**LAPORAN PRAKTIKUM STRUKTUR DATA**

ShellSort, QuickSort, MergeSort



Oleh :

FAYI AMATULLAH AZHARA

NIM 2311537001

MATA KULIAH

STRUKTUR DATA

DOSEN PENGAMPU : Dr.WAHYUDI ST, MT.

**FAKULTAS TEKNOLOGI INFORMASI**

**DEPARTEMEN INFORMATIKA**

**UNIVERSITAS ANDALAS**

1. **PENDAHULUAN**

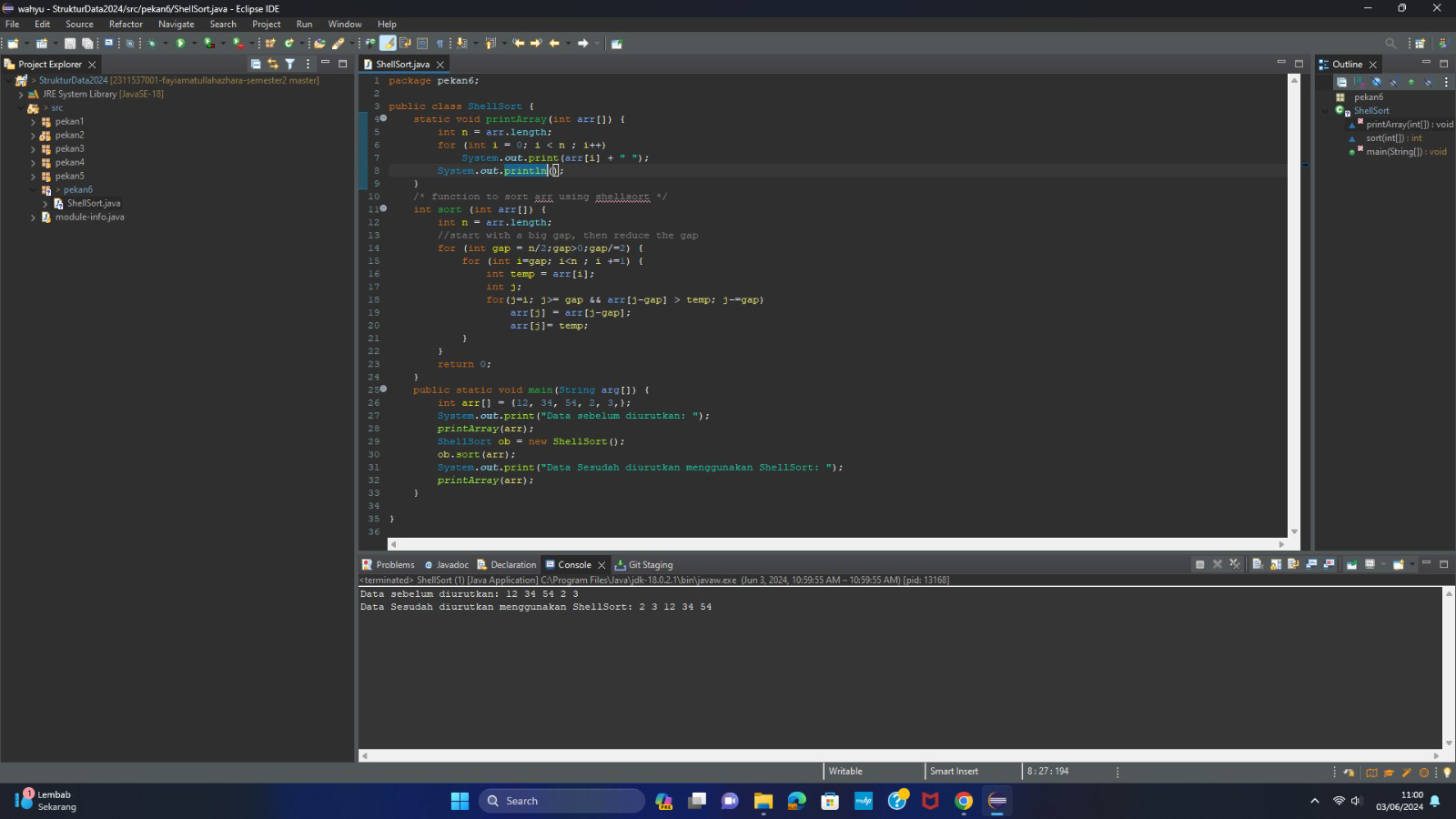
ShellSort, QuickSort, dan MergeSort adalah tiga algoritma pengurutan (sorting) yang digunakan untuk mengatur elemen-elemen dalam sebuah array atau struktur data lainnya dalam urutan tertentu. ShellSort adalah variasi dari Insertion Sort yang memperkenalkan konsep gap. Algoritma ini memulai dengan gap besar dan secara bertahap menguranginya hingga gap menjadi 1, pada tahap ini, algoritma bekerja seperti Insertion Sort. Dengan menggunakan gap yang lebih besar, elemen-elemen yang berjauhan bisa dibandingkan dan dipindahkan lebih awal, sehingga mempercepat proses pengurutan secara keseluruhan. ShellSort cenderung lebih efisien daripada Insertion Sort untuk array berukuran besar, terutama karena mengurangi jumlah pergeseran yang diperlukan untuk memasukkan elemen pada posisi yang benar.

QuickSort adalah salah satu algoritma pengurutan yang paling efisien dan banyak digunakan. Algoritma ini menggunakan pendekatan divide and conquer dengan memilih elemen pivot, kemudian mengatur elemen-elemen di sekitar pivot sedemikian rupa sehingga elemen yang lebih kecil dari pivot berada di sebelah kiri dan elemen yang lebih besar berada di sebelah kanan. Proses ini dilakukan secara rekursif pada sub-array yang terbentuk hingga seluruh array terurut. QuickSort terkenal karena kinerjanya yang cepat pada data acak dan memiliki kompleksitas waktu rata-rata O(n log n), meskipun pada kasus terburuk bisa mencapai O(n²). MergeSort, di sisi lain, juga menggunakan pendekatan divide and conquer dengan membagi array menjadi dua bagian, mengurutkan setiap bagian secara rekursif, dan kemudian menggabungkannya kembali menjadi satu array yang terurut. MergeSort menjamin kompleksitas waktu O(n log n) di semua kasus, menjadikannya sangat stabil dan efisien untuk data berukuran besar, meskipun memerlukan ruang tambahan untuk menyimpan salinan dari array selama proses penggabungan.

1. **TUJUAN**

Memahami Konsep dan Implementasi pengurutan

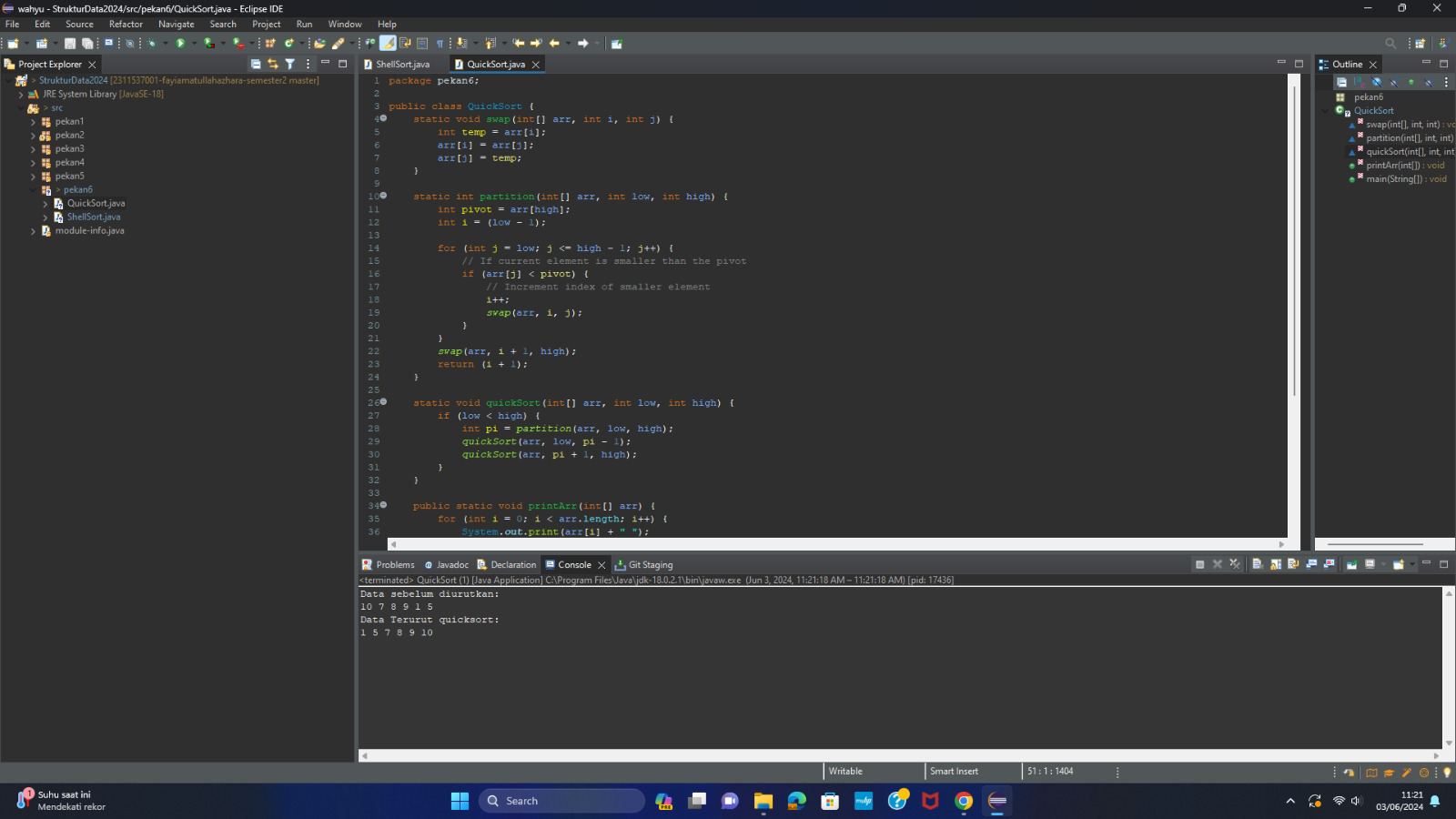
1. **LANGKAH-LANGKAH**
2. Membuka Eclipse IDE dan import proyek github dengan nama “STRUKTURDATA2024”
3. Membuat Package baru dengan nama “pekan6”
4. Membuat Kelas baru dengan nama “ShellSort”
5. Pada kelas “ShellSort”, masukkan kode seperti gambar di bawah

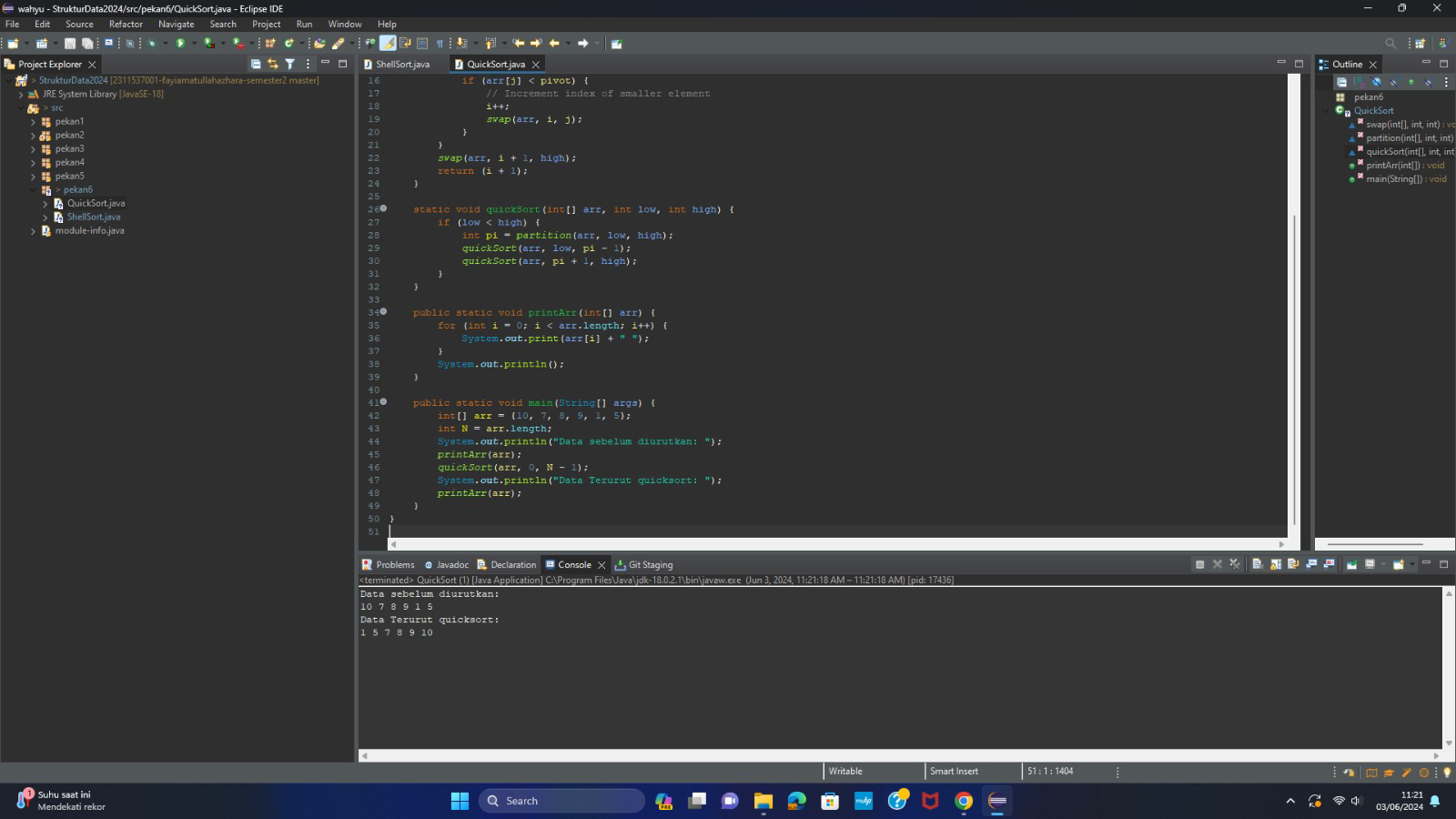


Program di atas adalah implementasi Shell Sort dalam bahasa Java. Program ini terdapat dalam package bernama "pekan6" dan memiliki sebuah kelas bernama "ShellSort." Kelas ini memiliki dua metode utama: printArray dan sort. Metode printArray digunakan untuk mencetak elemen-elemen array ke konsol. Metode sort digunakan untuk mengurutkan array menggunakan algoritma Shell Sort, yang dimulai dengan gap besar dan secara bertahap menguranginya. Dalam metode sort, elemen-elemen array dibandingkan dan ditukar berdasarkan nilai gap, sehingga akhirnya menghasilkan array yang terurut. Pada metode main, sebuah array contoh diinisialisasi, dicetak sebelum diurutkan, diurutkan menggunakan Shell Sort, dan kemudian dicetak lagi setelah diurutkan.

1. Membuat Kelas baru dengan nama “QuickSort”

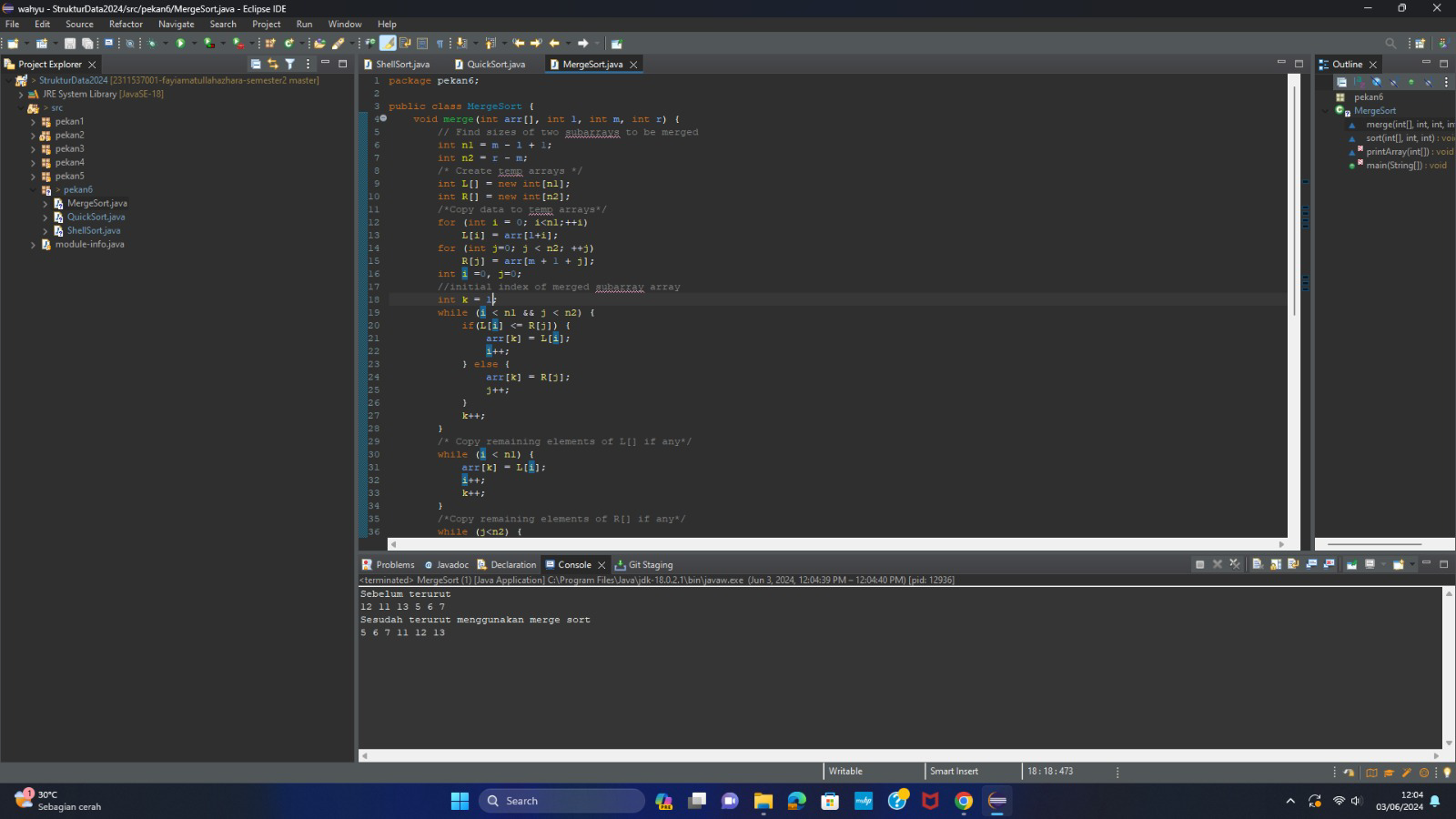
Pada kelas “QuickSort”, masukkan kode seperti gambar di bawah

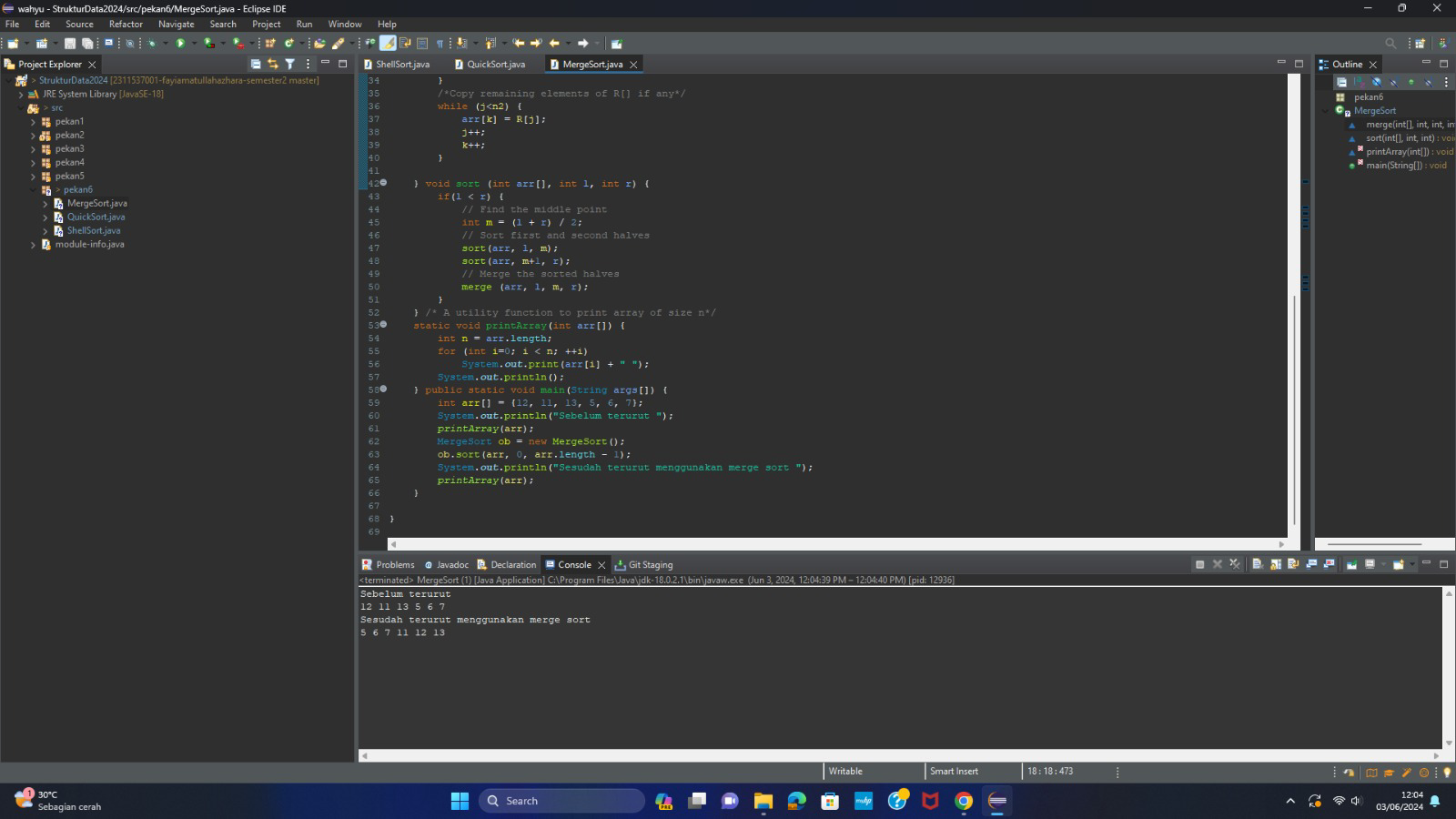




Program di atas adalah implementasi Quick Sort dalam bahasa Java yang terdapat dalam package bernama "pekan6" dengan kelas "QuickSort." Kelas ini memiliki beberapa metode utama: swap, partition, quickSort, dan printArr. Metode swap digunakan untuk menukar dua elemen dalam array. Metode partition memilih elemen pivot dari array dan mengatur elemen-elemen di sekitarnya sehingga elemen yang lebih kecil dari pivot berada di sebelah kiri dan yang lebih besar berada di sebelah kanan. Metode quickSort secara rekursif memanggil dirinya sendiri untuk mengurutkan sub-array yang terbentuk dari proses partition. Metode printArr digunakan untuk mencetak elemen-elemen array ke konsol. Pada metode main, sebuah array contoh diinisialisasi, dicetak sebelum diurutkan, diurutkan menggunakan Quick Sort, dan kemudian dicetak lagi setelah diurutkan.

1. Pada kelas “MergeSort”, masukkan kode seperti gambar di bawah





Program di atas adalah implementasi Merge Sort dalam bahasa Java yang terdapat dalam package bernama "pekan6" dengan kelas "MergeSort." Kelas ini memiliki beberapa metode utama: merge, sort, dan printArray. Metode merge digunakan untuk menggabungkan dua sub-array yang sudah diurutkan menjadi satu array yang terurut. Metode ini memisahkan array menjadi dua bagian, menyalin data ke dalam array sementara, dan kemudian menggabungkannya kembali ke dalam array asli dengan urutan yang benar. Metode sort digunakan untuk membagi array secara rekursif menjadi dua bagian hingga menjadi elemen tunggal, lalu menggabungkannya kembali menggunakan metode merge. Metode printArray digunakan untuk mencetak elemen-elemen array ke konsol. Pada metode main, sebuah array contoh diinisialisasi, dicetak sebelum diurutkan, diurutkan menggunakan Merge Sort, dan kemudian dicetak lagi setelah diurutkan.

**D. PENUTUP**

Dalam praktikum ini, saya telah mempelajari dan mengimplementasikan tiga algoritma pengurutan yang penting yaitu ShellSort, QuickSort, dan MergeSort. Masing-masing algoritma memiliki pendekatan dan karakteristik yang unik dalam mengurutkan elemen-elemen dalam sebuah array.

ShellSort merupakan peningkatan dari Insertion Sort dengan pengenalan konsep gap, yang memungkinkan elemen-elemen yang berjauhan untuk dibandingkan dan dipindahkan lebih awal, sehingga mempercepat proses pengurutan. QuickSort, dengan pendekatan divide and conquer dan penggunaan elemen pivot, terkenal karena efisiensinya pada data acak dan kompleksitas waktu rata-rata yang cepat. MergeSort juga menggunakan pendekatan divide and conquer dan menjamin kompleksitas waktu O(n log n) di semua kasus, meskipun memerlukan ruang tambahan.

Melalui implementasi algoritma-algoritma ini dalam bahasa Java, saya mendapatkan pemahaman yang lebih mendalam tentang cara kerja dan perbedaan dari ketiga algoritma tersebut. Praktikum ini memberikan pengalaman praktis dalam menerapkan konsep-konsep pengurutan dan meningkatkan pemahaman saya tentang efisiensi dan kompleksitas waktu dari berbagai metode pengurutan.

Dengan demikian, tujuan dari praktikum ini untuk memahami konsep dan implementasi pengurutan telah tercapai.