**LAPORAN TUGAS BESAR**

**PEMROGRAMAN BERORIENTASI OBJEK**

**SORTING ALGORITMA**

# HALAMAN JUDUL



**Disusun Oleh :**

**NAMA : AHMAD FAIZ SETIAWANTORO**

**NIM : 32602200034**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM SULTAN AGUNG**

**SEMARANG**

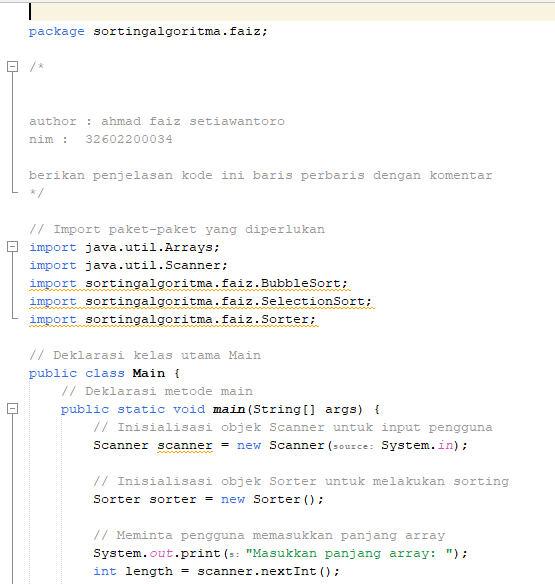
**2022**

**DAFTAR**

# BAB I Struktur Program

## **File Utama ( Main )**

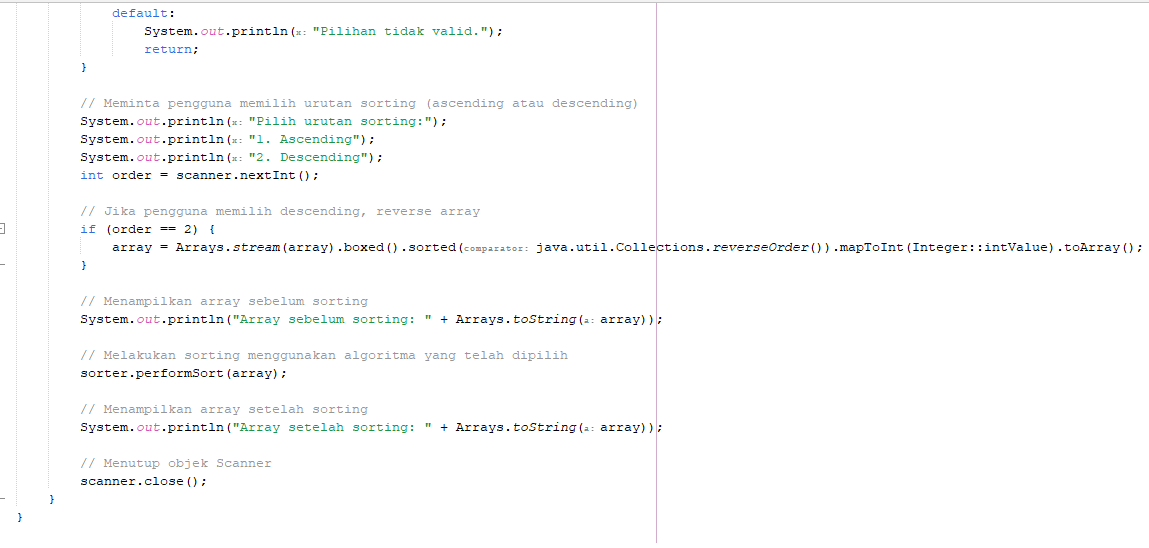
File utama berisi logika untuk pengguna memilih algoritma dan urutan sorting.



Gambar 1. 1 File Main 1



Gambar 1. 2 File Main 2



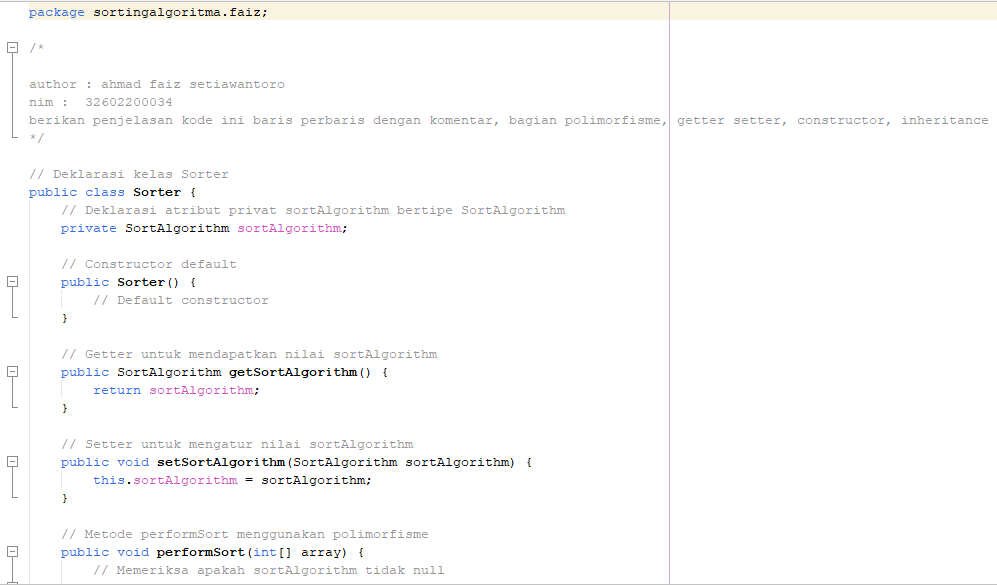
Gambar 1. 3 File Main 3

Penjelasan:

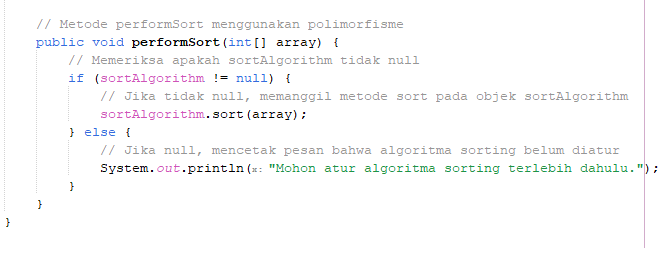
1. public class Main {: Mendeklarasikan kelas utama bernama Main.
2. public static void main(String[] args) {: Mendeklarasikan metode utama (main) yang akan dieksekusi saat program dijalankan.
3. Scanner scanner = new Scanner(System.in);: Membuat objek Scanner untuk membaca input dari pengguna.
4. Sorter sorter = new Sorter();: Membuat objek sorter dari kelas Sorter untuk melakukan proses sorting.
5. System.out.print("Masukkan panjang array: ");: Menampilkan pesan untuk meminta pengguna memasukkan panjang array.
6. int length = scanner.nextInt();: Membaca panjang array yang dimasukkan pengguna.
7. int[] array = new int[length];: Membuat array dengan panjang yang telah dimasukkan pengguna.
8. System.out.println("Masukkan elemen array:");: Menampilkan pesan untuk meminta pengguna memasukkan elemen-elemen array.
9. for (int i = 0; i < length; i++) {: Looping untuk meminta pengguna memasukkan nilai setiap elemen array.
10. System.out.print("Elemen ke-" + (i + 1) + ": ");: Menampilkan pesan untuk meminta pengguna memasukkan nilai elemen tertentu.
11. array[i] = scanner.nextInt();: Membaca nilai elemen array yang dimasukkan pengguna.
12. System.out.println("Pilih algoritma sorting:");: Menampilkan pesan untuk meminta pengguna memilih algoritma sorting.
13. System.out.println("1. Bubble Sort");: Menampilkan pilihan algoritma sorting Bubble Sort.
14. System.out.println("2. Selection Sort");: Menampilkan pilihan algoritma sorting Selection Sort.
15. int choice = scanner.nextInt();: Membaca pilihan algoritma sorting yang dimasukkan pengguna.
16. switch (choice) {: Memulai struktur switch untuk memilih algoritma sorting berdasarkan pilihan pengguna.
17. case 1:: Jika pilihan adalah 1 (Bubble Sort), maka melakukan langkah selanjutnya.
18. sorter.setSortAlgorithm(new BubbleSort());: Menetapkan algoritma sorting pada objek sorter menjadi Bubble Sort.
19. break;: Keluar dari struktur switch.
20. case 2:: Jika pilihan adalah 2 (Selection Sort), maka melakukan langkah selanjutnya.
21. sorter.setSortAlgorithm(new SelectionSort());: Menetapkan algoritma sorting pada objek sorter menjadi Selection Sort.
22. break;: Keluar dari struktur switch.
23. default:: Jika pilihan tidak valid, maka melakukan langkah selanjutnya.
24. System.out.println("Pilihan tidak valid.");: Menampilkan pesan bahwa pilihan tidak valid.
25. return;: Menghentikan eksekusi program.
26. }: Akhir dari struktur switch.
27. System.out.println("Pilih urutan sorting:");: Menampilkan pesan untuk meminta pengguna memilih urutan sorting.
28. System.out.println("1. Ascending");: Menampilkan pilihan urutan sorting ascending.
29. System.out.println("2. Descending");: Menampilkan pilihan urutan sorting descending.
30. int order = scanner.nextInt();: Membaca pilihan urutan sorting yang dimasukkan pengguna.
31. if (order == 2) {: Memulai struktur kondisional jika urutan sorting adalah descending.
32. array = Arrays.stream(array).boxed().sorted(java.util.Collections.reverseOrder()).mapToInt(Integer::intValue).toArray();: Melakukan reverse array untuk descending order.
33. }: Akhir dari struktur kondisional.
34. System.out.println("Array sebelum sorting: " + Arrays.toString(array));: Menampilkan array sebelum proses sorting.
35. sorter.performSort(array);: Melakukan sorting menggunakan algoritma yang telah dipilih.
36. System.out.println("Array setelah sorting: " + Arrays.toString(array));: Menampilkan array setelah proses sorting.
37. scanner.close();: Menutup objek Scanner untuk menghindari resource leak.

## **Sorter**

Kelas yang menggunakan inheritance dan encapsulation untuk melakukan sorting..



Gambar 1. 4 Kelas Sorter 1



Gambar 1. 5 Kelas Sorter 2

Penjelasan:

1. Polimorfisme:

Polimorfisme terlihat pada penggunaan antarmuka SortAlgorithm. Kelas Sorter memiliki atribut sortAlgorithm yang memiliki tipe data antarmuka SortAlgorithm. Ini memungkinkan objek dari kelas yang mengimplementasikan antarmuka SortAlgorithm untuk diatur pada atribut ini, sehingga metode sort dapat dipanggil pada objek tersebut melalui polimorfisme.

1. Getter dan Setter:

Getter (getSortAlgorithm): Metode ini digunakan untuk mendapatkan nilai dari atribut sortAlgorithm.

Setter (setSortAlgorithm): Metode ini digunakan untuk mengatur nilai atribut sortAlgorithm. Dengan adanya setter, kita dapat mengubah algoritma sorting yang digunakan oleh objek Sorter saat program berjalan.

1. Constructor:

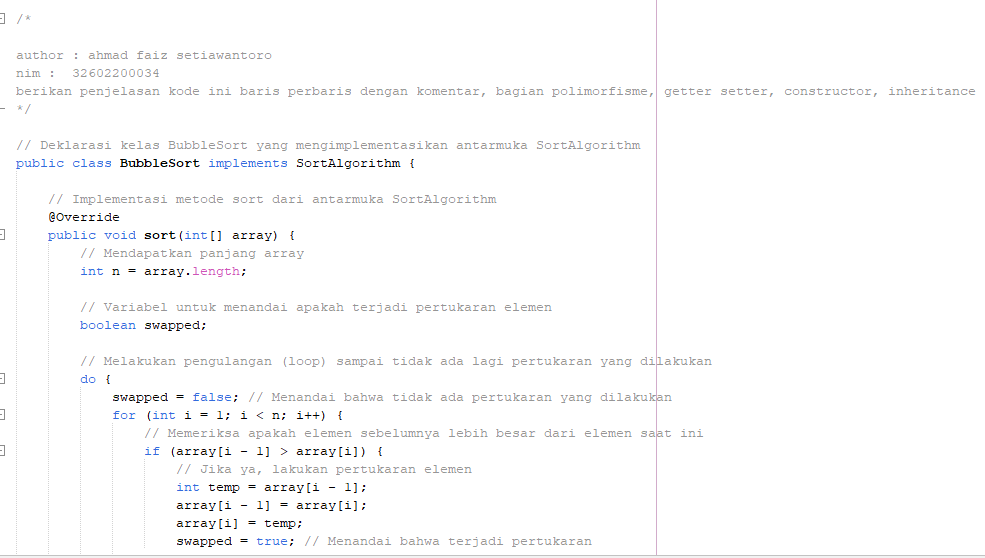
Constructor default (public Sorter()): Constructor ini tidak menerima parameter dan digunakan untuk membuat objek dari kelas Sorter. Dalam hal ini, constructor default hanya memberikan nilai awal yang sederhana pada objek.

1. Inheritance:

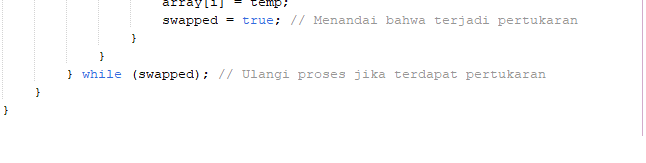
Tidak ada konsep inheritance yang terlihat dalam kelas Sorter. Namun, kelas Sorter menggunakan polimorfisme melalui antarmuka SortAlgorithm, yang dapat diimplementasikan oleh kelas-kelas lain seperti BubbleSort dan SelectionSort. Melalui antarmuka ini, objek Sorter dapat berinteraksi dengan berbagai algoritma sorting tanpa bergantung pada implementasi spesifik dari setiap algoritma.

## **BubbleSort**

Implementasi algoritma Bubble Sort yang mengimplementasikan SortAlgorithm.



Gambar 1. 6 Kelas BubleSort 1



Gambar 1. 7 Kelas BubleSort 2

Penjelasan:

1. Polimorfisme:

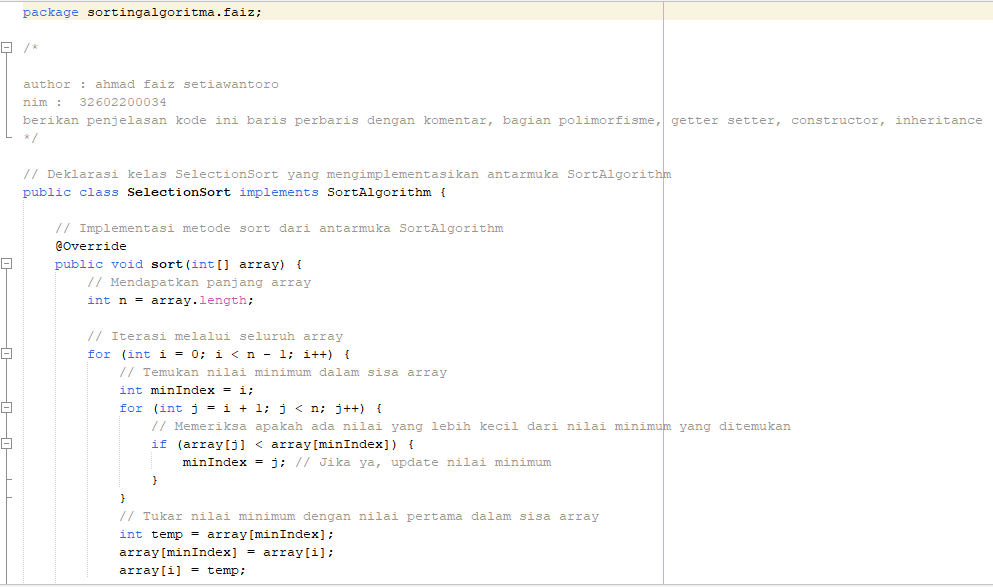
Kelas BubbleSort mengimplementasikan antarmuka SortAlgorithm, sehingga dapat digunakan sebagai objek dengan tipe data SortAlgorithm. Ini memungkinkan objek BubbleSort digunakan di mana pun objek SortAlgorithm dibutuhkan, seperti pada atribut sortAlgorithm dalam kelas Sorter.

1. Getter dan Setter, Constructor, Inheritance:

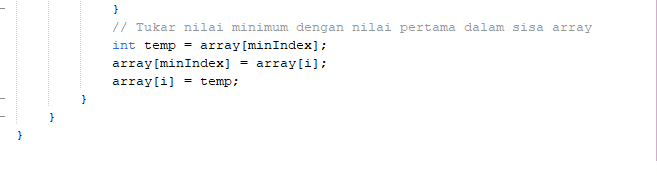
Tidak ada konsep getter, setter, atau constructor dalam kelas BubbleSort. Kelas ini fokus pada implementasi algoritma sorting dan tidak memiliki atribut yang memerlukan getter, setter, atau konstruktor. Inheritance terjadi melalui implementasi antarmuka SortAlgorithm.

## **SelectionSort**

Implementasi algoritma Selection Sort yang mengimplementasikan SortAlgorithm.



Gambar 1. 8 Kelas SelectionSort 1



Gambar 1. 9 Kelas SelectionSort 2

Penjelasan:

1. Polimorfisme:

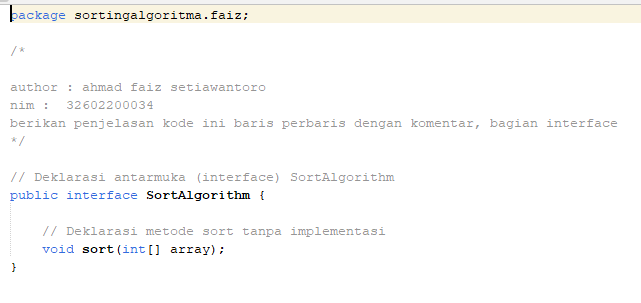
Kelas SelectionSort mengimplementasikan antarmuka SortAlgorithm, sehingga dapat digunakan sebagai objek dengan tipe data SortAlgorithm. Ini memungkinkan objek SelectionSort digunakan di mana pun objek SortAlgorithm dibutuhkan, seperti pada atribut sortAlgorithm dalam kelas Sorter.

1. Getter dan Setter, Constructor, Inheritance:

Tidak ada konsep getter, setter, atau constructor dalam kelas SelectionSort. Kelas ini fokus pada implementasi algoritma sorting dan tidak memiliki atribut yang memerlukan getter, setter, atau konstruktor. Inheritance terjadi melalui implementasi antarmuka SortAlgorithm.

## **Interface SortAlgorithm**

Interface yang menyediakan kerangka kerja untuk algoritma sorting.



Gambar 1. 10 Interface SortAlgorithm

Penjelasan:

1. Interface:

public interface SortAlgorithm: Mendeklarasikan sebuah antarmuka (interface) dengan nama SortAlgorithm. Interface adalah kumpulan dari metode-metode yang hanya memiliki deklarasi tanpa implementasi.

1. Metode dalam Interface:

void sort(int[] array);: Mendeklarasikan satu metode dalam antarmuka SortAlgorithm. Metode ini adalah sort yang menerima parameter array integer (int[] array) dan tidak memiliki implementasi di dalam antarmuka. Implementasi metode ini akan dilakukan oleh kelas-kelas yang mengimplementasikan antarmuka ini, seperti BubbleSort dan SelectionSort. Metode ini mempresentasikan aksi sorting yang akan dilakukan oleh kelas-kelas yang mengimplementasikannya.

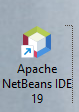
1. Tujuan Interface:

Antarmuka digunakan untuk menetapkan kontrak untuk kelas-kelas yang akan mengimplementasikannya. Dalam konteks ini, SortAlgorithm menjadi kontrak untuk kelas-kelas yang menyediakan implementasi algoritma sorting. Sebagai contoh, kelas BubbleSort dan SelectionSort harus menyediakan implementasi untuk metode sort yang dideklarasikan dalam antarmuka ini. Ini memastikan bahwa kelas-kelas tersebut mematuhi kontrak dan dapat digunakan secara interchangeably melalui tipe data SortAlgorithm

# BAB II Implementasi Program

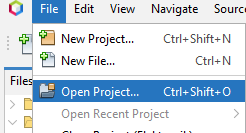
## **Menjalankan Program**

1. Untuk menjalankan program klick neatbeans

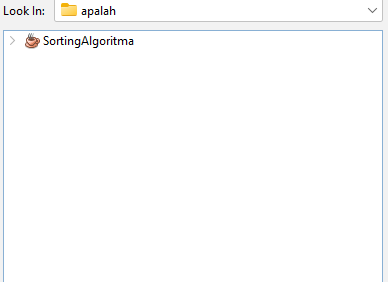


Gambar 2. 1 Open Neatbeans

1. Open new project, arahkan ke sortingalgoritma

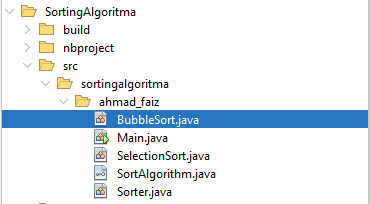


Gambar 2. 2 Open Projects



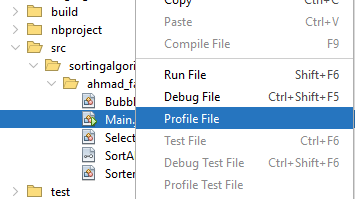
Gambar 2. 3 Arahkan ke project

1. Buka src/sortingalgoritma/ahmad\_faiz



Gambar 2. 4 Open struktur project

1. Klick file Main.java, lalu klick kanan run file atau shift + f6



Gambar 2. 5 Run program

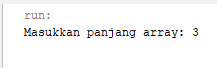
1. Tunggu program muncul



Gambar 2. 6 Tunggu program muncul

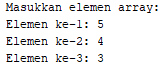
## **Mengoperasikan Sorting**

1. Isi berapa nilai Panjang array



Gambar 2. 7 Masukan Panjang array

1. Isi Elemen Array



Gambar 2. 8 Isi elemen array

1. Pilih Metode Bubble sort atau Selection Sort



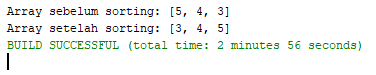
Gambar 2. 9 Pilih metode sorting

1. Pilih Urutan Ascending atau Descending



Gambar 2. 10 Pilih urutan sorting

1. Maka akan keluar hasil sorting



Gambar 2. 11 Hasil sorting array