LAPORAN PRAKTIKUM STRUKTUR DATA

APLIKASI GUI SORTING (INSERTION & SELECTION)

OLEH :

JOVANTRI IMMANUEL GULO

NIM 2411532014

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU :

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

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, 03 JUNI 2025

A. Pendahuluan

Sorting atau pengurutan data adalah salah satu konsep fundamental dalam struktur data dan algoritma. Pada praktikum ini, implementasi data yang akan dilakukan adalah implementasi data untuk insertion sort dan selection sort yang masing-masingnya memiliki karakteristiknya masing-masing dalam melakukan pengurutan. Insertion sort bekerja dengan cara menyisipkan elemen ke dalam posisi yang tepat dalam daftar yang telah diurutkan, sedangkan selection sort bekerja dengan cara memilih elemen terkecil dari daftar yang belum diurutkan dan menempatkannya pada posisi yang sesuai.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah

1. Mampu memahami cara mengimplementasikan insertion sort.
2. Mampu memahami cara mengimplementasikan selection sort.

C. Langkah – langkah Pengerjaan

Berikut adalah langkah-langkah dalam pengerjaan praktikum kali ini:

1. Buat package baru dengan nama **pekan7**
2. Klik kanan pada package dan klik new, kemudian klik **Other,** kemudian klik **Window Builder**, klik **Swing Designer**, dan klik **JFrame**, kemudian buat nama file barunya dengan nama **InsertionSortGUI**
3. Import package, modul yang dibutuhkan, kemudian buat juga public class untuk file InsertionSortGUI dengan extend dari JFrame.

```
1  package pekan7;
2
3  import java.awt.BorderLayout;
4  import java.awt.Color;
5  import java.awt.Dimension;
6  import java.awt.EventQueue;
7  import java.awt.FlowLayout;
8  import java.awt.Font;
9
10 import javax.swing.BorderFactory;
11 import javax.swing.JButton;
12 import javax.swing.JFrame;
13 import javax.swing.JLabel;
14 import javax.swing.JOptionPane;
15 import javax.swing.JPanel;
16 import javax.swing.JScrollPane;
17 import javax.swing.JTextArea;
18 import javax.swing.JTextField;
19 import javax.swing.SwingConstants;
20
21
22 public class InsertionSortGUI extends JFrame {
```

4. Kemudian, deklarasikan variabel yang diperlukan dalam project GUI ini.

```
24      private static final long serialVersionUID = 1L;
25      private int[] array;
26      private JLabel[] labelArray;
27      private JButton stepButton, resetButton, setButton;
28      private JTextField inputField;
29      private JPanel panelArray;
30      private JTextArea stepArea;
31
32      private int i = 1, j; // i adalah indeks elemen yang akan disisipkan
33      private boolean sorting = false;
34      private int stepCount = 1;
35
36      /**
37       * Launch the application.
38       */
```

5. Pada class main
    a. Menggunakan deklarasi EventQueue.invokeLater untuk menjalankan GUI pada thread khusus yang disebut dengan EDT (Event Dispatch Thread)
    b. public void run() digunakan untuk menjalankan method run pada EventQueue
    c. Membuat objek baru dari class InsertionSortGUI dengan nama **frame**, setVisible ke true supaya GUI dapat ditampilkan ke layar

```
39      public static void main(String[] args) {
40          EventQueue.invokeLater(new Runnable() {
41              public void run() {
42                  try {
43                      InsertionSortGUI frame = new InsertionSortGUI();
44                      frame.setVisible(true);
45                  } catch (Exception e) {
46                      e.printStackTrace();
47                  }
48              }
49          });
50      }
```

6. Pada konstruktor kelas InsertionSortGUI
    a. Menetapkan judul (title), ukuran, dan posisi jendela, serta mengatur layout utamanya sebagai **BorderLayout**

b. Mendeklarasikan inputField dan button setArray supaya dapat melakukan input array, panelnya akan ditaruh pada bagian north (utara)
c. Panel untuk menampilkan visualisasi array, yang akan diletakkan pada bagian tengah (center)
d. Panel kontrol untuk membuat tombol langkah selanjutnya sehingga langkah dari insertion sort akan maju selangkah dan juga menyediakan tombol reset untuk mengatur ulang semuanya, panel kontrol ini akan terletak pada bagian south (selatan)
e. Bagian text area output, stepArea akan menampilkan penjelasan text tiap langkah, scrollPane digunakan supaya window atau layar dapat discroll, bagian ini akan terletak pada bagian south (timur)
f. Menambahkan semua komponen pada frame
g. Membuat event listener untuk setButton, stepButton, dan resetButton

```
55    public InsertionSortGUI() {
56        setTitle("Insertion Sort Langkah per Langkah");
57        setSize(750, 400);
58        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
59        setLocationRelativeTo(null);
60        setLayout(new BorderLayout());
61
62        JPanel inputPanel = new JPanel(new FlowLayout());
63        inputField = new JTextField(30);
64        setButton = new JButton("Set Array");
65        inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma) :"));
66        inputPanel.add(inputField);
67        inputPanel.add(setButton);
68
69        panelArray = new JPanel();
70        panelArray.setLayout(new FlowLayout());
71
72        JPanel controlPanel = new JPanel();
73        stepButton = new JButton("Langkah Selanjutnya");
74        resetButton = new JButton("Reset");
75        stepButton.setEnabled(false);
76        controlPanel.add(stepButton);
77        controlPanel.add(resetButton);
78
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        add(inputPanel, BorderLayout.NORTH);
85        add(panelArray, BorderLayout.CENTER);
86        add(controlPanel, BorderLayout.SOUTH);
87        add(scrollPane, BorderLayout.EAST);
88
89        // Event set Array
90        setButton.addActionListener(e -> setArrayFromInput());
91
92        // Event langkah selanjutnya
93        stepButton.addActionListener(e -> performStep());
94
95        //Event reset
96        resetButton.addActionListener(e -> reset());
97    }
```

7. Pada method performStep
    a. Mendeklarasikan key sebagai elemen yang nantinya akan disisipkan ke posisi yang benar dan j digunakan untuk membandingkan elemen sebelumnya pada array yang ada

b. Terdapat bagian logging dan gambaran awal : append(key) berguna untuk menyimpan informasi bahwa key dipilih dan nantinya akan dibandingkan, serta pengecualian pada if (labelArray != null && i < labelArray.length) untuk memberi warna kuning pada elemen pada array yang sedang menjadi key
c. Original dideklarasikan sebagai j untuk menyimpan posisi awal dari key sehingga nantinya dapat kita bandingkan apakah posisinya berubah atau tidak. Boolean shifted dideklarasikan sebagai false untuk menandai apakah terjadi pergeseran elemen atau tidak
d. Pada bagian while (j >= 0 && array[j] > key) merupakan bagian operasi atau bagian inti dari insertion sort yaitu selama elemennya lebih besar dari key, maka elemen akan digeser ke kanan, serta pada bagian ini juga, akan disisipkan ruang untuk key. Pada bagian labelArray[j + 1].setBackground(Color.ORANGE) akan digunakan untuk memvisualisasi elemen yang digeser dengan menggunakan warna oranye\
e. Terdapat bagian untuk menyisipkan key dan logging : array[j + 1] = key akan menaruh posisi key pada posisi yang tepat setelah sesi geser-geser selesai dan pada bagian if(shifted) mendefinisikan apabila tidak terjadi pergeseran, maka key sudah berada pada posisi yang tepat
f. Pada method updateLabels() digunakan untuk memperbarui tampilan pada GUI berdasarkan data array yang baru. Untuk setOpaque dan setBackground pada labelArray, ini menandakan untuk memberi warna abu-abu muda pada elemen array yang sudah terurut
g. Pada bagian stepLog.append akan menampilkan hasil array sementara setelah langkah tertentu
h. Untuk i++ dan stepCount++ berguna untuk melanjutkan proses ke elemen atau langkah berikutnya dan juga menambah jumlah langkah
i. Pada bagian if (i >= array.length) menunjukkan bahwa apabila array telah disortir dengan benar, maka warna elemen di akhir akan menjadi warna hijau dan akan muncul text “Sorting Selesai!”

```
99    private void performStep() {
100       if(i < array.length && sorting) {
101           // PERBAIKAN: Ambil elemen pada indeks i sebagai key
102           int key = array[i];
103           j = i -1;
104
105           StringBuilder stepLog = new StringBuilder();
106           // Menambahkan informasi elemen yang sedang di-highlight atau menjadi 'key'
107           stepLog.append("Langkah ").append(stepCount).append(": Key = ").append(key).append(" (elemen ke-").append(i + 1).append(")\n");
108           stepLog.append("   Membandingkan ").append(key).append(" dengan elemen di sebelah kirinya.\n");
109
110
111           // Visualisasi: Highlight elemen 'key'
112           if (labelArray != null && i < labelArray.length) {
113               // Reset warna label sebelumnya
114               for (JLabel lbl : labelArray) {
115                   lbl.setOpaque(false); // Pastikan bisa diwarnai
116                   lbl.setBackground(null); // Warna default
117                   lbl.setForeground(Color.BLACK); // Warna teks default
118               }
119               labelArray[i].setOpaque(true);
120               labelArray[i].setBackground(Color.YELLOW); // Warna highlight untuk key
121           }
122
123           int originalJ = j; // Simpan nilai j awal untuk logging
124           boolean shifted = false;
125
126           while(j >= 0 && array[j] > key) {
127               stepLog.append("   → Geser ").append(array[j]).append(" (di posisi ").append(j+1).append(") ke kanan (posisi ").append(j+2).append(")\n");
128               array[j + 1] = array[j];
129               // Visualisasi pergeseran
130               if (labelArray != null && (j + 1) < labelArray.length) {
131                   labelArray[j + 1].setOpaque(true);
132                   labelArray[j + 1].setBackground(Color.ORANGE); // Warna untuk elemen yang digeser
133               }
134               j--;
135               shifted = true;
136           }
137           array[j + 1] = key;
138
139           if (shifted) {
140               stepLog.append("   Sisipkan ").append(key).append(" pada posisi ").append(j + 2).append(".\n");
141           } else if (originalJ >0) {
142                stepLog.append("   ").append(key).append(" sudah pada posisi yang benar atau lebih kecil dari ").append(array[originalJ]).append(".\n");
143           } else {
144                stepLog.append("   ").append(key).append(" adalah elemen terkecil sejauh ini, ditempatkan di awal.\n");
145           }
146
147           updateLabels(); // Update semua label setelah satu langkah selesai
148
149           // Setelah updateLabels, kembalikan warna elemen yang sudah terurut
150           for(int k_sorted = 0; k_sorted <= i; k_sorted++){
151               if(labelArray != null && k_sorted < labelArray.length){
152                   labelArray[k_sorted].setOpaque(true);
153                   labelArray[k_sorted].setBackground(Color.LIGHT_GRAY); // Warna untuk bagian yang sudah terurut
154               }
155           }
156
157
158           stepLog.append("Hasil sementara: ").append(arrayToString(array)).append("\n\n");
159           stepArea.append(stepLog.toString());
160
161           i++;
162           stepCount++;
163
164           if(i == array.length) {
165               sorting = false;
166               stepButton.setEnabled(false);
167               // Reset warna semua label ke final sorted color
168               for (JLabel lbl : labelArray) {
169                   lbl.setOpaque(true);
170                   lbl.setBackground(Color.GREEN); // Warna akhir setelah sorting selesai
171                   lbl.setForeground(Color.BLACK);
172               }
173               JOptionPane.showMessageDialog(this, "Sorting Selesai!");
174           }
175       }
176   }
```

8. Pada method setArrayFromInput
    a. Pada bagian if(text.isEmpty()) menunjukkan bahwa apabila text yang diinputkan ke dalam array adalah text kosong, maka akan muncul pesan “Input tidak boleh kosong!” dan maka akan memunculkan error
    b. Bagian array string : String[] parts = text.split(",") akan memisahkan tiap inputan pengguna untuk tiap koma dan bagian if (parts.length == 0 || (parts.length == 1 && parts[0].trim().isEmpty())) akan memeriksa apakah input hanya koma saja tanpa angka, jika iya akan memunculkan pesan error
    c. Mengubah string menjadi integer dengan Integer.parseInt, sehingga dari yang tadinya “6” pada input array, menjadi 6
    d. Deklarasi i = 1 menandakan bahwa kita akan memulai sorting dari indeks pertama, deklarasi stepCount = 1 menandakan bahwa dimulai dari langkah pertama, deklarasi sorting = true. Semuanya menandakan bahwa pada bagian ini proses sorting dipersiapkan
    e. Setiap angka ditampilkan sebagai label GUI dengan perulangan for dan panelArray.add (sekalian diset ukuran teksnya, warna, garis border, ukuran tetap, dan peletakan teks di tengah atau center)
    f. Menandai elemen pertama sebagai elemen yang sudah terurut sementara, dikarenakan dalam insertionSort, elemen pertama tidak dibandingkan,

melainkan sudah dianggap telah terurut, ditandai dengan background berwarna abu-abu

g. Method revalidate dan repaint digunakan untuk update tampilan pada GUI sehingga semua elemen dapat tampil

```
178    private void setArrayFromInput() {
179        String text = inputField.getText().trim();
180        if(text.isEmpty()){
181            JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!", "Error", JOptionPane.ERROR_MESSAGE);
182            return;
183        }
184        String[] parts = text.split(",");
185        if (parts.length == 0 || (parts.length == 1 && parts[0].trim().isEmpty())) {
186             JOptionPane.showMessageDialog(this, "Masukkan setidaknya satu angka!", "Error", JOptionPane.ERROR_MESSAGE);
187            return;
188        }
189        array = new int[parts.length];
190
191        try {
192            for(int k = 0; k < parts.length ; k++) {
193                array[k] = Integer.parseInt(parts[k].trim());
194            }
195        } catch (NumberFormatException e) {
196            JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
197            return;
198        }
199        i = 1; // Insertion sort dimulai dari elemen kedua (indeks 1)
200        stepCount = 1;
201        sorting = true;
202        stepButton.setEnabled(true);
203        stepArea.setText("Array awal: " + arrayToString(array) + "\n\n"); // Tampilkan array awal
204        panelArray.removeAll();
205        labelArray = new JLabel[array.length];
206        for(int k = 0; k < array.length; k++) {
207            labelArray[k] = new JLabel(String.valueOf(array[k]));
208            labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
209            labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
210            labelArray[k].setPreferredSize(new Dimension(50, 50));
211            labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
212            labelArray[k].setOpaque(true); // Agar background color terlihat
213            panelArray.add(labelArray[k]);
214        }
215        // Highlight elemen pertama sebagai bagian yang "sudah terurut" secara default
216        if (labelArray.length > 0) {
217            labelArray[0].setBackground(Color.LIGHT_GRAY);
218        }
219
220        panelArray.revalidate();
221        panelArray.repaint();
222    }
```

9. Method updateLabels
    a. Menggunakan loop for untuk mengakses seluruh elemen yang ada pada array, kemudian menggunakan labelArray setText untuk mengupdate setiap teks label sesuai dengan nilai terbaru dari array
    b. Melakukan reset pada background apabila belum terurut (background abu-abu) dan finish (background hijau)

```
224    private void updateLabels() {
225        for(int k = 0; k < array.length; k++) {
226            labelArray[k].setText(String.valueOf(array[k]));
227            // Reset warna dasar sebelum highlight langkah berikutnya, kecuali yang sudah terurut
228            if (! (labelArray[k].getBackground() == Color.LIGHT_GRAY || labelArray[k].getBackground() == Color.GREEN) ) {
229                labelArray[k].setBackground(null);
230                labelArray[k].setOpaque(false);
231            }
232        }
233    }
```

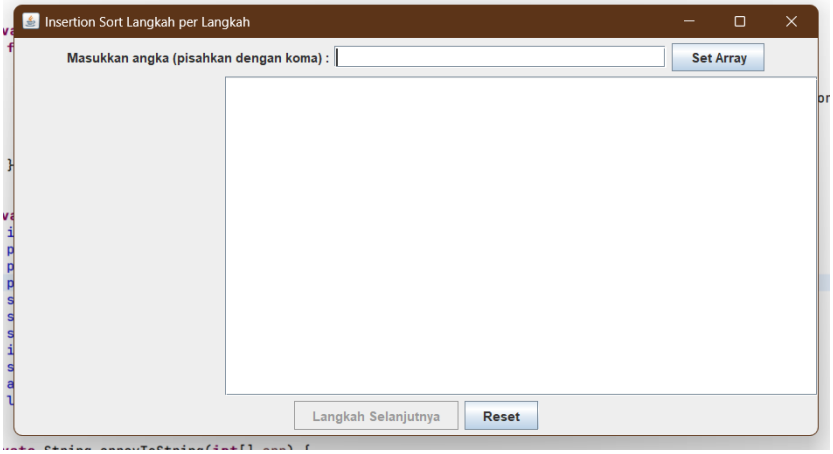
10. Method reset
    a. Mengosongkan input pada kolom input pada user atau pengguna dengan setText(“”)
    b. Menghapus seluruh label dari panel visualisasi dengan removeAll(), memperbaharui tampilan GUI sehingga langsung mencerminkan perubahan dengan revalidate() dan repaint()
    c. Mengosongkan area text pada area log atau langkah sorting
    d. Menonaktifkan tombol “Step” dikarenakan tidak ada array yang sudah diinput
    e. Reset status sorting menjadi false
    f. Indeks pada insertion sort direset menjadi indeks ke 1
    g. StepCount juga dilakukan reset
    h. Variabel utama dihapus isinya (array = null dan labelArray = null) supaya data sebelumnya tidak mempengaruhi input setelahnya

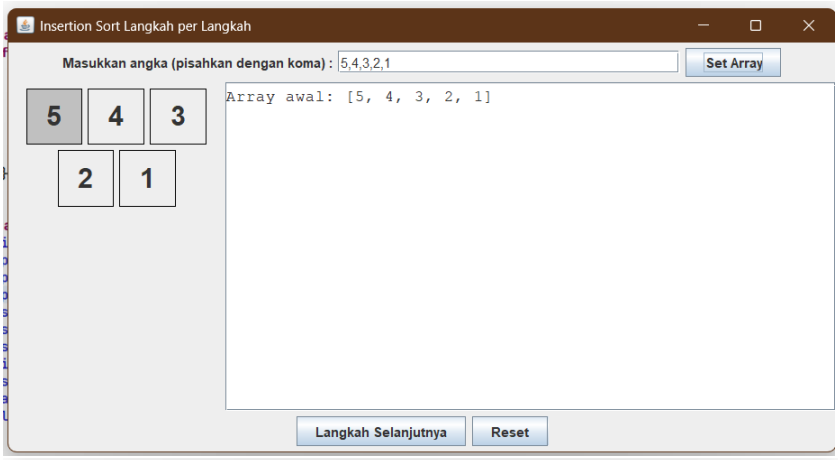
```
235    private void reset() {
236        inputField.setText("");
237        panelArray.removeAll();
238        panelArray.revalidate();
239        panelArray.repaint();
240        stepArea.setText("");
241        stepButton.setEnabled(false);
242        sorting = false;
243        i = 1;
244        stepCount = 1;
245        array = null;
246        labelArray = null;
247    }
```

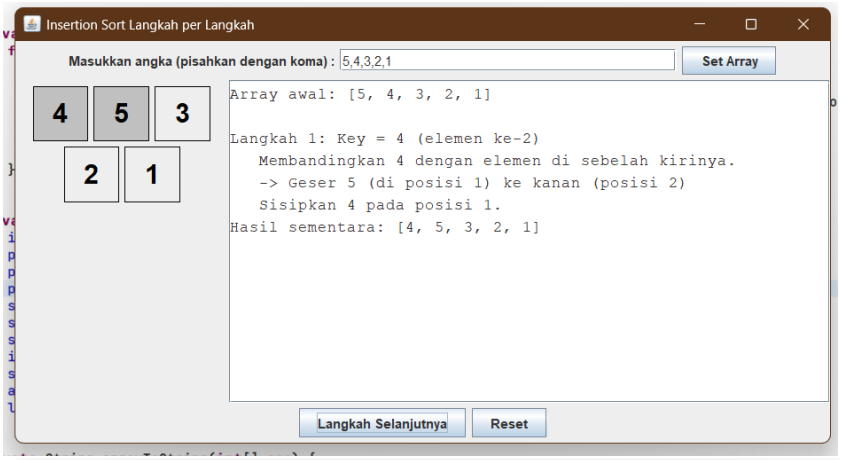
11. Method arrayToString
    a. Menangani apabila array nya bernilai null, maka return “” string kosong
    b. Membuat objek dari class StringBuilder dengan nama sb, untuk membuat teks lebih efisien, dimulai pada saat menambahkan siku terbuka “[“ ke dalam teks
    c. Melakukan perulangan for untuk tiap elemen array, sehingga tiap angka pada array akan ditambahkan ke dalam StringBuilder
    d. Mendeteksi apakah elemen merupakan elemen terakhir pada array yang sedang dibangun, apabila bukan, maka ditambahkan “, “ pada elemen setelahnya
    e. Menambahkan siku penutup “]” pada bagian akhir, menandakan array sudah tersedia dalam bentuk String tadi

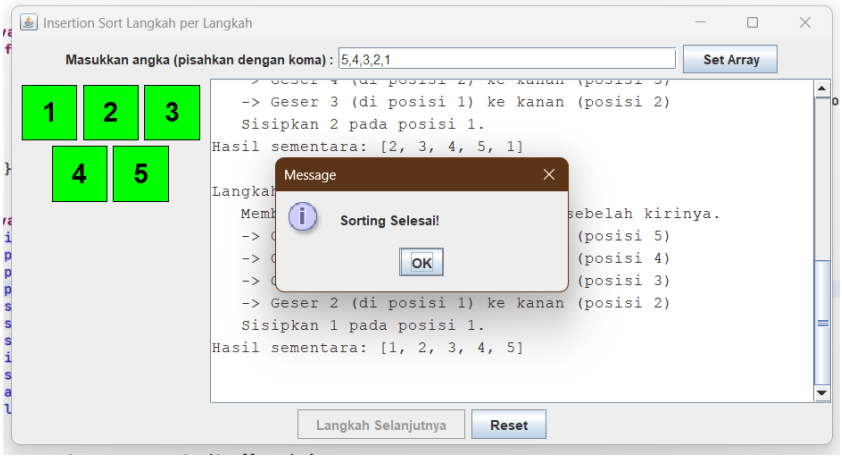
```
249    private String arrayToString(int[] arr) {
250        if (arr == null) return "";
251        StringBuilder sb = new StringBuilder();
252        sb.append("[");
253        for(int k = 0; k < arr.length; k++) {
254            sb.append(arr[k]);
255            if(k < arr.length - 1) sb.append(", ");
256        }
257        sb.append("]");
258        return sb.toString();
259    }
```

12. Berikut merupakan hasil dari GUI InsertionSort yang telah disusun tadi









13. Selanjutnya, kita akan melakukan GUI untuk SelectionSort, copy kode dari InsertionSort tadi kemudian buat file baru dengan nama **SelectionSortGUI**, hanya beberapa method yang akan ada perubahan.
14. Sama halnya dengan InsertionSortGUI, pada SelectionSortGUI hanya terdapat beberapa algoritma dan logika yang diubah, namun secara kerangka UI, InsertionSortGUI sama dengan SelectionSortGUI
15. Pada bagian deklarasi variabel pengendali algoritma
    a. Deklarasi integer i untuk indeks pass luar (menandakan batas antara bagian yang sudah terurut dan yang belum)
    b. Deklarasi integer j untuk indeks memindai sisa array atau inner loop (melakukan pemindaian atau pencarian pada bagian elemen yang belum terurut)
    c. Deklarasi integer minIndex untuk index dari elemen minimum yang ditemukan di bagian yang belum terurut (menyimpan posisi elemen terkecil yang ditemukan selama pencarian sorting)

```
32    private int i;          // Indeks pass luar (posisi yang akan diisi elemen minimum)
33    private int j;          // Indeks untuk memindai sisa array (inner loop)
34    private int minIndex;   // Indeks dari elemen minimum yang ditemukan di bagian belum terurut
```

16. Pada method **performStep** yang diubah/ditambahkan adalah

a. Pada bagian if (j < array.length) logika diubahkan, yaitu melakukan pemindaian nilai minimum, bagian ini menangani satu langkah pemindaian. Setiap tombol “Langkah Selanjutnya” di klik, maka proses ini akan membandingkan elemen minimum dengan elemen yang diperiksa selanjutnya
b. Pada bagian else { }, bagian ini merupakan bagian pertukaran (swap). Proses ini akan tereksekusi saat j sudah mencapai akhir array. Pada saat bagian ini dijalankan, elemen terkecil yang ditemukan akan ditukar dengan posisi i, setelah itu i dinaikkan dan variabel j dan minIndex akan direset untuk memulai langkah selanjutnya

```
private void performStep() {
    if (!sorting || array == null || array.length == 0) {
        return;
    }

    StringBuilder stepLog = new StringBuilder(); // Log dibuat baru untuk setiap klik "Langkah Selanjutnya"
    stepLog.append("Langkah ").append(stepCount).append(" (Pass ke-").append(i + 1).append("):\n");

    // Tahap 1: Memindai untuk menemukan elemen minimum dalam sisa array
    // Kondisi ini menangani satu langkah pemindaian (satu iterasi dari inner loop)
    if (j < array.length) {
        stepLog.append("   Membandingkan array[").append(j).append("] (").append(array[j]).append(") dengan kandidat minimum saat ini array[").append(minIndex).append("] (").append(array[minIndex]).append(").\n");
        if (array[j] < array[minIndex]) {
            minIndex = j; // Update minIndex jika ditemukan elemen yang lebih kecil
            stepLog.append("    → Ditemukan minimum baru sementara di indeks ").append(minIndex).append(" (nilai ").append(array[minIndex]).append(").\n");
        }
        visualizeStep(j, minIndex, false); // Visualisasikan langkah pemindaian saat ini
        j++; // Pindah ke elemen berikutnya untuk dipindai di klik selanjutnya
    }
    // Tahap 2: Pemindaian untuk pass i saat ini selesai (j sudah mencapai akhir array)
    // Lakukan pertukaran jika perlu, dan siapkan untuk pass i berikutnya.
    else { // j ≥ array.length
        stepLog.append("   Selesai memindai untuk pass ke-").append(i + 1).append(". Minimum ditemukan di indeks ").append(minIndex).append(" (nilai ").append(array[minIndex]).append(").\n");
        if (minIndex != i) { // Jika minimum bukan elemen pertama dari bagian yang belum terurut
            stepLog.append("   Menukar array[").append(i).append("] (").append(array[i]).append(") dengan array[").append(minIndex).append("] (").append(array[minIndex]).append(").\n");
            int temp = array[i];
            array[i] = array[minIndex];
            array[minIndex] = temp;
        } else {
            stepLog.append("   Elemen array[").append(i).append("] (").append(array[i]).append(") sudah merupakan minimum di posisinya, tidak ada pertukaran.\n");
        }
        updateLabelsText(); // Update teks label setelah potensi pertukaran
        visualizeStep(-1, -1, true); // Visualisasi setelah swap, tandai elemen ke-i sebagai sorted

        i++; // Pindah ke pass berikutnya (elemen berikutnya yang akan diisi)

        if (i < array.length - 1) { // Jika masih ada pass berikutnya yang perlu dijalankan
            minIndex = i;    // Reset minIndex ke awal bagian yang belum terurut untuk pass baru
            j = i + 1;       // Reset j untuk memulai pemindaian dari elemen setelah i yang baru
            stepLog.append("   Memulai pass baru untuk posisi ke-").append(i+1).append(". Kandidat minimum awal: array[").append(minIndex).append("] (").append(array[minIndex]).append(").\n");
            visualizeStep(-1, minIndex, false); // Highlight minIndex awal untuk pass baru
        } else { // Sorting selesai (semua elemen kecuali mungkin yang terakhir sudah diurutkan)
            sorting = false;
            stepButton.setEnabled(false);
            stepLog.append("\nSorting Selesai!\n");
            visualizeStep(-1,-1,true); // Visualisasi final state (semua hijau)
            JOptionPane.showMessageDialog(this, "Sorting selesai!");
        }
    }

    stepLog.append("Array saat ini: ").append(arrayToString(array)).append("\n\n");
    stepArea.append(stepLog.toString()); // Tambahkan log dari langkah ini ke JTextArea
    stepCount++;
}
```

17. Method **visualizeStep**
    a. Apabila labelArray belum dibuat, maka untuk menghindari error, method ini tidak akan jalan
    b. Menggunakan perulangan for untuk memeriksa seluruh elemen pada array dan memasukkannya sebagai label serta diwarnai sesuai dengan status atau kondisi terbarunya
    c. Memutuskan warna yang akan diberi pada tiap label, dengan menggunakan logika if-else, yaitu apabila k lebih kecil dari i (tidak termasuk dalam pemeriksaan, alias posisi fix), maka elemen tersebut diwarnai hijau, pada bagian else if akan dijalankan untuk menangani elemen yang baru saja dipindahkan ke posisi i-1 setelah satu putaran selesai. Ini memastikan elemen tersebut langsung menjadi hijau
    d. Untuk elemen yang belum terurut, elemen yang menjadi nilai minimum sementara diberi warna **kuning**, untuk elemen yang sedang dilakukan perbandingan diberi warna **oranye**, untuk elemen yang belum dilakukan pengurutan diberi warna **putih**
    e. Bagian if (!sorting && array != null && array.length > 0) dieksekusi ketika variabel global sorting bernilai false (artinya proses pengurutan telah selesai). Tujuannya adalah untuk memastikan semua elemen diwarnai hijau sebagai tanda bahwa array telah berhasil diurutkan dengan benar
    f. Pada penggunaan method panelArray.revalidate() dan panelArray.repaint() merupakan perintah standar dalam Java Swing/AWT. Setelah semua warna diatur, repaint() memerintahkan sistem untuk segera menggambar ulang panel yang berisi label-label tersebut agar perubahan warna bisa terlihat pada tampilan GUI di layar

```
175    private void visualizeStep(int currentScanIndex, int currentMinIndex, boolean passCompletedForI) {
176        if (labelArray == null) return;
177
178        for (int k = 0; k < labelArray.length; k++) {
179            labelArray[k].setOpaque(true); // Penting agar background terlihat
180            labelArray[k].setForeground(Color.BLACK); // Default warna teks
181
182            if (k < i) { // Bagian yang sudah terurut (sebelum pass i saat ini)
183                labelArray[k].setBackground(Color.GREEN);
184            } else if (passCompletedForI && k == i -1 && i > 0) { // Elemen yang baru saja disortir di pass sebelumnya (i sudah di-increment)
185                 labelArray[k].setBackground(Color.GREEN);
186            }
187            else { // Bagian yang belum terurut atau sedang diproses
188                if (k == currentMinIndex) {
189                    labelArray[k].setBackground(Color.YELLOW); // Kandidat minimum saat ini
190                } else if (k == currentScanIndex) {
191                    labelArray[k].setBackground(Color.ORANGE); // Elemen yang sedang dipindai/dibandingkan
192                } else {
193                    labelArray[k].setBackground(Color.WHITE); // Default untuk bagian belum terurut
194                }
195            }
196        }
197         // Jika sorting sudah selesai semua, pastikan semua hijau
198        if (!sorting && array != null && array.length > 0) {
199             for (JLabel label : labelArray) {
200                label.setOpaque(true);
201                label.setBackground(Color.GREEN);
202                label.setForeground(Color.BLACK);
203            }
204        }
205        panelArray.revalidate();
206        panelArray.repaint();
207    }
```

18. Method **updateLabelsText**
    a. Pada bagian if (labelArray == null || array == null) berperan sebagai penjaga, karena hal ini menghindari terjadinya program error, apabila labelArray dan array belum ada nilainya maka method ini akan berhenti
    b. Melakukan perulangan for dengan fungsi mengulang proses untuk setiap elemen yang ada pada data array dari index 0 hingga akhir array
    c. Pada bagian if(k < labelArray.length) memastikan bahwa kita tidak mencoba untuk melakukan akses pada nilai k yang tidak ada elemennya pada array, sehingga kita setText sesuai dengan nilai yang sudah kita ubah menjadi text tadi

```
165    private void updateLabelsText() {
166        if (labelArray == null || array == null) return;
167        for (int k = 0; k < array.length; k++) {
168            if (k < labelArray.length) { // Pastikan tidak out of bounds
169                labelArray[k].setText(String.valueOf(array[k]));
170            }
171        }
172    }
```

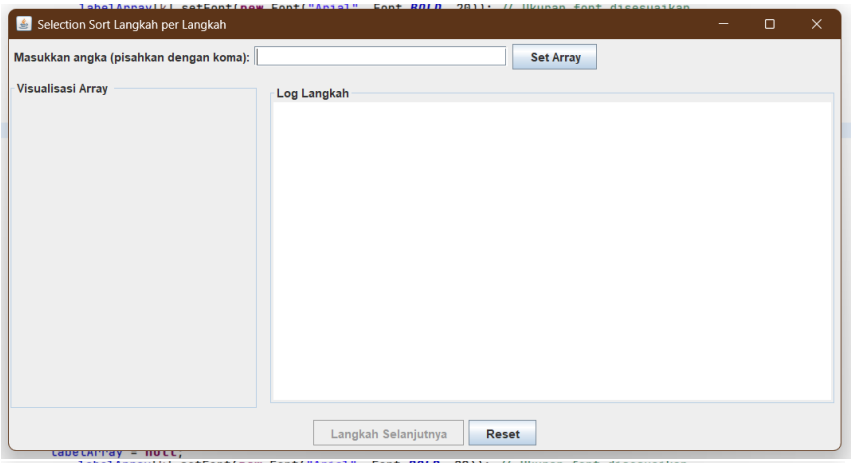
19. Method **setArrayFromInput**, yang berbeda adalah
    a. Deklarasi i = 0, untuk selection sort nilai i dimulai dari 0, nilai minIndex sama dengan i yaitu 0 dan nilai dari j adalah i ditambah 1 atau sama dengan nilai awalnya adalah 1

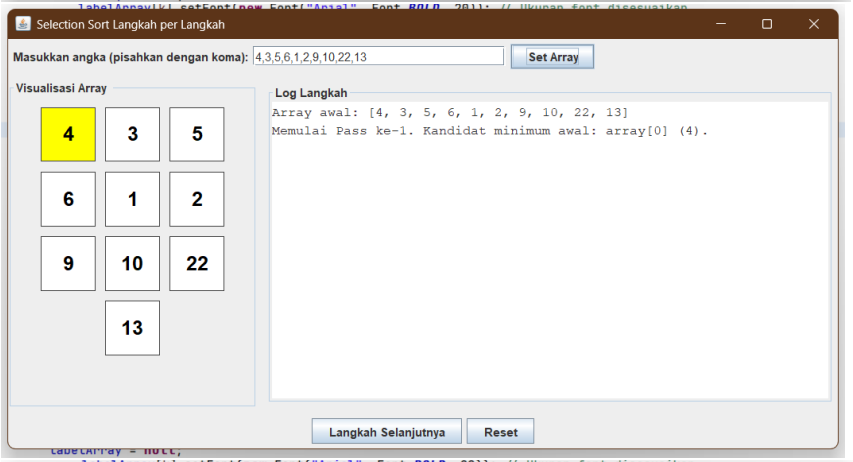
```java
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    String[] parts = text.split(",");
    if (parts.length == 0 || (parts.length == 1 && parts[0].trim().isEmpty())) {
        JOptionPane.showMessageDialog(this, "Masukkan setidaknya satu angka!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try {
        array = new int[parts.length];
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim());
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Masukkan hanya angka yang dipisahkan dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        array = null; // Reset array jika error
        return;
    }

    // Inisialisasi state untuk Selection Sort
    i = 0; // Pass luar dimulai dari indeks 0
    if (array.length > 0) { // Hanya jika array tidak kosong
        minIndex = i; // Awalnya, elemen pertama dianggap minimum untuk pass pertama
        j = i + 1;    // Mulai memindai dari elemen setelah i
    } else { // Jika array kosong, set j agar tidak error
        minIndex = -1; // atau nilai invalid lainnya
        j = 0;
    }


    stepCount = 1;
    sorting = true;
    stepButton.setEnabled(true);
    stepArea.setText("Array awal: " + arrayToString(array) + "\n");
    if (array.length > 0) {
         stepArea.append("Memulai Pass ke-1. Kandidat minimum awal: array["+minIndex+"] ("+array[minIndex]+").\n\n");
    }


    panelArray.removeAll();
    labelArray = new JLabel[array.length];
    for (int k = 0; k < array.length; k++) {
        labelArray[k] = new JLabel(String.valueOf(array[k]));
        labelArray[k].setFont(new Font("Arial", Font.BOLD, 20)); // Ukuran font disesuaikan
        labelArray[k].setOpaque(true); // Penting agar background terlihat
        labelArray[k].setBackground(Color.WHITE);
        labelArray[k].setBorder(BorderFactory.createLineBorder(Color.DARK_GRAY));
        labelArray[k].setPreferredSize(new Dimension(55, 55)); // Ukuran label
        labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
        panelArray.add(labelArray[k]);
    }

    if (array.length > 0) {
        visualizeStep(-1, minIndex, false); // Visualisasi awal, highlight minIndex pertama
    }

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

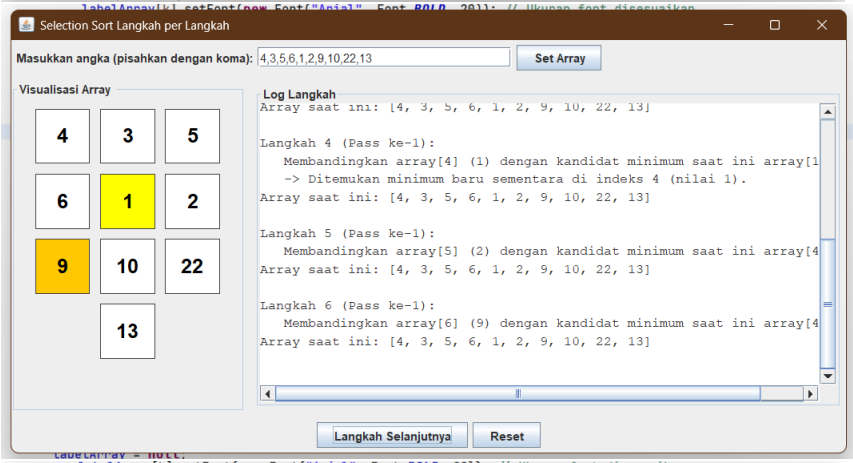
20. Method **reset**, yang berbeda adalah
    a. Melakukan deklarasi reset i kembali ke nilai awal yang benar, yaitu 0. (Berbeda pada insertionSort tadi, nilai i direset kembali ke nilai i = 1)

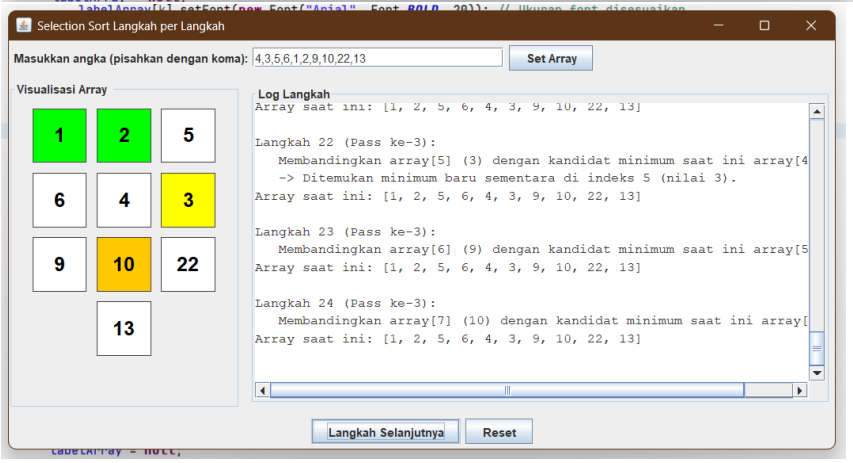
```java
private void reset() {
    inputField.setText("");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText("");
    stepButton.setEnabled(false);
    sorting = false;
    i = 0; // Reset i ke nilai awal yang benar
    // j dan minIndex akan di-set ulang saat setArrayFromInput dipanggil lagi
    stepCount = 1;
    array = null;
    labelArray = null;
}
```

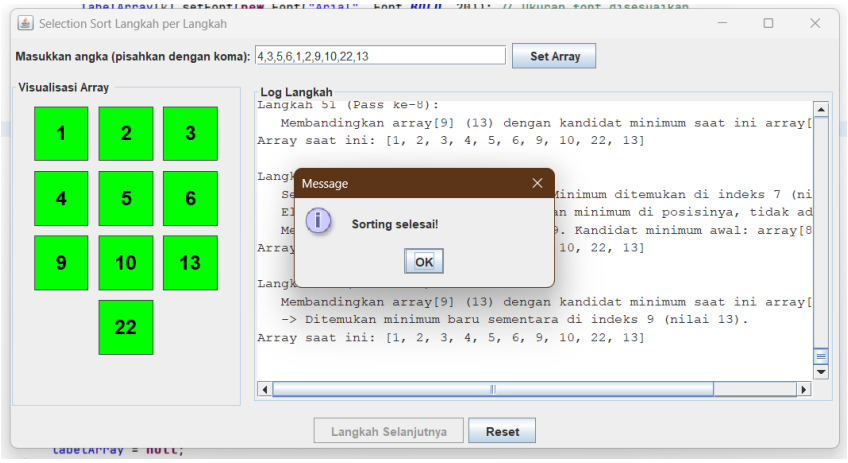
21. Beberapa perubahan kecil diantaranya adalah pada bagian desain GUI nya
    a. Pada judul frame, diubah menjadi "Selection Sort Langkah per Langkah"
    b. Untuk ukuran frame atau frame sizenya lebih besar sedikit daripada GUI InsertionSort yaitu (850, 450) untuk memberi ruang lebih pada area log
    c. Pada layout dan bordernya diberi sedikit jarak antar komponen dan ditambahkan TitledBorder pada panel array dan area log untuk memperjelas fungsinya
22. Berikut merupakan hasil dari GUI SelectionSort yang telah disusun tadi


Selection Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma):
Set Array
Visualisasi Array
Log Langkah
Langkah Selanjutnya
Reset


Selection Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma): 4,3,5,6,1,2,9,10,22,13
Set Array
Visualisasi Array
4
3
5
6
1
2
9
10
22
13
Log Langkah
Array awal: [4, 3, 5, 6, 1, 2, 9, 10, 22, 13]
Memulai Pass ke-1. Kandidat minimum awal: array[0] (4).
Langkah Selanjutnya
Reset


Selection Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma): 4,3,5,6,1,2,9,10,22,13
Set Array
Visualisasi Array
4
3
5
6
1
2
9
10
22
13
Log Langkah
Array saat ini: [4, 3, 5, 6, 1, 2, 9, 10, 22, 13]
Langkah 4 (Pass ke-1):
Membandingkan array[4] (1) dengan kandidat minimum saat ini array[1
-> Ditemukan minimum baru sementara di indeks 4 (nilai 1).
Array saat ini: [4, 3, 5, 6, 1, 2, 9, 10, 22, 13]
Langkah 5 (Pass ke-1):
Membandingkan array[5] (2) dengan kandidat minimum saat ini array[4
Array saat ini: [4, 3, 5, 6, 1, 2, 9, 10, 22, 13]
Langkah 6 (Pass ke-1):
Membandingkan array[6] (9) dengan kandidat minimum saat ini array[4
Array saat ini: [4, 3, 5, 6, 1, 2, 9, 10, 22, 13]
Langkah Selanjutnya
Reset


Selection Sort Langkah per Langkah
Masukkan angka (pisahkan dengan koma): 4,3,5,6,1,2,9,10,22,13
Set Array
Visualisasi Array
1
2
5
6
4
3
9
10
22
13
Log Langkah
Array saat ini: [1, 2, 5, 6, 4, 3, 9, 10, 22, 13]
Langkah 22 (Pass ke-3):
Membandingkan array[5] (3) dengan kandidat minimum saat ini array[4
-> Ditemukan minimum baru sementara di indeks 5 (nilai 3).
Array saat ini: [1, 2, 5, 6, 4, 3, 9, 10, 22, 13]
Langkah 23 (Pass ke-3):
Membandingkan array[6] (9) dengan kandidat minimum saat ini array[5
Array saat ini: [1, 2, 5, 6, 4, 3, 9, 10, 22, 13]
Langkah 24 (Pass ke-3):
Membandingkan array[7] (10) dengan kandidat minimum saat ini array[
Array saat ini: [1, 2, 5, 6, 4, 3, 9, 10, 22, 13]
Langkah Selanjutnya
Reset



D. Kesimpulan

Dari praktikum yang telah dilakukan, dapat diambil kesimpulan bahwa penerapan algoritma sorting, khususnya insertion sort dan selection sort, memberikan pemahaman tentang kemudahan dan detail dalam pengurutan data. Insertion sort bekerja dengan menyisipkan elemen ke posisi yang tepat dalam daftar yang sudah diurutkan, sementara selection sort memilih elemen terkecil dari daftar yang belum diurutkan dan menukarnya dengan elemen pertama. Kedua cara pengurutan ini memiliki keunggulan masing-masing, dengan insertion sort lebih optimal untuk data yang sebagian sudah terurut, sedangkan selection sort sederhana tetapi kurang efisien untuk jumlah data yang besar dikarenakan akan memakan langkah dan waktu yang lumayan banyak.