**LAPORAN HASIL PRAKTIKUM ALSD**

**JOBSHEET 9 STACK**



Faiva Puspa Sahara

244107020036

TI – 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

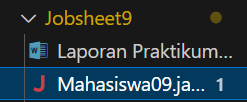
POLITEKNIK NEGERI MALANG

2025

**HASIL PRAKTIKUM**

* 1. **Percobaan 1 (Mahasiswa Mengumpulkan Tugas)**

1. **Class Mahasiswa09.java**
2. Buat folder baru bernama **Jobsheet9**. Buat file baru, beri nama **Mahasiswa09.java**



1. Lengkapi class **Mahasiswa09** dengan atribut yang telah digambarkan di dalam class diagram Mahasiswa, yang terdiri dari atribut **nama**, **nim**, **kelas**, dan **nilai**

public class Mahasiswa09 {

String nama;

String nim;

String kelas;

int nilai;

1. Tambahkan konstruktor berparameter pada class Mahasiswa sesuai dengan class diagram Mahasiswa. Berikan nilai default nilai = -1 sebagai nilai awal ketika tugas belum dinilai

public Mahasiswa09() {

}

public Mahasiswa09(String nama, String nim, String kelas) {

this.nama = nama;

this.nim = nim;

this.kelas = kelas;

nilai = -1;

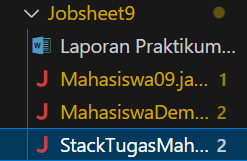
}

1. Tambahkan method **tugasDinilai()** yang digunakan untuk mengeset nilai ketika dilakukan penilaian tugas mahasiswa

void tugasDinilai (int nilai) {

this.nilai = nilai;

}

1. **Class StackTugasMahasiswa09.java**
2. Buat class **StackTugasMahasiswa09.java** tempat untuk mengelola tumpukan tugas.
3. Lengkapi class **StackTugasMahasiswa09** dengan atribut yang telah digambarkan di dalam class diagram StackTugasMahasiswa, yang terdiri dari atribut **stack**, **size**, dan **top**

public class StackTugasMahasiswa09 {

Mahasiswa09[] stack;

int size;

int top;

1. Tambahkan konstruktor berparameter untuk melakukan inisialisasi kapasitas maksimum data tugas mahasiswa yang dapat disimpan di dalam Stack, serta mengeset indeks awal dari pointer **top**

public StackTugasMahasiswa09 (int size) {

this.size = size;

stack = new Mahasiswa09[size];

top = -1;

}

1. Buat method **isFull** bertipe boolean untuk mengecek apakah tumpukan tugas mahasiswa sudah terisi penuh sesuai kapasitas

public boolean isFull() {

if (top == size -1) {

return true;

} else {

return false;

}

}

1. Buat method **isEmpty** bertipe boolean untuk mengecek apakah tumpukan tugas masih kosong

public boolean isEmpty() {

if (top == -1) {

return true;

} else {

return false;

}

}

1. Buat method **push**. Method ini menerima parameter **mhs** yang berupa object dari class **Mahasiswa09**

public void push(Mahasiswa09 mhs) {

if (!isFull()) {

top++;

stack[top] = mhs;

} else {

System.out.println("Stack penuh! Tidak bisa menambahkan tugas lagi.");

}

}

1. Penilaian tugas mahasiswa yang dilakukan oleh dosen dilakukan dengan menggunakan method **pop** untuk mengeluarkan tugas yang akan dinilai. Method ini tidak menerima parameter apapun, namun mempunyai nilai kembalian berupa object dari class **Mahasiswa09**

public Mahasiswa09 pop() {

if (!isEmpty()) {

Mahasiswa09 m = stack[top];

top--;

return m;

} else {

System.out.println("Stack kosong! Tidak ada tugas untuk dinilai.");

return null;

}

}

1. Buat method **peek** untuk dapat mengecek tumpukan tugas mahasiswa yang berada di posisi paling atas

public Mahasiswa09 peek() {

if (!isEmpty()) {

return stack[top];

} else {

System.out.println("Stack kosong! Tidak ada tugas yang dikumpulkan.");

return null;

}

}

1. Tambahkan method **print** untuk dapat menampilkan semua daftar tugas mahasiswa pada Stack

public void print() {

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

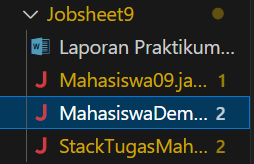
System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);

}

System.out.println("");

}

1. **Class Utama**
2. Buat file **MahasiswaDemo09.java**



1. Tuliskan struktur dasar Bahasa pemrograman java yang terdiri dari fungsi **main**

package Jobsheet9;

import java.util.Scanner;

public class MahasiswaDemo09 {

public static void main(String[] args) {

1. Di dalam fungsi **main,** lakukan instansiasi object StackTugasMahasiswa bernama **stack** dengan nilai parameternya adalah 5. Deklarasikan Scanner dengan nama variabel **scan** dan variabel **pilih** bertipe int

public static void main(String[] args) {

StackTugasMahasiswa09 stack = new StackTugasMahasiswa09(5);

Scanner sc = new Scanner(System.in);

int pilih;

1. Tambahkan menu untuk memfasilitasi pengguna dalam memilih operasi Stack dalam mengelola data tugas mahasiswa menggunakan struktur perulangan **do-while**

do {

System.out.println("\nMenu:");

System.out.println("1. Mengumpukan Tugas");

System.out.println("2. Menilai Tugas");

System.out.println("3. Melihat Tugas Teratas");

System.out.println("4. Melihat Daftar Tugas");

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

pilih = sc.nextInt();

switch (pilih) {

case 1 :

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

String nama = sc.next();

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

String nim = sc.next();

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

String kelas = sc.next();

Mahasiswa09 mhs = new Mahasiswa09(nama, nim, kelas);

stack.push(mhs);

System.out.printf("Tugas %s berhasil dikumpulkan\n", mhs.nama);

break;

case 2 :

Mahasiswa09 dinilai = stack.pop();

if (dinilai != null) {

System.out.println("Menilai tugas dari " + dinilai.nama);

System.out.print("Masukkan nilai (0-100): ");

int nilai = sc.nextInt();

dinilai.tugasDinilai(nilai);

System.out.printf("Nilai Tugas %s adalah %d\n", dinilai.nama, nilai);

}

break;

case 3 :

Mahasiswa09 lihat = stack.peek();

if (lihat != null) {

System.out.println("Tugas terakhir dikumpulkan oleh " + lihat.nama);

}

break;

case 4 :

System.out.println("Daftar semua tugas");

System.out.println("Nama\tNIM\tKelas");

stack.print();

break;

default:

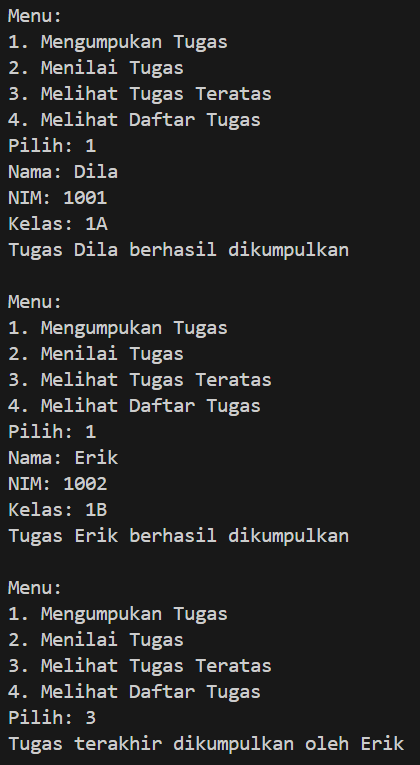
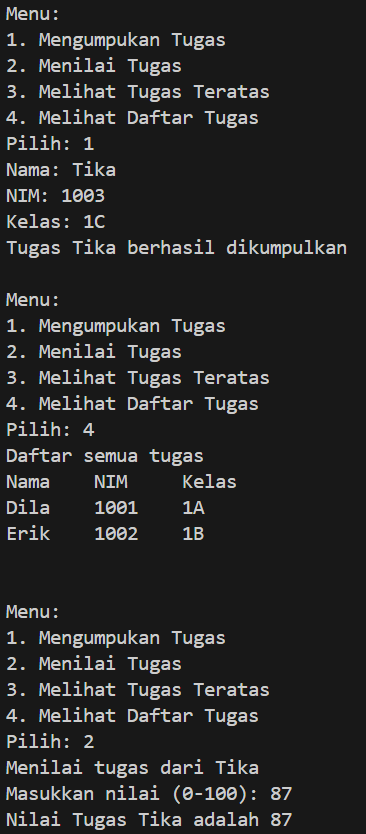
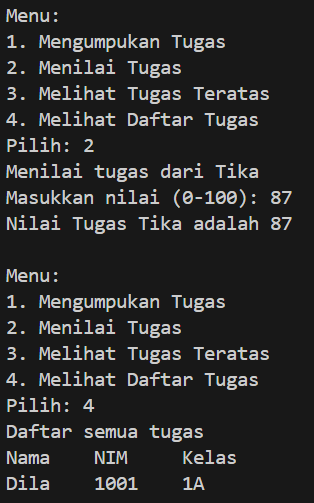
System.out.println("Pilihan tidak valid.");;

}

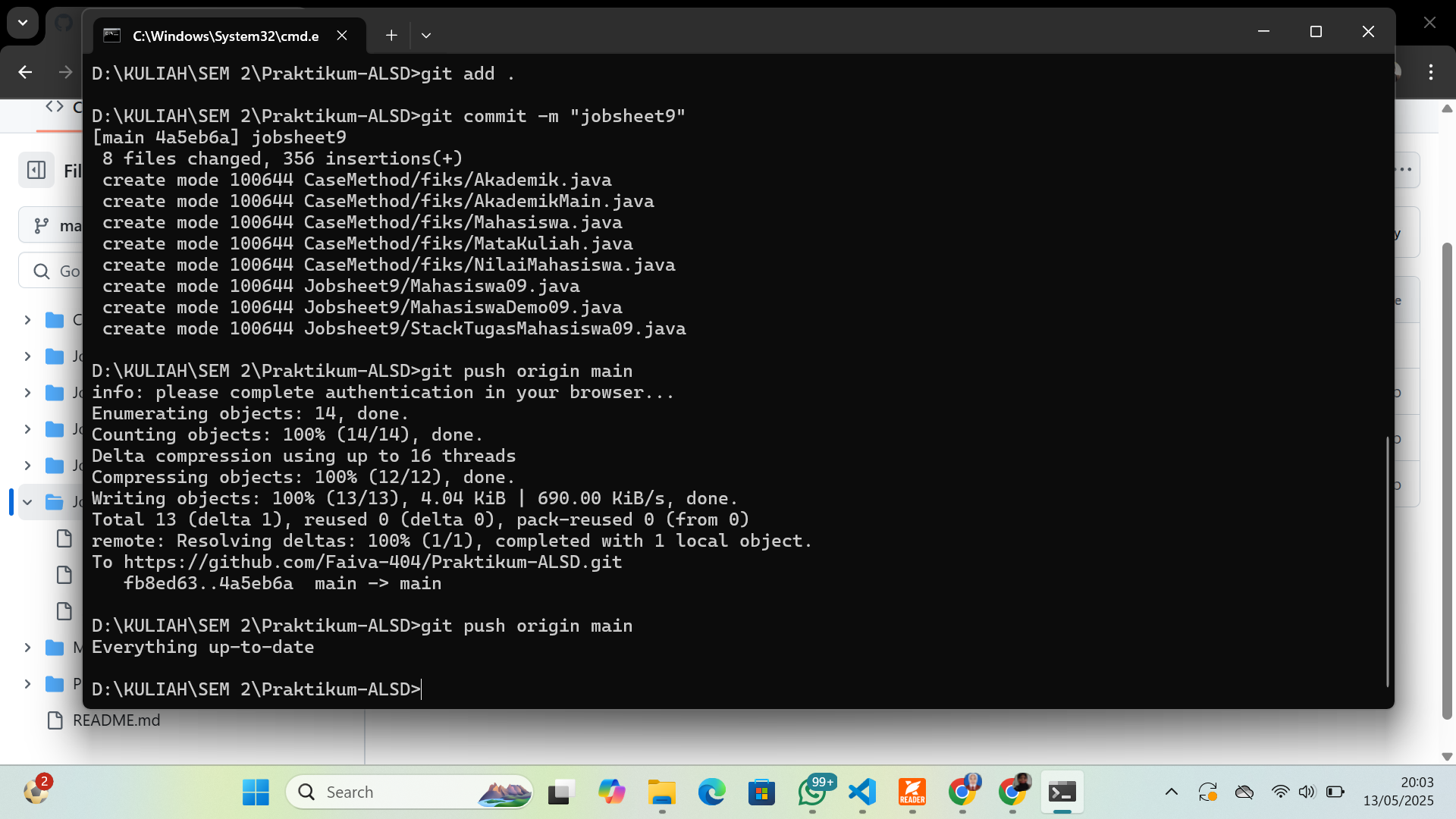
} while (pilih >= 1 && pilih <= 4);

}

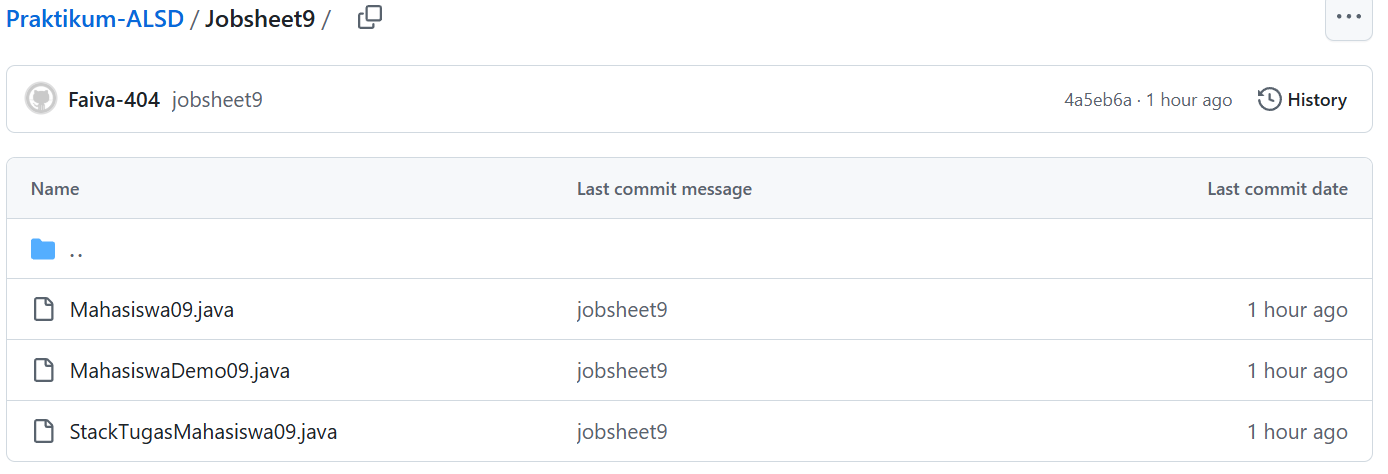
1. Run Program

1. **Commit dan push ke Github**



1. Compile dan run program



* **Pertanyaan**

1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana yang perlu diperbaiki?

* Kode Programnya

public void print() {

for (int i = 0; i >= 0; i++) {

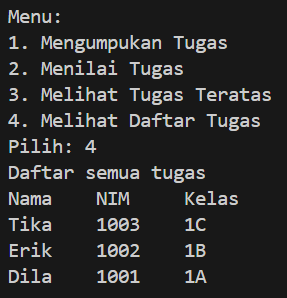
System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);

}

System.out.println("");

}

* Outputnya



1. Berapa banyak data tugas mahasiswa yang dapat ditampung di dalam Stack? Tunjukkan potongan kode programnya!

* Banyaknya data tugas mahasiswa yag ditampung, maksimal ada 5.

StackTugasMahasiswa09 stack = new StackTugasMahasiswa09(5);

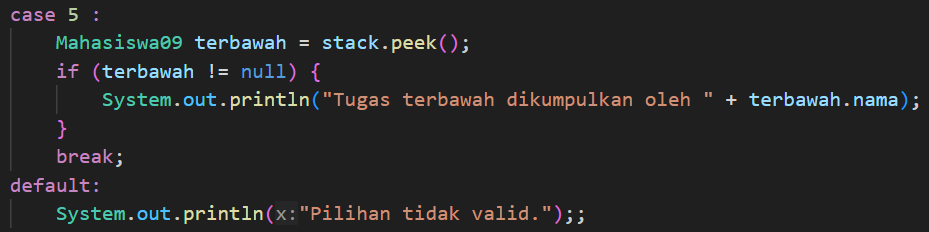
1. Mengapa perlu pengecekan kondisi **!isFull()** pada method **push**? Kalau kondisi if-else tersebut dihapus, apa dampaknya?

* Kalau pengecekan kondisi !**isFull** dihapus, dampaknya bisa fatal. Tanpa batasan, program akan tetap mencoba menambahkan elemen meskipun kapasitas sudah penuh.

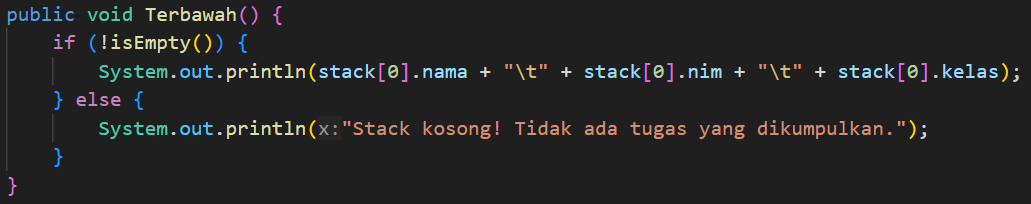
1. Modifikasi kode program pada class **MahasiswaDemo** dan **StackTugasMahasiswa** sehingga pengguna juga dapat melihat mahasiswa yang pertama kali mengumpulkan tugas melalui operasi lihat tugas terbawah!

* Kode program – pada class **MahasiswaDemo09** tambahkan menu 5 dan tambahkan case 5 di switch

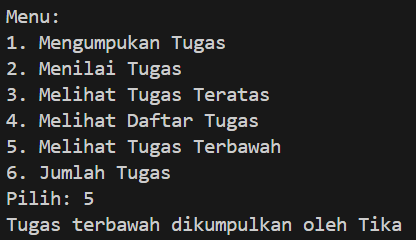




* Kode program – pada class **StackTugasMahasiswa09** tambakan method **void Terbawah**



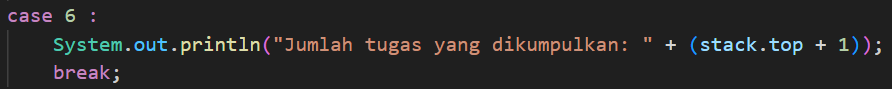
* Outputnya



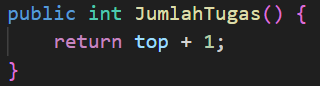
1. Tambahkan method untuk dapat menghitung berapa banyak tugas yang sudah dikumpulkan saat ini, serta tambahkan operasi menunya!

* Kode program – pada class **MahasiswaDemo09** tambahkan menu 6 dan tambahkan case 6 di switch





* Kode program – pada class **StackTugasMahasiswa09** tambakan method **JumlahTugas**



1. **Commit dan push kode program ke Github**

* 1. **Percobaan 2 (Searching – Binary Search)**

1. Tambahkan method **findBinarySearch** bertipe int pada class **MahasiswaBerprestasi09.** Deklarasikan isi method **findBinarySearch** dengan algoritma pencarian data menggunakan teknik binary searching.

int findBinarySearch(double cari, int left, int right) {

int mid;

if (right >= left) {

mid = (left+right)/2;

if (cari == listMhs[mid].ipk) {

return (mid);

} else if (listMhs[mid].ipk>cari) {

return findBinarySearch(cari, left, mid-1);

} else {

return findBinarySearch(cari, mid+1, right);

}

}

return -1;

}

1. Panggil method **findBinarySearch** terdapat pada class **MahasiswaBerprestasi09** di kelas **MahasiswaDemo09**. Kemudia panggil method **tampilPosisi** dan **tampilDataSearch.**

**//melakukan pencarian data Binary**

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

System.out.println("Pencarian data");

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

System.out.println("masukkan ipk mahasiswa yang dicari : ");

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

double cari = sc.nextDouble();

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

System.out.println("menggunakan sequential searching");

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

double posisi2 = list.findBinarySearch(cari, 0, jumMhs-1);

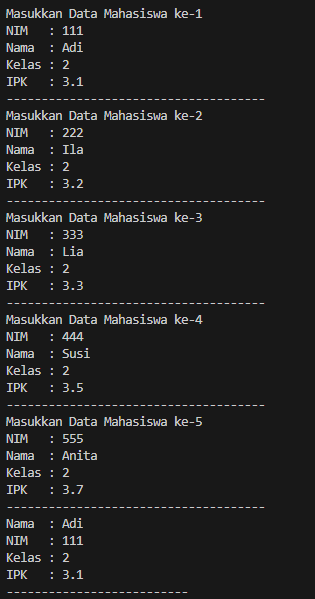
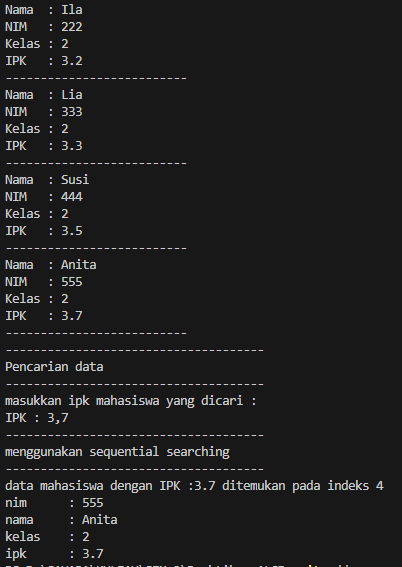
int pss2 = (int)posisi2;

list.tampilPosisi(cari, pss2);

list.tampilDataSearch(cari, pss2);

}

1. Run program



* **Pertanyaan**

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

* Proses **divide** dilakukan pada

mid = (left + right) / 2;

* Setelah nilai tengah (mid) dihitung, ruang pencarian dibagi menjadi dua. Jika nilai yang dicari lebih kecil, pencarian dilanjutkan ke kiri **findBinarySearch(cari, left, mid - 1)**. Jika lebih besar, pencarian ke kanan **findBinarySearch(cari, mid + 1, right).**

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

* Jika elemen yang dicari (cari) sama dengan elemen di indeks tengah (listMhs[mid].ipk), maka indeks tersebut dikembalikan (return (mid)), yang menandakan proses pencarian berhasil.

if (cari == listMhs[mid].ipk) {

return (mid);

}

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

* Tidak akan berjalan dengan benar jika data IPK yang dimasukkan **tidak** **urut**. Karena algoritma Binary Search **mengandalkan data yang sudah terurut** untuk membagi ruang pencarian menjadi dua bagian secara logis.
* Indeks tengah (mid) yang dihitung tidak mencerminkan posisi elemen yang lebih besar atau lebih kecil dalam data. Akibatnya, pencarian elemen menghasilkan salah atau gagal menemukan elemen yang sebenarnya ada dalam daftar.
* Untuk memastikan algoritma Binary Search berjalan benar, data perlu diurutkan dahulu menggunakan metode pengurutan **bubbleSort**, **selectionSort**, atau **insertionSort** sebelum memanggil fungsi pencarian Binary Search.

1. Jika IPK yang dimasukkan dari IPK terbesar ke terkecil (missal : 3.8, 3.7, 3.5, 3.4, 3.2) dan elemen yang dicari adalah 3.2. Bagaimana hasil dari binary search? Apakah sesuai? Jika tidak sesuai maka ubahlah kode program binary seach agar hasilnya sesuai.

* Jika IPK diurutkan dari terbesar ke terkecil, **binary search dalam kode program tidak memberikan hasil yang sesuai**. Algoritma binary search dalam kode dirancang untuk bekerja dengan data yang diurutkan secara **menaik (ascending)**, bukan **menurun (descending).**
* Ubah kode program

int findBinarySearch(double cari, int left, int right) {

int mid;

if (right >= left) {

mid = (left+right)/2;

if (cari == listMhs[mid].ipk) {

return (mid);

} else if (listMhs[mid].ipk>cari) { //ubah perbandingan

return findBinarySearch(cari, left, mid-1);

} else {

return findBinarySearch(cari, mid+1, right);

}

}

return -1;

}

1. Modifikasilah program diatas yang mana jumlah mahasiswa yang di inputkan sesuai dengan masukan dari keyboard.

* Pada class **MahasiswaBerprestasi09** modifikasi kode program

public class MahasiswaBerprestasi09 {

Mahasiswa09[] listMhs;

int idx;

MahasiswaBerprestasi09(int jumMhs) {

listMhs = new Mahasiswa09[jumMhs];

idx = 0;

}

* Pada class **MahasiswaDemo09** modifikasi kode program

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Masukkan jumlah mahasiswa: ");

int jumMhs = sc.nextInt();

MahasiswaBerprestasi09 list = new MahasiswaBerprestasi09(jumMhs);

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

System.out.println("Masukkan Data Mahasiswa ke-" + (i + 1));

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

String nim = sc.next();

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

String nama = sc.next();

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

String kelas = sc.next();

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

String ip = sc.next();

Double ipk = Double.parseDouble(ip);

sc.nextLine();

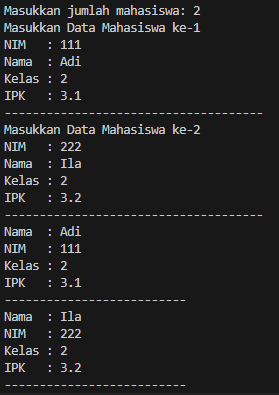
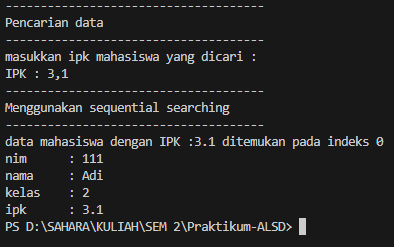
System.out.println("-------------------------------------");

Mahasiswa09 mhs = new Mahasiswa09(nim, nama, kelas, ipk);

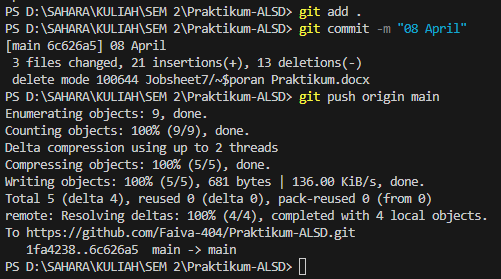
list.tambah(mhs);

}

* Run program



1. Commit dan push



* 1. **Latihan Praktikum**

1. Pada Latihan praktikum pertemuan sebelumnya pada Jobsheet 6 yang terdapat 3 class yaitu **Dosen09**, **DataDosen09**, dan **DosenDemo09**, tambahkan method:
   1. Pencarian **DataSequential09** : digunakan untuk mencari data dosen berdasarkan nama dengan algoritma sequential search.

void pencarianDataSequential09(String nama) {

int jumlahHasil = 0;

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

if (dataDosen09[i].nama.equalsIgnoreCase(nama)) {

dataDosen09[i].tampil();

jumlahHasil++;

}

}

if (jumlahHasil > 1) {

System.out.println("peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

} else if (jumlahHasil == 0) {

System.out.println("Data tidak ditemukan.");

}

}

* 1. Pencarian **DataBinary09**: digunakan untuk mencari data dosen berdasarkan usia dengan algoritma binary search.

void pencarianDataBinary09(int usia) {

sortingASC();

int kiri = 0, kanan = idx -1;

boolean ditemukan = false;

while (kiri <= kanan) {

int tengah = (kiri + kanan) / 2;

if (dataDosen09[tengah].usia == usia) {

dataDosen09[tengah].tampil();

ditemukan = true;

break;

} else if (dataDosen09[tengah].usia < usia) {

kiri = tengah + 1;

} else {

kanan = tengah -1;

}

}

if (!ditemukan) {

System.out.println("Data tidak ditemukan.");

}

}

* 1. Buat aturan untuk mendeteksi hasil pencarian lebih dari 1 hasil dalam bentuk kalimat peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!

while (pilihan != 7);

System.out.println("Peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

* Kode Program **dosen09.java**

package Jobsheet7;

public class dosen09 {

String kode;

String nama;

boolean jenisKelamin;

int usia;

dosen09 (String kd, String name, boolean jk, int age) {

kode = kd;

nama = name;

jenisKelamin = jk;

usia = age;

}

void tampil() {

System.out.println("Kode : " + kode);

System.out.println("Nama : " + nama);

System.out.println("Jenis Kelamin : " + jenisKelamin);

System.out.println("Usia : " + usia);

}

}

* Kode program **dataDosen09.java**

package Jobsheet7;

public class dataDosen09 {

dosen09[] dataDosen09 = new dosen09[9];

int idx;

void tambah(dosen09 dsn) {

if (idx < dataDosen09.length) {

dataDosen09[idx] = dsn;

idx++;

}else {

System.out.println("Data Dosen Sudah Penuh!");

}

}

//menampilkan data dosen

void tampil() {

if (idx == 0) {

System.out.println("Tidak Ada Data Dosen");

return;

}

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

dataDosen09[i].tampil();

System.out.println();

}

}

//A.

void pencarianDataSequential09(String nama) {

int jumlahHasil = 0;

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

if (dataDosen09[i].nama.equalsIgnoreCase(nama)) {

dataDosen09[i].tampil();

jumlahHasil++;

}

}

if (jumlahHasil > 1) {

System.out.println("Peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

} else if (jumlahHasil == 0) {

System.out.println("Data tidak ditemukan.");

}

}

//B.

void pencarianDataBinary09(int usia) {

sortingASC();

int kiri = 0, kanan = idx -1;

boolean ditemukan = false;

while (kiri <= kanan) {

int tengah = (kiri + kanan) / 2;

if (dataDosen09[tengah].usia == usia) {

dataDosen09[tengah].tampil();

ditemukan = true;

break;

} else if (dataDosen09[tengah].usia < usia) {

kiri = tengah + 1;

} else {

kanan = tengah -1;

}

}

if (!ditemukan) {

System.out.println("Data tidak ditemukan.");

}

}

//mengurutkan data dosen secara ascending (asc)

void sortingASC() {

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

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

if (dataDosen09[j].usia > dataDosen09[j+1].usia) {

dosen09 temp = dataDosen09[j];

dataDosen09[j] = dataDosen09[j+1];

dataDosen09[j+1] = temp;

}

}

}

}

//mengurutkan data dosen secara descending (dsc)

void sortingDSC() {

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

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

if (dataDosen09[j].usia < dataDosen09[j+1].usia) {

dosen09 temp = dataDosen09[j];

dataDosen09[j] = dataDosen09[j+1];

dataDosen09[j+1] = temp;

}

}

}

}

}

} else if (jumlahHasil == 0) {

System.out.println("Data tidak ditemukan.");

}

}

//B.

void pencarianDataBinary09(int usia) {

sortingASC();

int kiri = 0, kanan = idx -1;

boolean ditemukan = false;

while (kiri <= kanan) {

int tengah = (kiri + kanan) / 2;

if (dataDosen09[tengah].usia == usia) {

dataDosen09[tengah].tampil();

ditemukan = true;

break;

} else if (dataDosen09[tengah].usia < usia) {

kiri = tengah + 1;

} else {

kanan = tengah -1;

}

}

if (!ditemukan) {

System.out.println("Data tidak ditemukan.");

}

}

//mengurutkan data dosen secara ascending (asc)

void sortingASC() {

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

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

if (dataDosen09[j].usia > dataDosen09[j+1].usia) {

dosen09 temp = dataDosen09[j];

dataDosen09[j] = dataDosen09[j+1];

dataDosen09[j+1] = temp;

}

}

}

}

//mengurutkan data dosen secara descending (dsc)

void sortingDSC() {

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

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

if (dataDosen09[j].usia < dataDosen09[j+1].usia) {

dosen09 temp = dataDosen09[j];

dataDosen09[j] = dataDosen09[j+1];

dataDosen09[j+1] = temp;

}

}

}

}

}

* Kode program **dataDemo09.java**

package Jobsheet7;

import java.util.Scanner;

public class dosenDemo09 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

dataDosen09 datadsn = new dataDosen09();

int pilihan;

do {

System.out.println("\nMenu");

System.out.println("1. Tambah Data Dosen");

System.out.println("2. Tampilkan Data Dosen");

System.out.println("3. Sorting ASC (Usia Termuda ke Tertua)");

System.out.println("4. Sorting DSC (Usia Tertua ke Termuda)");

System.out.println("5. Mencari Nama Dosen");

System.out.println("6. Mencari Usia Dosen");

System.out.println("7. Keluar");

System.out.print("\nPilih Menu : ");

pilihan = sc.nextInt();

sc.nextLine();

switch (pilihan) {

case 1:

for (int i = 0; i < 10; i++) {

System.out.println("Masukkan data dosen ke-" + (i+1) + ":");

System.out.print("Masukkan kode dosen: ");

String kode = sc.nextLine();

System.out.print("Masukkan nama dosen: ");

String nama = sc.nextLine();

System.out.print("Masukkan jenis Kelamin (P / L) : ");

char jk = sc.next().charAt(0);

boolean jenisKelamin = (jk == '1' || jk == 'p');

System.out.print("Masukkan usia dosen: ");

int usia = sc.nextInt();

sc.nextLine();

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

dosen09 dsn = new dosen09(kode, nama, jenisKelamin, usia);

datadsn.tambah(dsn);

}

break;

case 2:

datadsn.tampil();

break;

case 3:

datadsn.sortingASC();

System.out.println("Data Dosen Telah Diurutkan Secara Ascending");

datadsn.tampil();

break;

case 4:

datadsn.sortingDSC();

System.out.println("Data Dosen Telah Diurutkan Secara Descending");

datadsn.tampil();

break;

case 5:

System.out.print("Masukkan nama dosen yang dicari: ");

String namaCari = sc.nextLine();

datadsn.pencarianDataSequential09(namaCari);

break;

case 6:

System.out.print("Masukkan usia dosen yang dicari: ");

int usiaCari = sc.nextInt();

sc.nextLine();

datadsn.pencarianDataBinary09(usiaCari);

break;

default:

System.out.println("Pilihan Tidak Valid. Silahkan Coba Lagi.");

}

}

while (pilihan != 7);

System.out.println("Peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

}

}

System.out.println("1. Tambah Data Dosen");

System.out.println("2. Tampilkan Data Dosen");

System.out.println("3. Sorting ASC (Usia Termuda ke Tertua)");

System.out.println("4. Sorting DSC (Usia Tertua ke Termuda)");

System.out.println("5. Mencari Nama Dosen");

System.out.println("6. Mencari Usia Dosen");

System.out.println("7. Keluar");

System.out.print("\nPilih Menu : ");

pilihan = sc.nextInt();

sc.nextLine();

switch (pilihan) {

case 1:

for (int i = 0; i < 10; i++) {

System.out.println("Masukkan data dosen ke-" + (i+1) + ":");

System.out.print("Masukkan kode dosen: ");

String kode = sc.nextLine();

System.out.print("Masukkan nama dosen: ");

String nama = sc.nextLine();

System.out.print("Masukkan jenis Kelamin (P / L) : ");

char jk = sc.next().charAt(0);

boolean jenisKelamin = (jk == '1' || jk == 'p');

System.out.print("Masukkan usia dosen: ");

int usia = sc.nextInt();

sc.nextLine();

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

dosen09 dsn = new dosen09(kode, nama, jenisKelamin, usia);

datadsn.tambah(dsn);

}

break;

case 2:

datadsn.tampil();

break;

case 3:

datadsn.sortingASC();

System.out.println("Data Dosen Telah Diurutkan Secara Ascending");

datadsn.tampil();

break;

case 4:

datadsn.sortingDSC();

System.out.println("Data Dosen Telah Diurutkan Secara Descending");

datadsn.tampil();

break;

case 5:

System.out.print("Masukkan nama dosen yang dicari: ");

String namaCari = sc.nextLine();

datadsn.pencarianDataSequential09(namaCari);

break;

case 6:

System.out.print("Masukkan usia dosen yang dicari: ");

int usiaCari = sc.nextInt();

sc.nextLine();

datadsn.pencarianDataBinary09(usiaCari);

break;

default:

System.out.println("Pilihan Tidak Valid. Silahkan Coba Lagi.");

}

}

while (pilihan != 7);

System.out.println("Peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

}

}

Silahkan Coba Lagi.");

}

}

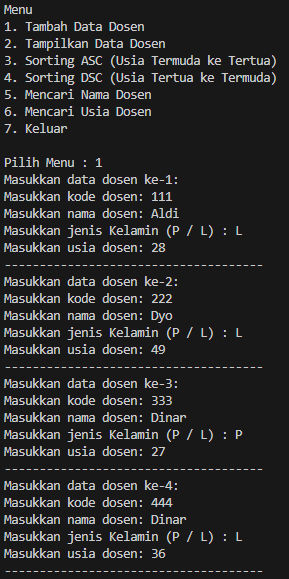
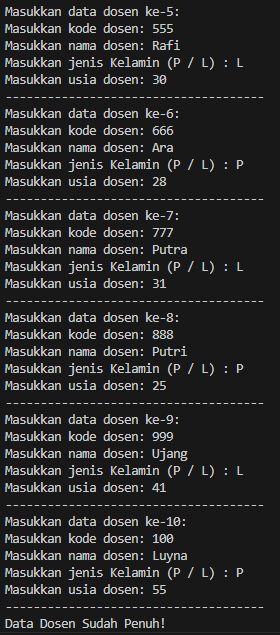
while (pilihan != 7);

System.out.println("Peringatan! Pastikan algoritma yang diterapkan sesuai dengan kasus yang diberikan!");

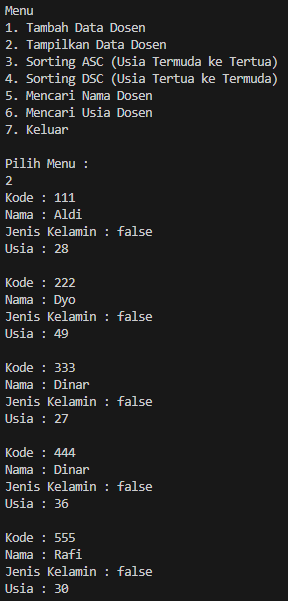
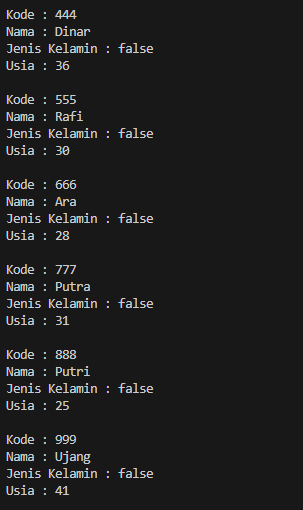
}

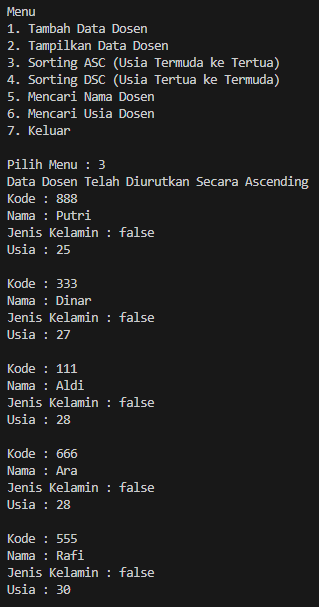
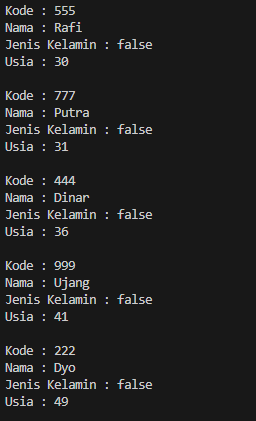
}

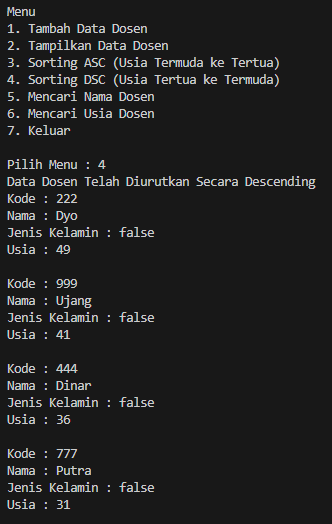
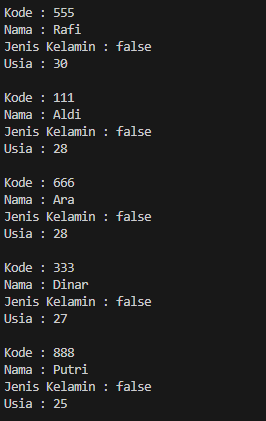
* Output Menu 1

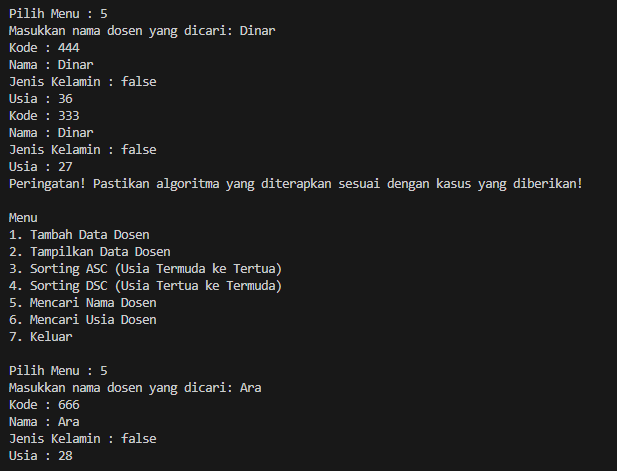


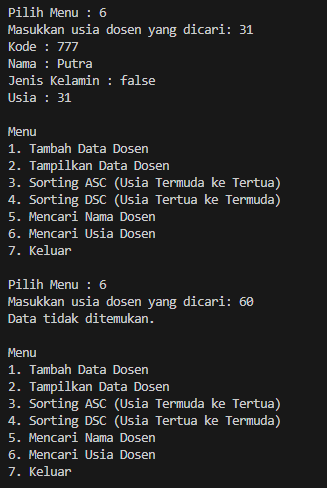
* Output Menu 2

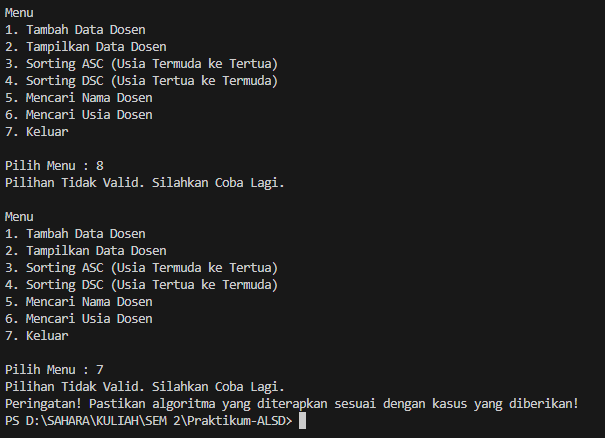
* Output Menu 3
* Output Menu 4

* Output Menu 5
* Output Menu 6



* Output Menu 7



* Commit dan Push

