LAPORAN PRATIKUM STRUKTUR DATA PEKAN 7

IMPLEMENTASI ALGORITMA INSERTION SORT DAN SELECTION SORT DALAM APLIKASI GUI JAVA

MATA KULIAH

DOSEN PENGAMPU :

DR. WAHYUDI, S.T, M.T

OLEH:

AUFAN TAUFIQURRAHMAN

NIM 2411532011

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. LATAR BELAKANG

Algoritma pengurutan (sorting) merupakan dasar penting dalam ilmu komputer dan pemrograman. Salah satu penerapannya adalah dalam pembuatan aplikasi untuk mengurutkan data. Praktikum ini bertujuan untuk menerapkan dua algoritma sorting klasik, yaitu Insertion Sort dan Selection Sort, dalam bentuk aplikasi GUI berbasis Java menggunakan Swing.

B. TUJUAN

- Memahami dan mengimplementasikan algoritma Insertion Sort dan Selection Sort.
- Membangun antarmuka pengguna berbasis GUI menggunakan Java Swing.
- Mengintegrasikan logika algoritma ke dalam program yang interaktif.

C. DASAR TEORI

- **Insertion Sort:** Algoritma ini bekerja dengan menyisipkan elemen ke posisi yang sesuai dalam bagian array yang telah terurut. Kompleksitas waktu: $O(n^2)$ pada kasus terburuk.
- **Selection Sort:** Algoritma ini mencari elemen terkecil dari array yang belum terurut, lalu menukarnya dengan elemen pertama yang belum terurut. Kompleksitas waktu: $O(n^2)$.
- **GUI (Graphical User Interface):** Antarmuka pengguna grafis yang dibangun dengan Java Swing untuk memberikan pengalaman interaktif.

D. PROSEDUR KERJA

1. Class InsertionSortGUI,java
    a. Deklarasi Class dan Variabel

```
public class InsertionSortGUI extends JFrame {

    // Aufan_Taufiqurrahman
    // 2411532011
    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField textField;
    private JPanel panelArray;
    private JTextArea stepArea;

    private int i = 1, j;
    private boolean sorting = false;
    private int stepCount = 1;
```

b. Method Main

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

c. Constructor

```
public InsertionSortGUI() {
    setTitle(title:"Insertion Sort Langkah per langkah");
    setSize(width:750, height:400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(c:null);
    setLayout(new BorderLayout());
```

d. Pembuatan Panel Input

```
JPanel textPanel = new JPanel(new FlowLayout());
textField = new JTextField(columns:30);
setButton = new JButton(text:"Set array");
textPanel.add(new JLabel(text:"Input array (pisahkan dengan koma):"));
textPanel.add(textField);
textPanel.add(setButton);
```

e. Pembuatan Panel Array dan Kontrol

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

JPanel controlPanel = new JPanel();
stepButton = new JButton(text:"Langkah selanjutnya");
resetButton = new JButton(text:"Reset");
stepButton.setEnabled(b:false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

f. Pembuatan Text Area untuk Log

```
stepArea = new JTextArea(rows:8, columns:60);
stepArea.setEditable(b:false);
stepArea.setFont(new Font(name:"Monospaced", Font.PLAIN, size:14));
JScrollPane scrollPane = new JScrollPane(stepArea);
```

g. Penambahan Event Listeners

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

h. Method setArrayFromtext()

```
private void setArrayFromtext() {
    String text = textField.getText().trim();
    if (text.isEmpty()) return;
    String[] parts = text.split(regex:",");
    array = new int[parts.length];

    try {
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim());
        }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, message:"Masukkan hanya angka yang dipisahkan dengan koma!",
        title:"error", JOptionPane.ERROR_MESSAGE);
        return;
    }
```

i. Inisialisasi GUI Array

```
    i = 1;
    stepCount = 1;
    sorting = true;
    stepButton.setEnabled(b:true);
    stepArea.setText(t:"");
    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(name:"Arial", Font.BOLD, size:24));
        labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
        labelArray[k].setPreferredSize(new Dimension(width:50, height:30));
        labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
        panelArray.add(labelArray[k]);
    }
```

j. Method performStep() - Langkah Utama Insertion Sort

```
private void performStep() {
    if (i < array.length && sorting) {
        int key = array[i];
        j = i - 1;

        StringBuilder stepLog = new StringBuilder();
        stepLog.append(str:"Langkah ").append(stepCount).append(str:": Memasukkan ").append(key).append(str:"\n");

        while (j >= 0 && array[j] > key) {
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = key;
```

k. Update Display dan Log

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

i++;
stepCount++;

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

2. Class SelectionSortGUI.java

a. Perbedaan Utama dalam Deklarasi Variabel

```
public class SelectionSortGUI extends JFrame {

    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField textField;
    private JPanel panelArray;
    private JTextArea stepArea;
    private int minIndex;
```

- Tambahan variabel minIndex untuk menyimpan indeks elemen minimum

b. Method setArrayFromInput() - Inisialisasi Selection Sort

```
i = 0;
j = j + 1;
stepCount = 1;
sorting = true;
```

c. Styling Label dengan Highlight

```
for (int k = 0; k < array.length; k++) {
    labelArray[k] = new JLabel(String.valueOf(array[k]));
    labelArray[k].setFont(new Font(name:"Arial", Font.BOLD, size:24));
    labelArray[k].setOpaque(isOpaque:true);
    labelArray[k].setBackground(Color.WHITE);
    labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
    labelArray[k].setPreferredSize(new Dimension(width:50, height:50));
    labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
    panelArray.add(labelArray[k]);
}
```

d. Method performStep() Selection Sort - Algoritma Utama

```
private void performStep() {
    if (i < array.length - 1 && sorting) {
        StringBuilder stepLog = new StringBuilder();
        if (j == i + 1) {
            minIndex = i;
        }

        if (j < array.length) {
            if (array[j] < array[minIndex]) {
                minIndex = j;
            }
            j++;
        }
```

e. Proses Pertukaran Elemen

```
if (j == array.length) {
    if (minIndex != i) {
        int temp = array[i];
        array[i] = array[minIndex];
        array[minIndex] = temp;
        stepLog.append(str:"Langkah ").append(stepCount).append(str:": Menukar elemen ke- ").append(i).append(str:" (").
    } else {
        stepLog.append(str:"Langkah ").append(stepCount).append(str:": Tidak ada pertukaran (elemen ke- ").append(i).app
    }
    stepLog.append(str:"hasil: ").append(arrayToString(array)).append(str:"\n\n");
    stepArea.append(stepLog.toString());

    i++;
    j = i + 1;
    stepCount++;
}
```

f. Method Highlight untuk Visualisasi

```
private void highlightMinIndex() {
    resetHighlight();
    if (minIndex >= 0 && minIndex < labelArray.length) {
        labelArray[minIndex].setBackground(Color.YELLOW);
    }
}

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

g. Method Utility yang Sama

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

private void reset() {
    textField.setText(t:"");
    panelArray.removeAll();
    panelArray.revalidate();
    panelArray.repaint();
    stepArea.setText(t:"");
    stepButton.setEnabled(b:false);
    sorting = false;
    i = 1;
    stepCount = 1;
}

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

E. KESIMPULAN

Berdasarkan implementasi dan analisis kedua program GUI sorting, dapat disimpulkan:

1. Perbedaan Algoritma
   - Insertion Sort: Memasukkan setiap elemen ke posisi yang tepat dalam bagian yang sudah terurut, dimulai dari elemen kedua (indeks 1)
   - Selection Sort: Mencari elemen minimum dan menukarnya dengan elemen di posisi yang sesuai, dimulai dari elemen pertama (indeks 0)
2. Kompleksitas dan Efisiensi

- Kedua algoritma memiliki time complexity $O(n^2)$, tetapi insertion sort lebih adaptif untuk data yang hampir terurut
- Selection sort selalu melakukan n-1 pertukaran, sedangkan insertion sort bisa melakukan lebih sedikit operasi untuk data yang sudah hampir terurut

3. Implementasi GUI
   - Java Swing terbukti efektif untuk membuat visualisasi algoritma sorting Penggunaan event-driven programming memungkinkan kontrol step-by-step yang interaktif
   - Visualisasi dengan highlighting dan logging membantu pemahaman proses algoritma
4. Kelebihan dan Kekurangan Implementasi

   Kelebihan:
   - Interface yang user-friendly dengan input validation
   - Visualisasi real-time yang memudahkan pemahaman
   - Log langkah-langkah yang detail
   - Kontrol manual untuk pembelajaran step-by-step

   Kekurangan:
   - Terdapat beberapa bug minor dalam implementasi Selection Sort (inisialisasi variabel)
   - Tidak ada fitur auto-play atau kontrol kecepatan
   - Layout yang bisa diperbaiki untuk array yang besar
5. Pembelajaran

   Implementasi GUI untuk algoritma sorting memberikan pemahaman yang lebih mendalam tentang:
   - Cara kerja internal algoritma sorting
   - Perbedaan karakteristik antar algoritma
   - Penggunaan Java Swing untuk aplikasi edukatif
   - Pentingnya visualisasi dalam pembelajaran algoritma

Praktikum ini berhasil mendemonstrasikan bahwa kombinasi teori algoritma dengan implementasi visual dapat significantly meningkatkan pemahaman konsep sorting algorithms.