

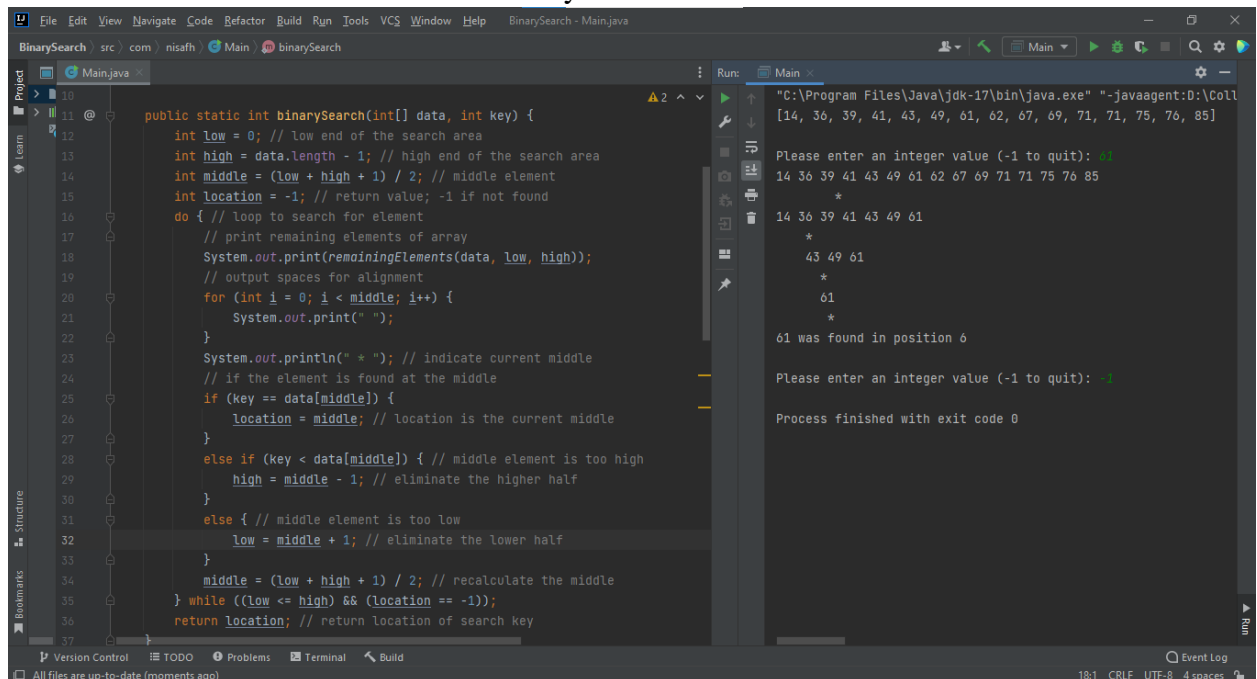
Nama : Nisa Fauziyyah Handari

NIM : 6706213138

Kelas : 4505

a. Penjelasan koding

Binary Search



```
public static int binarySearch(int[] data, int key) {
    int low = 0; // low end of the search area
    int high = data.length - 1; // high end of the search area
    int middle = (low + high + 1) / 2; // middle element
    int location = -1; // return value; -1 if not found
    do { // loop to search for element
        // print remaining elements of array
        System.out.print(remainingElements(data, low, high));
        // output spaces for alignment
        for (int i = 0; i < middle; i++) {
            System.out.print(" ");
        }
        System.out.println(" * "); // indicate current middle
        // if the element is found at the middle
        if (key == data[middle]) {
            location = middle; // location is the current middle
        }
        else if (key < data[middle]) { // middle element is too high
            high = middle - 1; // eliminate the higher half
        }
        else { // middle element is too low
            low = middle + 1; // eliminate the lower half
        }
        middle = (low + high + 1) / 2; // recalculate the middle
    } while ((low <= high) && (location == -1));
    return location; // return location of search key
}
```

Binary search merupakan algoritma yang lebih efisien daripada linear search, tetapi algoritma ini hanya dapat digunakan pada data yang telah terurut. Dalam hal ini, data yang disimpan pada array telah terurut. Pada iterasi pertama, dilakukan pengujian pada elemen tengah array. Jika elemen ini ternyata sama dengan nilai yang dicari, algoritma akan berhenti.

Jika tidak, dengan asumsi bahwa array terurut dalam susunan ascending, jika nilai yang dicari (search key) lebih kecil daripada nilai elemen tengah, maka tidak mungkin nilai yang dicari tersebut ada di setengah bagian atas dari array. Karenanya, pencarian akan dilanjutkan pada setengah bagian awal (bawah) dari array. Hal sebaliknya terjadi jika search key lebih besar daripada nilai tengah array.

Pada screenshot diatas saya memasukan angka bernilai 61 yang pada array list di screenshot di awal angka 61 berada di posisi ke 7 atau di dalam array list adalah posisi 6. Tanda * yang di ditampilkan merupakan hasil eliminasi yang ada di codingan yang berujung pada nilai akhir 19.

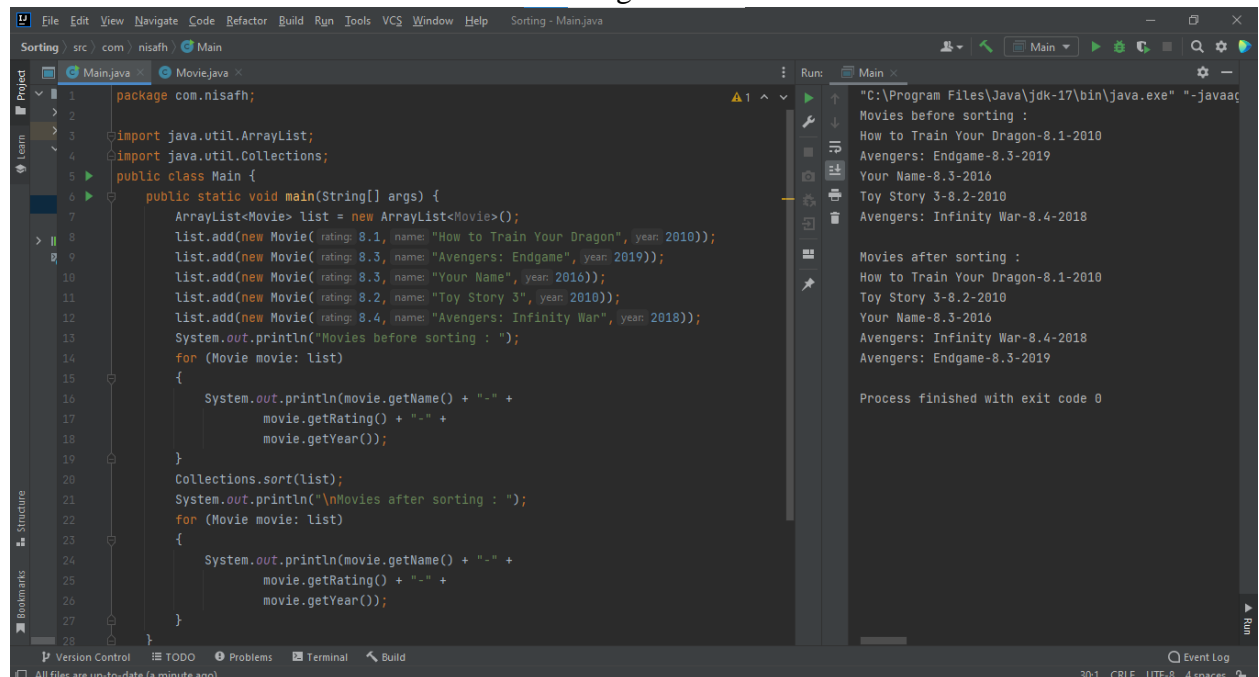
Pada setiap iterasi, pengetesan dilakukan pada nilai tengah dari bagian array yang tersisa. Jika search key tidak sesuai dengan nilai elemen, algoritma ini akan menghilangkan setengah bagian lagi dari elemen yang tersisa.

Algoritma ini akan berhenti pada dua keadaan; elemen yang dicari ditemukan, atau subarray yang tersisa telah menjadi nol. Algoritma binary search merupakan algoritma yang digunakan pada Java Collections framework. Method searching tersebut pun menggunakan nama yang sama dengan algoritmanya, yaitu method binary search. Method ini dapat digunakan baik untuk LinkedList maupun ArrayList.

Dengan rumus :

- Jika $value < cari$ maka, $awal = tengah + 1$
- Jika $value > cari$ maka, $akhir = tengah - 1$
- Jika $value == cari$ maka, pencarian berakhir

Sorting



```
package com.nisafh;

import java.util.ArrayList;
import java.util.Collections;

public class Main {

    public static void main(String[] args) {
        ArrayList<Movie> list = new ArrayList<Movie>();
        list.add(new Movie( ratings: 8.1, name: "How to Train Your Dragon", year: 2010));
        list.add(new Movie( ratings: 8.3, name: "Avengers: Endgame", year: 2019));
        list.add(new Movie( ratings: 8.3, name: "Your Name", year: 2016));
        list.add(new Movie( ratings: 8.2, name: "Toy Story 3", year: 2010));
        list.add(new Movie( ratings: 8.4, name: "Avengers: Infinity War", year: 2018));
        System.out.println("Movies before sorting : ");
        for (Movie movie: list)
        {
            System.out.println(movie.getName() + "-" +
                                movie.getRating() + "-" +
                                movie.getYear());
        }
        Collections.sort(list);
        System.out.println("\nMovies after sorting : ");
        for (Movie movie: list)
        {
            System.out.println(movie.getName() + "-" +
                                movie.getRating() + "-" +
                                movie.getYear());
        }
    }
}
```

Movies before sorting :

How to Train Your Dragon-8.1-2010
Avengers: Endgame-8.3-2019
Your Name-8.3-2016
Toy Story 3-8.2-2010
Avengers: Infinity War-8.4-2018

Movies after sorting :

How to Train Your Dragon-8.1-2010
Toy Story 3-8.2-2010
Your Name-8.3-2016
Avengers: Infinity War-8.4-2018
Avengers: Endgame-8.3-2019

Process finished with exit code 0

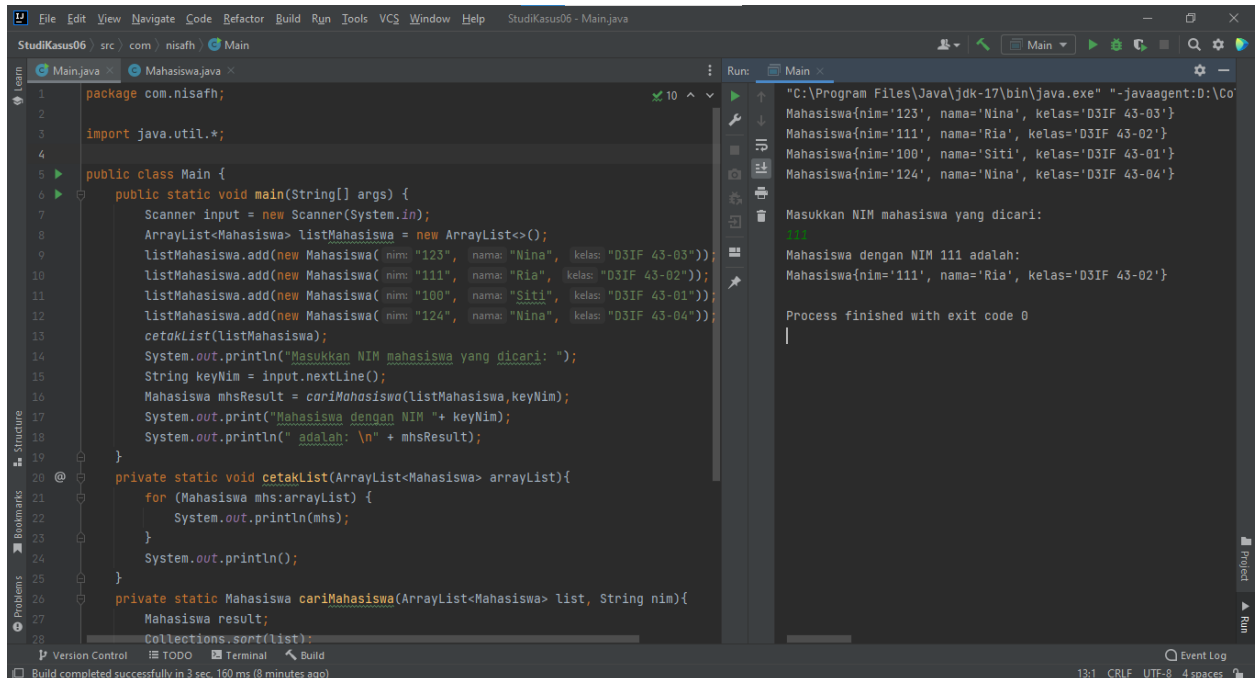
Algoritma sorting akan menyusun ulang elemen dari suatu kumpulan (koleksi) sehingga elemen-elemen tersebut akan tersimpan dalam susunan yang sudah terurut.

Hal penting yang harus diingat dalam melakukan sorting adalah bahwa hasil akhir dari sorting (array yang sudah terurut) akan sama, apapun algoritma sorting yang digunakan untuk melakukan pengurutan.

Data yang akan diurutkan adalah data Movie dengan atribut rating, name dan year. Pengurutan akan dilakukan berdasarkan tahun (year). Pada kelas ini kita tidak menggunakan method toString() karena pencetakan objek dilakukan dengan menggunakan

method `get` (baik untuk atribut `name`, `rating`, maupun `year`). Perhatikan, method `compareTo()` mengembalikan nilai dari `year` dengan menggunakan method `getYear` (tidak langsung mengakses atribut `year`). Dengan demikian, kita akan melakukan pengurutan berdasarkan `year`. Namun, kekurangan penggunaan `Comparable` ini adalah, kita hanya dapat melakukan perbandingan pada satu atribut saja (tidak bisa digunakan untuk atribut `year`, `name`, dan `rating` sekaligus). Jika ingin untuk melakukan sorting berdasarkan lebih dari satu atribut, maka harus digunakan `Comparator`.

Studi Kasus



```
package com.nisafh;

import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        ArrayList<Mahasiswa> listMahasiswa = new ArrayList<>();
        listMahasiswa.add(new Mahasiswa("123", "Nina", "D3IF 43-03"));
        listMahasiswa.add(new Mahasiswa("111", "Ria", "D3IF 43-02"));
        listMahasiswa.add(new Mahasiswa("100", "Siti", "D3IF 43-01"));
        listMahasiswa.add(new Mahasiswa("124", "Nina", "D3IF 43-04"));
        cetakList(listMahasiswa);
        System.out.println("Masukkan NIM mahasiswa yang dicari: ");
        String keyNim = input.nextLine();
        Mahasiswa mhsResult = cariMahasiswa(listMahasiswa, keyNim);
        System.out.print("Mahasiswa dengan NIM " + keyNim);
        System.out.println(" adalah: \n" + mhsResult);
    }

    private static void cetakList(ArrayList<Mahasiswa> arrayList) {
        for (Mahasiswa mhs:arrayList) {
            System.out.println(mhs);
        }
        System.out.println();
    }

    private static Mahasiswa cariMahasiswa(ArrayList<Mahasiswa> list, String nim){
        Mahasiswa result;
        Collections.sort(list);
    }
}
```

Run: "C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:D:\Co...
Mahasiswa{nim='123', nama='Nina', kelas='D3IF 43-03'}
Mahasiswa{nim='111', nama='Ria', kelas='D3IF 43-02'}
Mahasiswa{nim='100', nama='Siti', kelas='D3IF 43-01'}
Mahasiswa{nim='124', nama='Nina', kelas='D3IF 43-04'}

Masukkan NIM mahasiswa yang dicari:
111
Mahasiswa dengan NIM 111 adalah:
Mahasiswa{nim='111', nama='Ria', kelas='D3IF 43-02'}

Process finished with exit code 0

Mahasiswa memiliki atribut NIM, nama dan kelas. Karena binary search hanya bisa diimplementasikan pada data yang telah terurut, maka objek Mahasiswa harus dapat diurutkan, dengan demikian kelas Mahasiswa harus mengimplementasikan interface `Comparable`. Mahasiswa akan diurutkan berdasarkan NIM, yang terlihat pada implementasi method `compareTo` di kelas Mahasiswa.

Saat pembentukan objek kelas Mahasiswa dan pemanggilan method binary search, kata kunci (keyword) yang dimasukkan adalah NIM. Keyword untuk binary search dan metode pengurutan pada method `compareTo` haruslah berasal dari atribut yang sama (dalam kasus ini, NIM). Kemudian, karena binary search hanya dapat mencari berdasarkan tipe data elemen yang disimpan dalam array, maka key untuk binary search seharusnya adalah elemen dengan tipe data Mahasiswa. Karena itu, ketika method binary search dari Java Collections dipanggil, yang dimasukkan adalah suatu objek Mahasiswa baru, namun dengan isi dari konstruktor hanya untuk NIM saja.

- b. Insight Anda mengenai materi pada modul 06. Apa yang Anda pahami dari materi Sorting dan Searching? Apakah ada kesulitan dalam memahami materi? Silahkan juga dituliskan hal-hal lain yang Anda inginkan.

Materi yang saya pahami adalah pengertian, penggunaan, serta fungsi dari **Binary Search dan Sorting**.

Kesulitan saya dalam memahami materi adalah sulit memahami alur logika yang ada. Sehingga, sedikit telat dalam berpikir dan menyerap informasi yang tersedia.