JOBSHEET VI  
SEARCHING

1. 1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Menjelaskan mengenai algoritma Searching.
2. Membuat dan mendeklarasikan struktur algoritma Searching.
3. Menerapkan dan mengimplementasikan algoritma Searching.
4. Searching / Pencarian Menggunakan Algoritma Sequential Search
   * 1. **Sequential Search Menggunakan Array**
5. Buat folder baru dengan nama Praktikum06. Buat file dengan nama Sorting.java
6. Tambahkan method sequentialSearch() yang melakukan pencarian data bertipe integer di dalam array of integer

A screen shot of a computer code

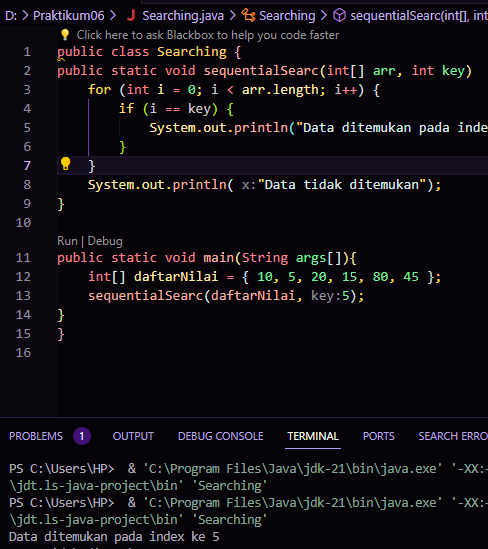
Description automatically generated

1. Tambahkan fungsi main sebagai berikut

A computer code with text

Description automatically generated with medium confidence

1. Compile dan run program



* + 1. **Sequential Search Menggunakan Array of Object**

Perhatikan diagram class Mahasiswa di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program class Mahasiswa.

|  |
| --- |
| Mahasiswa |
| Nim: int  nama: String  umur: int  ipk: double |
| Mahasiswa(ni:int, n: String, u: int, i: double)  tampil(): void |

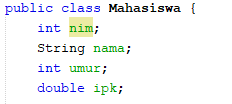
Berdasarkan class diagram di atas, akan dibuat class Mahasiswa yang berfunsi untuk membuat objek mahasiswa yang akan dimasukan ke dalam sebuah array. Terdapat sebuah konstruktor berparameter dan juga fungsi tampil() untuk menampilkan semua attribute yang ada.

|  |
| --- |
| PencarianMhs |
| listMhs: Mahasiswa[5]  idx: int |
| tambah(mhs: Mahasiswa): void  tampil(): void  FindSeqSearch(int cari): int  Tampilpoisisi(int x,int pos): void  TampilData(int x,int pos) :void |

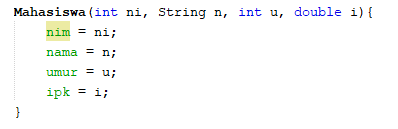
Selanjutnya class diagram di atas merupakan representasi dari sebuah class yang berfungsi untuk melakukan operasi-operasi dari objek array mahasiswa, misalkan untuk menambahkan objek mahasiswa, menampilkan semua data mahasiswa, untuk melakukan pencarian berdasarkan NIM menggunakan algoritma Sequential Search, menampilkan posisi dari data yang dicari, serta menampilkan data mahasiswa yang dicari.

* Langkah-langkah Percobaan Sequential Search

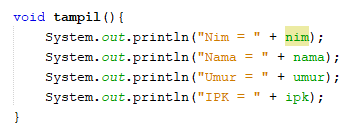
1. Buatlah Project baru pada Netbeans dengan nama **TestSearching**
2. Kemuadian buat packages baru dengan nama **minggu7**.
3. Buat class **Mahasiswa**, kemudian deklarasikan atribut berikut ini:



1. Buatlah konstruktor dengan nama **Mahasiswa** dengan parameter (**int ni, String n, int u, double i**) kemudian Isi konstruktor tersebut dengan kode berikut!



1. Buatlah method **tampil** bertipe void.



1. Buat class baru dengan nama **PencarianMhs** seperti di bawah ini!

|  |
| --- |
|  |

1. Tambahkan method **tambah()** di dalam class tersebut! Method **tambah()** digunakan untuk menambahkan objek dari class Mahasiswa ke dalam atribut listMhs.

|  |
| --- |
|  |

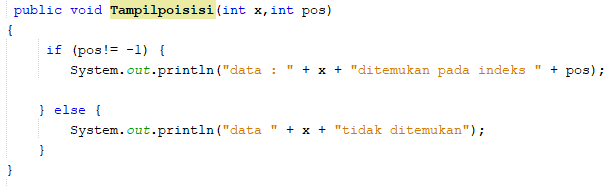
1. Tambahkan method **tampil()** di dalam class **PencarianMhs**! Method **tampil()** digunakan untuk menampilkan semua data mahasiswa-mahasiswa yang ada di dalam class tersebut! Perhatikan penggunaan sintaks for yang agak berbeda dengan for yang telah dipelajari sebelumnya, meskipun secara konsep sebenarnya mirip.

|  |
| --- |
|  |

1. Tambahkan method **FindSeqSearch** bertipe integer dengan parameter **cari** bertipe integer. Kemudian Deklarasikan isi method **FindSeqSearch** dengan algoritma pencarian data menggunakan teknik sequential searching.

|  |
| --- |
|  |

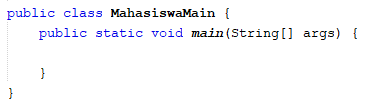
1. Buatlah method **Tampilpoisisi** bertipe void dan Deklarasikan isi dari method **Tampilpoisisi.**



1. Buatlah method **TampilData** bertipe void dan Deklarasikan isi dari method **TampilData.**

|  |
| --- |
|  |

1. Buatlah class baru dengan nama **MahasiswaMain** tambahkan method **main** seperti pada gambar berikut!



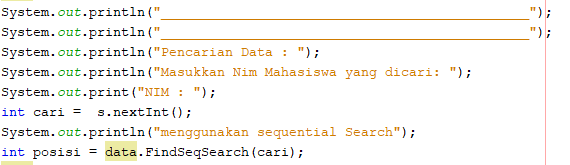
1. Di dalam method **main(),** buatlah sebuah objek PencarianMhs dan buatlah 5 objek mahasiswa kemudian tambahkan semua objek mahasiswa tersebut dengan memanggil fungsi tambah pada objek PencarianMhs.

|  |
| --- |
|  |

1. Panggil method **tampil()** untuk melihat semua data yang telah dimasukan.

|  |
| --- |
|  |

1. Untuk melakukan pencarian berdasarkan NIM mahasiswa. Buatlah variable **cari** yang dapat menampung masukan dari keyboard lalu panggil method **FindSeqSearch** dengan isi parameternya adalah variable cari.



1. Lakukan pemanggilan method **Tampilposisi** dari class **PencarianMhs.**

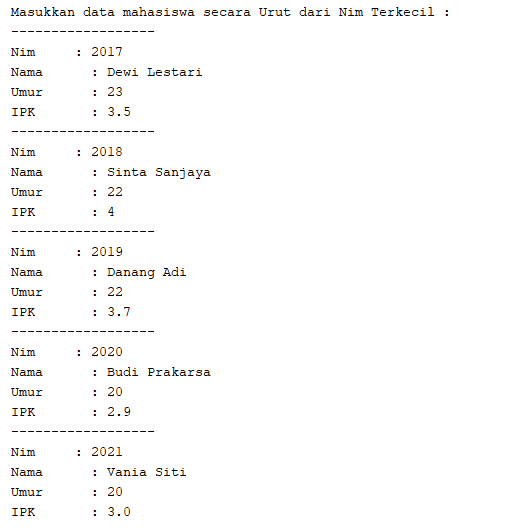


1. Lakukan pemanggilan method **TampilData** dari class **PencarianMhs.**

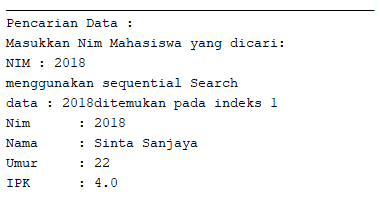


1. Jalankan dan amati hasilnya.
2. **Verifikasi Hasil Percobaan**

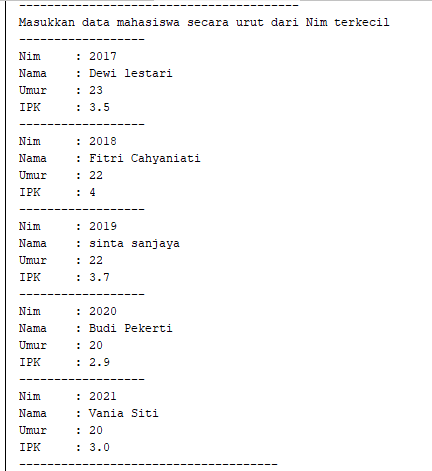
Cocokkan hasil kode program anda dengan gambar berikut ini.

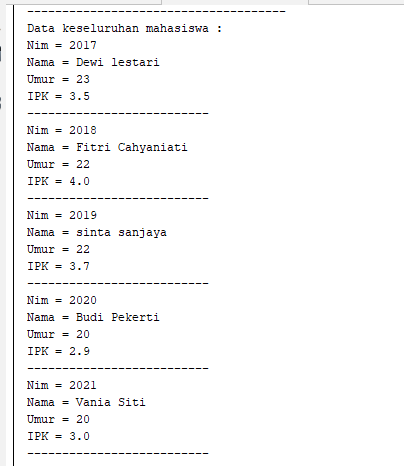


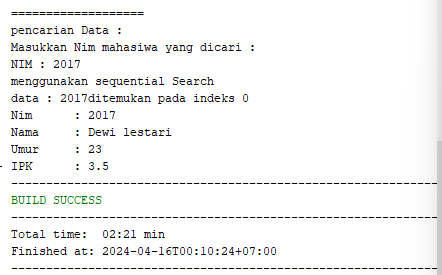




**Hasil RUN :**



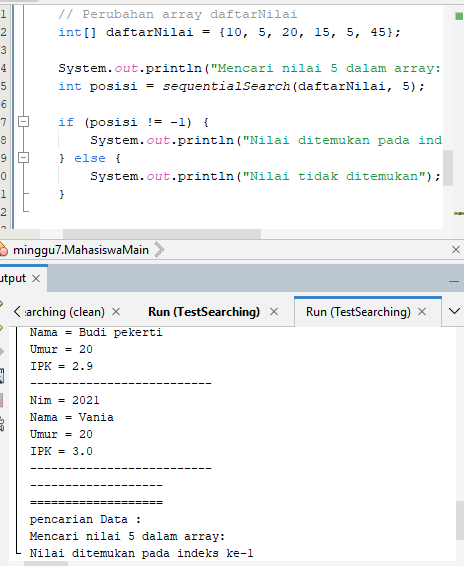




1. **Pertanyaan**
2. Lakukan perubahan array daftarNilai pada fungsi main().

A computer code with text

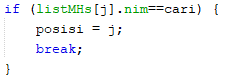
Description automatically generated with medium confidence



1. Jelaskan perbedaan metod **TampilData** dan **Tampilposisi** pada class PencarianMhs

**Metode `Tampilposisi` memberikan informasi apakah data ditemukan atau tidak beserta indeksnya, sedangkan `TampilData` memberikan informasi detail dari data yang ditemukan, seperti NIM, Nama, Umur, dan IPK.**

1. Jelaskan fungsi **break** pada kode program dibawah ini!

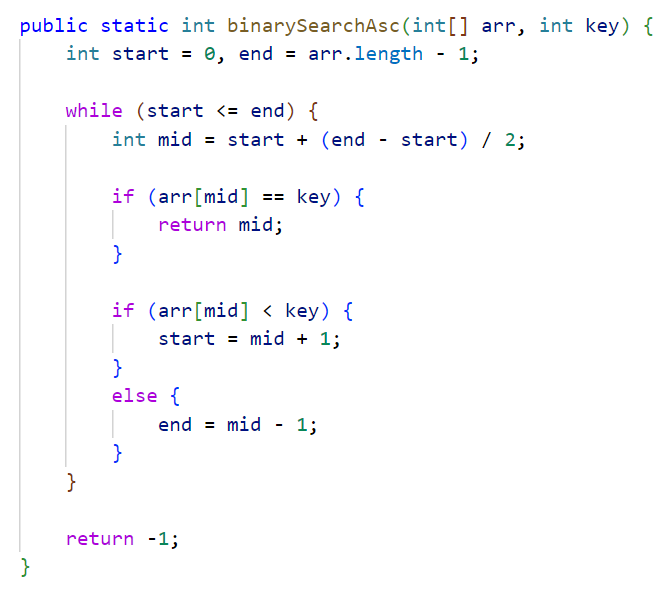


**break digunakan setelah posisi ditemukan (posisi = j;) ketika nim yang dicari (cari) sama dengan nim dari listMHs[j].nim.** `**break` digunakan untuk menghentikan perulangan secara paksa ketika kondisi tertentu terpenuhi. Dalam contoh tersebut, `break` digunakan setelah posisi ditemukan, sehingga iterasi berhenti dan pencarian data selesai.**

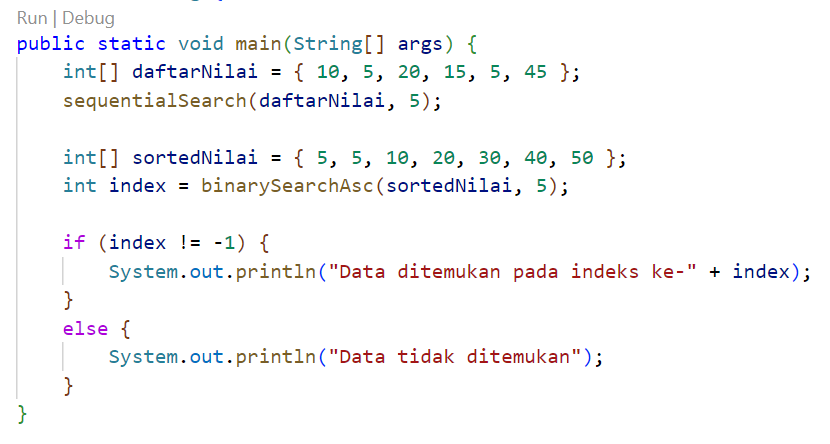
1. Jika Data Nim yang dimasukkan tidak terurut dari kecil ke besar. Apakah program masih dapat berjalan? Apakah hasil yang dikeluarkan benar? Mengapa demikian!

**Jika data Nim yang dimasukkan tidak terurut dari kecil ke besar, program masih dapat berjalan tetapi hasil yang dikeluarkan mungkin tidak benar. Hal ini karena metode pencarian sequential search memerlukan data terurut untuk memberikan hasil yang benar. Jika data tidak terurut, program mungkin tidak dapat menemukan data yang dicari pada posisi yang seharusnya. Salah satu solusinya adalah dengan mengurutkan data terlebih dahulu sebelum melakukan pencarian.**

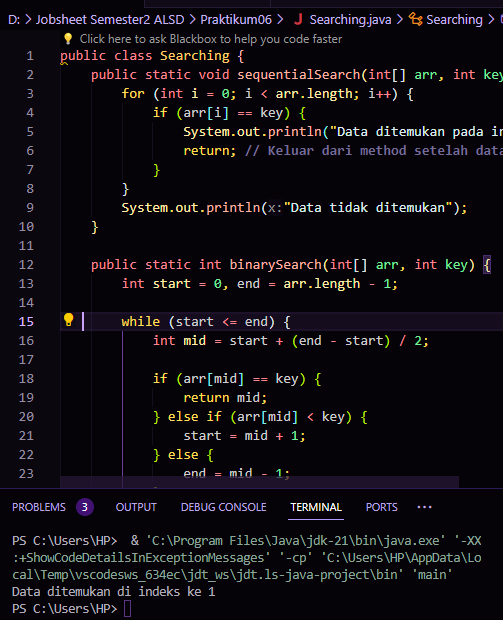
1. Searching / Pencarian Menggunakan Binary Search
2. Langkah-langkah Percobaan Binary Search menggunakan Array
3. Tambahkan method binarySearchAsc() pada file Sorting.java



1. Tambahkan baris program untuk menguji method binarySearchAsc() pada fungsi main()



1. Run dan compile program



1. Langkah-langkah Percobaan Binary Search menggunakan Array of Object
2. Pada percobaan 6.2.2 (sequential search) tambahkan method **FindBinarySearch** bertipe integer pada class **PencarianMhs**. Kemudian Deklarasikan isi method **FindBinarySearch** dengan algoritma pencarian data menggunakan teknik binary searching.

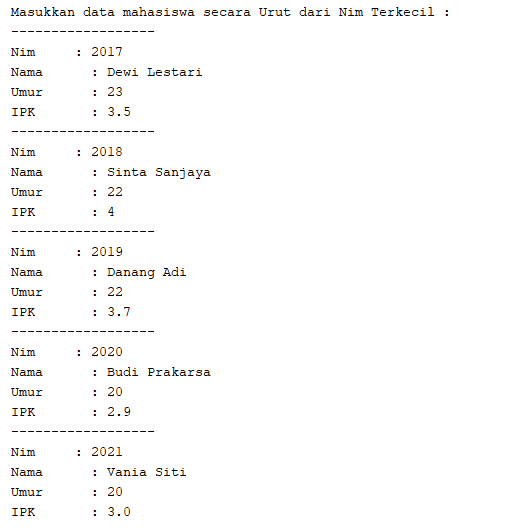
|  |
| --- |
|  |

1. Panggil method **FindBinarySearch** terdapat pada class **PencarianMhs** di kelas **Mahasiswamain.** Kemudia panggil **method tampilposisi** dan **tampilData**

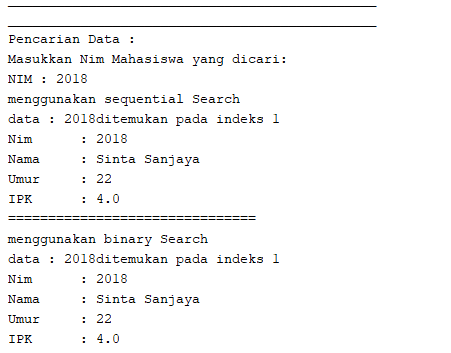
|  |
| --- |
|  |

1. Jalankan dan amati hasilnya.
2. **Verifikasi Hasil Percobaan**

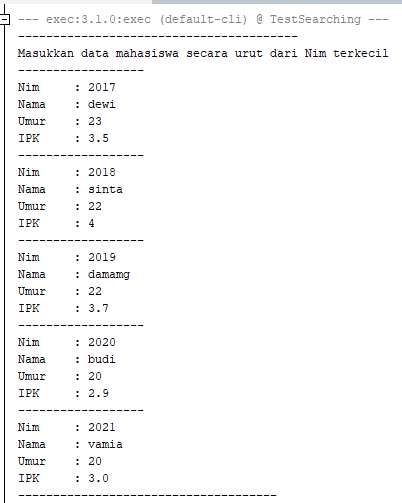
Cocokkan hasil kode program anda dengan gambar berikut ini.



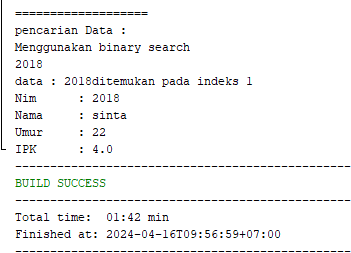




**Hasil RUN :**







1. **Pertanyaan**
2. Tunjukkan pada kode program yang mana proses divide dijalankan!

**public int FindBinarySearch(int cari, int left, int right){**

**if (right >= left){**

**int mid = (left + right) / 2;**

**if (cari == listMHs[mid].nim)**

**return mid;**

**if (listMHs[mid].nim > cari)**

**return FindBinarySearch(cari, left, mid - 1);**

**return FindBinarySearch(cari, mid + 1, right);**

**}**

**return -1;**

**}**

**Menghitung titik tengah array dengan rumus (left + right) / 2.**

**Membandingkan nilai yang dicari dengan nilai di titik tengah.**

**Berdasarkan hasil perbandingan, pemilihan pembagian dilakukan pada setengah kiri atau kanan array.**

**Proses ini terus berlanjut secara rekursif hingga elemen yang dicari ditemukan atau rentang pencarian sudah habis.**

1. Tunjukkan pada kode program yang mana proses conquer dijalankan!

**public int FindBinarySearch(int cari, int left, int right){**

**if (right >= left){**

**int mid = (left + right) / 2;**

**if (cari == listMHs[mid].nim)**

**return mid; // Proses conquer: elemen ditemukan**

**if (listMHs[mid].nim > cari)**

**return FindBinarySearch(cari, left, mid - 1); // Proses conquer: pemanggilan rekursif**

**return FindBinarySearch(cari, mid + 1, right); // Proses conquer: pemanggilan rekursif**

**}**

**return -1; // Proses conquer: rentang pencarian habis**

**}**

**Dalam metode FindBinarySearch, proses "conquer" terjadi saat:**

**Elemen yang dicari ditemukan di indeks mid.**

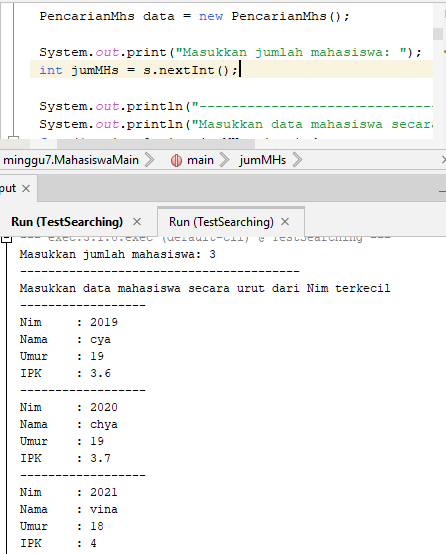
**Rentang pencarian sudah habis (right < left), yang menunjukkan bahwa elemen tidak ditemukan dalam array.**

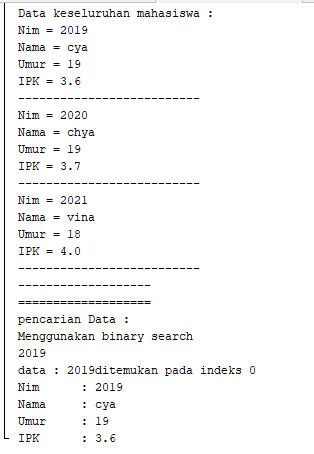
**Dalam kedua kasus tersebut, nilai indeks mid atau -1 dikembalikan sebagai hasil pencarian, menunjukkan bahwa proses "conquer" telah selesai.**

1. Jika data Nim yang dimasukkan tidak urut. Apakah program masih dapat berjalan? Mengapa demikian!

**Jika data Nim yang dimasukkan tidak dalam urutan yang sesuai, program tidak akan memberikan hasil yang benar. Algoritma binary search memerlukan bahwa data telah diurutkan sebelumnya. Jika tidak diurutkan, hasil pencarian dapat menjadi tidak konsisten atau bahkan salah karena algoritma tidak dapat membagi rentang pencarian dengan benar. Oleh karena itu, urutan data sangat penting untuk memastikan keberhasilan binary search.**

1. Jika Nim yang dimasukkan dari NIM terbesar ke terkecil (missal : 20215, 20214, 20212, 20211, 20210) dan elemen yang dicari adalah 20210. Bagaimana hasil dari binary search? Apakah sesuai? Jika tidak sesuai maka ubahlah kode program binary seach agar hasilnya sesuai
2. Modifikasilah program diatas yang mana jumlah mahasiswa yang di inputkan sesuai dengan masukan dari keyboard.





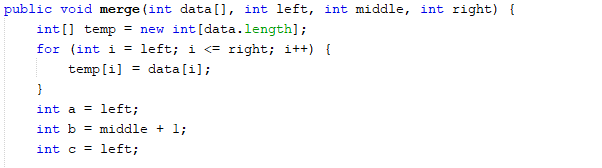
1. Percobaan Pengayaan Divide and Conquer
2. Langkah-langkah Percobaan Merge Sort
3. Buatlah Package baru pada NetBeans dengan nama **MergeSortTest**
4. Tambahkan class **MergeSorting** pada package tersebut
5. Pada class **MergeSorting** buatlah method **mergeSort** yang menerima parameter data array yang akan diurutkan

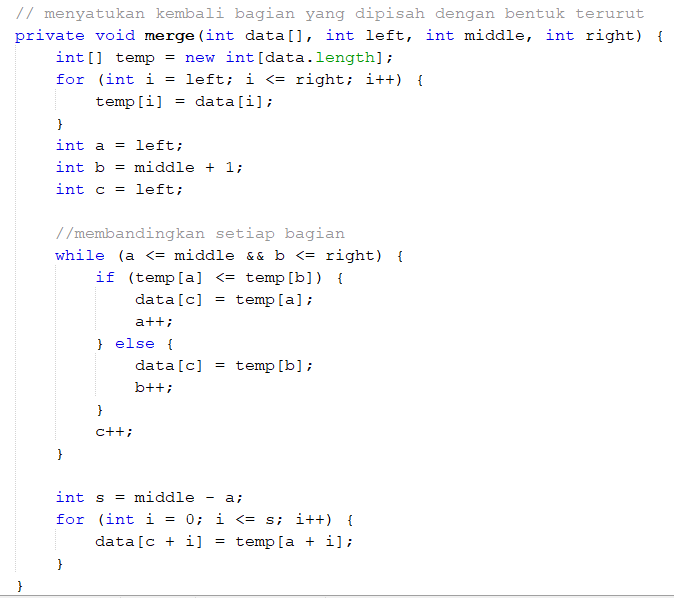


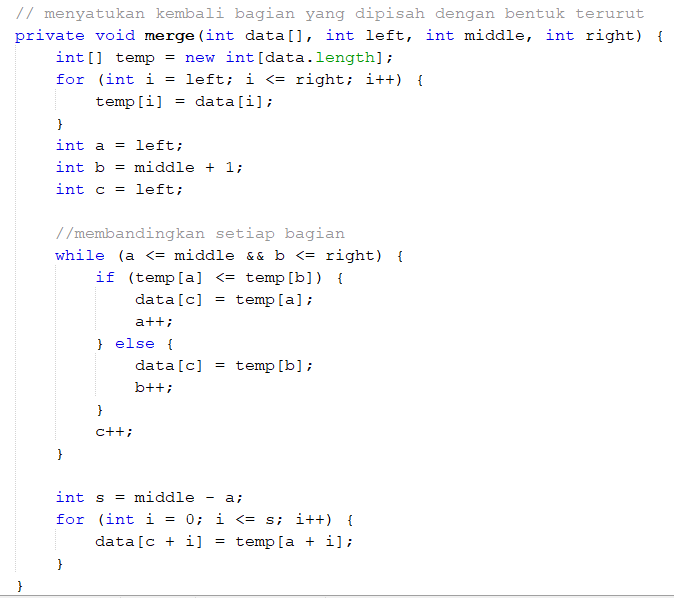
1. Buatlah method **merge** untuk melakukan proses penggabungan data dari bagian kiri dan kanan.



1. Implementasikan proses **merge** sebagai berikut.



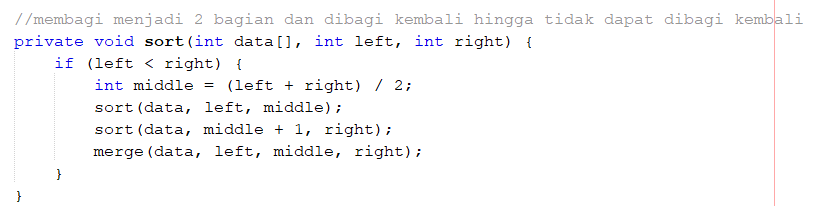




1. Buatlah method **sort**



1. Implementasikan kode berikut pada method **sort**



1. Pada method **mergeSort**, panggil method **sort** dengan parameter data yang ingin diurutkan serta range data awal sampai dengan akhir.
2. Tambahkan method **printArray**

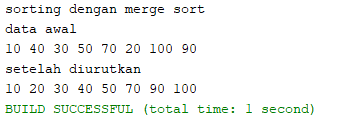
|  |
| --- |
|  |

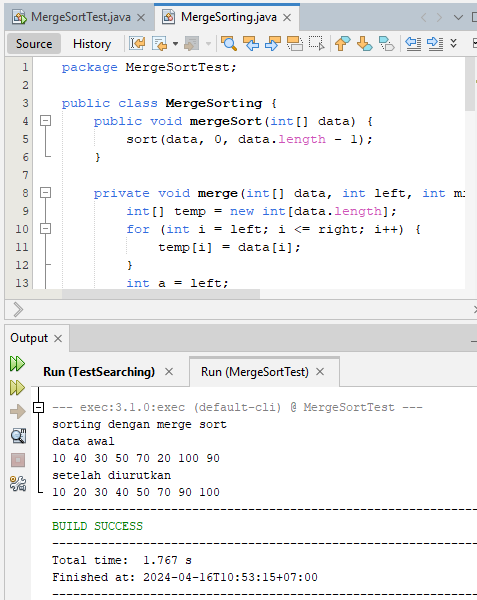
1. Sebagai langkah terakhir, deklarasikan data yang akan diurutkan kemudian panggil proses sorting pada class SortMain

|  |
| --- |
|  |

1. **Verifikasi Hasil Percobaan**

Cocokkan hasil compile kode program anda dengan gambar berikut ini.





1. Latihan Praktikum
2. Modifikasi percobaan searching diatas yang menggunakan Searching array of object dengan ketentuan berikut ini

* Pencarian dilakukan berdasarkan Nama Mahasiswa (gunakan Algoritma binary Search)
* Buat aturan untuk mendeteksi hasil pencarian lebih dari 1 hasil dalam bentuk kalimat peringatan!

