LAPORAN PRAKTIKUM STRUKTUR DATA

SORTING LANJUTAN

OLEH:

AHMAD ZAHRAN

2411532004

DOSEN PENGAMPU:

DR. WAHYUDI, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Struktur data merupakan fondasi penting dalam dunia pemrograman, khususnya dalam pengolahan dan pengurutan data. Salah satu bentuk implementasinya adalah penggunaan algoritma pengurutan (sorting), seperti *Bubble Sort, Merge Sort, Quick Sort* dan *Shell Sort*. Dalam praktikum ini, kita menggabungkan pemahaman algoritma dengan visualisasi menggunakan antarmuka grafis berbasis *Java Swing*.

B. Tujuan

Tujuan dilakukannya praktikum ini adalah sebagai berikut:

1. Memahami Struktur Data Sorting pada java
2. Mengimplementasikan Sorting pada java dalam bentuk GUI

C. Langkah Kerja Praktikum

a. Kelas BubbleSort

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

1. Pertama buat class InsertionSortGUI, lalu buatlah array yang bertipe data Integer, dan deklarasikan fitur-fitur yang ingin dipakai pada GUI seperti JLabel, JButton, JTextField, JPanel, JTextArea, lalu buat variabel untuk penanda proses dalam insertion sort, dan buat penanda posisi dalam insertion sort.

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

2. lalu buat penanda apakah sortingnya sedang berlangsung

```
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();
            }
```

3. Method main, untuk menjalankan program GUI di java agar dapat berjalan dalam thread GUI

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

4. Membuat window utama untuk Jframe

```
// 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);
```

5. Untuk menambahkan Panel untuk menginputkan Array

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

6. Menambahkan panel untuk menampilkan nilai-nilai array

```
JPanel controlPanel = new JPanel();
stepButton = new JButton("Langkah Selanjutnya");
resetButton = new JButton("Reset");
stepButton.setEnabled(false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

7. Menambahkan panel untuk mengontrol proses sorting yang mana disini terdiri dari dua tombol yaitu StepButton untuk menjalankan langkah dari sorting dan resetButton untuk mereset.

```
stepArea = new JTextArea(8, 60);
stepArea.setEditable(false);
stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
JScrollPane scrollPane = new JScrollPane(stepArea);
```

8. Menambahkan JtextArea untuk menampilkan langkah-langkah dari sorting

```
// Tambahkan Panel ke Frame
getContentPane().add(inputPanel, BorderLayout.NORTH);
getContentPane().add(panelArray, BorderLayout.CENTER);
getContentPane().add(controlPanel, BorderLayout.SOUTH);
getContentPane().add(scrollPane, BorderLayout.EAST);
```

9. Lalu terakhir tambahkan semua panel ke Jframe sesuai dengan posisinya

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

10. Kode ini untuk menambahkan EventListener agar Tombol bias berfungsi

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

11. Buat method setArrayFromInput, dengan penjelasan sebagai berikut pertama untuk mengambil teks dari inputField, dan pisakhlan teks berdasarkan koma menjadi array dengan typedata String, dan buat array integer.

```
try {
    for (int k = 0; k < parts.length; k++) {
        array[k] = Integer.parseInt(parts[k].trim());
    }
} catch (NumberFormatException e) {
```

12. Kode ini memproses tiap-tiap elemen untuk menghapus spasinya dan mengubah menjadi integer

```
 JOptionPane.showMessageDialog(this, "Masukkan hanya angka "
         + "yang dipisahkan koma!", "Error", JOptionPane.ERROR_MESSAGE);
 return;
```

13. Untuk menampilkan pesan jika input tidak valid

```
i = 0;
j = 0;
stepCount = 1;
sorting = true;
stepButton.setEnabled(true);
stepArea.setText("");
panelArray.removeAll();
```

14. Untuk membuat variabel okntrol agar bias memulai sorting dari awal dan aktifkan stepButton, lalu menghapus log langkah sebelumnya di stepArea dan hapus seluruh elemen visual array dari panel

```
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]);
```

15. Membuat Label untuk Menampilkan Visual Elemen array

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

16. Untuk merefresh tampilan panel

```
private void performStep() {
    if (!sorting || i >= array.length - 1) {
        sorting = false;
        stepButton.setEnabled(false);
        JOptionPane.showMessageDialog(this, "Sorting selesai!");
        return;
```

17. MethodperformStep, ini untuk menjalankan setiap langkah dari Bubble sort, pertama cek apakah sorting masih berlangsung dan ‘i’ >= dari panjang array

```
resetHighlights();
StringBuilder stepLog = new StringBuilder();
labelArray[j].setBackground(Color.CYAN);
labelArray[j + 1].setBackground(Color.CYAN);
```

18. Untuk mereset kembali warna backround, dan menyusun stepLog yang akan ditampilkan di stepArea dan membri warna pada elemen array yang sedang dibandingkan.

```
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);
```

19. Untuk mengecek apakah elemen ‘j’ apakah lebih besar dari ‘j+1’, maka harus ditukar, lalu tukar dua elemen array untuk langkah utama dalam Bubble Sort, dan jika ada pertukaran, beri warna merah pada elemen yang ditukar

```
        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");
```

20. Menambahkan catatan ke stepLog bahwa pertukaran terjadi, lengkap dengan posisi dan nilai yang ditukar. Jika tidak ada pertukaran, dicatat juga bahwa perbandingan tidak menghasilkan perubahan posisi.

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

21. Menambahkan hasil array sementara setelah langkah tersebut ke dalam area log stepArea. Memperbarui teks JLabel agar menampilkan nilai array yang terbaru setelah pertukaran. Lanjutkan ke pasangan indeks berikutnya .

```
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!");
```

22. Cek apakah j sudah sampai, lalu tambahkan penghitung dari langkah-langkahnya, dan Setelah semua elemen diproses dan array dianggap terurut, proses dihentikan dan pengguna diberi notifikasi.

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

23. Method updateLabels, untuk mengupdate teks dari Jlabel di ‘panelArray’ untuk membuat kondisi array terbaru

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

24. Method resetHighLightts menjaga agar tampilan GUI tetap bersih dan jelas**,** hanya menampilkan highlight untuk elemen yang sedang dibandingkan atau ditukar pada setiap langkah.

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

25. Mehod reset, untuk mereset seluruh tampilan dan data agar bias mengulang proses dari awal

b. Kelas MergeSort

```
public class MergeSort extends JFrame {
//Ahmad Zahran :)
    private static final long serialVersionUID = 1L;
    private int[] array;
    private JLabel[] labelArray;
    private JButton stepButton, resetButton, setButton;
    private JTextField inputField;
    private JPanel panelArray;
    private JTextArea stepArea;

    private int stepCount = 1;
    private Queue<int[]> mergeQueue;
    private boolean isMerging = false;
    private boolean copying = false;
```

1. Sama seperti kelas-kelas sebelumnya untuk GUI dari Merge Sort ini masih sama, Pertama buat class MergeSort, lalu buatlah array yang bertipe data Integer, dan deklarasikan fitur-fitur yang ingin dipakai pada GUI seperti JLabel, JButton, JTextField, JPanel, JTextArea, lalu buat variabel untuk penanda proses dalam insertion sort, dan buat penanda posisi dalam insertion sort, lalu buat penanda apakah sortingnya sedang berlangsung

```
private int left, mid, right;
private int i, j, k;
private int[] temp;
```

2. Buat variabel unuk pembantu mergeSort yang mana berisi indeks area merge, indeks dari kiri dan kanan sub array

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

3. Method main, untuk menjalankan program GUI di java agar dapat berjalan dalam thread GUI

```
public MergeSort() {
    setTitle("Merge Sort Langkah per Langkah");
    setSize(850, 500);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(10, 10));
```

4. Membuat window utama untuk Jframe

```
JPanel topPanel = new JPanel(new BorderLayout());
JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
inputField = new JTextField(30);
setButton = new JButton("Set Array");
inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma):"));
inputPanel.add(inputField);
inputPanel.add(setButton);
topPanel.add(inputPanel, BorderLayout.CENTER);
```

5. Untuk menambahkan Panel untuk menginputkan Array dan mengaktifkan button SetArray

```
panelArray = new JPanel();
panelArray.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
```

6. Menambahkan panel untuk menampilkan nilai-nilai array

```
JPanel controlPanel = new JPanel();
stepButton = new JButton("Langkah Selanjutnya");
resetButton = new JButton("Reset");
stepButton.setEnabled(false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

7. Menambahkan panel untuk mengontrol proses sorting yang mana disini terdiri dari dua tombol yaitu StepButton untuk menjalankan langkah dari sorting dan resetButton untuk mereset.

```
JPanel logPanel = new JPanel(new BorderLayout());
logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));
stepArea = new JTextArea(8, 60);
stepArea.setEditable(false);
stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
JScrollPane scrollPane = new JScrollPane(stepArea);
logPanel.add(scrollPane, BorderLayout.CENTER);
```

8. Menambahkan panel untuk menampilkan langkah-langkah dari sorting

```
add(topPanel, BorderLayout.NORTH);
add(panelArray, BorderLayout.CENTER);
add(controlPanel, BorderLayout.SOUTH);
add(logPanel, BorderLayout.EAST);
```

9. Lalu terakhir tambahkan semua panel ke Jframe sesuai dengan posisinya

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

10. Kode ini untuk menambahkan EventListener agar Tombol bisa berfungsi

```
private void performStep() {
    resetHighlights();
    if (!isMerging && !mergeQueue.isEmpty()) {
        int[] range = mergeQueue.poll();
        left = range[0];
        mid = range[1];
        right = range[2];
```

11. MethodperformStep, dijalankan setiap kali pengguna menekan tombol “Langkah Berikutnya”, untuk menjalankan 1 langkah Merge Sort sekaligus memperbarui warna, log, dan visual array, untuk pertama panggil method resetHighlights dan cek apakah proses dan masih ada langkah merge, lalu ambil data tandai batas-batas kiri, tengah,dan kanan

```
temp = new int[right - left + 1];
i = left;
j = mid + 1;
k = 0;
copying = false;
isMerging = true;
```

12. Buat variabel untuk proses merge

```
logStep("Mulai merge range [" + left + ".." + right + "]");
highlightRange(left, right);
return;
```

13. Lalu kode ini Tampilkan pesan awal langkah ini, Warnai label dari left ke right (range sedang diproses).

```
if (isMerging && !copying) {
    highlightRange(left, right);

    if (i <= mid && j <= right) {
        highlightCompare(i, j);
        if (array[i] <= array[j]) {
            logStep("Bandingkan " + array[i] + " <= " + array[j] + ". Salin " + array[i] + " ke temp.");
            temp[k++] = array[i++];
        } else {
            logStep("Bandingkan " + array[i] + " > " + array[j] + ". Salin " + array[j] + " ke temp.");
            temp[k++] = array[j++];
        }
        stepArea.append("  -> Temp: " + Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
        return;
```

14. Proses merging, pertama cek apakah sedang melakukan dan belum menyalin array, lalu cek apakah ‘j’ dan ‘I’ masih dalam batas sub array, dan bandingkan dua elemen lalu nilai yang lebih kecil salin ke dalam temp, lalu tampilkan isi temp saat ini

```
if (i <= mid) {
    logStep("Salin sisa dari kiri: " + array[i]);
    temp[k++] = array[i++];
    stepArea.append("  -> Temp: " + Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
    return;
}
if (j <= right) {
    logStep("Salin sisa dari kanan: " + array[j]);
    temp[k++] = array[j++];
    stepArea.append("  -> Temp: " + Arrays.toString(Arrays.copyOf(temp, k)) + "\n\n");
    return;
```

15. Salin sisa dari kiri dan kanan, sampai salah satu sub array sudah habis lanjut ke sisi lainnya

```
copying = true;
k = 0;
logStep("Selesai mengisi temp. Mulai menyalin kembali ke array utama.");
stepArea.append("  -> Temp Lengkap: " + Arrays.toString(temp) + "\n");
return;
```

16. Jika semua data sudah ada pada temp, maka siap untuk disalin kembali ke array

```
if (isMerging && copying) {
    if (k < temp.length) {
        logStep("Salin " + temp[k] + " dari temp ke posisi " + (left + k));
        array[left + k] = temp[k];
        updateLabels();
        highlightCopy(left + k);
        k++;
        return;
    } else {
        isMerging = false;
        copying = false;
        logStep("Selesai merge range [" + left + ".." + right + "]");
```

17. Lalu salin satu per satu dari temp ke array dan tampilkan proses indikasi penyalinan, lalu update tampilan label, dan tandai selesai proses merge untuk sub array ini

```
if (mergeQueue.isEmpty() && !isMerging) {
    stepArea.append("\n--- MERGE SORT SELESAI ---\n");
    stepArea.append("Hasil Akhir: " + Arrays.toString(array) + "\n");
    stepButton.setEnabled(false);
    JOptionPane.showMessageDialog(this, "Merge Sort selesai!");
    for (JLabel label : labelArray)
        label.setBackground(Color.LIGHT_GRAY);
```

18. Lalu jika semua merge selesai, tampilkan log hasil, dan tandai semua label sebagai tanda finalisasi

```
private void setArrayFromInput() {
    String text = inputField.getText().trim();
    if (text.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Input tidak boleh kosong!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
```

19. Method setArrayFromInput, pertama ambil teks dari inputField, dan jika input kosong, tampilkan pesan salah dan berhentikan proses

```
String[] parts = text.split(",");
array = new int[parts.length];
try {
    for (int i = 0; i < parts.length; i++) {
        array[i] = Integer.parseInt(parts[i].trim());
    }
} catch (NumberFormatException e) {
    JOptionPane.showMessageDialog(this, "Masukkan hanya angka!", "Error", JOptionPane.ERROR_MESSAGE);
    return;
```

20. Pecahkan input menggunakan koma, dan buat array untuk menampung hasil, lalu lakukan proses konversi elemen menjadi integer, dan jika inputan ada yang bukan angka, maka tampilkan pesan eror

```
stepCount = 1;
isMerging = false;
copying = false;
if (mergeQueue != null) {
    mergeQueue.clear();
```

21. Untuk mereset status langkah dan flag agar mulai dari awal lagi, lalu kosongkan antrian merge sebelumnya

```
panelArray.removeAll();
stepArea.setText("");
```

22. Untuk mereset tampilan visual array, hapus semua elemen lama dari panelArray dan kosongkan log langkah di stepArea

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

23. Untuk menampilkan label untuk setiap elemen array

```
generateMergeSteps(0, array.length - 1);

stepButton.setEnabled(true);
stepArea.setText("Array Awal: " + Arrays.toString(array) + "\n\n");

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

24. Untuk menyiapkan langkah-langkah merge, aktifkan stepButton dan tampilkan array awal, lalu terakhir refresh panel

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

25. Method generateMergeSteps, untuk membuat urutan langkah-langkah merge, pertama buat basis dan hitungindeks tengah, lalu pecah bagian kiri hingga sampai tinggal satu elemen , lalu pecah bagian kanan, dan simpan langkah merge di mergeQueue

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

26. Method resetHighlights untuk menghapus kembali warna yang ditandai sebelumya

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

27. Method updateLabels untuk mengupdate isi dari labelArray

```
private void highlightRange(int l, int r) {
    for (int i = l; i <= r; i++) {
        labelArray[i].setBackground(Color.ORANGE);
    }
```

28. Method highlightRange, untuk Menyorot rentang elemen dari indeks ‘l’ sampai ‘r’ (inklusif) dengan warna oren**.**

```
private void highlightCompare(int i, int j) {
    labelArray[i].setBackground(Color.CYAN);
    labelArray[j].setBackground(Color.CYAN);
}
```

29. Method highlightCompare, Untuk menyorot dua elemen yang sedang dibandingkan dengan warna cyan

```
private void highlightCopy(int i) {
    labelArray[i].setBackground(Color.GREEN);
}
```

30. Method highlightCopy, untuk Menyorot satu elemen saat sedang disalin dari array sementara (temp[]**)** kembali ke array utama**.**

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

31. Method reset, untuk mereset ulang seluruh GUI agar bias memulai dari awal

```
private void logStep(String message) {
    stepArea.append("Langkah " + stepCount + ": " + message + "\n");
    if (array != null) {
        stepArea.append("  Array: " + Arrays.toString(array) + "\n\n");
    }
    stepArea.setCaretPosition(stepArea.getDocument().getLength());
    stepCount++;
```

32. Method logStepuntuk mencatat langkah-langkah yang sedang dilakukan algoritma ke area teks (stepArea), lengkap dengan isi array saat itu.

c. QuickSort

```
public class QuickSort extends JFrame {

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

    private int i = 0, j = 0;
    private boolean sorting = false;
    private int stepCount = 1;
```

1. Sama seperti kelas-kelas sebelumnya untuk GUI dari Quick Sort ini masih sama, Pertama buat class QuickSort, lalu buatlah array yang bertipe data Integer, dan deklarasikan fitur-fitur yang ingin dipakai pada GUI seperti JLabel, JButton, JTextField, JPanel, JTextArea, lalu buat variabel untuk penanda proses dalam insertion sort, dan buat penanda posisi dalam insertion sort, lalu buat penanda apakah sortingnya sedang berlangsung

```
private Stack<int[]> stack;
private int low, high, pivot;
private boolean partitioning = false;
```

2. Buat variabel untuk QuickSort, pertama buat stack untuk menyimpan rentang partisi, lalu buat batas indeks partisi dan nilai pivot, terakhir buat index ‘i’, ‘j’ untuk iterasi partisi, dan buat partitioning untuk melihat status apakah dalam proses partisi

```
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();
            }
```

3. Method main, untuk menjalankan program GUI di java agar dapat berjalan dalam thread GUI

```
public QuickSort() {
    setTitle("Quick Sort Langkah per Langkah");
    setSize(850, 450);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    setLayout(new BorderLayout(10, 10));

    stack = new Stack<>();
```

4. Membuat window utama untuk Jframe, dan membuat stack untuk menyimpan rantang dari partisi

```
JPanel topPanel = new JPanel(new BorderLayout());
JPanel inputPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
inputField = new JTextField(30);
setButton = new JButton("Set Array");
inputPanel.add(new JLabel("Masukkan angka (pisahkan dengan koma):"));
inputPanel.add(inputField);
inputPanel.add(setButton);
topPanel.add(inputPanel, BorderLayout.CENTER);
```

5. Untuk menambahkan Panel untuk menginputkan Array dan mengaktifkan button SetArray

```
panelArray = new JPanel();
panelArray.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
```

6. Menambahkan panel untuk menampilkan nilai-nilai array

```
JPanel controlPanel = new JPanel();
stepButton = new JButton("Langkah Selanjutnya");
resetButton = new JButton("Reset");
stepButton.setEnabled(false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

7. Menambahkan panel untuk mengontrol proses sorting yang mana disini terdiri dari dua tombol yaitu StepButton untuk menjalankan langkah dari sorting dan resetButton untuk mereset.

```
JPanel logPanel = new JPanel(new BorderLayout());
logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));

stepArea = new JTextArea(8, 60);
stepArea.setEditable(false);
stepArea.setFont(new Font("Monospaced", Font.PLAIN, 14));
JScrollPane scrollPane = new JScrollPane(stepArea);
logPanel.add(scrollPane, BorderLayout.CENTER);
```

8. Menambahkan panel untuk menampilkan langkah-langkah dari sorting, Menampilkan log proses Quick Sort langkah demi langkah, Menjaga log bisa dibaca dan discroll, Mencegah log diedit oleh pengguna.

```
add(topPanel, BorderLayout.NORTH);
add(panelArray, BorderLayout.CENTER);
add(controlPanel, BorderLayout.SOUTH);
add(logPanel, BorderLayout.EAST);
```

9. Lalu terakhir tambahkan semua panel ke Jframe sesuai dengan posisinya

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

10. Kode ini untuk menambahkan EventListener agar Tombol bisa berfungsi

```
private void performStep() {

    if (stack.isEmpty() && !partitioning) {
        if(sorting){
            sorting = false;
            stepButton.setEnabled(false);
            resetHighlights(true); // Warnai semua jadi abu-abu
            stepArea.append("\nQuick Sort selesai.\n");
            JOptionPane.showMessageDialog(this, "Quick Sort selesai!");
        }
        return;
```

11. MethodperformStep, dijalankan setiap kali pengguna menekan tombol “Langkah Berikutnya”, untuk menjalankan 1 langkah QuickSort dan mendemonstrasikan partisi array berdasarkan algoritma Quick Sort. Pertama cek kondisi sorting jika stack kosong maka panggil method `resetHighlights(false);` untuk menghapus highlight sementara

```
if (!partitioning) {
    int[] range = stack.pop();
    low = range[0];
    high = range[1];

    if(low >= high) {
        performStep();
        return;
```

12. Lalu mulai proses partisi baru, ambil subarray terbaru dari stack, lalu buat batas kiri dan kanan dari subarray, dan jika hanya 1 elemen atau tidak ada elemen, lewati dan lanjut ke langkah berikutnya

```
pivot = array[high];
i = low - 1;
j = low;
partitioning = true;
stepArea.append("--- Mulai Partisi ---\n");
logStep("Range: [" + low + ".." + high + "], Pivot: " + pivot + " (di indeks " + high +")");
highlightPivot(high);
return;
```

13. Lalu set pivot ke elemen paling kanan, Inisialisasi variabel partisi i adalah indeks terakhir elemen kecil j adalah indeks pemindai, Tampilkan log dan highlight pivot.

```
if (j < high) {
    highlightCompare(j, high);
    if (array[j] <= pivot) {
        i++;
        logStep("Cek " + array[j] + " <= " + pivot + ". Benar. Pindahkan ke bagian kiri.");
        if (i != j) {
            logStep(" -> Tukar arr[" + i + "]=" + array[i] + " dengan arr[" + j + "]=" + array[j]);
            swap(i, j);
            updateLabels();
        }
    } else {
        logStep("Cek " + array[j] + " <= " + pivot + ". Salah. Lewatkan.");
    }
    j++;
    return;
```

14. Proses perbandingan elemen dengan pivot, pertama bandingkan ‘j’ dengan pivot dan Highlight elemen yang sedang dibandingkan, dan jika ‘j’ lebih kecil dari pivot maka tambah i, lalu tukar arr[i] dan arr[j] (memindahkan elemen ke sisi kiri), dan updateLabel, dan jika lebih besar, lewati.

```
int pivotFinalIndex = i + 1;
logStep("Partisi selesai. Pindahkan pivot " + pivot + " ke posisi finalnya di indeks " + pivotFinalIndex);
swap(pivotFinalIndex, high);
updateLabels();

partitioning = false;

labelArray[pivotFinalIndex].setBackground(Color.LIGHT_GRAY);

stepArea.append("--- Selesai Partisi ---\n\n");

stack.push(new int[] {low, pivotFinalIndex - 1});
stack.push(new int[] {pivotFinalIndex + 1, high});
```

15. Setelah selesai scan semua elemen, tempatkan pivot di posisi yang benar, dan tandai partisi yang selesai, lalu tambahkan subarray kiri dan kanan (jika ada) ke stack untuk diproses selanjutnya.

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

16. Method highlightPivot, untuk menandai elemen pivot saat proses partisi

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

17. Method highlightCompare, untuk menandai elemen yang sedang dibandingkan dengan pivot

```
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 < 2) {
        JOptionPane.showMessageDialog(this, "Masukkan setidaknya dua angka!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
```

18. Method setArrayFromInput, pertama ambil teks dari inputField, dan jika input kosong, tampilkan pesan salah dan berhentikan proses, Pecahkan input menggunakan koma, dan buat array untuk menampung hasil dan jika inputan ada yang bukan angka, maka tampilkan pesan eror

```
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();
```

19. Membuat array int[] sebanyak jumlah inputan, dan konversi setiap elemen string menjadi integer, dan jika terjadi kesalahan dalam input maka tampilkan pesan eror,

dan reset tampilan visual array, hapus semua elemen lama dari panelArray dan kosongkan log langkah di stepArea

```
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].setBorder(BorderFactory.createLineBorder(Color.BLACK));
    labelArray[k].setPreferredSize(new Dimension(50, 50));
    labelArray[k].setHorizontalAlignment(SwingConstants.CENTER);
    labelArray[k].setOpaque(true);
    labelArray[k].setBackground(Color.WHITE);
    panelArray.add(labelArray[k]);
```

20. Untuk menampilkan label untuk setiap elemen array

```
stack.clear();
stack.push(new int[] {0, array.length - 1});
sorting = true;
partitioning = false;
stepCount = 1;
stepArea.setText("Array Awal: " + java.util.Arrays.toString(array) + "\n\n");
stepButton.setEnabled(true);

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

21. Untuk menghapus isi stack sebelumnya dan push array dengan dua elemen ke dalam stack sebagai batas awal array dan inisialisasi ulang langkah, dan Memerintahkan Swing untuk menghitung ulang tata letak panel dan menggambar ulang isi panel.

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

22. Method updateLabels untuk mengupdate isi dari labelArray

```
private void resetHighlights(boolean markAsSorted) {
    if (labelArray == null) return;
    for (JLabel label : labelArray) {
        if (markAsSorted || label.getBackground() == Color.LIGHT_GRAY) {
            label.setBackground(Color.LIGHT_GRAY);
        } else {
            label.setBackground(Color.WHITE);
        }
```

23. Method resetHighlights untuk menghapus kembali warna yang ditandai sebelumya

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

24. Method Swap, untuk menukar elemen array a dan b

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

25. Method reset, untuk mereset ulang seluruh GUI agar bias memulai dari awal

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

26. Method logStepuntuk mencatat langkah-langkah yang sedang dilakukan algoritma ke area teks (stepArea), lengkap dengan isi array saat itu.

d. ShellSort

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

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

1. Sama seperti kelas-kelas sebelumnya untuk GUI dari Quick Sort ini masih sama, Pertama buat class QuickSort, lalu buatlah array yang bertipe data Integer, dan deklarasikan fitur-fitur yang ingin dipakai pada GUI seperti JLabel, JButton, JTextField, JPanel, JTextArea, lalu buat variabel untuk penanda proses dalam insertion sort, dan buat penanda posisi dalam Shell sort, lalu buat penanda apakah sortingnya sedang berlangsung

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

2. Method main, untuk menjalankan program GUI di java agar dapat berjalan dalam thread GUI

```
public ShellSort() {
    //Ahmad Zahran sudah lelah :(
    setTitle("Shell Sort Langkah per Langkah");
    setSize(750, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLocationRelativeTo(null);
    getContentPane().setLayout(new BorderLayout());
```

3. Membuat window utama untuk Jframe.

```
// 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);
```

4. Untuk menambahkan Panel untuk menginputkan Array dan mengaktifkan button SetArray

```
// Panel Array visual
panelArray = new JPanel();
panelArray.setLayout(new FlowLayout());
```

5. Menambahkan panel untuk menampilkan nilai-nilai array

```
// Panel Kontrol
JPanel controlPanel = new JPanel();
stepButton = new JButton("Langkah Selanjutnya");
resetButton = new JButton("Reset");
stepButton.setEnabled(false);
controlPanel.add(stepButton);
controlPanel.add(resetButton);
```

6. Menambahkan panel untuk mengontrol proses sorting yang mana disini terdiri dari dua tombol yaitu StepButton untuk menjalankan langkah dari sorting dan resetButton untuk mereset.

```
JPanel logPanel = new JPanel(new BorderLayout());
logPanel.setBorder(BorderFactory.createTitledBorder("Log Langkah"));

// 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);
```

7. Menambahkan panel untuk menampilkan langkah-langkah dari sorting, Menampilkan log proses Quick Sort langkah demi langkah, Menjaga log bisa dibaca dan discroll, Mencegah log diedit oleh pengguna.

```
// Tambahkan Panel ke Frame
getContentPane().add(inputPanel, BorderLayout.NORTH);
getContentPane().add(panelArray, BorderLayout.CENTER);
getContentPane().add(controlPanel, BorderLayout.SOUTH);
getContentPane().add(scrollPane, BorderLayout.EAST);
```

8. Lalu terakhir tambahkan semua panel ke Jframe sesuai dengan posisinya

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

9. Kode ini untuk menambahkan EventListener agar Tombol bisa berfungsi

```
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 koma!", "Error", JOptionPane.ERROR_MESSAGE);
        return;
```

10. Method setArrayFromInput, pertama ambil teks dari inputField, dan jika input kosong fungsi akan langsung di return, Pecahkan input menggunakan koma, dan buat array untuk menampung hasil dan jika inputan ada yang bukan angka, maka tampilkan pesan eror

```
gap = array.length / 2;
i = gap;
sorting = true;
stepCount = 1;
stepArea.setText("");
stepButton.setEnabled(true);
panelArray.removeAll();
```

11. Kode ini digunakan sebagai bagian dari inisialisasi awal sebelum memulai proses sorting langkah per langkah

```
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]);
```

12. Kode ini untuk membuat dan menampilkan setiap elemen yang diwakili oleh label

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

13. Lalu hitung ulang setiap langkah, dan memastikan bahwa perubahan langsung di tampilkan di panel array

```
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;
```

14. Method performStep, untuk melakukan satu langkah proses Shell Sort pada array, pertama panggil method resetHighlights, laluu cek kondsi apakah proses sorting sudah selesai, jika sudah Tambahkan teks ke area log stepArea bahwa sorting selesai, Disable tombol stepButton agar tidak bisa ditekan lagi, Tampilkan hasil akhir array pada stepArea, keluar dari fungsi, tidak ada langkah berikutnya.

```
if (i < array.length) {
    if (!isSwapping) {
        temp = array[i];
        j = i;
        isSwapping = true;
```

15. Jika indeks i masih dalam batas panjang array, maka proses sorting dengan gap saat ini masih berjalan, dan Jika belum mulai swapping simpan nilai ke temp dan set j = I sebagai indeks untuk pergeseran elemen

```
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;
```

16. Kode ini untuk proses pergeseran elemen pada shell sort

```
} else {
    array[j] = temp; // letakkan nilai temp
    updateLabels();
    logStep("Tempatkan " + temp + " di posisi " + j);
    i++;
    isSwapping = false;
```

17. Jika kondisi di atas tidak terpenuhi (artinya tidak perlu geser lagi)

```
} else {
    gap /= 2;
    i = gap;
    isSwapping = false;
    stepArea.append("Langkah " + stepCount++ + ": Kurangi gap menjadi " + gap + "\n\n");
```

18. Jika i sudah melewati panjang array, artinya satu putaran Shell Sort dengan gap saat ini sudah selesai, Kurangi gap menjadi setengahnya (gap /= 2) untuk putaran berikutnya, sesuai algoritma Shell Sort.

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

19. Method logStepuntuk mencatat langkah-langkah yang sedang dilakukan algoritma ke area teks (stepArea), lengkap dengan isi array saat itu.

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

20. Method updateLabels untuk mengupdate isi dari labelArray

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

21. Method resetHighlights untuk menghapus kembali warna yang ditandai sebelumya

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

22. Method reset, untuk mereset ulang seluruh GUI agar bias memulai dari awal

D. Kesimpulan

Melalui praktikum ini, mahasiswa memperoleh pemahaman yang lebih dalam tentang bagaimana struktur data Sorting bekerja. Dan bagaimana cara kerja sorting bekerja dengan visualisasi tiap-tiap langkah dari sorting tersebut