# LAPORAN PRAKTIKUM PEKAN 7 STRUKTUR DATA
# INSERTION SORT & SELECTION SORT


OLEH :

FADIL INSANUS SIDDIK

(2411532013)

MATA KULIAH : PRAKTIKUM STRUKTURR DATA

DOSEN PENGAMPU : Dr. Wahyudi, S.T, M.T.

FAKULTAS TEKNOLOGI INFORMASI

PROGRAM STUDI INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, JUNI 2025

# A. PENDAHULUAN

Struktur data merupakan salah satu fondasi utama dalam dunia pemrograman dan ilmu komputer. Dalam banyak kasus, data yang tidak terstruktur akan sulit diolah secara efisien. Salah satu metode dasar dalam pengolahan data adalah sorting atau pengurutan, yakni proses menyusun elemen-elemen dalam suatu koleksi berdasarkan urutan tertentu, seperti menaik (ascending) atau menurun (descending).

Beberapa algoritma pengurutan klasik yang sering digunakan dan dipelajari adalah Bubble Sort, Merge Sort, Quick Sort, dan Shell Sort. Masing-masing memiliki cara kerja, kompleksitas, dan efisiensi yang berbeda.

- Bubble Sort adalah algoritma sederhana yang membandingkan elemen bersebelahan dan menukarnya jika salah urut. Proses ini diulang hingga seluruh array terurut.
- Merge Sort adalah algoritma divide and conquer yang memecah array menjadi bagian kecil, mengurutkannya, lalu menggabungkannya kembali secara terurut.
- Quick Sort juga merupakan algoritma divide and conquer, namun bekerja dengan memilih pivot dan mempartisi array ke kiri dan kanan berdasarkan pivot, lalu mengurutkannya secara rekursif.
- Shell Sort adalah pengembangan dari Insertion Sort yang menggunakan gap (jarak) untuk membandingkan dan menyisipkan elemen, mempercepat proses pengurutan terutama untuk dataset berukuran besar.

Dalam praktikum ini, seluruh algoritma diimplementasikan dalam bentuk aplikasi GUI interaktif menggunakan bahasa pemrograman Java dengan pustaka Swing. Hal ini bertujuan untuk membantu pemahaman konsep secara visual dan mendalam.

.

# B. TUJUAN PRAKTIKUM

Adapun tujuan dari praktikum ini adalah sebagai berikut:

1. Mempelajari dan memahami konsep dasar berbagai algoritma sorting, yaitu Bubble Sort, Merge Sort, Quick Sort, dan Shell Sort.
2. Mengimplementasikan algoritma sorting menggunakan Java, khususnya dengan pendekatan berbasis GUI (Graphical User Interface) untuk mempermudah visualisasi langkah-langkah sorting.
3. Mengembangkan kemampuan logika algoritmik dan pemrograman interaktif melalui simulasi proses sorting yang divisualisasikan secara real-time.
4. Membandingkan performa dan cara kerja masing-masing algoritma sorting berdasarkan visualisasi dan jumlah langkah yang dibutuhkan.

5. Menumbuhkan kemampuan debugging dan pemahaman struktur kontrol, terutama pada proses rekursif (seperti Merge Sort dan Quick Sort) serta iteratif (Bubble dan Shell Sort).

## C. LANGKAH-LANGKAH

1. BubbleSort

```
public class BubbleSort extends JFrame {
```

- Mendefinisikan kelas public bernama BubbelSort yang mewarisi JFrame, artinya aplikasi ini merupakan program GUI berbasis Java Swing.

```
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private int[] array;
    private JPanel panelArray;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JTextArea stepArea;
    private int i = 1, j;
    private boolean sorting = false;
    private int stepCount = 1;
```

- inputField : untuk input angka oleh pengguna (dipisahkan koma).
- stepArea : area teks untuk log proses bubble sort.
- labelArray : array label untuk menampilkan nilai dalam GUI.
- stepButto, resetButton, setButton : tombol-tombol control.
- panelArray : panel visual tempat array ditampilkan.
- array : array integer yang akan diurutkan.
- i, j : variable loop untuk bubble sort.
- sorting : penanda apakah proses sorting sedang berjalan.
- stepCount : menghitung jumlah langkah sorting.

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

- Kelas utama yang mnejdai penanda saat program mulai dijalankan.

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

    // panel input
            JPanel inputPanel = new JPanel (new FlowLayout());
            inputField = new JTextField(30);
            setButton = new JButton("Set Array");
            inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
            inputPanel.add(inputField);
            inputPanel.add(setButton);

            // panel array visual
            panelArray = new JPanel();
            panelArray.setLayout (new FlowLayout());

            // panel kontrol
            JPanel controlPanel = new JPanel();
            stepButton = new JButton("Langkah selanjutnya");
            resetButton = new JButton("Reset");
            stepButton.setEnabled(false);
            controlPanel.add(stepButton);
            controlPanel.add(resetButton);
```

```
    // area teks untuk log langkah-langkah
    stepArea = new JTextArea(8, 60);
    stepArea.setEditable(false);
    stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
    JScrollPane scrollPane = new JScrollPane(stepArea);

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

    // event set array
    setButton.addActionListener(e -> setArrayFromInput());

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

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

}
```

Membuat dan mengatur semua komponen GUI, termasuk:

- Label input dan tombol.
- Panel array.
- Area log proses.
- LIistener untuk tombol:
    - setButton → memanggil setArrayFromInput()
    - stepButton → memanggil performStep()
    - resetButton → memanggil reset()

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

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

- Membaca input teks dan menginovasinya menjadi int[].
- Membuat array label visual (labelArray).
- Menampilkan array ke panelArray.
- Mengatur i, j, dan stepCount ke kondisi awal.

```
private void performStep() {
    if (!sorting || i >= array.length - 1) {
        sorting = false;
        stepButton.setEnabled(false);
        JOptionPane.showMessageDialog(this, "Sorting selesai!");
        return; }
    resetHighlights();
    StringBuilder stepLog = new StringBuilder();
    labelArray[j].setBackground(Color.CYAN);
    labelArray[j + 1].setBackground(Color.CYAN);
    if (array[j] > array[j + 1]) {
        // swap
        int temp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temp;
        labelArray[j].setBackground(Color.RED);
        labelArray[j + 1].setBackground(Color.RED);
        stepLog.append("Langkah ").append(stepCount).append(": Menukar elemen ke-")
               .append(j).append(" (").append(array[j + 1]).append(") dengan ke-")
               .append(j + 1).append(" (").append(array[j]).append(")\n");
```

```
} else {
    stepLog.append("Langkah ").append(stepCount).append(": Tidak ada pertukaran antara ke-")
           .append(j).append(" dan ke-").append(j + 1).append("\n");
    stepLog.append("Hasil: ").append(arrayToString(array)).append("\n\n");
    stepArea.append(stepLog.toString());
    updateLabels();
    j++;
    if (j >= array.length - i - 1) {
        j = 0;
        i++; }
    stepCount++;
    if (i >= array.length -1) {
        sorting = false;
        stepButton.setEnabled(false);;
        JOptionPane.showMessageDialog(this,  "Sorting selesai!"); }}

        }
```

- Menjalankan satu langkah dari algoritma Bubble Sort.
- Menukar elemen bersebelahan jika salah urut.
- Jika j mencapai batas, reset j dan naikkan i.
- Jika i sudah mencapai akhir array → sorting selesai.
- Semua langkah dicatat dalam stepArea.
- Update tampilan array dengan updateLabels().

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

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

- Mengubah nilai teks di semua JLabel agar mencerminkan isi terbaru dari array[].

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

- Mengosongkan semua komponen dan log.
- Reset semua variabel dan tampilan GUI ke awal.

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

- Fungsi untuk mengubah array ke dalam format string.

2. MergeSort

```
public class MergeSort extends JFrame {
```

- Mendeklarasikan jika kelas bernama MergeSort yang mewarisi Jframe.

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

- panelArray : menampilkan array sebagai label.
- inputField : tempat pengguna mengetikkan angka.
- stepButton : untuk melanjutkan ke langkah merge berikutnya.
- resetButton : mengulang proses.
- stetButton : mengatur input array.
- stepArea : log langkah-langkah proses merge.

```
private int i, j, k, left, mid, right;
private boolean isMerging = false;
private boolean copying = false;
private int[] temp;
private int stepCount = 1;
private Queue<int[]> mergeQueue = new LinkedList<>();
```

- i, j, k : indeks saat penggabungan.
- left, mid, right : batas kiri, tengah, kanan subarray.
- temp : array sementara hasil merge.
- mergeQueue : menyimpan antrian bagian-bagian array yang akan digabung.

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

- Kelas utama yang menjadi penanda ketika program mulai dijalankan.

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

    // panel input
    JPanel inputPanel = new JPanel (new FlowLayout());
    inputField = new JTextField(30);
    setButton = new JButton("Set Array");
    inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
    inputPanel.add(inputField);
    inputPanel.add(setButton);

    // panel array visual
    panelArray = new JPanel();
    panelArray.setLayout (new FlowLayout());

    // panel kontrol
    JPanel controlPanel = new JPanel();
    stepButton = new JButton("Langkah selanjutnya");
    resetButton = new JButton("Reset");
    stepButton.setEnabled(false);
    controlPanel.add(stepButton);
    controlPanel.add(resetButton);
```

```
// area teks untuk log langkah-langkah
stepArea = new JTextArea(8, 60);
stepArea.setEditable(false);
stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
JScrollPane scrollPane = new JScrollPane(stepArea);

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

// event set array
setButton.addActionListener(e -> setArrayFromInput());

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

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

- Penjelasan tampilan ketika program dijalankan.

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;
    String[] parts = text.split(",");
    array = new int[parts.length];
    try {
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim()); }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this,  "Masukkan hanya angka yang dipisahkan "
                + "dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return; }

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

```
    mergeQueue.clear();
    generateMergeSteps(0, array.length - 1);
    stepButton.setEnabled(true);
    stepArea.setText("");
    stepCount = 1;
    isMerging = false;
    panelArray.revalidate();
    panelArray.repaint();
    }
```

- Input dipecah berdasarkan koma.
- Setiap nilai di-prase menjadi integer.
- Label dibuat untuk masing-masing elemen array.
- Memanggil generateMergeSteps(0, array.length – 1) untuk membentuk antrian merge berdasarkan rekursi.

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

    if (!isMerging && !mergeQueue.isEmpty()) {
        int[] range = mergeQueue.poll();
        left = range[0];
        mid = range[1];
        right = range[2];
        temp = new int[right - left + 1];
        i = left;
        j = mid + 1;
        k = 0;
        copying = false;
        isMerging = true;
        stepArea.append("Langkah " + stepCount++ + ": Mulai merge dari " + left + " ke " + righ
        return;
    }
```

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

```
if (copying && k < temp.length) {
    array[left + k] = temp[k];
    labelArray[left + k].setText(String.valueOf(temp[k]));
    labelArray[left + k].setBackground(Color.GREEN);
    k++;
    stepArea.append("Langkah " + stepCount++ + ": Tempelkan ke array utama\n");
    return;
}

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

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

}
```

- Mengambil rentang merge dari antrian.
- Menyiapkan array sementara temp.
- Bandingkan dan salin elemen jika salah satu sisi sudah habis.
- Menyimpan sisa elemen jika salah satu sisi sudah habis.
- Elemen pada temp disalin kembali ke array[]
- Label juga diperbarui dengan nilai baru.

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

- Menghapus warna latar belakang pada semua label.

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

- Membersihkan seluruh isi array dan tampilan.
- Mengatur semua status ke awal.

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

- Memperbarui nilai-nilai pada label sesuai isi array.

```
    private void generateMergeSteps(int left, int right) {
        if (left >= right) return;
        int mid = (left + right) / 2;
        generateMergeSteps(left, mid);
        generateMergeSteps(mid + 1, right);
        mergeQueue.add(new int[]{left, mid, right});
    }

}
```

- Membagi array secara rekursif menjadi bagian-bagian kecil.
- Menambahkan rencana merge ke dalam antrian (mergeQueue).

3. QuickSort

```
public class QuickSort extends JFrame {
```

- Program dengan kelas bernama QuickSort merupakan turunan dari JFrame, artinya aplikasi GUI.
- Menggunakan Swing GUI untuk membuat antarmuka visual.

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

- array: array yang akan diurutkan.
- labelArray: label untuk setiap elemen array (tampil di GUI).
- inputField: tempat pengguna memasukkan angka.
- stepArea: area teks untuk mencatat log proses sorting.
- panelArray: panel visual untuk menampilkan elemen array.
- Tombol kontrol: “Set Array”, “Langkah Selanjutnya”, dan “Reset”.

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

- stack: menyimpan indeks partisi yang belum diproses.
- low, high: batas bawah dan atas partisi saat ini.
- i, j: indeks saat ini dalam proses partisi.
- pivot: nilai poros untuk partisi.
- partitioning: menandai apakah sedang dalam proses partisi.
- sorting: status proses sorting.

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

- Kelas utama pada kelas ini yang berarti program mulai dijalankan.

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

- Digunakan untuk menyusun tampilan dari GUI.

```
    // panel input
    JPanel inputPanel = new JPanel (new FlowLayout());
    inputField = new JTextField(30);
    setButton = new JButton("Set Array");
    inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
    inputPanel.add(inputField);
    inputPanel.add(setButton);

    // panel array visual
    panelArray = new JPanel();
    panelArray.setLayout (new FlowLayout());

    // panel kontrol
    JPanel controlPanel = new JPanel();
    stepButton = new JButton("Langkah selanjutnya");
    resetButton = new JButton("Reset");
    stepButton.setEnabled(false);
    controlPanel.add(stepButton);
    controlPanel.add(resetButton);
```

- Menambahkan panel array, panel kontrol dan log, listener.

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;

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

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

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

```
    stack.clear();
    stack.push(new int[] {0, array.length - 1});
    sorting = true;
    partitioning = false;
    stepCount = 1;
    stepArea.setText("");;
    stepButton.setEnabled(true);

    panelArray.revalidate();
    panelArray.repaint();

    }
```

- Mengambil input dari inputField, memecahnya dengan koma.
- Mem-prase ke integer → disimpan ke array[].
- Membuat JLabel untuk setiap angka.
- Menambahkan semua label ke panelArray.
- Mengatur ulang stack dan mulai proses sorting.

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

    resetHighlights();

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

        highlightPivot(high);
        return;
```

```
if (j <= high - 1) {
    highlightCompare(j, high);
    if (array[j] <= pivot) {
        i++;
        if (i != j) {
            stepArea.append("Langkah " + stepCount++ + ": Tukar " + array[j] + " dan " + array[i]
            swap(i, j);
        } else {
            stepArea.append("Langkah " + stepCount++ + ": " + array[j] + " sudah di posisi yang b
        }
        updateLabels();
    } else {
        stepArea.append("Langkah " + stepCount++ + ": Lewatkan " + array[j] + " (lebih besar dari
        }
    j++;
    return;
}

if (!pivotSwapped) {
    swap(i + 1, high);
    updateLabels();
    stepArea.append("Langkah " + stepCount++ + ": Pindahkan pivot ke posisi tengah (index " + (i

    int p = i + 1;
    partitioning = false;

    if (p - 1 > low) stack.push(new int[] {low, p - 1});
    if (p + 1 < high) stack.push(new int[] {p + 1, high});

    pivotSwapped = true;
    resetHighlights();
    return;
```

- Fungsi ini menjalankan satu langkah Quick Sort per klik.
- Mengecek apakah sorting selesai.
- Mengambil partisi jika belum ada.
- Melakukan perbandingan dan pertukaran.
- Jika partisi selesai, tempatkan pivot dan tambahkan subpartisi.
- Menempatkan pivot di tempat yang benar.

- Menambahkan dua rentang partisi baru ke stack.

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

- Memberi warna kuning pada elemen pivot.

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

- Memberi warna pada elemen yang dibandingkan dan pivot.

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

- Mengemablikan semua warna label ke putih.

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

- Menukar dua elemen array dan memperbarui label GUI.

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

- Memperbarui nilai di JLabel agar sesuai isi array terbaru.

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

- Mengosongkan panel, array, label, dan area log.
- Menghapus semua sate sorting.

4. ShellSort

```
public class ShellSort extends JFrame {
```

- Mendeklarasikan kelas dengan nama ShellSort yang mewarisi JFrame, berarti ini adalah aplikasi berbasis GUI (Java Swing).

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

    private int temp;
    private int gap;
    private int i = 1, j;
    private boolean sorting = false;
    private int stepCount = 1;
    private boolean isSwapping = false;
    private JPanel contentPane;
```

- array: menyimpan data angka.
- panelArray: panel tempat elemen array ditampilkan.

- labelArray: array label yang menunjukkan elemen array secara visual.
- stepButton, resetButton, setButton: tombol kontrol.
- inputField: tempat user mengetik angka.
- stepArea: area teks untuk mencatat langkah-langkah.
- gap, i, j: variabel utama dalam proses Shell Sort.
- sorting: status proses sorting.
- isSwapping: menandakan apakah sedang dalam proses penukaran/penyisipan.
- stepCount: menghitung jumlah langkah.

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

- Menyiapkan seluruh komponen GUI.

```
// panel input
JPanel inputPanel = new JPanel (new FlowLayout());
inputField = new JTextField(30);
setButton = new JButton("Set Array");
inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma): "));
inputPanel.add(inputField);
inputPanel.add(setButton);

// panel array visual
panelArray = new JPanel();
panelArray.setLayout (new FlowLayout());

// panel kontrol
JPanel controlPanel = new JPanel();
stepButton = new JButton("Langkah selanjutnya");
resetButton = new JButton("Reset");
stepButton.setEnabled(false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

```
    // area teks untuk log langkah-langkah
    stepArea = new JTextArea(8, 60);
    stepArea.setEditable(false);
    stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
    JScrollPane scrollPane = new JScrollPane(stepArea);

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

    // event set array
    setButton.addActionListener(e -> setArrayFromInput());

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

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

- Menginput panel untuk memasukkan angka (dipisahkan koma) dan menyetelnya.
- Menampilkan elemen array dalam bentuk JLabel.
- Digunakan untuk menjalankan proses sorting satu langkah demi satu.
- Menampilkan proses/langkah-langkah yang sedang dijalankan.
- Mengatur tindakan saat tombol ditekan.

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) return;
    String[] parts = text.split(",");
    array = new int[parts.length];
    try {
        for (int k = 0; k < parts.length; k++) {
            array[k] = Integer.parseInt(parts[k].trim()); }
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this,  "Masukkan hanya angka yang dipisahkan "
                + "dengan koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return; }
```

```
gap = array.length/ 2;
i = gap;
sorting = true;
stepCount = 1;
stepArea.setText("");
stepButton.setEnabled(true);
panelArray.removeAll();
labelArray = new JLabel[array.length];
for (int k = 0; k < array.length; k++) {
    labelArray[k] = new JLabel(String.valueOf(array[k]));
    labelArray[k].setFont(new Font("Arial", Font.BOLD, 24));
    labelArray[k].setOpaque(true);
    labelArray[k].setBackground(Color.WHITE);;
    labelArray[k].setBorder(BorderFactory.createLineBorder(Color.BLACK));
    labelArray[k].setPreferredSize(new Dimension(50, 50));
    labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
    panelArray.add(labelArray[k]);
    }

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

- Membaca angka dari input user, memisahkan dengan koma, dan mengonversi ke int[].
- Inisialisasi nilai gap = array.length / 2.
- Menyiapkan label visual.
- Menampilkan array ke GUI.
- Mengaktifkan tombol stepButton.

```
private void performStep() {
    resetHighlights();
    if (!sorting || gap == 0) {
        stepArea.append("Shell Sort selesai.\n");;
        stepButton.setEnabled(false);;
        JOptionPane.showMessageDialog(this, "Shell Sort selesai!");
        stepArea.append("Hasil akhir: " + java.util.Arrays.toString(array) + "\n\n");
        return;
    }
    if (i < array.length) {
        if (!isSwapping) {
            temp = array[i];
            j = i;
            isSwapping = true;
        }
```

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

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

- Menjalankan 1 langkah ShellSort.

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

- Menambahkan pesan ke stepArea serta kondisi array saat itu.

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

- Mengubah nilai teks JLabel agar sesuai dengan nilai array saat ini.

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

- Mengembalikan warna latar belakang semua label ke putih.

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

- Menghapus seluruh isi input dan tampilan GUI.
- Reset kondisi agar siap digunakan kembali.

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

- Menjalankan GUI menggunakan thread event Swing.

# D. KESIMPULAN

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. Setiap algoritma sorting memiliki keunggulan dan kelemahan masing-masing:
    - Bubble Sort mudah dipahami namun lambat ($O(n^2)$)
    - Merge Sort stabil dan cepat pada data besar ($O(n \log n)$) namun membutuhkan memori tambahan
    - Quick Sort sangat cepat dalam praktik namun tidak stabil dan kurang baik pada data hampir terurut
    - Shell Sort memberikan efisiensi lebih baik daripada Bubble/Insertion, terutama pada data besar
2. Penggunaan Java GUI sangat efektif dalam membantu mahasiswa memahami langkah-langkah internal algoritma, terutama dengan tampilan visual, log teks, dan interaksi tombol.
3. Praktikum ini memberikan pengalaman langsung dalam mengembangkan simulasi algoritma serta memperkuat pemahaman tentang bagaimana data diproses dan diubah selama proses pengurutan.

4. Simulasi langkah demi langkah melalui tombol “Langkah Selanjutnya” memberikan kesempatan bagi mahasiswa untuk menganalisis dan mempelajari proses sorting secara bertahap, bukan hanya dari hasil akhir