

Laporan Hasil Praktikum
Algoritma dan Struktur Data
Jobsheet 9 Stack



Muhammad Firman Aditiasmara

244107020094

TI-1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Membuat struktur data Stack
2. Menerapkan algoritma Stack ke dalam program Java

2. Praktikum

2.1 Percobaan 1: Mahasiswa Mengumpulkan Tugas

2.1.1 Langkah-langkah Percobaan

A. Class Mahasiswa

1. Menambahkan deklarasi pada class Mahasiswa18

```
public class Mahasiswa18 {  
  
    String nim;  
    String nama;  
    String kelas;  
    int nilai;  
}
```

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

```
public Mahasiswa18(String nama, String nim, String kelas) {  
    this.nama = nama;  
    this.nim = nim;  
    this.kelas = kelas;  
    nilai = -1;  
}
```

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

```
void tugasDinilai(int nilai) {  
    this.nilai = nilai;  
}
```

B. Class StackTugasMahasiswa

4. Mengisi class StackTugasMahasiswa dengan atribut yang telah digambarkan di dalam class diagram StackTugasMahasiswa, yang terdiri

```
Mahasiswa18[] stack;  
int top;  
int size;
```

dari atribut stack, size, dan top

5. Menambahkan konstruktor berparameter pada class StackTugasMahasiswa untuk melakukan inisialisasi kapasitas maksimum data, serta mengeset indeks awal dari pointer top

```

public StackTugasMahasiswa18(int size) {
    this.size = size;
    stack = new Mahasiswa18[size];
    top = -1;
}

```

6. Membuat 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;
    }
}

```

7. Membuat method `isEmpty` bertipe boolean untuk mengecek apakah tumpukan tugas masih kosong

```

public boolean isEmpty() {
    if (top == -1) {
        return true;
    } else {
        return false;
    }
}

```

8. Membuat method `push`. Method ini menerima parameter `mhs` yang berupa object dari class `Mahasiswa`

```

public void push(Mahasiswa18 mhs) {
    if (!isFull()) {
        top++;
        stack[top] = mhs;
    } else {
        System.out.println("Stack penuh! Tidak bisa menambahkan tugas lagi.");
    }
}

```

9. Membuat method `pop` untuk mengeluarkan tugas yang akan dinilai.

```

public Mahasiswa18 pop() {
    if (!isEmpty()) {
        Mahasiswa18 m = stack[top];
        top--;
        return m;
    } else {
        System.out.println("Stack kosong! Tidak ada tugas untuk dinilai.");
        return null;
    }
}

```

10. Membuat method peek untuk dapat mengecek tumpukan tugas mahasiswa yang berada di posisi paling atas

```
public Mahasiswa18 peek() {  
    if (!isEmpty()) {  
        return stack[top];  
    } else {  
        System.out.println("Stack kosong! Tidak ada  
tugas yang dikumpulkan.");  
        return null;  
    }  
}
```

11. Menambahkan 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("");  
}
```

C. Class Utama (Kelas Main)

1. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi main, di dalam fungsi main, lakukan instansiasi object StackTugasMahasiswa bernama stack dengan nilai parameternya adalah 5. Dan mendeklarasikan Scanner dengan nama variabel scan dan variabel pilih bertipe int

```
import java.util.Scanner;  
  
public class MahasiswaDemo18 {  
  
    public static void main(String[] args) {  
        StackTugasMahasiswa18 stack = new  
StackTugasMahasiswa18(5);  
        Scanner scan = new Scanner(System.in);  
        int pilih;
```

2. Menambahkan menu untuk memfasilitasi pengguna dalam memilih operasi Stack dalam mengelola data tugas mahasiswa menggunakan struktur perulangan do-while

```

import java.util.Scanner;

public class MahasiswaDemol8 {
    public static void main(String[] args) {
        StackTugassMahasiswa18 stack = new StackTugassMahasiswa18(5);
        Scanner scan = new Scanner(System.in);
        int pilih;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Mengumpulkan Tugas");
            System.out.println("2. Menilai Tugas");
            System.out.println("3. Melihat Tugas");
            System.out.println("4. Melihat Daftar Tugas");
            System.out.print("Pilih: ");
            pilih = scan.nextInt();
            scan.nextLine();

            switch (pilih) {
                case 1:
                    System.out.print("Nama: ");
                    String nama = scan.nextLine();
                    System.out.print("NIM: ");
                    String nim = scan.nextLine();
                    System.out.print("Kelas: ");
                    String kelas = scan.nextLine();
                    Mahasiswa18 mhs = new Mahasiswa18(nama, nim, kelas);
                    stack.push(mhs);
                    System.out.printf("Tugas %s berhasil dikumpulkan\n", mhs.nama);
                    break;
                case 2:
                    Mahasiswa18 dinilai = stack.pop();
                    if (dinilai != null) {
                        System.out.println("Menilai tugas dari " + dinilai.nama);
                        System.out.print("Masukkan nilai (0-100): ");
                        int nilai = scan.nextInt();
                        dinilai.tugasDinilai(nilai);
                        System.out.printf("Nilai Tugas %s adalah %d\n",
dinilai.nama, nilai);
                    }
                    break;
                case 3:
                    Mahasiswa18 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);
    }
}

```

2.1.2 Verifikasi Hasil Percobaan

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 1

Nama: Dila

NIM: 1001

Kelas: 1A

Tugas Dila berhasil dikumpulkan

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 1

Nama: Erik

NIM: 1002

Kelas: 1B

Tugas Erik berhasil dikumpulkan

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 3

Tugas terakhir dikumpulkan oleh Erik

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 1

Nama: Tika

NIM: 1003

Kelas: 1C

Tugas Tika berhasil dikumpulkan

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 2

Menilai tugas dari Tika

Masukkan nilai (0-100): 87

Nilai Tugas Tika adalah 87

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas

Pilih: 4

Daftar semua tugas

Nama	NIM	Kelas
------	-----	-------

Dila	1001	1A
------	------	----

Erik	1002	1B
------	------	----

2.1.3 Pertanyaan

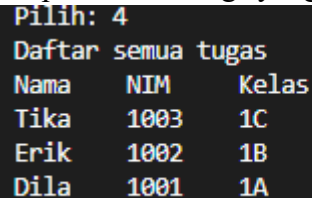
1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana yang perlu diperbaiki?

Jawab :

Bagian yang perlu diperbaiki adalah pada bagian method print

```
public void print() {  
    for (int i = top; i >= 0; i--) {  
        System.out.println(stack[i].nama + "\t" +  
stack[i].nim + "\t" + stack[i].kelas);  
    }  
    System.out.println("");  
}
```

Dengan perbaikan kode tersebut maka akan menghasilkan output tampilan daftar tugas yang sesuai dengan verifikasi hasil percobaan



```
Pilih: 4  
Daftar semua tugas  
Nama    NIM    Kelas  
Tika    1003   1C  
Erik    1002   1B  
Dila    1001   1A
```

Setelah diubah maka hasil outputnya akan menampilkan daftar tugas dari tugas yang terakhir kali diinputkan

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

Jawab : Banyak tugas mahasiswa yang dapat ditampung adalah 5 data

```
StackTugasMahasiswa18 stack = new  
StackTugasMahasiswa18(5);
```

Stack tersebut menampung 5 data sesuai dengan lebar size yang kita tentukan

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

Jawab :

Pengecekan !isFull() berfungsi untuk mencegah penambahan data ke stack yang sudah penuh. Jika kondisi if-else tersebut dihapus maka program akan error saat stack penuh, dan proses selanjutnya gagal dijalankan.

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

Jawab :

- Modifikasi class MahasiswaDemoo

1. Penambahan menu

```
do {  
    System.out.println("\nMenu:");  
    System.out.println("1. Mengumpulkan  
Tugas");  
    System.out.println("2. Menilai  
Tugas");  
    System.out.println("3. Melihat Tugas  
Teratas");  
    System.out.println("4. Melihat  
Daftar Tugas");  
    System.out.println("5. Melihat Tugas  
Terbawah"); // Pertanyaan nomor 4 percobaan 1  
    System.out.print("Pilih: ");  
    pilih = scan.nextInt();  
    scan.nextLine();  
}
```

2. Penambahan case 5 pada switch-case untuk kondisi jika dipilih menu 5

```
case 5:  
    Mahasiswa18 seek =  
stack.peekBot();  
    if (seek != null) {  
        System.out.println("Tuga  
s pertama dikumpulkan oleh " + seek.nama);  
    }
```

3. Perubahan Batasan pada while untuk menjalankan program jika pilihan diantara 1 sampai 5

```
} while (pilih >= 1 && pilih <= 5);
```

- Modifikasi class StackTugasMahasiswa

Penambahan method peekBot untuk melihat tugas dari yang terbawah


```

public Mahasiswa18 peekBot() {
    if (!isEmpty()) {
        return stack[0];
    } else {
        System.out.println("Stack kosong!
Tidak ada tugas yang dikumpulkan.");
        return null;
    }
}

```

Hasil compile

```

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
Pilih: 4
Daftar semua tugas
Nama    NIM    Kelas
Tika    1003    1C
Erik    1002    1B
Dila    1001    1A

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
Pilih: 3
Tugas terakhir dikumpulkan oleh Tika

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
Pilih: 5
Tugas pertama dikumpulkan oleh Dila
Jumlah tugas yang dikumpulkan: 3

```

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

Jawab :

- Modifikasi pada class MahasiswaDemo
 1. Penambahan menu untuk pilihan menghitung tugas

```

do {
    System.out.println("\nMenu:");
    System.out.println("1. Mengumpulkan Tugas");
    System.out.println("2. Menilai Tugas");
    System.out.println("3. Melihat Tugas Teratas");
    System.out.println("4. Melihat Daftar Tugas");
    System.out.println("5. Melihat Tugas Terbawah");
    // Pertanyaan nomor 4 percobaan 1
    System.out.println("6. Melihat Jumlah Tugas ");
    // Pertanyaan nomor 5 percobaan 1
    System.out.print("Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
}

```

2. Penambahan switch-case jika dipilih case 6

```

case 6:
    int jumlah =
stack.jumlahTugas();
    System.out.println("Jumlah
tugas yang dikumpulkan: " + jumlah);
    break;

```

- Modifikasi pada class StackTugasMahasiswa
Penambahan method jumlahTugas untuk menghitung jumlah tugas yang ada berdasarkan stack

```

public int jumlahTugas() {
    return top + 1;
}

```

Hasil modifikasi kode program

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
6. Melihat Jumlah Tugas

Pilih: 4

Daftar semua tugas

Nama	NIM	Kelas
Erik	1002	1B
Dila	1001	1A

Menu:

1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
6. Melihat Jumlah Tugas

Pilih: 6

Jumlah tugas yang dikumpulkan: 2

2.2 Percobaan 2: Konversi Nilai Tugas ke Biner

2.2.1 Langkah-langkah Percobaan

1. Menambahkan method `konversiDesimalKeBiner` pada class `StackTugasMahasiswa` dengan menerima parameter kode bertipe `int`

```
public String konversiDesimalKeBiner(int nilai) {
    StackKonversi18 stack = new
StackKonversi18();
    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;
}
```

2. Membuat file `StackKonversi` dan menambahkan empat method yaitu `isEmpty`, `isFull`, `push`, dan `pull` sebagai operasi utama Stack pada class `StackKonversi`

```
public class StackKonversi18 {
    int[] tumpukBiner;
    int size;
    int top;

    public StackKonversi18(){
        this.size = 32;
        tumpukBiner = 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()) {
            System.out.println("Stack penuh");
        } else {
            top++;
            tumpukBiner[top] = data;
        }
    }

    public int pop (){
        if (isEmpty()) {
            System.out.println("Stack kosong");
            return -1;
        } else {
```

```

    int data = tumpukBiner[top];
        top--;
        return data;
    }
}

```

3. Menambahkan baris kode program pada method pop di class MahasiswaDemo, agar nilai tugas mahasiswa dikonversi ke dalam bentuk biner setelah dilakukan penilaian

```

case 2:
    Mahasiswa18 dinilai = stack.pop();
    if (dinilai != null) {
        System.out.println("Menilai
tugas dari " + dinilai.nama);
        System.out.print("Masukkan
nilai (0-100): ");
        int nilai = scan.nextInt();
        dinilai.tugasDinilai(nilai);
        System.out.printf("Nilai Tugas
%s adalah %d\n", dinilai.nama, nilai);
        // percobaan 2
        String biner =
stack.konversiDesimalKeBiner(nilai);
        System.out.println("Nilai
Biner Tugas: " + biner);
    }
    break;

```

2.2.2 Verifikasi Hasil Percobaan

```

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Tugas Terbawah
6. Melihat Jumlah Tugas
Pilih: 2
Menilai tugas dari Tika
Masukkan nilai (0-100): 87
Nilai Tugas Tika adalah 87
Nilai Biner Tugas: 1010111

```

2.2.3 Pertanyaan

1. Jelaskan alur kerja dari method konversiDesimalKeBiner!
Jawab : Method konversiDesimalKeBiner bertujuan untuk mengubah bilangan desimal menjadi bilangan biner menggunakan

```
mid = (left + right) / 2;
```

stack, method tersebut akan melakukan pembagian dengan membagi nilai dengan 2, jika sisa hasil pembagian adalah 0 atau 1, sisa tersebut disimpan di stack, proses ini diulang ulang hingga nilainya menjadi 0. Setelah pembagian selesai, program ini akan mengambil isi stack secara berurutan dari yang paling atas sampai bawah kemudian disusun menjadi hasil konversi berupa bilangan biner

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

Jawab :

Jika kondisi perulangan diubah menjadi !=0 maka kode akan tetap berjalan dan hasilnya tidak berubah karena program akan tetap berjalan namun perbedaan pada kode ini, program akan tetap berjalan jika kondisi bilangan positif saja, jika kondisi bilangan negative maka Program akan berhenti

2.3 Latihan Praktikum

- Kode program class StackSurat

```
public class StackSurat18 {  
  
    Surat18[] stack;  
    int top;  
    int size;  
  
    public StackSurat18(int size) {  
        this.size = size;  
        stack = new Surat18[size];  
        top = -1;  
    }  
  
    public boolean isFull() {  
        if (top == size - 1) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public boolean isEmpty() {  
        if (top == -1) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
  
    public void push(Surat18 srt) {
```

```

        if (!isFull()) {
            top++;
            stack[top] = srt;
        } else {
            System.out.println("Stack penuh! Tidak dapat
menambahkan surat lagi.");
        }
    }

    public Surat18 pop() {
        if (!isEmpty()) {
            Surat18 s = stack[top];
            top--;
            return s;
        } else {
            System.out.println("Stack kosong! Tidak ada surat
untuk diproses");
            return null;
        }
    }

    public Surat18 peek() {
        if (!isEmpty()) {
            return stack[top];
        } else {
            System.out.println("Stack kosong! Tidak ada surat
yang masuk");
            return null;
        }
    }

    public Surat18 cariSurat(String namaMahasiswa) {
        for (int i = 0; i <= top; i++) {
            if
(stack[i].namaMahasiswa.equalsIgnoreCase(namaMahasiswa)) {
                return stack[i];
            }
        }
        return null;
    }
}

```

- Kode program class Surat18

```
public class Surat18 {  
  
    String idSurat;  
    String namaMahasiswa;  
    String kelas;  
    char jenisIzin;  
    int durasi;  
  
    public Surat18() {  
  
    }  
  
    public Surat18(String idSurat, String namaMahasiswa, String  
kelas, char jenisIzin, int durasi) {  
        this.idSurat = idSurat;  
        this.namaMahasiswa = namaMahasiswa;  
        this.kelas = kelas;  
        this.jenisIzin = jenisIzin;  
        this.durasi = durasi;  
  
    }  
  
}
```

- Kode program class SuratDemo

```
import java.util.Scanner;  
  
public class SuratDemo18 {  
  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        StackSurat18 surat = new StackSurat18(5);  
        int pilih;  
  
        do {  
            System.out.println("\nMenu:");  
            System.out.println("1. Terima Surat Izin");  
            System.out.println("2. Proses Surat Izin");  
            System.out.println("3. Lihat Surat Izin Terakhir");  
            System.out.println("4. Cari Surat");  
            System.out.print("Pilih: ");  
            pilih = input.nextInt();  
            input.nextLine();  
  
            switch (pilih) {  
                case 1:  
                    System.out.print("ID Surat: ");  
                    String id = input.nextLine();  
                    System.out.print("Nama: ");  
                    String nama = input.nextLine();  
                    System.out.print("Kelas: ");  
                    String kelas = input.nextLine();  
                    System.out.print("Jenis Izin (S/I): ");
```



```

char jenis = input.nextLine().toUpperCase().charAt(0);
        System.out.print("Durasi Izin (hari): ");
        int durasi = input.nextInt();
        input.nextLine();
        Surat18 srt = new Surat18(id, nama, kelas, jenis,
durasi);

        surat.push(srt);
        System.out.printf("Surat %s berhasil diterima\n",
srt.namaMahasiswa);
        break;
    case 2:
        Surat18 diproses = surat.pop();
        if (diproses != null) {
            System.out.println("Memproses surat dari " +
diproses.namaMahasiswa);
        }
        break;
    case 3:
        Surat18 lihat = surat.peek();
        if (lihat != null) {
            System.out.println("Surat terakhir dikirimkan
oleh " + lihat.namaMahasiswa);
        }
        break;
    case 4:
        System.out.print("Masukkan nama mahasiswa: ");
        String cari = input.nextLine();
        Surat18 ditemukan = surat.cariSurat(cari);
        if (ditemukan != null) {
            System.out.println("Surat ditemukan:");
            System.out.println("ID Surat: " +
ditemukan.idSurat);
            System.out.println("Nama Mahasiswa: " +
ditemukan.namaMahasiswa);
            System.out.println("Kelas: " +
ditemukan.kelas);
            System.out.println("Jenis Izin: " +
(ditemukan.jenisIzin == 'S' ? "Sakit" : "Izin"));
            System.out.println("Durasi Izin: " +
ditemukan.durasi + " hari");
        } else {
            System.out.println("Tidak ada surat dari " +
cari );
        }
        break;

    default:
        System.out.println("Pilihan menu tidak
valid ");
        break;
    }
} while (pilih >=1 && pilih <=4);
}

```

Penjelasan kode program:

Program tersebut merupakan program untuk sistem surat, program tersebut memiliki 4 menu yang berfungsi untuk memasukkan surat, mencari surat, memproses surat, dan melihat surat terakhir. Program ini menggunakan konsep stack

Hasil running kode program

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 1

ID Surat: SRT01

Nama: Aditiasmara

Kelas: 1A

Jenis Izin (S/I): S

Durasi Izin (hari): 2

Surat Aditiasmara berhasil diterima

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 1

ID Surat: SRT02

Nama: Fara

Kelas: 1B

Jenis Izin (S/I): I

Durasi Izin (hari): 2

Surat Fara berhasil diterima

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 2

Memproses surat dari Fara

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 3

Surat terakhir dikirimkan oleh Aditiasmara

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 4

Masukkan nama mahasiswa: Aditiasmara

Surat ditemukan:

ID Surat: SRT01

Nama Mahasiswa: Aditiasmara

Kelas: 1A

Jenis Izin: Sakit

Durasi Izin: 2 hari

Menu:

1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat

Pilih: 4

Masukkan nama mahasiswa: Dimas

Tidak ada surat dari Dimas