Laporan Praktikum

Struktur Data



Disusun Oleh :

**MUHAMMAD FARREL GIOVANNI**

**NIM. 2311533006**

Dosen Pengampu : Dr. Wahyudi, S.T, M.T.

Departemen Informatika

Fakultas Teknologi Informasi

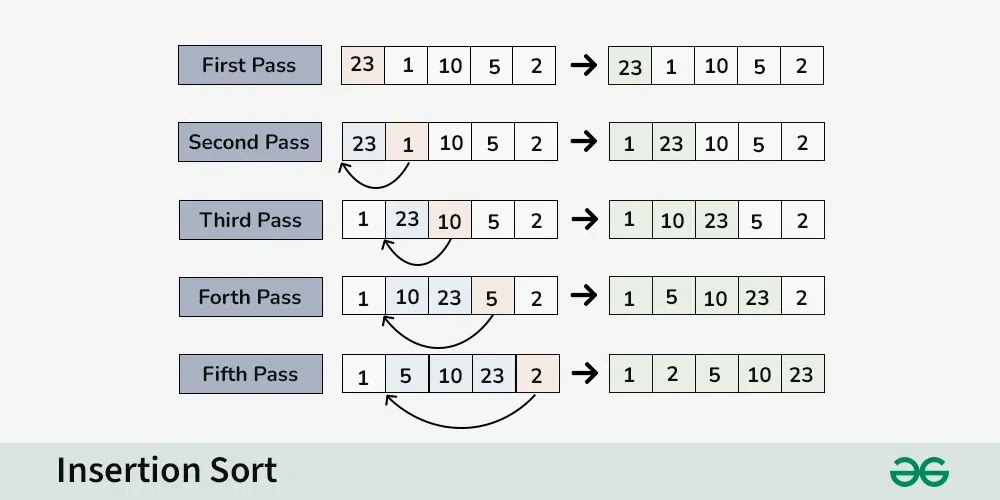
Universitas Andalas

Tahun 2024

**A. Pendahuluan**

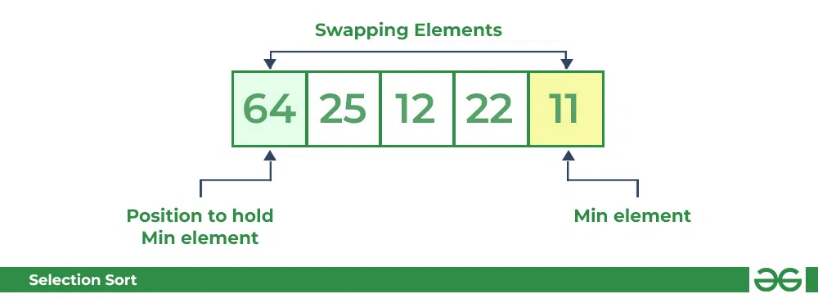
1. Insertion Sort

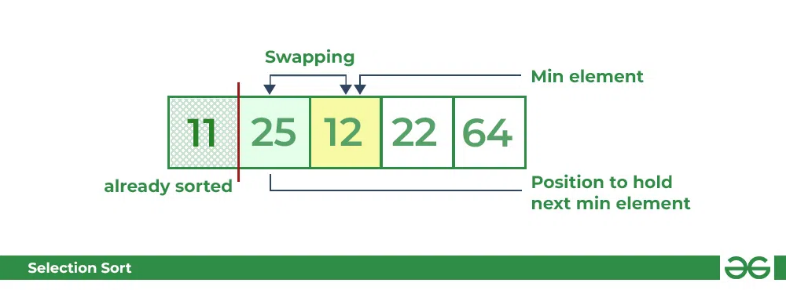
Metode penyisipan (Insertion sort) bertujuan untuk menjadikan bagian sisi kiri array terurutkan sampai dengan seluruh array berhasil diurutkan. Metode ini mengurutkan bilangan-bilangan yang telah dibaca dan berikutnya secara berulang akan menyisipkan bilangan- bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut.

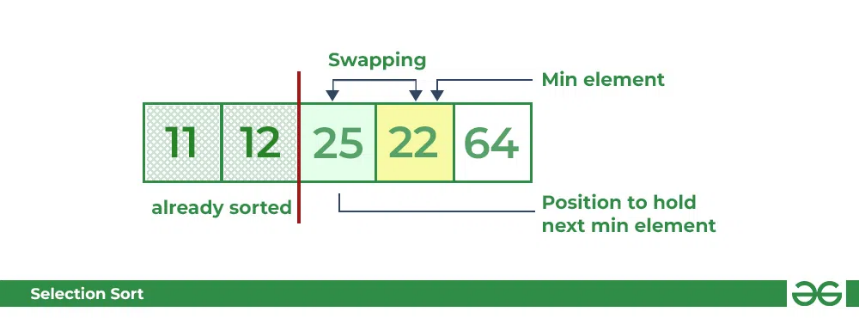


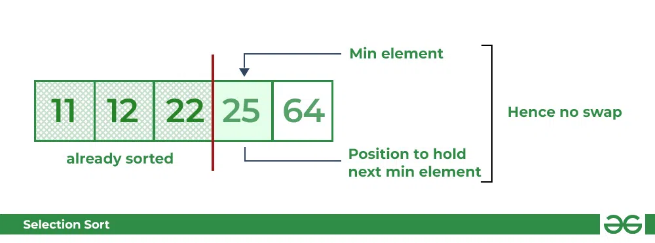
2. Selection Sort

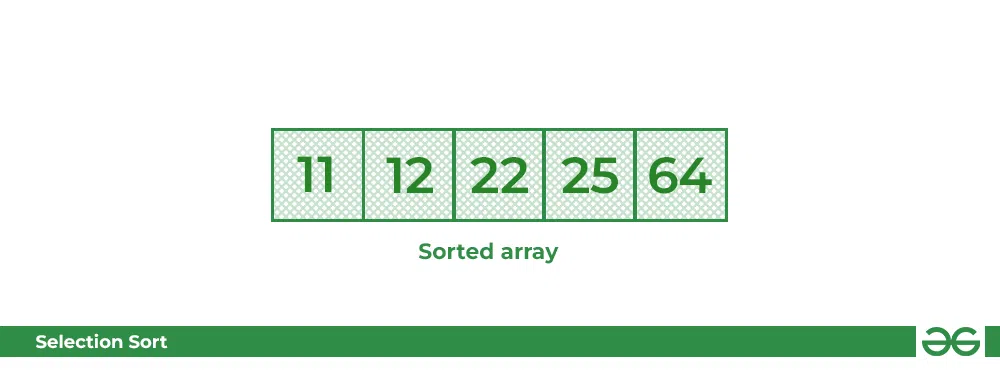
Algoritma berulang kali memilih elemen terkecil (atau terbesar) dari bagian daftar yang tidak diurutkan dan menukarnya dengan elemen pertama dari bagian daftar yang tidak diurutkan. Proses ini diulangi untuk sisa bagian yang belum disortir hingga seluruh daftar diurutkan.







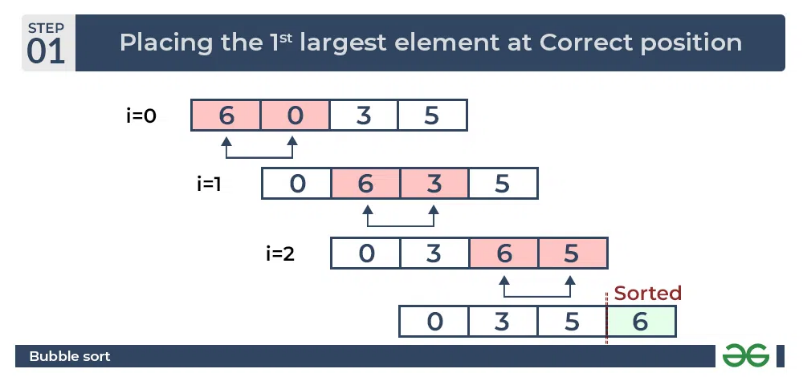


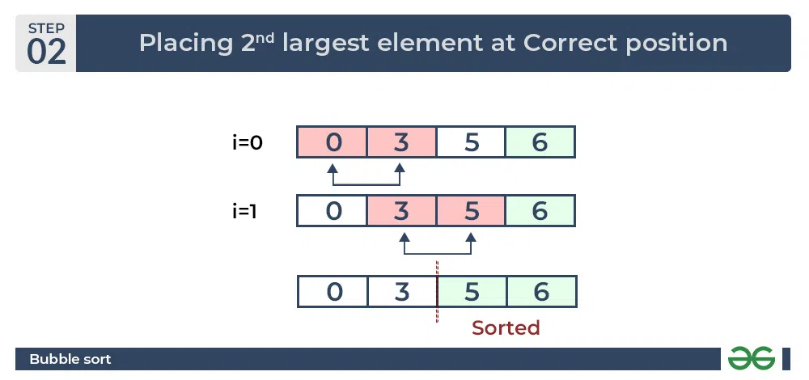


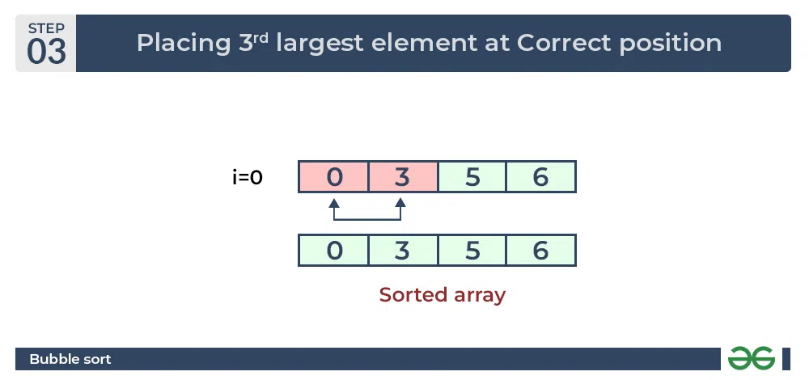
3. Bubble Sort

Metode gelembung (bubble sort) disebut dengan metode penukaran (exchange sort) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu.

Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.







**B. Tujuan**

1. Mengetahui cara menggunakan algoritma Insertion Sort pada java.

2. Mengetahui cara menggunakan algoritma Selection Sort pada java.

3. Mengetahui cara menggunakan algoritma Bubble Sort pada java.

**C. Metode Praktikum**

1. Insertion Sort metode printArray

public class InsertionSort {

//Function to sort array using insertion sort

void sort(int arr[]) {

int n = arr.length;

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

//Move elements of arr(0..i-1), that are greater then key

//to one position ahead of their current position

while (j >= 0 && arr[j] > key) {

arr[j+1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

static void printArray(int arr[]) {

int n = arr.length;

for (int i = 0; i < n; ++i) {

System.***out***.print(arr[i] + " ");

}

System.***out***.println();

}

//Driver method

public static void main(String[] args) {

int arr[] = {3,10,4,6,8,9,7,2,1,5};

InsertionSort ob = new InsertionSort();

ob.sort(arr);

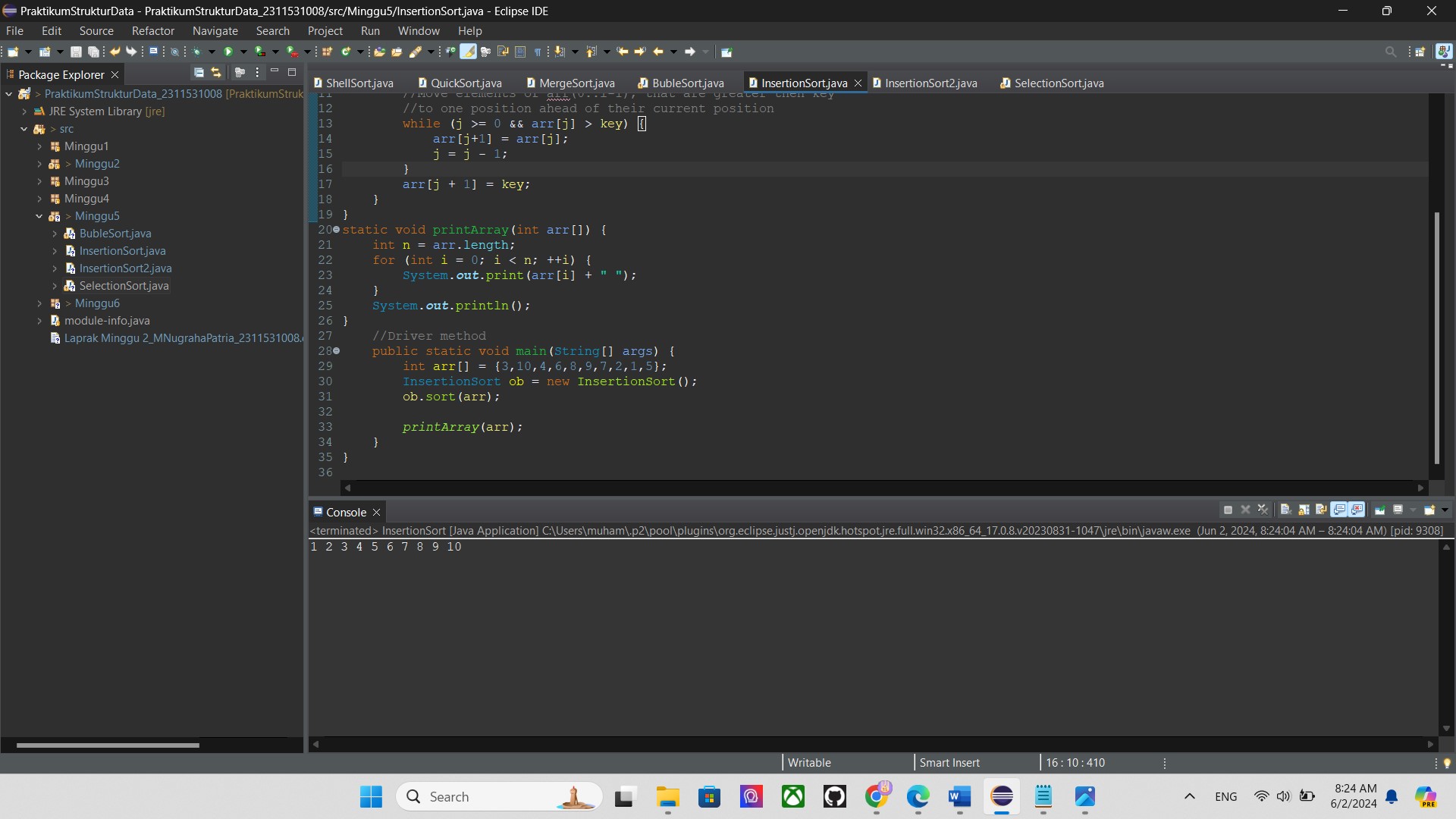
*printArray*(arr);

}

}

a. Penjelasan Algoritma  
Metode ini menerima array integer **arr** sebagai argumen dan mengurutkannya menggunakan algoritma Insertion Sort. Variabel **n** menyimpan panjang array. **Loop luar (for (int i = 1; i < n; i++))** literasi dimulai dari elemen kedua hingga elemen terakhir array. Variabel **key** menyimpan elemen saat ini yang akan ditempatkan pada posisi yang benar dalam bagian yang sudah diurutkan dari array. Variabel **j** digunakan untuk melacak indeks elemen yang dibandingkan dengan key. **Loop while (while (j >= 0 && arr[j] > key))** menggeser elemen-elemen yang lebih besar dari key ke satu posisi di depan posisi mereka saat ini. Setelah menemukan posisi yang benar untuk key, elemen tersebut ditempatkan pada posisi **arr[j + 1]**.

b. Inisialisasi  
Metode ini menerima array integer arr sebagai argumen dan mencetak setiap elemen array dengan spasi di antara elemen-elemen tersebut. Variabel **n** menyimpan panjang array. **Loop for** mencetak setiap elemen array. Pada metode **main**, inisialisasi array arr dengan beberapa nilai integer. Membuat objek InsertionSort dengan nama **ob**. Memanggil metode **sort** pada objek ob untuk mengurutkan **array arr**. Memanggil metode **printArray** untuk **mencetak array** yang sudah diurutkan.

Output:  


2. Insertion Sort metode main

public class InsertionSort2 {

public static void insertionSort(int[] arr){

int n = arr.length;

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

}

Metode ini menerima array integer **arr** sebagai argumen dan mengurutkannya menggunakan algoritma Insertion Sort. Variabel **n** menyimpan panjang array. **Loop luar (for (int i = 1; i < n; i++))** literasi dimulai dari elemen kedua hingga elemen terakhir array. Variabel **key** menyimpan elemen saat ini yang akan ditempatkan pada posisi yang benar dalam bagian yang sudah diurutkan dari array. Variabel **j** digunakan untuk melacak indeks elemen yang dibandingkan dengan key. **Loop while (while (j >= 0 && arr[j] > key))** menggeser elemen-elemen yang lebih besar dari **key** ke satu posisi di depan posisi mereka saat ini. Setelah menemukan posisi yang benar untuk **key**, elemen tersebut ditempatkan pada posisi **arr[j + 1]**.

public static void main(String[] args) {

int arr[] = {23, 78, 45, 8, 32, 56, 1};

int n = arr.length;

System.***out***.printf("array yang belum terurut:\n");

for (int i = 0; i < n; i++)

System.***out***.print(arr[i] + " ");

System.***out***.println("");

*insertionSort*(arr);

System.***out***.printf("array yang terurut:\n");

for (int i = 0; i < n; i++) {

System.***out***.printf(arr[i] + " ");

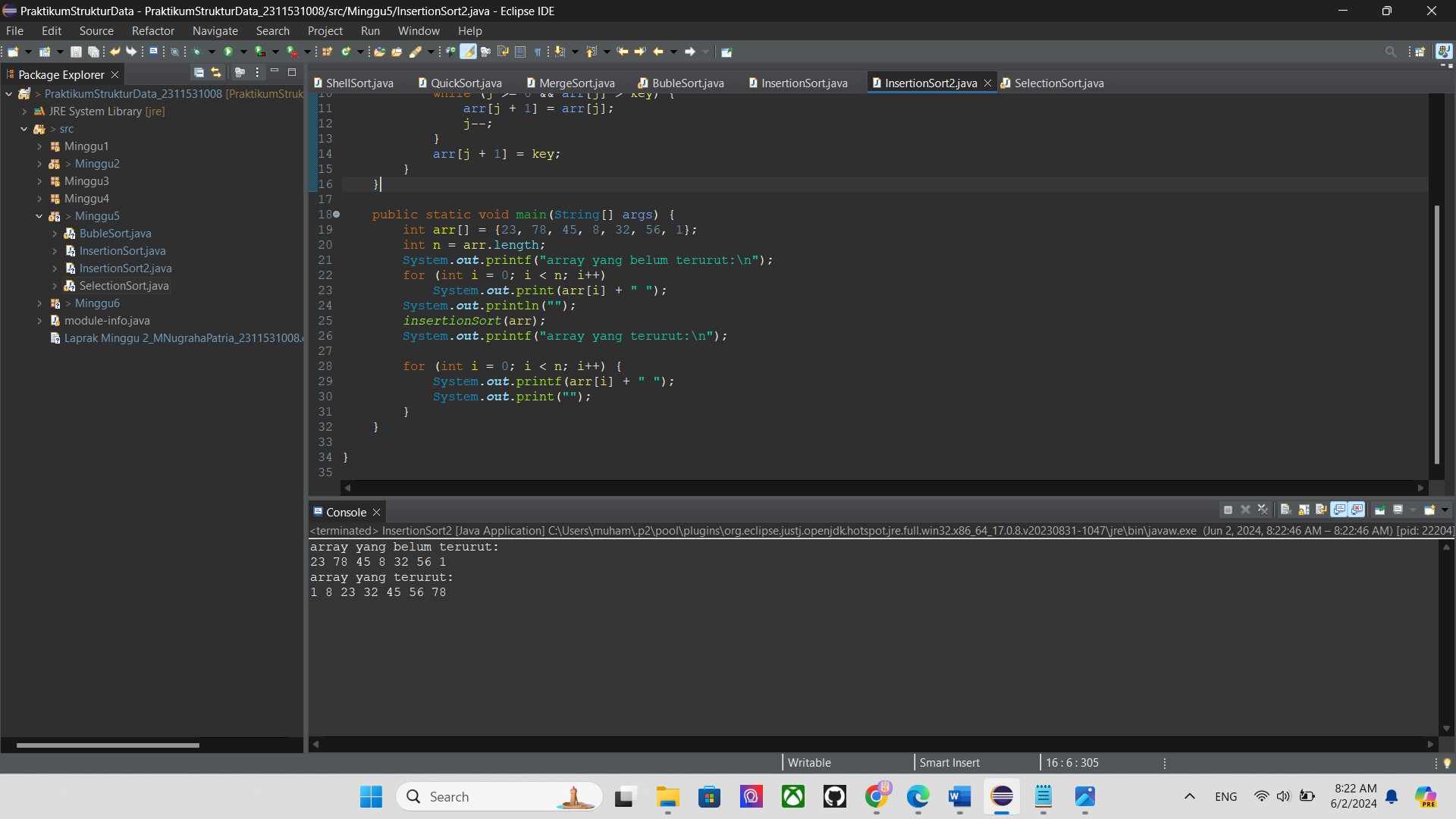
System.***out***.print("");

}

}

}

Inisialisasi array **arr** dengan beberapa nilai integer. Variabel **n** menyimpan panjang array. Mencetak judul "**array yang belum terurut:"**. **Loop for pertama** mencetak setiap elemen array yang belum diurutkan. Memanggil metode **insertionSort** untuk mengurutkan array **arr**. Mencetak judul **"array yang terurut:"**. **Loop for kedua** mencetak setiap elemen array yang sudah diurutkan.

Output:  


3. Selection Sort

public class SelectionSort {

public static void SelectionSort(int [] arr) {

int n = arr.length;

for (int i = 0; i < n; i++) {

int minIndex = i;

for (int j = i+1; j < n; j++) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

}

Metode ini menerima array integer **arr** sebagai argumen. Variabel **n** menyimpan panjang array. **Loop luar (for (int i = 0; i < n; i++))** mengiterasi elemen-elemen array. **Variabel minIndex** digunakan untuk melacak indeks elemen terkecil yang ditemukan dalam iterasi. **Loop dalam (for (int j = i + 1; j < n; j++))** mencari elemen terkecil di subarray yang dimulai dari indeks **i**. Jika elemen pada **arr[j] lebih kecil dari elemen pada arr[minIndex]**, **minIndex diperbarui ke j**. Setelah menemukan elemen terkecil, elemen tersebut ditukar dengan elemen pada indeks i menggunakan variabel sementara **temp**.

public static void main(String[] args) {

int arr[] = {23, 78, 45, 8, 32, 56, 1};

int n = arr.length;

System.***out***.println("array yang belum terurut:\n");

for (int i = 0; i < n; i++)

System.***out***.print(arr[i] + " ");

System.***out***.println("");

*SelectionSort*(arr);

System.***out***.printf("array yang terurut:\n");

for (int i = 0; i < n; i++)

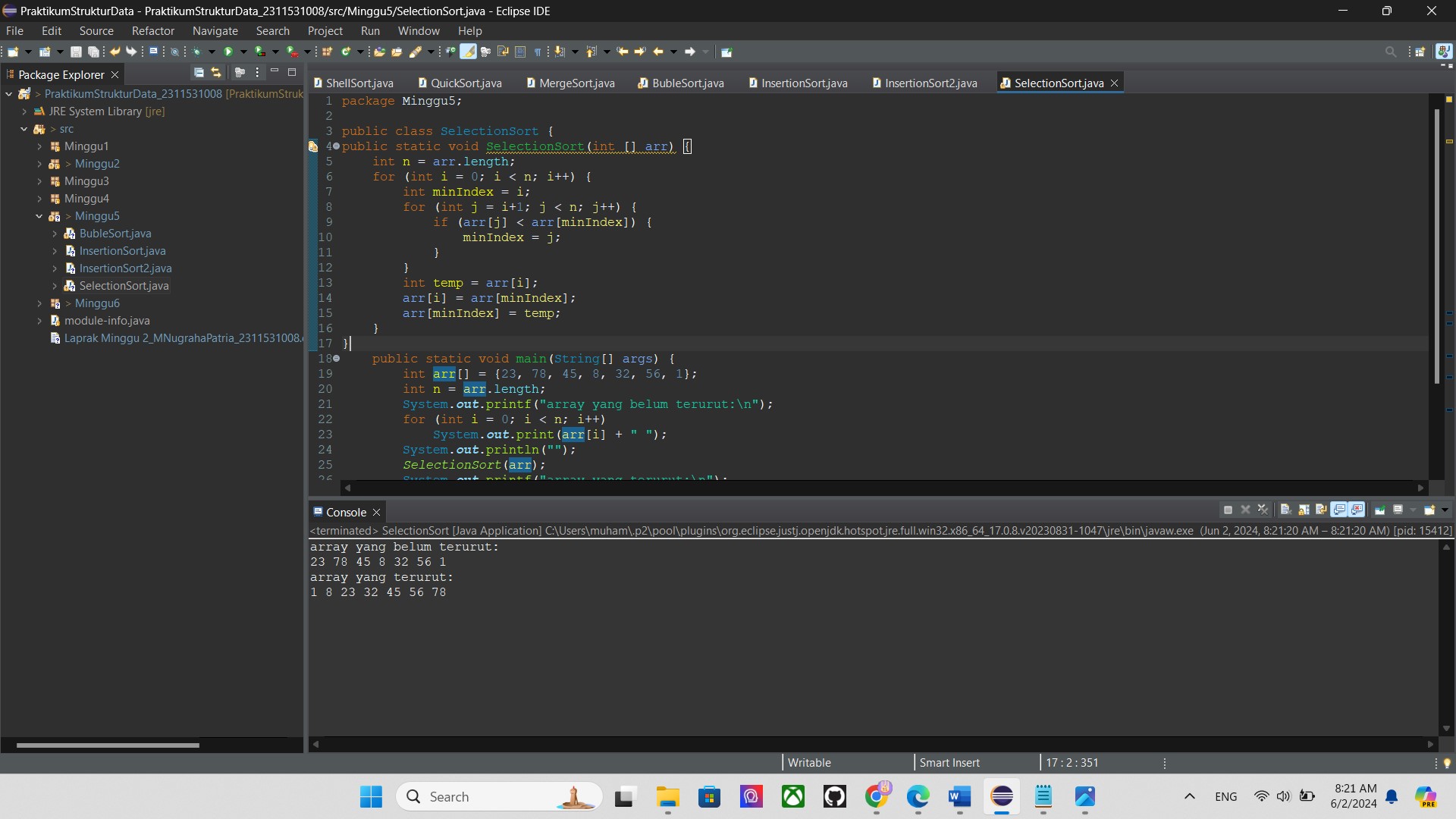
System.***out***.print(arr[i] + " ");

System.***out***.println("");

}

}

Array **arr** diinisialisasi dengan beberapa nilai integer. **Loop for pertama** mencetak setiap elemen array yang belum diurutkan. **System.out.println("array yang belum terurut:\n");** mencetak judul. **SelectionSort(arr);** memanggil metode SelectionSort untuk mengurutkan array arr. **Loop for kedua** mencetak setiap elemen array yang sudah diurutkan. **System.out.printf("array yang terurut:\n");** mencetak judul.

Output:  


4. Bubble Sort

public class BubleSort {

public static void BubleSort (int[] arr) {

int n = arr.length;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

Metode ini menerima array integer **arr** sebagai argumen. Variabel **n** menyimpan panjang array. Dua loop for digunakan untuk mengiterasi elemen-elemen array, **Loop luar (for (int i = 0; i < n; i++))** menentukan berapa banyak elemen terakhir yang sudah terurut dan tidak perlu diperiksa lagi. Loop dalam **(for (int j = 0; j < n-i-1; j++))** membandingkan elemen bertetangga dan menukar mereka jika elemen kiri lebih besar daripada elemen kanan. Jika elemen **arr[j]** lebih besar dari **arr[j+1]**, mereka ditukar menggunakan variabel sementara **temp**.

public static void main(String[] args) {

int arr[] = {23, 78, 45, 8, 32, 56, 1};

int n = arr.length;

System.***out***.printf("array yang belum terurut:\n");

for (int i = 0; i < n; i++)

System.***out***.print(arr[i] + " ");

System.***out***.println("");

//minMaxSelectionSort(arr, n);

*BubleSort*(arr);

System.***out***.printf("array yang terurut:\n");

for (int i = 0; i < n; i++)

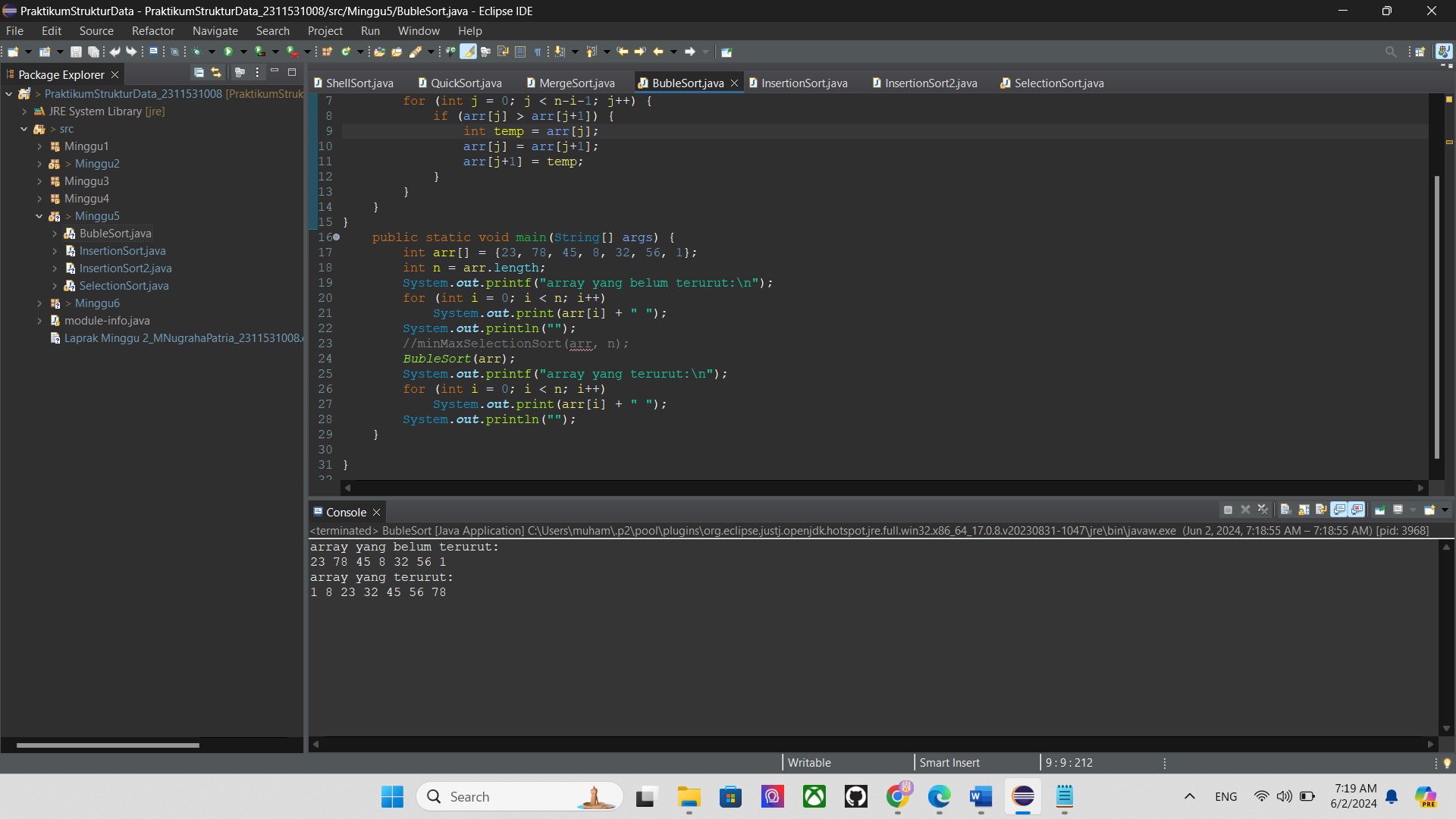
System.***out***.print(arr[i] + " ");

System.***out***.println("");

}

}

Array arr diinisialisasi dengan beberapa nilai integer. **System.out.printf("array yang belum terurut:\n");** mencetak judul. **Loop for** **pertama** mencetak setiap elemen array yang belum diurutkan. **BubleSort(arr);** memanggil metode BubleSort untuk mengurutkan array **arr**. **System.out.printf("array yang terurut:\n");** mencetak judul. **Loop for kedua** mencetak setiap elemen array yang sudah diurutkan.

Output :  


**D. Kesimpulan**

Metode Insertion sort merupakan metode mengurutkan bilangan-bilangan yang telah dibaca dan berikutnya secara berulang akan menyisipkan bilangan- bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut. Metode Selection Sort adalah metode memilih elemen terkecil (atau terbesar) dari bagian daftar yang tidak diurutkan dan menukarnya dengan elemen pertama dari bagian daftar yang tidak diurutkan. Metode Bubble Sort adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu.