {
155                 j = 0;
156                 i++;
157             }
```

Update Step dan Cek Akhir, counter StepCount tambah satu. Jika I sudah sampai akhir array, maka proses sorting di hentikan, stepButton false, lalu tampilkan pesan bahwa sorting telah selesei.

```
158         stepCount++;
159         if (i >= array.length - 1) {
160             sorting = false;
161             stepButton.setEnabled(false);
162             JOptionPane.showMessageDialog(this, "Sorting selesei!");
163         }
164     }
```

Penjelasan method performStep() : membandingkan dua elemen bersebelahan dalam array, menukarnya jika perlu, lalu memperbarui tampilan dan log

langkah sorting secara bertahap hingga array tersortir. Proses ini diulang setiap tombol langkah ditekan.

10. **Method updateLabels(),** berfungsi unutk memperbarui tampilan angka pada komponen GUI(labelArray) sesuai dengan isi terkini dari array. Melakukan loop untuk tiap elemen dalam array, mengubah teks dari labek ke-k agar menampilkan angka baru setelah proses sorting. Mengubah integer menjadi string agar bisa ditampilkan sebagai teks. Dipakai setelah perubahan nilai array seperti saat proses sprting berlangsung.

```
150     //Function
151     private void updateLabels() {
152         for (int k = 0; k < array.length; k++) {
153             labelArray[k].setText(String.valueOf(array[k]));
154         }
155     }
```

11. **Method resetHighlights(),** berfungsi unutk **menghapus warna highlight sebelumnya**, agar proses highlight baru (seperti di highlightMinIndex()) tidak menumpuk atau membuat tampilan membingungkan. dipanggil setiap kali sebelum memberi highlight baru, supaya hanya satu elemen yang diberi warna khusus.

```
174     private void resetHighLights() {
175         for (JLabel label : labelArray) {
176             label.setBackground(Color.WHITE);
177         }
178     }
```

12. **Method reset()**, berfungsi untuk mengatur ulang tampilan dan variabel ke kondisi awal (menghapus semua inputan dan proses). Mengatur ulang index dan penghitung langkah.

```
158     private void reset () {
159         inputField.setText("");
160         panelArray.removeAll();
161         panelArray.revalidate();
162         panelArray.repaint();
163         stepArea.setText("");
164         stepButton.setEnabled(false);
165         sorting = false;
166         i = 1;
167         stepCount = 1;
168     }
```

13. **Method arrayToString(int[] arr)**, berfungsi mengubah sebuah array integer menjadi string berformat seperti 1, 2, 3. Menampilkan array ke dalam stepArea selama proses sorting.

```
171     private String arrayToString(int[] arr) {
172         StringBuilder sb = new StringBuilder();
173         for (int k = 0; k < arr.length; k++) {
174             sb.append(arr[k]);
175             if (k < arr.length - 1) sb.append(", ");
176         }
177         return sb.toString();
178     }
179 }
```

14. Output program : Program BubbleSortGUI adalah aplikasi Java Swing yang mengimplementasikan algoritma BubbleSort dengan visualisasi step-by-step.

Program ini memiliki interface yang terdiri dari text field untuk input array (angka dipisahkan koma), panel visual yang menampilkan elemen array dalam bentuk kotak-kotak berlabel, tombol kontrol untuk menjalankan langkah sorting, dan area teks untuk menampilkan log setiap langkah proses.

Alur kerja program bekerja dengan cara menerima input angka dari pengguna yang dipisahkan koma, lalu menampilkannya dalam bentuk label di antarmuka. Saat tombol "Langkah" ditekan, program menjalankan satu langkah algoritma Bubble Sort, yaitu membandingkan dua elemen yang berdekatan dan menukarnya jika diperlukan. Setiap langkah ditampilkan secara visual dengan highlight warna dan log teks. Proses ini terus dilakukan hingga seluruh data terurut, kemudian tombol akan dinonaktifkan dan pesan "Sorting selesai" ditampilkan.

d. Program ShellSortGUI

1. Klik kanan pada package pekan8, new, other, windowBuilder lalu Jframe dan beri nama file GUI "**ShellSortGUI**".
2. Import package java util unutk menipulasi array dan struktur data, java awt unutk komponen GUI dasar berupa layout, java swing unutk membuat komponen GUI seperti Jbutton dan Jlabel, dan java eventQueue untuk menjalankan GUI secara thread-safe (agar aman dan konsisten).

```
1 package pekan8;
2 import java.util.*;
3 import java.awt.*;
4 import javax.swing.*;
```

3. Deklarasi class "ShellSortGUI" dengan memnggunakna JFFrame (jendela utama aplikasi Swing).

```
6 public class ShellSortGUI extends JFrame {
```

4. Deklarasi variabel. Menyimpan data array, elemen visual GUI, tombol interaktif GUI, dan variabel kontrol sorting. Variabel I sebagai index utama iterasi luar pada algortima sorting, variabel j digunakan unutk inner loop saat membandingkan dan menukar elemen. Boolean sorting sebagai penanda proses sorting sedang berlangsung atau tidak. StepCount unutk menghitung

proses sorting dilakukan. minIndex khusu unutk selection sort, menyimpan index dari nilai terkecil yang ditemukan selama satu putara iterasi.

```
 9      private static final long serialVersionUID = 1L;
10      private int[] array;
11      private JLabel[] labelArray;
12      private JButton stepButton, resetButton, setButton;
13      private JTextField inputField;
14      private JPanel panelArray;
15      private JTextArea stepArea;
```

Variabel kontrol proses sorting: indeks i dan j, gap (jarak antar elemen Shell Sort), temp untuk menyimpan sementara saat swap, sorting status berjalan, dan stepCount menghitung langkah.

```
16      private int i = 1, j;
17      private int gap;
18      private int temp;
19      private boolean sorting = false;
20      private int stepCount = 1;
```

5. **Main method**. Menjalankan GUI. EventQueue unutk menjalakan kode GUI, menjalakannya di thread yang benar. new SelectionSortGUI(i) membuat object baru dari kelas SelectionSortGUI turunan Jframe. frame.setVisible(true); untuk menampilkan GUI ke display

```
25●     public static void main(String[] args) {
26●         EventQueue.invokeLater(new Runnable() {
27●             public void run() {
28                  try {
29                      ShellSortGUI frame = new ShellSortGUI();
30                      frame.setVisible(true);
31                  } catch (Exception e) {
32                      e.printStackTrace();
33                  }
34              }
35          });
36      }
```

6. **Konstruktor GUI,** menentukan model/tampilan dari GUInya berupa ukuran, layout dan semua komponen GUI.

```
41●     public ShellSortGUI() {
42          setTitle("Shell Sort Langkah per Langkah");
43          setSize(750, 400);
44          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45          setLocationRelativeTo(null);
46          setLayout(new BorderLayout());
```

Panel input, terdiri dari inputField unutk tempat user mengetik angka dan setButton tombol yang memproses input dna menampilkan array.

```
45          //Panel input
46          JPanel inputPanel = new JPanel (new FlowLayout());
47          inputField = new JTextField(30);
48          setButton = new JButton("Set Array");
49          inputPanel.add(new JLabel("Masukan angka (pisahkan dengan koma):"));
50          inputPanel.add(inputField);
51          inputPanel.add(setButton);
```

Panel array visual digunakna unutk menampilkan elemen array sebagai label(Jlabel) secara horizontal.

```
53          //Panel array visual
54          panelArray = new JPanel();
55          panelArray.setLayout(new FlowLayout());
```

Panel tombol kontrol terdiri dari tombol step dan reset button.

```
57          //Panel kontrol
58          JPanel controlPanel = new JPanel();
59          stepButton = new JButton("Langkah selanjutnya");
60          resetButton = new JButton ("Reset");
61          stepButton.setEnabled(false);
62          controlPanel.add(stepButton);
63          controlPanel.add(resetButton);
```

Area teks yang menampilkan log langkah-langkah sorting dari tiap penyeleseian array.

```
65          //Area teks unutk log langkah-langkah
66          stepArea = new JTextArea(8, 60);
67          stepArea.setEditable(false);
68          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
69          JScrollPane scrollPane = new JScrollPane(stepArea);
```

Menambahkan panel ke frame, fungsi add() unutk menempatkan panel ke dalam frame menggunakan layout manager “Border layout”, yang membagi frame menjadi 5 bagain utama yaitu north,south,east, west dan center.

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

Menambahkan event listener ke tombol, yaitu event handler ke setButton, yang ketika ditekan akan menjalankan method.

```
77          //Event Set Array
78          setButton.addActionListener(e -> setArrayFromInput());
79
80          //Event Langkah Selanjutnya
81          stepButton.addActionListener(e -> performStep());
82
83          //Event reset
84          resetButton.addActionListener(e -> reset());
85      }
```

7. **Method setArrayFromInput()** berfungsi mengambil input array dari user, memprosesnya, dan menampilkannya secara visual. Modifier “private” artinya method ini hanya bisa dipanggil di dalam kelas ini. “void” mathod tidak mengembalikan nilai.

```
87          //Function setArray
88          private void setArrayFromInput() {
```

Mengambil input teks input field, ‘.trim()’ berfugnsi menghapus spasi di awal/akhir. Jika input kosong maka method berhenti.

```
89              String text = inputField.getText().trim();
90              if (text.isEmpty()) return;
```

Memisahkan input menjadi array string. ‘split(“,”)’ berfugnsi memisahkan input berdasarkan koma. ‘array = new int[parts.length];’ Membuat array integer sebanyak jumlah angka yang diinput user.

```
91          String[] parts = text.split(",");
92          array = new int[parts.length];
```

Try{} and catch {} berfungsi unutk menangani kesalahn/eror saat program berjalan, agar program tidak langsung berhenti/crash melainkan menampilkna pesan eror ke user.

Konversi String ke Integer dari array “parts” menggunakan for loop yang blok kode nya akan mengubah tiap elemen hasil looping menjadi integer ‘Integer.parseInt()’. ‘trim()’ menghapus spasi jika ada.

‘catch (NumberFormatException e)’ jika eror terjadi, maka program tidak akan langusng crash, melainkan tampil popup peringatan ke user menggunakan ‘JOptionPane.showMessageDialog’, lalu menghentikan ekskusi method dengan ‘return;’ agar tidak lanjut ke bagian lain.

Block try-catch memastikan input tidak valid yang menyebabkan program eror, dengan memberitahu tahu user dengan popup message, proses parsing hanya dilakukan jika semua input valid. Membuat program lebih user-frienly dan dapat menghandle eror dengan baik.

```
93          try {
94              for (int k = 0; k < parts.length; k++) {
95                  array[k] = Integer.parseInt(parts[k].trim()); }
96          } catch (NumberFormatException e) {
97              JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan "
98              + "dengan koma!", "Eror", JOptionPane.ERROR_MESSAGE);
99              return;
100         }
```

Inisialisasi variabel gap sebagai jarak antar elemen yang akan dibandingkan dan ditukar dalam Shell Sort, variabel i menunjukan posisi elemen yang sedang dibandingkan. stepCount berfungsi menghitung banyaknya proses sorting dilakukan. Sorting boolean true sebagai tanda bahwa proses sorting berjalan.

```
104             gap = array.length / 2;
105             i = gap;
106             sorting = true;
107             stepCount = 1;
```

Mengaktifkan button yaitu tombol unutk proses sorting satu per satu setelah input berhasil. StepArea dikosongkan agar siap menampilkan log sorting baru.

```
104             stepButton.setEnabled (true);
105             stepArea.setText("");
```

Reset tampilan panel array. Menghapus semau komponen lama dari panelArray, siapkan array Jlabel untuk menampilkan elemen array ke user.

```
106             panelArray.removeAll();
107             labelArray = new JLabel [array.length];
```

Tampilkan elemen array ke GUI, loop setiap elemen array, buat Jlabel berisi angka array.

```
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].setBorder(BorderFactory.createLineBorder(Color.BLACK));
112             labelArray[k].setPreferredSize(new Dimension(50,50));
113             labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
114             panelArray.add(labelArray[k]);
115         }
```

Refresh tampilan panel. revalidate(), Memberitahu layout manager untuk memperbarui posisi/ukuran komponen. repaint(), Memaksa komponen untuk di-redraw ulang, agar perubahan terlihat.

```
115             }
116             panelArray.revalidate();
117             panelArray.repaint();
118         }
```

Penjelasan method setArrayFromInput() : Mengambil input angka dari user dalam bentuk string, mengonversinya ke array integer (parsing), Menampilkan array tersebut sebagai deretan kotak angka di tampilan GUI, mengatur ulang kondisi agar siap untuk sorting visual secara bertahap. Sebagai langkah awal sebleum proses sorting dilakukan.

8. **Method performStep(),** berfungsi melakukan satu langkah sorting menggunakan algoritma Bubble Sort, lalu menampilkan hasil perbandingan dan/atau pertukaran elemen dalam array secara visual di layar (GUI).
   Menghapus/mengembalikan warna highlight elemen array sebelumnya agar tampilan visualnya bersih untuk langkah selanjutnya.

```
129             resetHighLights();
```

Pengecekan status sorting, jika proses sorting tidak aktif (!sorting) atau interai sudah mencapi akhir array (artinya sudah selesei), maka sorting boolean di set false, stepButton dinonaktifkan, Menampilkan pesan sorting telah selesei lalu keluar dari fungsi(return).

```
130         if (!sorting || gap == 0) {
131             stepArea.append("Shell sort selesai.\n");
132             stepButton.setEnabled(false);
133             JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
134             stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
135             return;
136         }
```

Mengecek apakah indeks i masih dalam batas array untuk diproses.

```
137             if (i < array.length) {
```

Jika belum memulai proses swap (penempatan), maka Ambil elemen array[i] dan simpan di temp untuk disisipkan, lalu Atur j = i dan tandai bahwa proses penyisipan (swapping) sedang dilakukan.

```
138                 if (!isSwapping) {
139                     temp = array[i];
140                     j = i;
141                     isSwapping = true;
142                 }
```

Jika j cukup besar untuk digeser dan elemen sebelumnya (array[j - gap]) lebih besar dari temp, maka Geser elemen tersebut ke posisi j, Warnai elemen sebagai visualisasi (hijau dan cyan), Panggil updateLabels() untuk memperbarui tampilan GUI, Catat langkah dengan logStep(), dan Geser j ke kiri dengan jarak gap untuk melanjutkan proses.

```
143             if (j >= gap && array[j - gap] > temp) {
144                 array[j] = array[j - gap]; // geser ke kanan
145                 labelArray[j].setBackground(Color.GREEN);
146                 labelArray[j - gap].setBackground(Color.CYAN);
147                 updateLabels();
148                 logStep("Geser elemen " + array[j] + " ke kanan");
149                 j -= gap;
150                 return;
```

Jika tidak perlu menggeser lagi, maka letakkan nilai ‘temp’ di posisi saat ini (j), update tampilan GUI dan log langkah, dan naikkan indeks i ke elemen berikutnya dan reset status isSwapping.

```
151             } else {
152                 array[j] = temp; // letakkan nilai temp
153                 updateLabels();
154                 logStep("Tempatkan " + temp + " di posisi " + j);
155                 i++;
156                 isSwapping = false;
157             }
```

Jika semua elemen pada gap saat ini selesai diproses, maka kurangi nilai gap (inti dari Shell Sort), reset i ke nilai awal baru (gap), dan reset status swap, lalu catat pergantian gap pada log.

```
158         } else {
159             gap /= 2;
160             i = gap;
161             isSwapping = false;
162             stepArea.append("Langkah " + stepCount++ + ": Kurangi gap menjadi " + gap + "\n\n");
163         }
164     }
```

Penjelasan **Method performStep() :** bekerja dengan cara memproses satu elemen array per langkah, menyisipkan (insertion) elemen tersebut ke posisi yang sesuai dalam sublist yang ditentukan oleh gap, lalu secara bertahap mengurangi gap hingga 0. Proses ini divisualisasikan dan dicatat langkah demi langkah.

9. **Method logStep(String message),** berfungsi digunakan untuk menampilkan log proses sorting di area teks (stepArea) setiap kali pengguna melakukan satu langkah sorting (step-by-step)

```
168     private void logStep(String message) {
169         stepArea.append("Langkah " + stepCount++ + ": " + message + "\n");
170         stepArea.append("Array: " + java.util.Arrays.toString(array) + "\n\n");
171     }
```

10. **Method resetHighlights(),** berfungsi unutk **menghapus warna highlight sebelumnya**, agar proses highlight baru (seperti di highlightMinIndex()) tidak menumpuk atau membuat tampilan membingungkan. dipanggil setiap kali sebelum memberi highlight baru, supaya hanya satu elemen yang diberi warna khusus.

```
183     private void resetHighLights() {
184         if (labelArray == null)
185         return;
186         for (JLabel label : labelArray) {
187             label.setBackground(Color.WHITE);
188         }
189     }
```

11. **Method updateLabels(),** digunakan setiap kali ada perubahan pada isi array, seperti saat proses penukaran (swap) dalam sorting. Tujuannya untuk menyinkronkan tampilan GUI dengan data array yang terbaru.

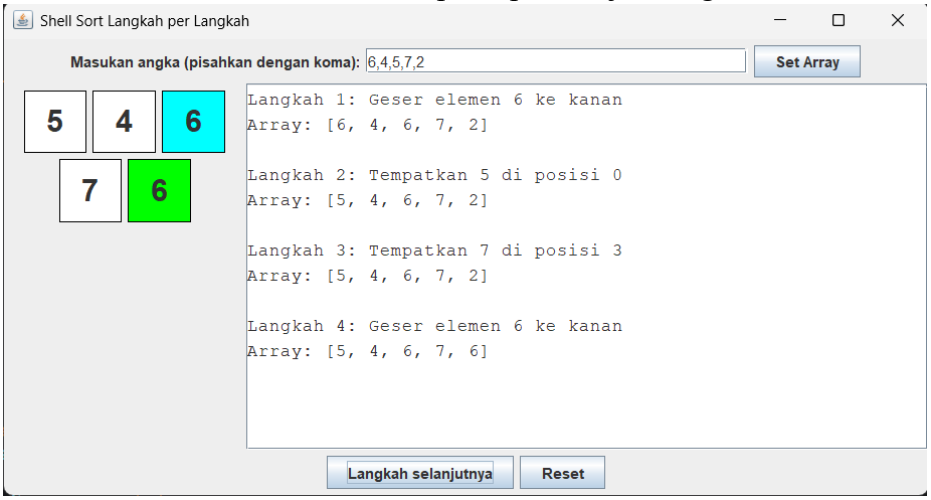
```
189     private void updateLabels() {
190         for(int k = 0; k < array.length; k++) {
191             labelArray[k].setText(String.valueOf(array[k]));
192         }
193     }
```

12. **Method reset(),** digunakan untuk **mengembalikan tampilan dan variabel program ke kondisi awal**, seolah-olah pengguna belum memasukkan data apa pun. Biasanya dipakai setelah proses sorting selesai atau saat pengguna ingin mengulang.

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

13. Output program : Program ShellSortGUI berbasis GUI berfungsi untuk menampilkan proses pengurutan data menggunakan algoritma Shell Sort secara visual dan interaktif. Pengguna dapat memasukkan deretan angka yang dipisahkan koma, lalu menekan tombol “Step” untuk melihat tiap langkah pengurutan berdasarkan nilai gap (jarak antar elemen yang dibandingkan). Program akan menyisipkan elemen ke posisi yang sesuai dalam sublist berdasarkan gap tersebut, memperlihatkan proses perpindahan data secara bertahap.

    Dengan pewarnaan elemen (seperti hijau dan cyan) dan log langkah yang dicetak di area teks, program ini membantu pengguna memahami bagaimana elemen-elemen berpindah posisi selama penyortiran berlangsung. Nilai gap akan berkurang setengah setiap kali seluruh array selesai diproses, hingga akhirnya menjadi 0 yang menandakan proses selesai. Melalui tampilan visual dan interaksi langkah demi langkah, program ini menjadi alat edukatif yang efektif dalam memahami prinsip kerja algoritma Shell Sort.



e. Program QuickSortGUI

1. Klik kanan pada package pekan8, new, other, windowBuilder lalu Jframe dan beri nama file GUI "**QuickSortGUI**".
2. Import package java util unutk menipulasi array dan struktur data, java awt unutk komponen GUI dasar berupa layout, java swing unutk membuat komponen GUI seperti Jbutton dan Jlabel, dan java eventQueue untuk menjalankan GUI secara thread-safe (agar aman dan konsisten).

```
1 package pekan8;
2 import java.util.*;
3 import java.awt.*;
4 import javax.swing.*;
```

3. Deklarasi class "QuickSortGUI" dengan memnggunakna JFFrame (jendela utama aplikasi Swing).

```
21 public class QuickSortGUI extends JFrame {
```

4. Deklarasi variabel. Menyimpan data array, elemen visual GUI, tombol interaktif GUI, dan variabel kontrol sorting. Variabel I sebagai index utama iterasi luar pada algortima sorting, variabel j digunakan unutk inner loop saat membandingkan dan menukar elemen. Boolean sorting sebagai penanda proses sorting sedang berlangsung atau tidak. StepCount unutk menghitung proses sorting dilakukan. minIndex khusu unutk selection sort, menyimpan index dari nilai terkecil yang ditemukan selama satu putara iterasi.

```
 9     private static final long serialVersionUID = 1L;
10     private int[] array;
11     private JLabel[] labelArray;
12     private JButton stepButton, resetButton, setButton;
13     private JTextField inputField;
14     private JPanel panelArray;
15     private JTextArea stepArea;
```

5. **Main method**. Menjalankan GUI. EventQueue unutk menjalakan kode GUI, menjalakannya di thread yang benar. new SelectionSortGUI(i) membuat object baru dari kelas SelectionSortGUI turunan Jframe. frame.setVisible(true); untuk menampilkan GUI ke display

```
25    public static void main(String[] args) {
26        EventQueue.invokeLater(new Runnable() {
27            public void run() {
28                try {
29                    ShellSortGUI frame = new ShellSortGUI();
30                    frame.setVisible(true);
31                } catch (Exception e) {
32                    e.printStackTrace();
33                }
34            }
35        });
36    }
```

6. **Konstruktor GUI,** menentukan model/tampilan dari GUInya berupa ukuran, layout dan semua komponen GUI.

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

Panel input, terdiri dari inputField unutk tempat user mengetik angka dan setButton tombol yang memproses input dna menampilkan array.

```
45        //Panel input
46        JPanel inputPanel = new JPanel (new FlowLayout());
47        inputField = new JTextField(30);
48        setButton = new JButton("Set Array");
49        inputPanel.add(new JLabel("Masukan angka (pisahkan dengan koma):"));
50        inputPanel.add(inputField);
51        inputPanel.add(setButton);
```

Panel array visual digunakna unutk menampilkan elemen array sebagai label(Jlabel) secara horizontal.

```
53          //Panel array visual
54          panelArray = new JPanel();
55          panelArray.setLayout(new FlowLayout());
```

Panel tombol kontrol terdiri dari tombol step dan reset button.

```
57          //Panel kontrol
58          JPanel controlPanel = new JPanel();
59          stepButton = new JButton("Langkah selanjutnya");
60          resetButton = new JButton ("Reset");
61          stepButton.setEnabled(false);
62          controlPanel.add(stepButton);
63          controlPanel.add(resetButton);
```

Area teks yang menampilkan log langkah-langkah sorting dari tiap penyeleseian array.

```
65          //Area teks unutk log langkah-langkah
66          stepArea = new JTextArea(8, 60);
67          stepArea.setEditable(false);
68          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
69          JScrollPane scrollPane = new JScrollPane(stepArea);
```

Menambahkan panel ke frame, fungsi add() unutk menempatkan panel ke dalam frame menggunakan layout manager “Border layout”, yang membagi frame menjadi 5 bagain utama yaitu north,south,east, west dan center.

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

Menambahkan event listener ke tombol, yaitu event handler ke setButton, yang ketika ditekan akan menjalankan method.

```
77          //Event Set Array
78          setButton.addActionListener(e -> setArrayFromInput());
79
80          //Event Langkah Selanjutnya
81          stepButton.addActionListener(e -> performStep());
82
83          //Event reset
84          resetButton.addActionListener(e -> reset());
85      }
```

7. **Method setArrayFromInput()** berfungsi mengambil input array dari user, memprosesnya, dan menampilkannya secara visual. Modifier “private” artinya method ini hanya bisa dipanggil di dalam kelas ini. “void” mathod tidak mengembalikan nilai.

```
87          //Function setArray
88          private void setArrayFromInput() {
```

Mengambil input teks input field, ‘.trim()’ berfugnsi menghapus spasi di awal/akhir. Jika input kosong maka method berhenti.

```
181         if (text.isEmpty()) {
182             JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!",
183                     "Error", JOptionPane.ERROR_MESSAGE);
184             return;
185         }
```

Memeriksa panjang dari array minimal 2 karakter, lalu return dengan menghentikan method.

```
186         if (parts.length < 2) {
187             JOptionPane.showMessageDialog(this, "Masukkan setidaknya dua angka!",
188                     "Error", JOptionPane.ERROR_MESSAGE);
189             return;
190         }
```

Memisahkan input menjadi array string. ‘split(“,”)’ berfugnsi memisahkan input berdasarkan koma. ‘array = new int[parts.length];’ Membuat array integer sebanyak jumlah angka yang diinput user.

```
91          String[] parts = text.split(",");
92          array = new int[parts.length];
```

Try{} and catch {} berfungsi unutk menangani kesalahn/eror saat program berjalan, agar program tidak langsung berhenti/crash melainkan menampilkna pesan eror ke user.

Konversi String ke Integer dari array “parts” menggunakan for loop yang blok kode nya akan mengubah tiap elemen hasil looping menjadi integer ‘Integer.parseInt()’. ‘trim()’ menghapus spasi jika ada.

‘catch (NumberFormatException e)’ jika eror terjadi, maka program tidak akan langusng crash, melainkan tampil popup peringatan ke user menggunakan ‘JOptionPane.showMessageDialog’, lalu menghentikan ekskusi method dengan ‘return;’ agar tidak lanjut ke bagian lain.

Block try-catch memastikan input tidak valid yang menyebabkan program eror, dengan memberitahu tahu user dengan popup message, proses parsing hanya dilakukan jika semua input valid. Membuat program lebih user-frienly dan dapat menghandle eror dengan baik.

```
93          try {
94              for (int k = 0; k < parts.length; k++) {
95                  array[k] = Integer.parseInt(parts[k].trim()); }
96          } catch (NumberFormatException e) {
97              JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan "
98              + "dengan koma!", "Eror", JOptionPane.ERROR_MESSAGE);
99              return;
100         }
```

Inisialisasi object array dengan panjang array. Menghilangkan semua tampilan dari array sebelumnya.

```
201             labelArray = new JLabel[array.length];
202             panelArray.removeAll();
```

Mengaktifkan button yaitu tombol unutk proses sorting satu per satu setelah input berhasil. StepArea dikosongkan agar siap menampilkan log sorting baru.

```
104             stepButton.setEnabled (true);
105             stepArea.setText("");
```

Reset tampilan panel array. Menghapus semau komponen lama dari panelArray, siapkan array Jlabel untuk menampilkan elemen array ke user.

```
106             panelArray.removeAll();
107             labelArray = new JLabel [array.length];
```

Tampilkan elemen array ke GUI, loop setiap elemen array, buat Jlabel berisi angka array.

```
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].setBorder(BorderFactory.createLineBorder(Color.BLACK));
112             labelArray[k].setPreferredSize(new Dimension(50,50));
113             labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
114             panelArray.add(labelArray[k]);
115         }
```

Refresh tampilan panel. revalidate(), Memberitahu layout manager untuk memperbarui posisi/ukuran komponen. repaint(), Memaksa komponen untuk di-redraw ulang, agar perubahan terlihat.

```
115             }
116             panelArray.revalidate();
117             panelArray.repaint();
118         }
```

Penjelasan method setArrayFromInput() : Mengambil input angka dari user dalam bentuk string, mengonversinya ke array integer (parsing), Menampilkan array tersebut sebagai deretan kotak angka di tampilan GUI, mengatur ulang kondisi agar siap untuk sorting visual secara bertahap. Sebagai langkah awal sebleum proses sorting dilakukan.

8. **Method performStep(),** berfungsi melakukan satu langkah sorting menggunakan algoritma Quick Sort, lalu menampilkan hasil perbandingan dan/atau pertukaran elemen dalam array secara visual di layar (GUI).
   Menghapus/mengembalikan warna highlight elemen array sebelumnya agar tampilan visualnya bersih untuk langkah selanjutnya.

```
129         resetHighLights();
```

Pengecekan status sorting, jika proses sorting tidak aktif (!sorting) atau interai sudah mencapi akhir array (artinya sudah selesei), maka sorting boolean di set

false, stepButton dinonaktifkan, Menampilkan pesan sorting telah selesei lalu keluar dari fungsi(return).

```
130         if (!sorting || gap == 0) {
131             stepArea.append("Shell sort selesai.\n");
132             stepButton.setEnabled(false);
133             JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
134             stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
135             return;
136         }
```

Mengecek apakah indeks i masih dalam batas array untuk diproses.

```
137         if (i < array.length) {
```

Jika belum memulai proses swap (penempatan), maka Ambil elemen array[i] dan simpan di temp untuk disisipkan, lalu Atur j = i dan tandai bahwa proses penyisipan (swapping) sedang dilakukan.

```
138             if (!isSwapping) {
139                 temp = array[i];
140                 j = i;
141                 isSwapping = true;
142             }
```

Jika j cukup besar untuk digeser dan elemen sebelumnya (array[j - gap]) lebih besar dari temp, maka Geser elemen tersebut ke posisi j, Warnai elemen sebagai visualisasi (hijau dan cyan), Panggil updateLabels() untuk memperbarui tampilan GUI, Catat langkah dengan logStep(), dan Geser j ke kiri dengan jarak gap untuk melanjutkan proses.

```
143             if (j >= gap && array[j - gap] > temp) {
144                 array[j] = array[j - gap]; // geser ke kanan
145                 labelArray[j].setBackground(Color.GREEN);
146                 labelArray[j - gap].setBackground(Color.CYAN);
147                 updateLabels();
148                 logStep("Geser elemen " + array[j] + " ke kanan");
149                 j -= gap;
150                 return;
```

Jika tidak perlu menggeser lagi, maka letakkan nilai ‘temp’ di posisi saat ini (j), update tampilan GUI dan log langkah, dan naikkan indeks i ke elemen berikutnya dan reset status isSwapping.

```
151             } else {
152                 array[j] = temp; // letakkan nilai temp
153                 updateLabels();
154                 logStep("Tempatkan " + temp + " di posisi " + j);
155                 i++;
156                 isSwapping = false;
157             }
```

Penjelasan **Method performStep() :** bekerja dengan cara memproses satu elemen array per langkah, menyisipkan (insertion) elemen tersebut ke posisi yang sesuai dalam sublist yang ditentukan oleh gap, lalu secara bertahap mengurangi gap hingga 0. Proses ini divisualisasikan dan dicatat langkah demi langkah.

9. **Method logStep(String message),** berfungsi digunakan untuk menampilkan log proses sorting di area teks (stepArea) setiap kali pengguna melakukan satu langkah sorting (step-by-step)

```
270●    private void logStep(String message) {
271         stepArea.append("Langkah " + stepCount + ": " + message + "\n");
272         stepArea.setCaretPosition(stepArea.getDocument().getLength());
273         stepCount++;
274     }
275 }
```

10. **Method highlightPivot(int index)**, berfungsi untuk menyorot pivot.

```
166●    private void highlightPivot(int index) {
167         labelArray[index].setBackground(Color.ORANGE);
168     }
```

11. **Method highlightCompare(int jIndex, int pivotIndex)**, berfungsi unutk menyorot elemen yang dibandingkan.

```
171●    private void highlightCompare(int jIndex, int pivotIndex) {
172         labelArray[jIndex].setBackground(Color.CYAN);
173         if(labelArray[pivotIndex].getBackground() != Color.LIGHT_GRAY) {
174             labelArray[pivotIndex].setBackground(Color.ORANGE);
175         }
176     }
```

12. **Method resetHighlights(),** berfungsi unutk **menghapus warna highlight sebelumnya**, agar proses highlight baru (seperti di highlightMinIndex()) tidak menumpuk atau membuat tampilan membingungkan. dipanggil setiap kali sebelum memberi highlight baru, supaya hanya satu elemen yang diberi warna khusus.

```
183●    private void resetHighLights() {
184         if (labelArray == null)
185         return;
186         for (JLabel label : labelArray) {
187             label.setBackground(Color.WHITE);
188         }
189     }
```

13. **Method updateLabels(),** digunakan setiap kali ada perubahan pada isi array, seperti saat proses penukaran (swap) dalam sorting. Tujuannya untuk menyinkronkan tampilan GUI dengan data array yang terbaru.

```
189●    private void updateLabels() {
190         for(int k = 0; k < array.length; k++) {
191             labelArray[k].setText(String.valueOf(array[k]));
192         }
193     }
```

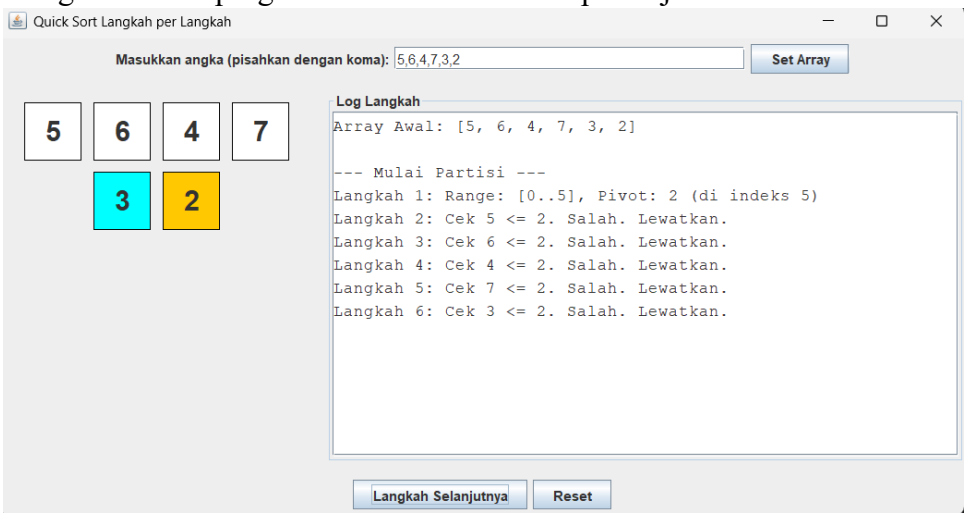
14. **Method swap(int a, int b),** berfungsi unutk menukar elemen array.

```
247●    private void swap(int a, int b) {
248         int temp = array[a];
249         array[a] = array[b];
250         array[b] = temp;
251     }
```

15. **Method reset(),** digunakan untuk **mengembalikan tampilan dan variabel program ke kondisi awal**, seolah-olah pengguna belum memasukkan data apa pun. Biasanya dipakai setelah proses sorting selesai atau saat pengguna ingin mengulang.

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

16. Output program : Alur kerja dari program ini yaitu Setiap langkah dieksekusi secara bertahap mulai dari menentukan range dan pivot (disorot oranye), lalu membandingkan elemen satu per satu (yang sedang dibandingkan disorot cyan), menukar posisi elemen jika diperlukan, dan terakhir memindahkan pivot ke posisi akhirnya. Setelah partisi selesai, program membagi array menjadi sub-array kiri dan kanan pivot, lalu mengulangi proses untuk masing-masing sub-array menggunakan stack. Setiap aksi dicatat secara detail di panel log sebelah kanan, termasuk perbandingan nilai dan pertukaran elemen. Proses berlanjut hingga seluruh array terurut, yang ditandai dengan semua label berwarna abu-abu dan notifikasi penyelesaian. Tombol "Reset" tersedia untuk mengembalikan program ke keadaan awal kapan saja.



f. Program MergeSortGUI

1. Klik kanan pada package pekan8, new, other, windowBuilder lalu Jframe dan beri nama file GUI "**MergeSortGUI**".
2. Import package java util unutk menipulasi array dan struktur data, java awt unutk komponen GUI dasar berupa layout, java swing unutk membuat komponen GUI seperti Jbutton dan Jlabel, dan java eventQueue untuk menjalankan GUI secara thread-safe (agar aman dan konsisten).

```
1 package pekan8;
2 import java.util.*;
3 import java.awt.*;
4 import javax.swing.*;
```

3. Deklarasi class "MergeSortGUI" dengan memnggunakna JFFrame (jendela utama aplikasi Swing).

```
21 public class QuickSortGUI extends JFrame {
```

4. Deklarasi variabel. Menyimpan data array, elemen visual GUI, tombol interaktif GUI, dan variabel kontrol sorting. Variabel I sebagai index utama iterasi luar pada algortima sorting, variabel j digunakan unutk inner loop saat membandingkan dan menukar elemen. Boolean sorting sebagai penanda proses sorting sedang berlangsung atau tidak. StepCount unutk menghitung proses sorting dilakukan. minIndex khusu unutk selection sort, menyimpan index dari nilai terkecil yang ditemukan selama satu putara iterasi.

```
 9      private static final long serialVersionUID = 1L;
10      private int[] array;
11      private JLabel[] labelArray;
12      private JButton stepButton, resetButton, setButton;
13      private JTextField inputField;
14      private JPanel panelArray;
15      private JTextArea stepArea;
```

5. **Main method**. Menjalankan GUI. EventQueue unutk menjalakan kode GUI, menjalakannya di thread yang benar. new SelectionSortGUI(i) membuat object baru dari kelas SelectionSortGUI turunan Jframe. frame.setVisible(true); untuk menampilkan GUI ke display

```
25      public static void main(String[] args) {
26          EventQueue.invokeLater(new Runnable() {
27              public void run() {
28                  try {
29                      ShellSortGUI frame = new ShellSortGUI();
30                      frame.setVisible(true);
31                  } catch (Exception e) {
32                      e.printStackTrace();
33                  }
34              }
35          });
36      }
```

6. **Konstruktor GUI,** menentukan model/tampilan dari GUInya berupa ukuran, layout dan semua komponen GUI.

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

Panel input, terdiri dari inputField unutk tempat user mengetik angka dan setButton tombol yang memproses input dna menampilkan array.

```
45          //Panel input
46          JPanel inputPanel = new JPanel (new FlowLayout());
47          inputField = new JTextField(30);
48          setButton = new JButton("Set Array");
49          inputPanel.add(new JLabel("Masukan angka (pisahkan dengan koma):"));
50          inputPanel.add(inputField);
51          inputPanel.add(setButton);
```

Panel array visual digunakna unutk menampilkan elemen array sebagai label(Jlabel) secara horizontal.

```
53          //Panel array visual
54          panelArray = new JPanel();
55          panelArray.setLayout(new FlowLayout());
```

Panel tombol kontrol terdiri dari tombol step dan reset button.

```
57          //Panel kontrol
58          JPanel controlPanel = new JPanel();
59          stepButton = new JButton("Langkah selanjutnya");
60          resetButton = new JButton ("Reset");
61          stepButton.setEnabled(false);
62          controlPanel.add(stepButton);
63          controlPanel.add(resetButton);
```

Area teks yang menampilkan log langkah-langkah sorting dari tiap penyeleseian array.

```
65          //Area teks unutk log langkah-langkah
66          stepArea = new JTextArea(8, 60);
67          stepArea.setEditable(false);
68          stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
69          JScrollPane scrollPane = new JScrollPane(stepArea);
```

Menambahkan panel ke frame, fungsi add() unutk menempatkan panel ke dalam frame menggunakan layout manager “Border layout”, yang membagi frame menjadi 5 bagain utama yaitu north,south,east, west dan center.

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

Menambahkan event listener ke tombol, yaitu event handler ke setButton, yang ketika ditekan akan menjalankan method.

```
77          //Event Set Array
78          setButton.addActionListener(e -> setArrayFromInput());
79
80          //Event Langkah Selanjutnya
81          stepButton.addActionListener(e -> performStep());
82
83          //Event reset
84          resetButton.addActionListener(e -> reset());
85      }
```

7. **Method setArrayFromInput()** berfungsi mengambil input array dari user, memprosesnya, dan menampilkannya secara visual. Modifier “private” artinya method ini hanya bisa dipanggil di dalam kelas ini. “void” mathod tidak mengembalikan nilai.

```
87          //Function setArray
88          private void setArrayFromInput() {
```

Mengambil input teks input field, ‘.trim()’ berfugnsi menghapus spasi di awal/akhir. Jika input kosong maka method berhenti.

```
181         if (text.isEmpty()) {
182             JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!",
183                     "Error", JOptionPane.ERROR_MESSAGE);
184             return;
185         }
```

Memeriksa panjang dari array minimal 2 karakter, lalu return dengan menghentikan method.

```
186         if (parts.length < 2) {
187             JOptionPane.showMessageDialog(this, "Masukkan setidaknya dua angka!",
188                     "Error", JOptionPane.ERROR_MESSAGE);
189             return;
190         }
```

Memisahkan input menjadi array string. ‘split(“,”)’ berfugnsi memisahkan input berdasarkan koma. ‘array = new int[parts.length];’ Membuat array integer sebanyak jumlah angka yang diinput user.

```
91          String[] parts = text.split(",");
92          array = new int[parts.length];
```

Try{} and catch {} berfungsi unutk menangani kesalahn/eror saat program berjalan, agar program tidak langsung berhenti/crash melainkan menampilkna pesan eror ke user.

Konversi String ke Integer dari array “parts” menggunakan for loop yang blok kode nya akan mengubah tiap elemen hasil looping menjadi integer ‘Integer.parseInt()’. ‘trim()’ menghapus spasi jika ada.

‘catch (NumberFormatException e)’ jika eror terjadi, maka program tidak akan langusng crash, melainkan tampil popup peringatan ke user menggunakan ‘JOptionPane.showMessageDialog’, lalu menghentikan ekskusi method dengan ‘return;’ agar tidak lanjut ke bagian lain.

Block try-catch memastikan input tidak valid yang menyebabkan program eror, dengan memberitahu tahu user dengan popup message, proses parsing hanya dilakukan jika semua input valid. Membuat program lebih user-frienly dan dapat menghandle eror dengan baik.

```
93          try {
94              for (int k = 0; k < parts.length; k++) {
95                  array[k] = Integer.parseInt(parts[k].trim()); }
96          } catch (NumberFormatException e) {
97              JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan "
98              + "dengan koma!", "Eror", JOptionPane.ERROR_MESSAGE);
99              return;
100         }
```

Inisialisasi object array dengan panjang array. Menghilangkan semua tampilan dari array sebelumnya.

```
201         labelArray = new JLabel[array.length];
202         panelArray.removeAll();
```

Mengaktifkan button yaitu tombol unutk proses sorting satu per satu setelah input berhasil. StepArea dikosongkan agar siap menampilkan log sorting baru.

```
104             stepButton.setEnabled (true);
105             stepArea.setText("");
```

Reset tampilan panel array. Menghapus semau komponen lama dari panelArray, siapkan array Jlabel untuk menampilkan elemen array ke user.

```
106             panelArray.removeAll();
107             labelArray = new JLabel [array.length];
```

Tampilkan elemen array ke GUI, loop setiap elemen array, buat Jlabel berisi angka array.

```
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].setBorder(BorderFactory.createLineBorder(Color.BLACK));
112             labelArray[k].setPreferredSize(new Dimension(50,50));
113             labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
114             panelArray.add(labelArray[k]);
115         }
```

Refresh tampilan panel. revalidate(), Memberitahu layout manager untuk memperbarui posisi/ukuran komponen. repaint(), Memaksa komponen untuk di-redraw ulang, agar perubahan terlihat.

```
115         }
116         panelArray.revalidate();
117         panelArray.repaint();
118     }
```

Penjelasan method setArrayFromInput() : Mengambil input angka dari user dalam bentuk string, mengonversinya ke array integer (parsing), Menampilkan array tersebut sebagai deretan kotak angka di tampilan GUI, mengatur ulang kondisi agar siap untuk sorting visual secara bertahap. Sebagai langkah awal sebleum proses sorting dilakukan.

8. **Method performStep(),** berfungsi melakukan satu langkah sorting menggunakan algoritma Quick Sort, lalu menampilkan hasil perbandingan dan/atau pertukaran elemen dalam array secara visual di layar (GUI).
   Menghapus/mengembalikan warna highlight elemen array sebelumnya agar tampilan visualnya bersih untuk langkah selanjutnya.

```
129         resetHighLights();
```

Pengecekan status sorting, jika proses sorting tidak aktif (!sorting) atau interai sudah mencapi akhir array (artinya sudah selesei), maka sorting boolean di set false, stepButton dinonaktifkan, Menampilkan pesan sorting telah selesei lalu keluar dari fungsi(return).

```
130     if (!sorting || gap == 0) {
131         stepArea.append("Shell sort selesai.\n");
132         stepButton.setEnabled(false);
133         JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
134         stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
135         return;
136     }
```

Mengecek apakah indeks i masih dalam batas array untuk diproses.

```
137         if (i < array.length) {
```

Jika j cukup besar untuk digeser dan elemen sebelumnya (array[j - gap]) lebih besar dari temp, maka Geser elemen tersebut ke posisi j, Warnai elemen sebagai visualisasi (hijau dan cyan), Panggil updateLabels() untuk memperbarui tampilan GUI, Catat langkah dengan logStep(), dan Geser j ke kiri dengan jarak gap untuk melanjutkan proses.

```
143             if (j >= gap && array[j - gap] > temp) {
144                 array[j] = array[j - gap]; // geser ke kanan
145                 labelArray[j].setBackground(Color.GREEN);
146                 labelArray[j - gap].setBackground(Color.CYAN);
147                 updateLabels();
148                 logStep("Geser elemen " + array[j] + " ke kanan");
149                 j -= gap;
150                 return;
```

Jika tidak perlu menggeser lagi, maka letakkan nilai ‘temp’ di posisi saat ini (j), update tampilan GUI dan log langkah, dan naikkan indeks i ke elemen berikutnya dan reset status isSwapping.

```
151             } else {
152                 array[j] = temp; // letakkan nilai temp
153                 updateLabels();
154                 logStep("Tempatkan " + temp + " di posisi " + j);
155                 i++;
156                 isSwapping = false;
157             }
```

Penjelasan **Method performStep() :** bekerja dengan cara memproses satu elemen array per langkah, menyisipkan (insertion) elemen tersebut ke posisi yang sesuai dalam sublist yang ditentukan oleh gap, lalu secara bertahap mengurangi gap hingga 0. Proses ini divisualisasikan dan dicatat langkah demi langkah.

9. **Method logStep(String message),** berfungsi digunakan untuk menampilkan log proses sorting di area teks (stepArea) setiap kali pengguna melakukan satu langkah sorting (step-by-step)

```
270     private void logStep(String message) {
271         stepArea.append("Langkah " + stepCount + ": " + message + "\n");
272         stepArea.setCaretPosition(stepArea.getDocument().getLength());
273         stepCount++;
274     }
275 }
```

10. **Method highlightPivot(int index)**, berfungsi untuk menyorot pivot.

```
166     private void highlightPivot(int index) {
167         labelArray[index].setBackground(Color.ORANGE);
168     }
```

11. **Method highlightCompare(int jIndex, int pivotIndex**), berfungsi unutk menyorot elemen yang dibandingkan.

```
171     private void highlightCompare(int jIndex, int pivotIndex) {
172         labelArray[jIndex].setBackground(Color.CYAN);
173         if(labelArray[pivotIndex].getBackground() != Color.LIGHT_GRAY) {
174             labelArray[pivotIndex].setBackground(Color.ORANGE);
175         }
176     }
```

12. **Method resetHighlights(),** berfungsi unutk **menghapus warna highlight sebelumnya**, agar proses highlight baru (seperti di highlightMinIndex()) tidak menumpuk atau membuat tampilan membingungkan. dipanggil setiap kali sebelum memberi highlight baru, supaya hanya satu elemen yang diberi warna khusus.

```
183     private void resetHighLights() {
184         if (labelArray == null)
185         return;
186         for (JLabel label : labelArray) {
187             label.setBackground(Color.WHITE);
188         }
189     }
```

13. **Method updateLabels(),** digunakan setiap kali ada perubahan pada isi array, seperti saat proses penukaran (swap) dalam sorting. Tujuannya untuk menyinkronkan tampilan GUI dengan data array yang terbaru.

```
189     private void updateLabels() {
190         for(int k = 0; k < array.length; k++) {
191             labelArray[k].setText(String.valueOf(array[k]));
192         }
193     }
```

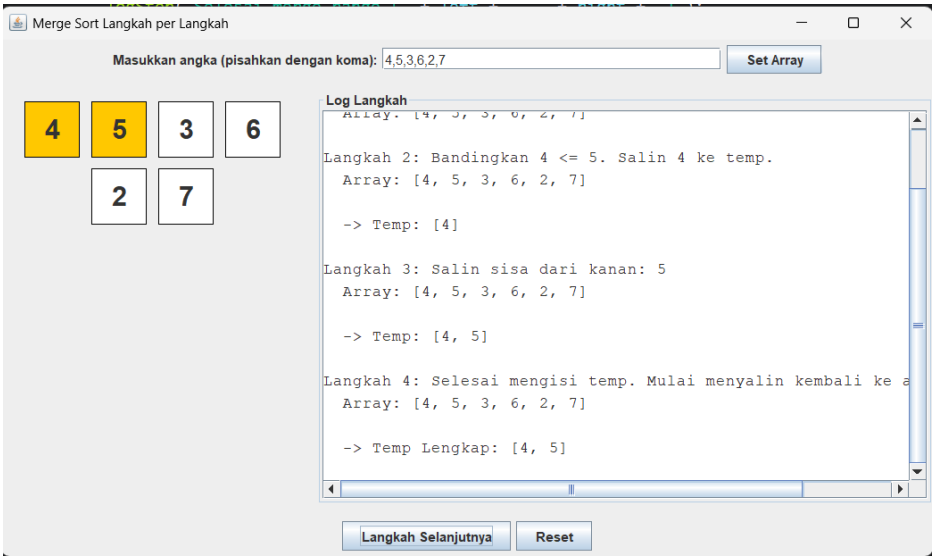
14. **Method swap(int a, int b),** berfungsi unutk menukar elemen array.

```
247     private void swap(int a, int b) {
248         int temp = array[a];
249         array[a] = array[b];
250         array[b] = temp;
251     }
```

15. **Method reset(),** digunakan untuk **mengembalikan tampilan dan variabel program ke kondisi awal**, seolah-olah pengguna belum memasukkan data apa pun. Biasanya dipakai setelah proses sorting selesai atau saat pengguna ingin mengulang.

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

16. Output program : Alur kerja program dimulai dari pembacaan input angka, lalu array dipecah menjadi bagian-bagian kecil (rekursif), kemudian setiap bagian tersebut digabung kembali dalam urutan yang terurut (merge). Proses merge dilakukan secara bertahap dan visual, di mana label yang mewakili elemen array akan berubah posisi dan warna sesuai proses yang terjadi. Setiap aksi dicatat dalam log teks agar pengguna memahami langkah per langkah proses merge sort. Karena program ini berbasis GUI, pengguna dapat mengamati dengan jelas bagaimana algoritma Merge Sort bekerja secara real-time dan interaktif.



D. Kesimpulan

Setelah melakukan praktikum disimpulkan implementasi algoritma sorting Selection Sort dan Insertion Sort dalam pemrograman Java untuk memahami proses pengurutan data secara lebih lebih visual dan interaktif, terutama dengan penggunaan GUI (Graphical User Interface). Bubble Sort cocok digunakan untuk pembelajaran dasar namun tidak efisien untuk big data. Shell Sort sebagai peningkatan dari Insertion Sort dan cocok digunakan untuk dataset. Quick Sort unggul dari sisi kecepatan dan banyak digunakan dalam pustaka bahasa pemrograman. Merge Sort stabil dan cocok untuk data besar atau yang membutuhkan konsistensi performa. Dapat disimpulkan bahwa pemilihan algoritma sorting tergantung pada kebutuhan spesifik, ukuran data, dan batasan memori.