**LAPORAN HASIL PRAKTIKUM ALSD**

**JOBSHEET 9 STACK**



Faiva Puspa Sahara

244107020036

TI – 1E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

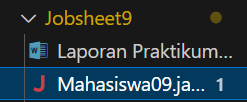
POLITEKNIK NEGERI MALANG

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;

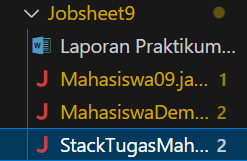
}

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++) {

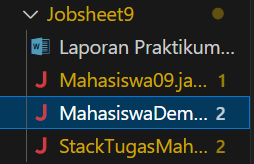
System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);

}

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:

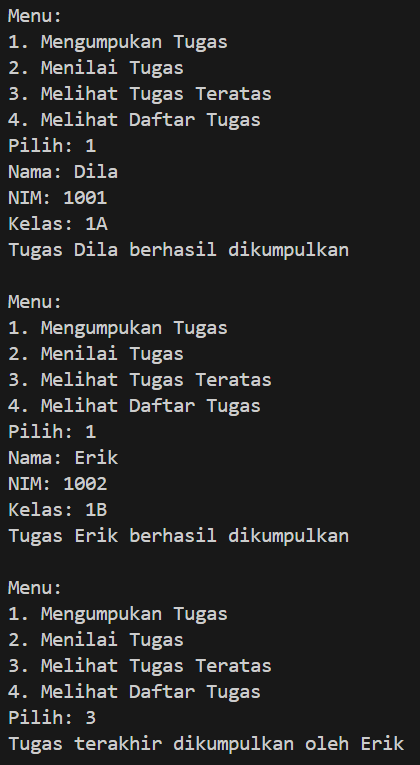
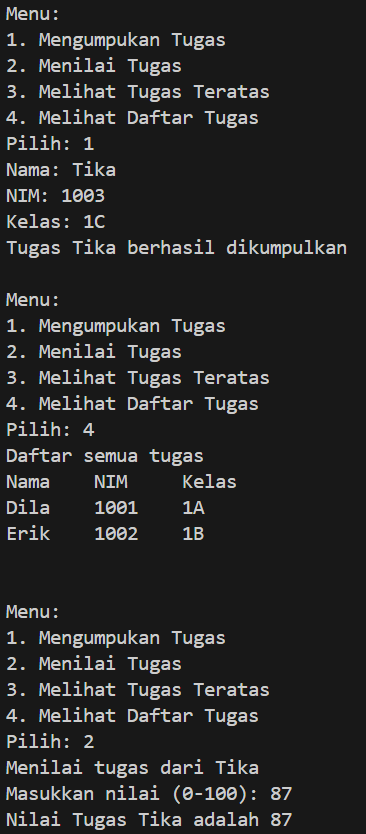
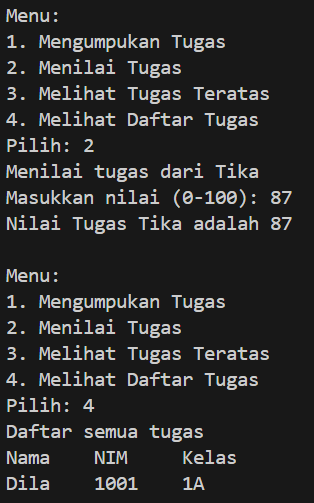
System.out.println("Pilihan tidak valid.");;

}

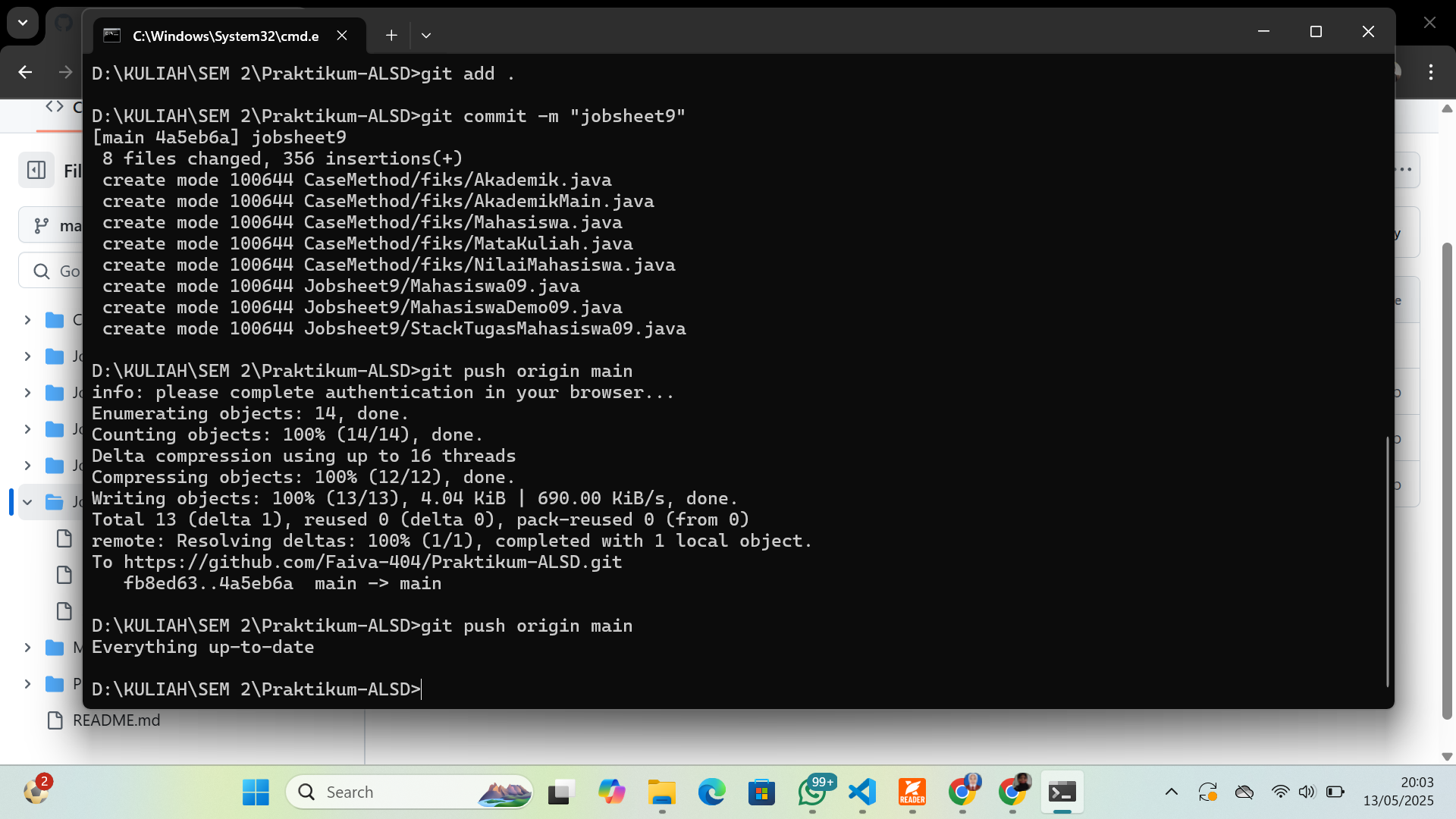
} while (pilih >= 1 && pilih <= 4);

}

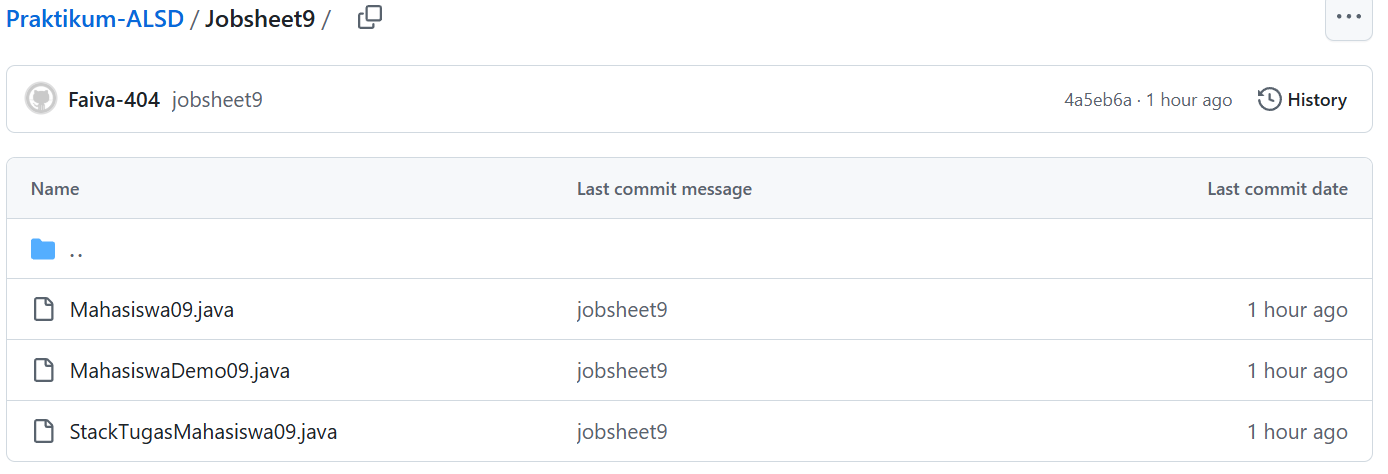
1. Run Program

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++) {

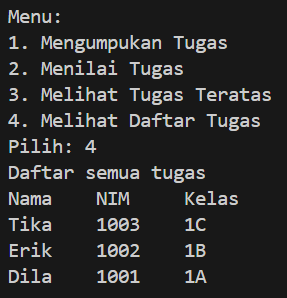
System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);

}

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);

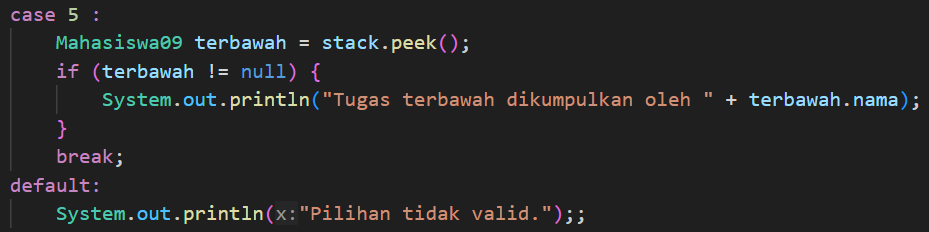
1. Mengapa perlu pengecekan kondisi **!isFull()** pada method **push**? Kalau kondisi if-else tersebut dihapus, apa dampaknya?

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

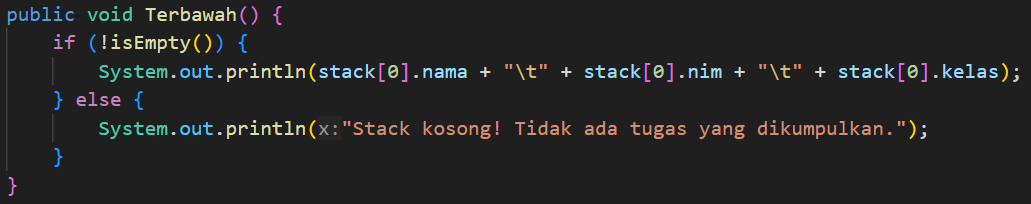
1. Modifikasi kode program pada class **MahasiswaDemo** dan **StackTugasMahasiswa** sehingga pengguna juga dapat melihat mahasiswa yang pertama kali mengumpulkan tugas melalui operasi lihat tugas terbawah!

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

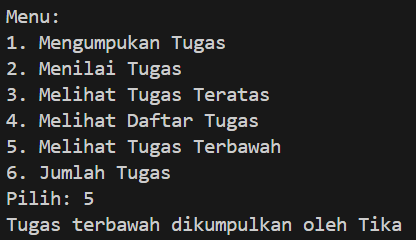




* Kode program – pada class **StackTugasMahasiswa09** tambakan method **void Terbawah**



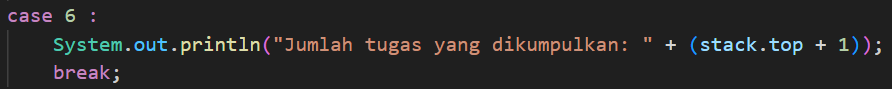
* Outputnya



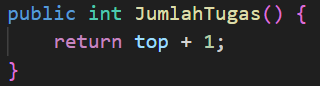
1. Tambahkan method untuk dapat menghitung berapa banyak tugas yang sudah dikumpulkan saat ini, serta tambahkan operasi menunya!

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

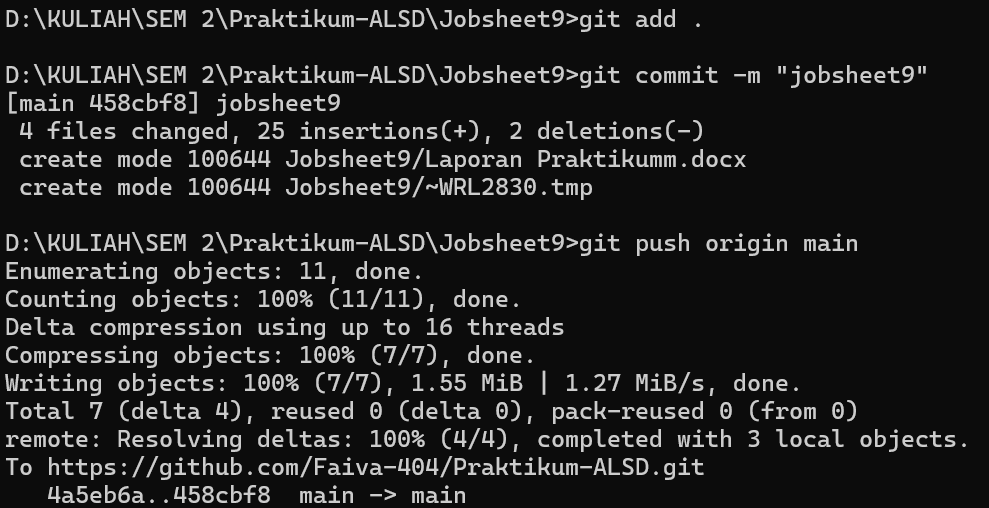




* Kode program – pada class **StackTugasMahasiswa09** tambakan method **JumlahTugas**

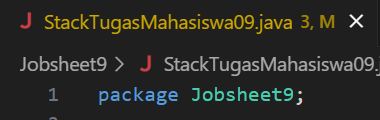


1. **Commit dan push kode program ke Github**



* 1. **Percobaan 2 (Konversi Nilai Tugas ke Biner)**

1. Buka kembali file **StackTugasMahasiswa09.java**



1. Tambahkan method **konversiDesimalKeBiner** dengan menerima parameter **kode** bertipe int

public String konversiDesimalKeBiner(int nilai) {

StackKonversi09 stack = new StackKonversi09();

while (nilai > 0) {

int sisa = nilai % 2;

stack.push(sisa);

nilai = nilai / 2;

}

String biner = new String();

while (!stack.isEmpty()) {

biner += stack.pop();

}

return biner;

}

1. Tambahkan empat method yaitu **isEmpty**, **isFull**, **push**, dan **pull** sebagai operasi utama Stack pada class **StackKonversi09**

package Jobsheet9;

public class StackKonversi09 {

int[] tumpukanBiner;

int size;

int top;

public StackKonversi09() {

this.size = 32;

tumpukanBiner = new int[size];

top = -1;

}

public boolean isEmpty() {

return top == -1;

}

public boolean isFull() {

return top == size - 1;

}

public void push(int data) {

if (!isFull()) {

top++;

tumpukanBiner[top] = data;

} else {

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

}

}

public int pop() {

if (!isEmpty()) {

int data = tumpukanBiner[top];

top--;

return data;

} else {

System.out.println("Stack kosong! Tidak ada data untuk diambil.");

return -1;

}

}

}

public int pop() {

if (!isEmpty()) {

int data = tumpukanBiner[top];

top--;

return data;

} else {

System.out.println("Stack kosong! Tidak ada data untuk diambil.");

return -1;

}

}

}

1. Agar nilai tugas mahasiswa dikonversi ke dalam bentuk biner setelah dilakukan penilaian, maka tambahkan baris kode program pada method **pop** di class **MahasiswaDemo09**

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);

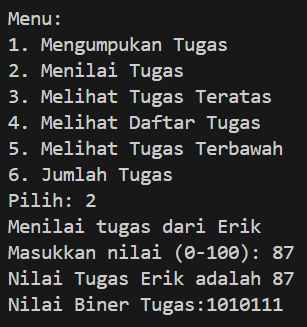
String biner = stack.konversiDesimalKeBiner(nilai);

System.out.println("Nilai Biner Tugas:" + biner);

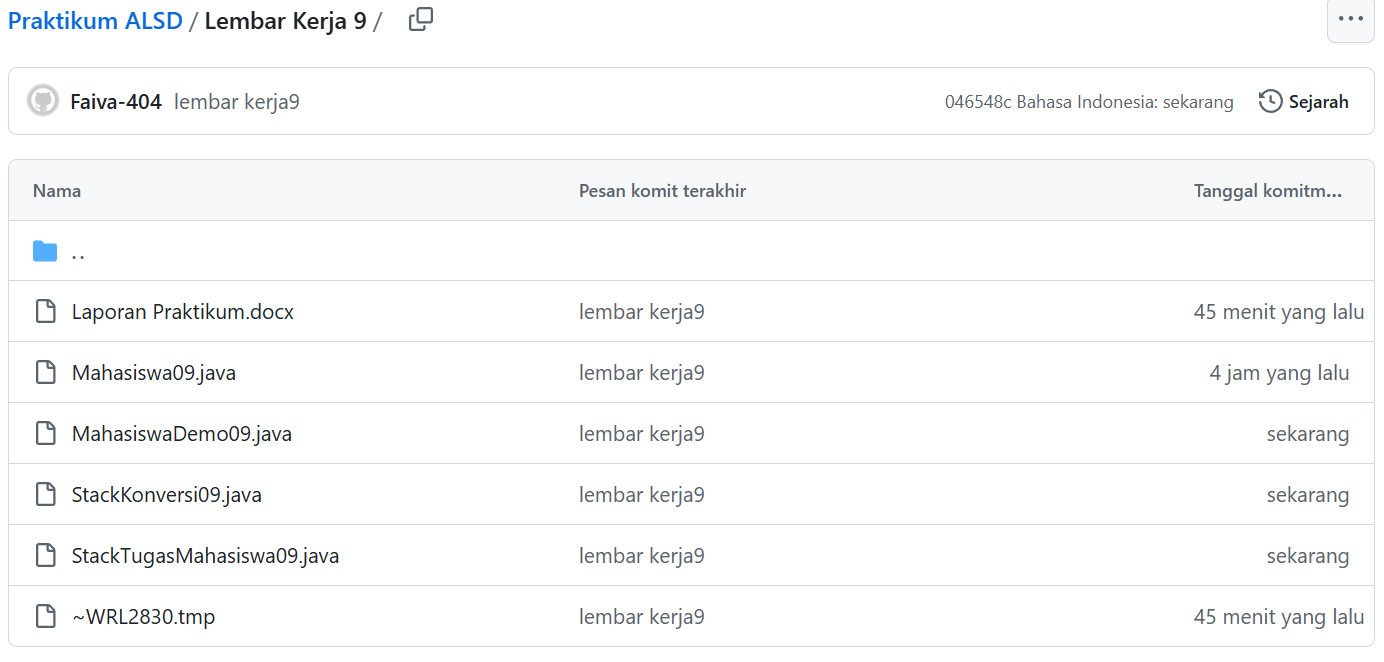
}

break;

1. Run Program



1. Commit dan Push ke Github



* **Pertanyaan**

1. Jelaskan alur kerja dari method **konversiDesimalKeBiner**!

* **Membagi angka secara berulang dengan 2** sambil mencatat **sisa hasil bagi** di setiap langkah. Sisa pembagian ini nantinya akan membentuk angka biner.

1. Pada method **konversiDesimalKeBiner**, ubah kondisi perulangan menjadi **while (kode != 0)**, bagaimana hasilnya? Jelaskan alasannya!

* Kode Programnya

public String konversiDesimalKeBiner(int nilai) {

StackKonversi09 stack = new StackKonversi09();

while (kode != 0) {

int sisa = nilai % 2;

stack.push(sisa);

nilai = nilai / 2;

}

String biner = new String();

while (!stack.isEmpty()) {

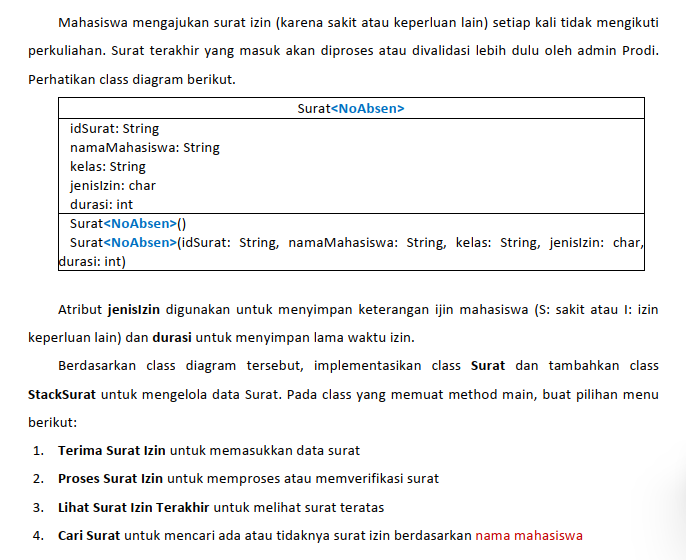
biner += stack.pop();

}

return biner;

}

* Penjelasan :
* program akan mengalami error karena variabel **kode** **tidak dideklarasikan atau tidak ada dalam metode.**

* 1. **Latihan Praktikum**