TUGAS BKPM 6

MATA KULIAH WORKSHOP SISTEM INFORMASI BERBASIS DESKTOP



Disusun Oleh:

FATKHUL HIDAYAH

(E41212105)

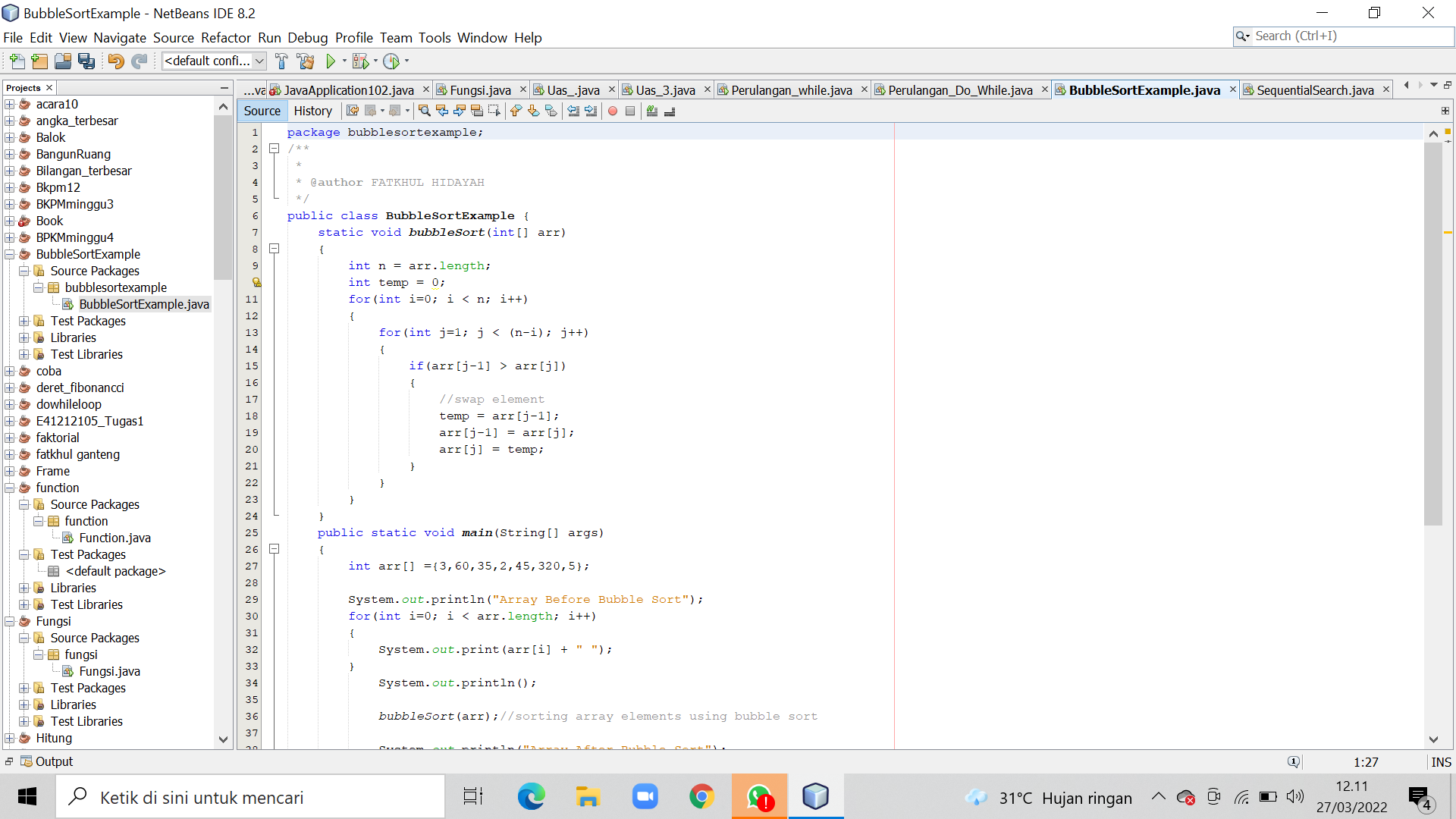
**PROGRAM STUDI TEKNIK INFORMATIKA PSDKU SIDOARJO**

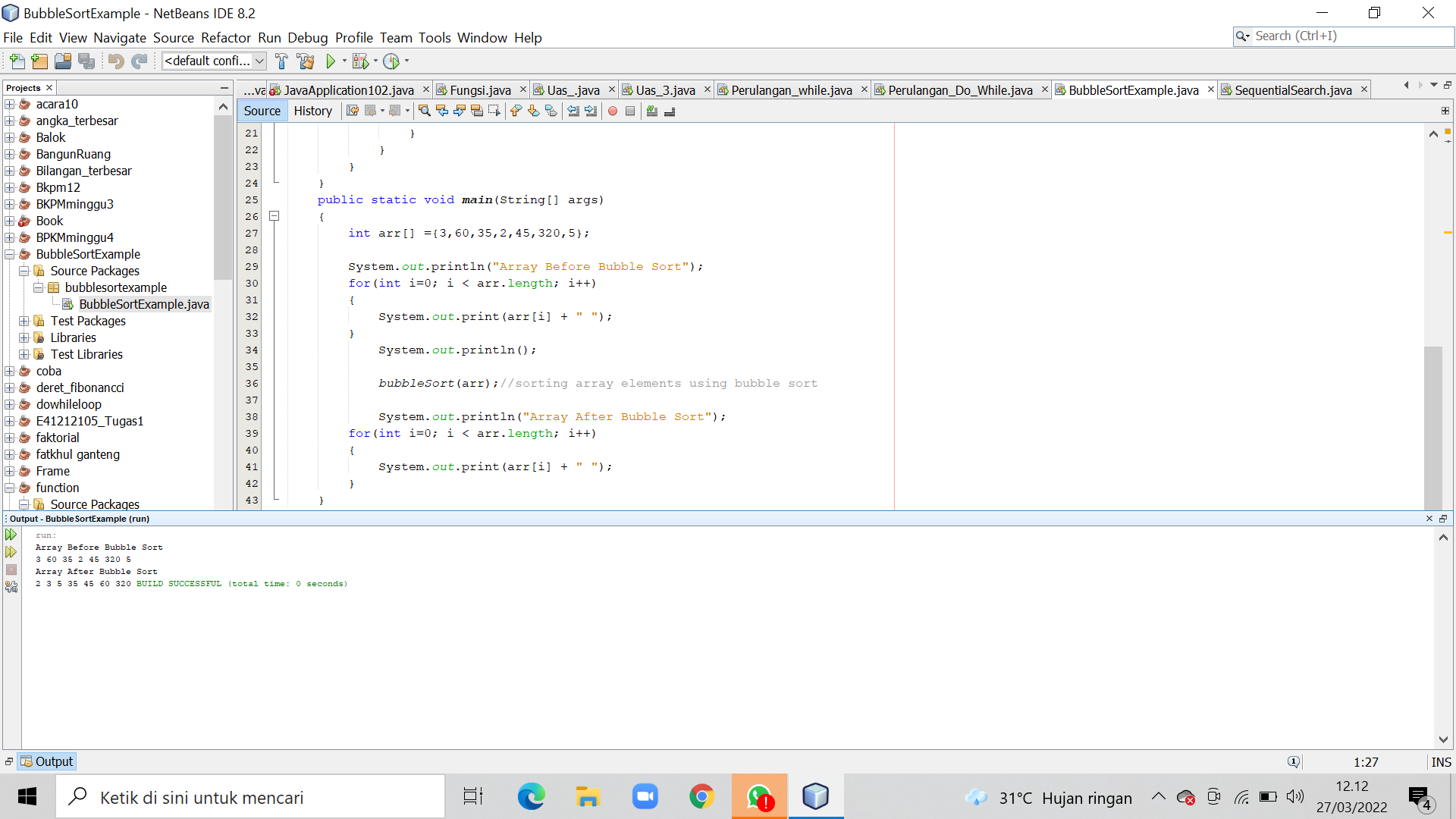
**JURUSAN TEKNOLOGI INFORMASI**

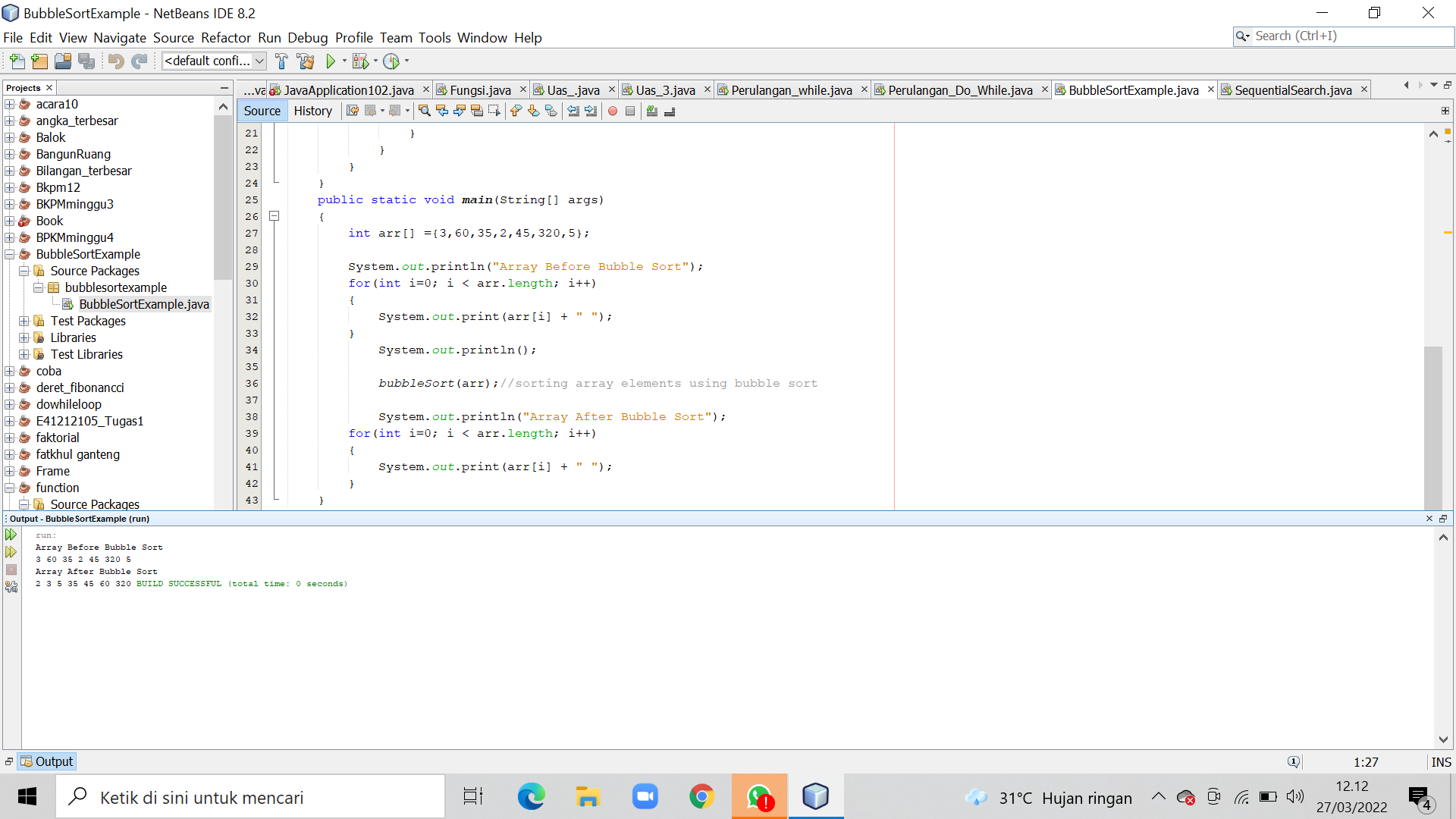
**POLITEKNIK NEGERI JEMBER**

**2022**

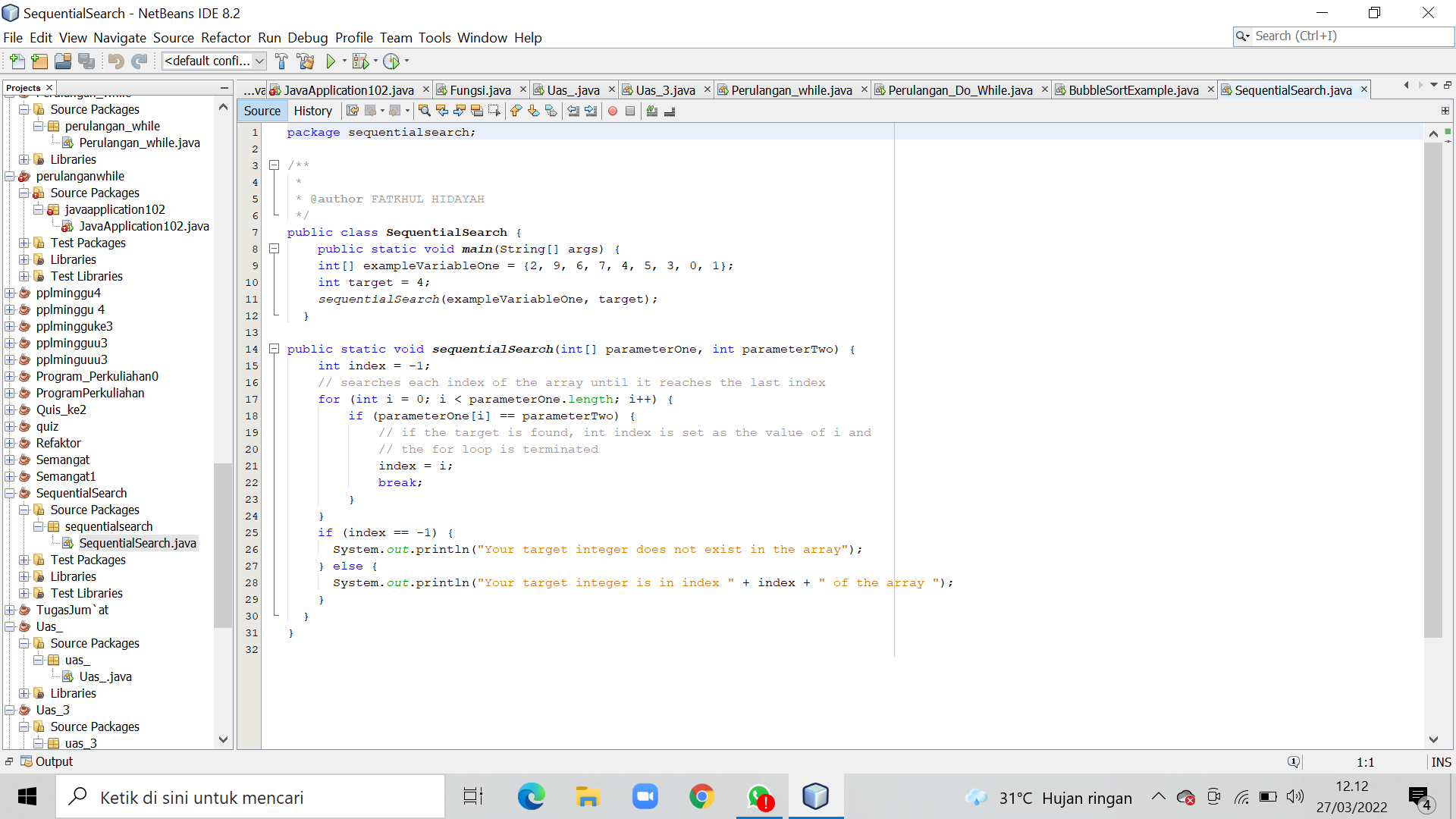
* **SYNTAX BUBBLE SORT**

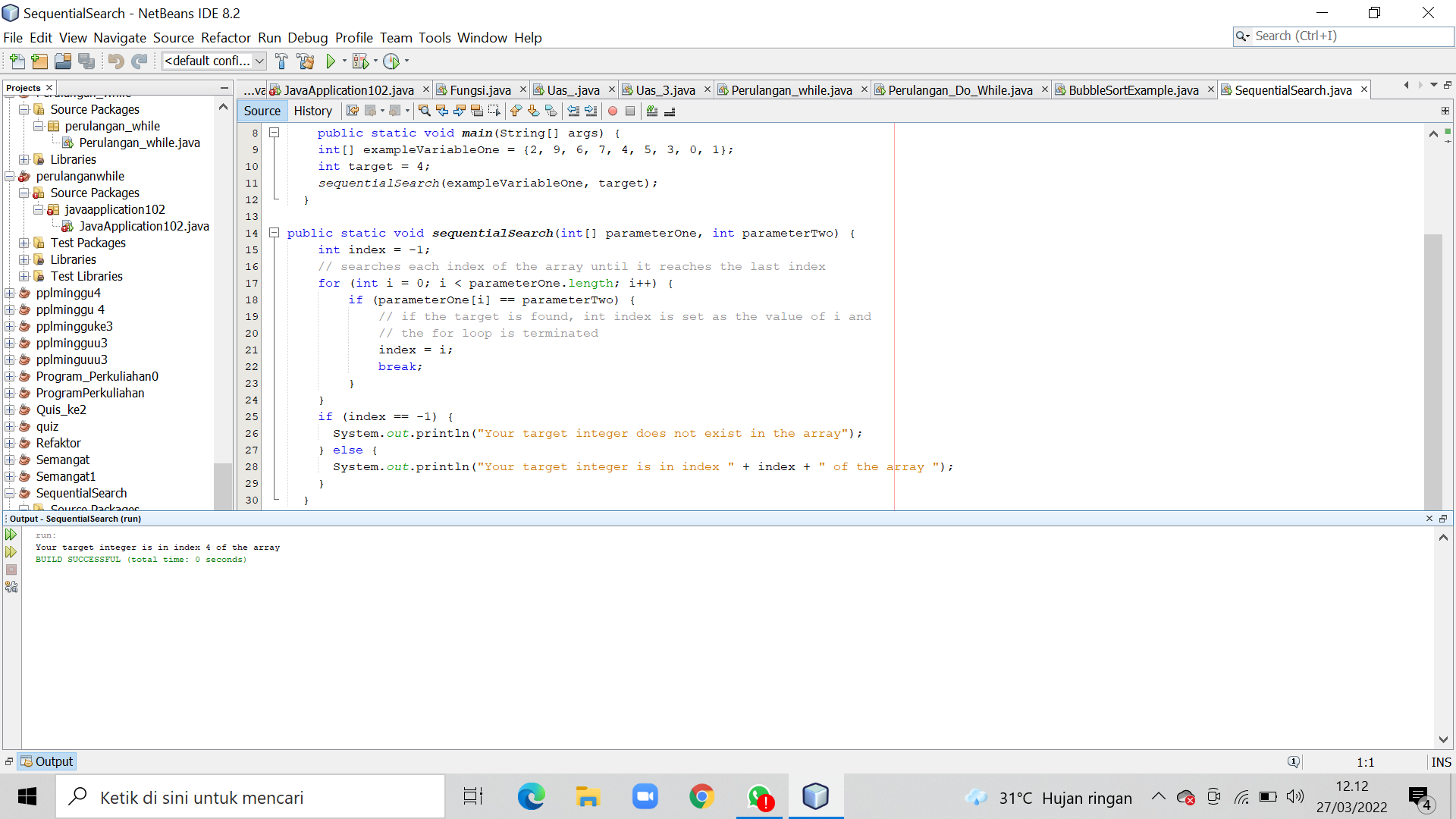






* **SYNTAX SEQUENTAL SEARCH**





* **SELECTION SORT**

1. PENGENALAN SORTING DAN INSERTION SORT

            Sorting adalah proses menyusun elemen – elemen dengan tata urut tertentu dan proses tersebut terimplementasi dalam bermacam aplikasi. Kita ambil contoh pada aplikasi perbankan. Aplikasi tersebut mampu menampilkan daftar account yang aktif.

Hampir seluruh pengguna pada sistem akan memilih tampilan daftar berurutan secara ascending demi kenyamanan dalam penelusuran data.

1. ILUSTRASI PENGURUTAN DENGAN SELECTION SORT

1)      Pengecekan dimulai dari data 1 sampai dengan data ke n

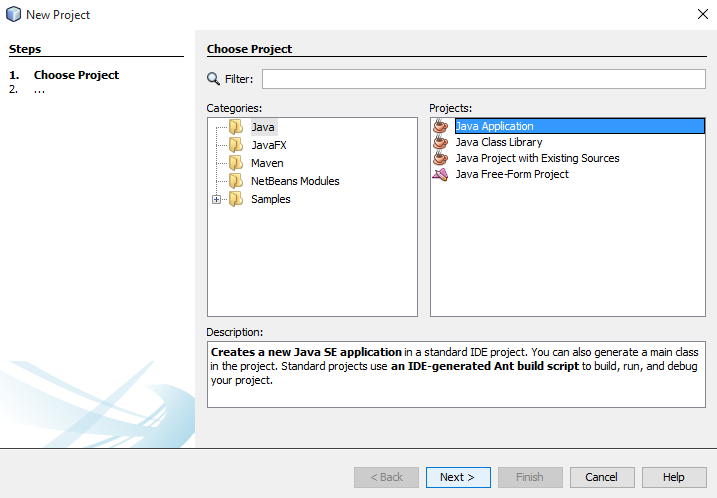
2)      Tentukan bilangan dengan index terkecil dari data bilangan tersebut

3)      Tukar bilangan dengan index terkecil tersebut dengan bilngan pertam (i=1) dari data bilngan tersebut.

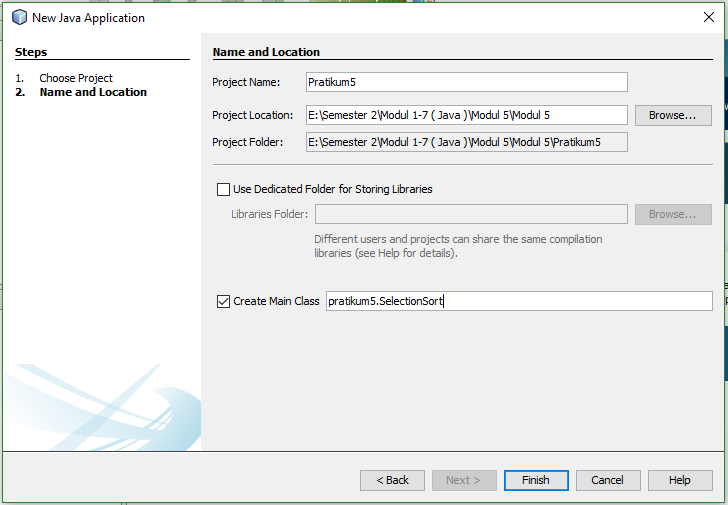
4)      Lekukan langkah 2 dan  3 untuk bilangan berikutnya (i=i+1) sampia di dapatkan data yang optimal

      3.      PRATIKUM

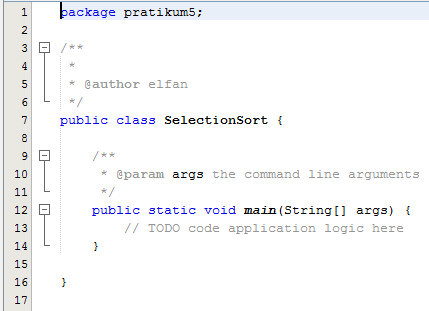
1)      Bukalah Netbeans dan buat project berupa java application:



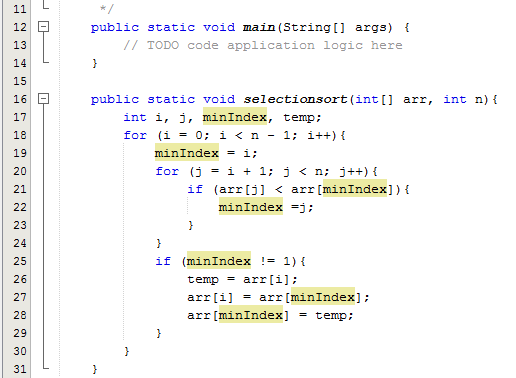
2)   Perhataikan apa yang harus diisi sesuai gambar berikut : project Name : diisi Pratikum5. Create Main Class diisi Pratikum5.selectionSort. beri tanda cek pada Set as Main Project



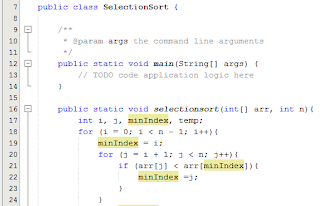
3) Perhatikan kode program yang sudah disediakan. Ada klausa package, ada kelas SelectionSort dan di dalam kelas sudah disediakan fungsi main() yang masih kosong



4)     Posisikan kursor di bawah fungsi main(), setelah kurung kurawal penutup fungsi, dan buatlah fungsi selectioSort() seperti berikut (pertikan letakanya):



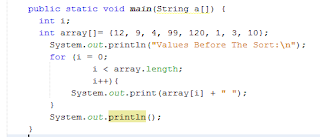
5)      Setelah fungsi selesai fungsi dibuat, pindahlah ke atas, ke fungsi main() yang masih kosong:



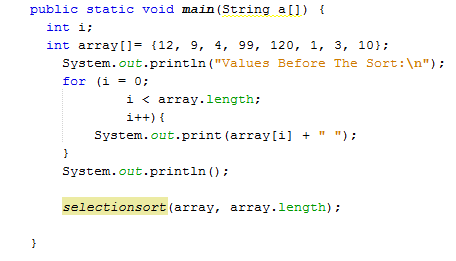
6)      Ubahlah judul fungsi main() sehingga memiliki perameter String a  (seperti berikut)



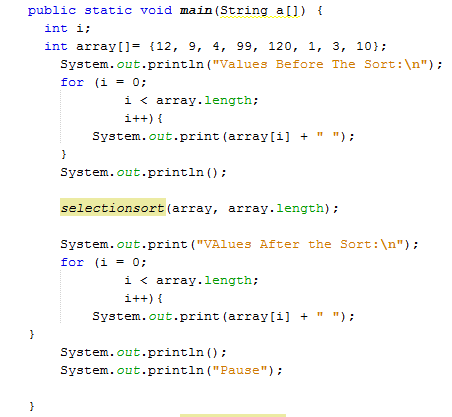
7)  Isilah variacel-variabel penctat perulangan (int i) dan variabel array integer yang akan diurutkan (), lalu tampilakn isi array dengan menggunakan perulangna (tanda kotak kedua ada gambar berikut)



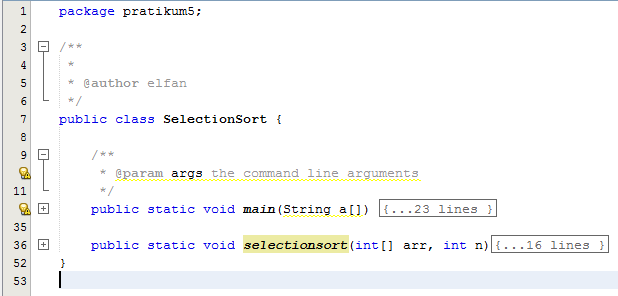
8)    Setelah isi array aslinya ditampilakan, urutkan dengan memanggil fungsi selectionSort() yang tadi sudah dibuat (perjatikan tanda kotak berikut yang menunjukkan cara pemanggilan fungsi)



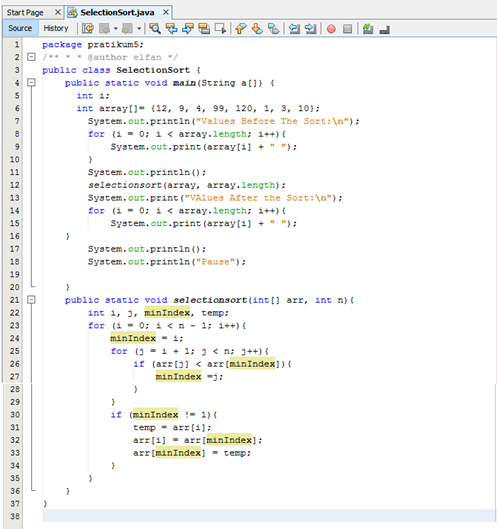
9)   Setelah diurutan, tapilkaan lagi dengan perulangan, mirip pada langkah sebelum memanggil fungsi pengurutan (perhatikan tanda kotak berikut)



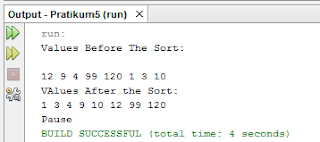
10)  Penulisan kode program anda selesai. Secara keseluruhan, struktur kode anda akan seperti ini (gambar berikut, fungsi-fungsi dicollape/disembunyikan agar mudah dilihat)



11)  Atau jika diexpand/diperlihatkan semua, struktur lengkapnya akan seperti ini (paket, kelas, fungsi )



12) Jalankan kode progam anda dengan mengklik-kanan area kaode, lalu pilih Run File (perhatikan hasilnya)



* **BINARY SEARCH**

Binary Searching merupakan salah satu metode pencarian yang kompleksitasnya cukup baik untuk melakukan pencarian data. Ada banyak metode pencarian yang ada namun, pada kesempatan ini saya akan bahas bagaimana algoritma dari Binary Searching. Pada dasarnya perhitungan pencarian Binary Searching cukup gampang dilakukan karena, sekumpulan data tersebut di urutkan terlebih dahulu secara Ascending kemudian, untuk melakukan pencariannya tiap iterasi membagi banyaknya data yang ada sehingga ditemukanlah data yang dicari.

**Proses Binary Search:**

* Iterasi 1  
  Awal: 0 --> (ingat, bahwa ini nilai index bukan nilai data dan tiap iterasi 1 pasti Awal == 0)  
  Akhir: 6 --> (nilai index yang terakhir. Perhatikan tabel sebelumnya)  
  Tengah: (Awal + Akhir) / 2 = (0 + 6) / 2 = 6 / 2 = 3  
  Value: Index 3 = 6 --> (Index 3 == 6. Perhatikan tabel sebelumnya. nilai 3 itu didapat dari Nilai Tengah)  
  Cek: **6 < 15** (berarti, value < cari maka, pada iterasi berikutnya nilai Awal = Tengah + 1)
* Iterasi 2  
  Awal: 3 + 1 = 4 (3 merupakan nilai Tengah sebelumnya. Lihat rumus diatas dan proses iterasi sebelumnya)  
  Akhir: 6 (Tetap 6)  
  Tengah: (Awal + Akhir) / 2 = (4 + 6) / 2 = 10 / 2 = 5  
  Value: Index 5 = 10 (Perhatikan tabel sebelumnya)  
  Cek: **10 < 15** (berarti, value < cari maka, pada iterasi berikutnya nilai Awal = Tengah + 1)
* Iterasi 3  
  Awal: 5 + 1 = 6  
  Akhir: 6 (Ingat, nilai Akhir akan berubah apabila rumus **value > cari** terpenuhi)  
  Tengah: (Awal + Akhir) / 2 = (6 + 6) / 2 = 12 / 2 = 6  
  Value: Index 6 = 15  
  Cek: **15 == 15** (value == cari. Hore,,,  akhirnya ketemu data yang dicari. Data ditemukan pada iterasi 3).

**Proses Binary Search**

* Iterasi 1  
  Awal: 0  
  Akhir: 6  
  Tengah: 3  
  Value: Index 3 = 6  
  Cek: 6 > 5 (value > cari maka, pada iterasi berikutnya nilai **Akhir = Tengah - 1**)
* Iterasi 2  
  Awal: 0  
  Akhir: 3 - 1 = 2  
  Tengah: 1  
  Value: Index 1 = 2  
  Cek: 2 < 5 (value < cari maka, pada iterasi berikutnya nilai **Awal = Tengah + 1**)
* Iterasi 3  
  Awal: 1 + 1 = 2  
  Akhir: 2  
  Tengah: 2  
  Value: Index 2 = 5  
  Cek: 5 == 5 (value == cari. Data ditemukan pada iterasi 3)

Dalam binary searching, jika Data yang dicari tidak ditemukan maka, Data tersebut worst case namun, apabila Data tersebut ditemukan pada iterasi pertama maka, Data tersebut merupakan best case.

**Proses Binary Search**

* Iterasi 1  
  Awal: 0  
  Akhir: 6  
  Tengah: 3  
  Value: Index 3 = 6  
  Cek: 6 < 12 (value < cari maka, pada iterasi berikutnya **Awal = Tengah + 1**)
* Iterasi 2  
  Awal: 3 + 1 = 4  
  Akhir: 6  
  Tengah: 5  
  Value: Index 5 = 10  
  Cek: 10 < 12 (value < cari maka, pada iterasi berikutnya **Awal = Tengah + 1**)
* Iterasi 3  
  Awal: 5 + 1 = 6  
  Akhir: 6  
  Tengah: 6  
  Value: Index 6 = 15  
  Cek: 15 > 12 (value > cari maka, pada iterasi berikutnya **Akhir = Tengah - 1**)
* Iterasi 4  
  Awal: 6  
  Akhir: 6 - 1 = 5  
  Tengah: 5 ( (6 + 5) / 2 = 11 / 2 = 5 --> hitung integer atau tanpa koma)  
  Value: Index 5 = 10  
  Cek: 10 < 12 (value < cari maka, pada iterasi berikutnya **Awal = Tengah + 1**)
* Iterasi 5  
  Awal: 5 + 1 = 6  
  Akhir: 5  
  Tengah: 5  
  Value: Index 5 = 10  
  Cek: 10 < 12 (value < cari maka, pada iterasi berikutnya **Awal = Tengah + 1**)
* Iterasi 6  
  Awal: 5 + 1 = 6  
  Akhir: 5  
  Tengah: 5  
  Value: Index 5 = 10  
  Cek: 10 < 12 (value < cari maka,pada iterasi beriutnya **Awal = Tengah + 1**)

terus proses iterasi berikutnya karena, nilai yang dicari memang tidak ditemukan maka, diakhir proses binary searching akan terus melakukan perulangan yang sama secara terus menerus dengan nilai yang sama sehingga system akan error. Namun, dalam program hal itu bisa Anda hindari dengan memberi batasan perulangannya apabila terjadi worst case seperti contoh kasus diatas.

**Berikut source code nya di Jawa :**

import java.util.Scanner;

/\*\*

\*

\* @author Yudi Setiawan

\*

\*/

public class MetodeBinarySearching

{

public static int[] data = null;

public static int awal, tengah, akhir, temp, count;

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

// Input jumlah data

System.out.print("Jumlah data : ");

int jlh = scan.nextInt();

// Input tiap nilai dan simpan ke array

data = new int[jlh];

for(int x = 0; x < data.length; x++)

{

System.out.print("Masukkan Data ke-"+(x+1)+" : ");

data[x] = scan.nextInt();

}

// Menampilkan data sebelum di sorting

System.out.print("\nData : ");

for(int x = 0; x < data.length; x++)

System.out.print(data[x]+" ");

// Proses sorting

sorting();

// Menampilkan Data setelah di sorting

System.out.println();

System.out.print("Sorting : ");

for(int x = 0; x < data.length; x++)

System.out.print(data[x]+" ");

// Input data yang dicari

System.out.print("\nData yang dicari : ");

int cari = scan.nextInt();

// Proses Metode Pencarian Binary Searching

System.out.println();

boolean temu = false;

awal = 0;

akhir = data.length - 1;

temp = 0;

count = 0;

int iterasi = 0;

System.out.println("It Aw Ak Te Ni");

while(temu != true)

{

tengah = (awal + akhir) / 2;

iterasi++;

// value == cari

if(data[tengah] == cari)

{

System.out.print(iterasi+" ");

System.out.print(awal+" ");

System.out.print(akhir+" ");

System.out.print(tengah+" ");

System.out.print(data[tengah]+"\n");

temu = true;

break;

}

// value < cari

else if(data[tengah] < cari)

{

System.out.print(iterasi+" ");

System.out.print(awal+" ");

System.out.print(akhir+" ");

System.out.print(tengah+" ");

System.out.print(data[tengah]+"\n");

awal = tengah + 1;

}

// value > cari

else if(data[tengah] > cari)

{

System.out.print(iterasi+" ");

System.out.print(awal+" ");

System.out.print(akhir+" ");

System.out.print(tengah+" ");

System.out.print(data[tengah]+"\n");

akhir = tengah - 1;

}

// Cek Worst Case

if(temp != data[tengah])

temp = data[tengah];

else

count++;

// batasan untuk worst case

if(count == 3)

break;

}

// Output

if(temu == true)

System.out.println("\nData "+cari+" ditemukan pada index ke-"+tengah+"\n"+

"dan Iterasi ke-"+iterasi);

else

System.out.println("\nData "+cari+" tidak ditemukan");

}

// Sorting Ascending

public static void sorting()

{

int temp = 0;

for(int x = 0; x < data.length; x++)

{

for(int y = 0; y < data.length; y++)

{

if(x == y)

continue;

else

{

if(data[x] < data[y])

{

temp = data[y];

data[y] = data[x];

data[x] = temp;

}

}

}

}

}

}