# LAPORAN JOBSHEET 10

# KONSEP DASAR PEMROGRAMAN

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**


**Alfreda Dhaifullah Mahezwara**
**244107020219**
**Kelas : 1A**
**Absen : 04**

**PROGRAM STUDI TEKNIK INFORMATIKA JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG TAHUN 2025**

# Percobaan 1

## MAIN

```
package Pertemuan_11;

import java.util.Scanner;
public class QueueMain {
    public static void menu () {
        System.out.println("Menu Queue");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Print ");
        System.out.println("4. Peek");
        System.out.println("5. Clear ");
    }

    public static void main (String[]args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        Queue Q = new Queue(n);
        int pilih;
        do {
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    System.out.println("Masukan data baru: ");
                    int dataMasuk = sc.nextInt();
                    Q.enQueue(dataMasuk);
                    break;
                case 2:
                    int dataKeluar = Q.deQueue();
                    if (dataKeluar != 0) {
                        System.out.println("Data yang dikeluarkan: " + dataKeluar);
                    }
                    break;
                case 3:
                    Q.print();
                    break;
```

```
                case 4:
                    Q.peek();
                    break;
                case 5:
                    Q.clear();
                    break;
                default:
                    break;
            }

        } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 || pilih == 5);
    }
}
```

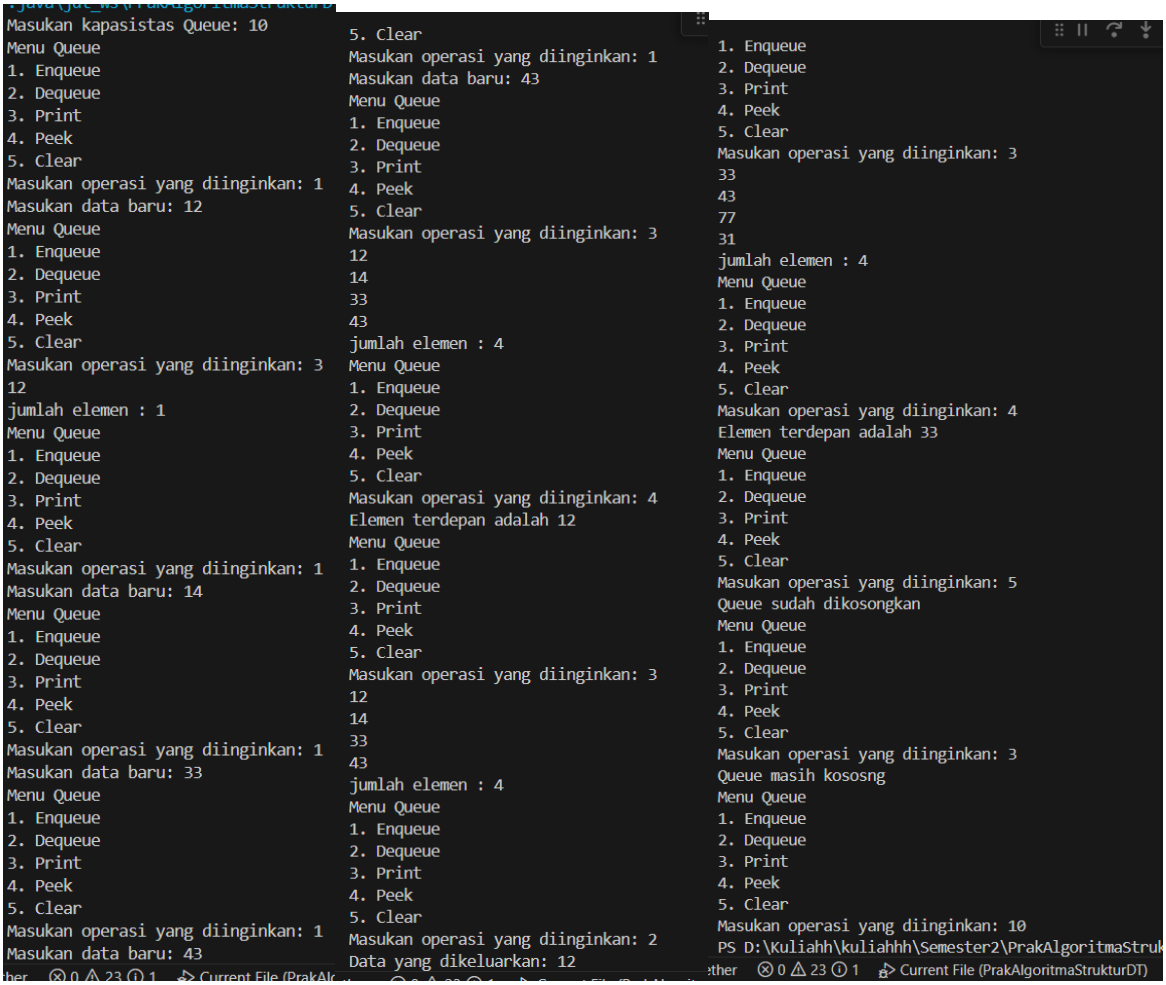
## QUEUE

```
package Pertemuan_11;

public class Queue {
    int [] data;
    int front;
    int rear;
    int maxSize;
    int size;

    // konstruktor Queue
    public Queue (int n) {
        maxSize = n;
        data = new int[maxSize];
        size = 0;
        front = rear = -1;

    }

    // method untuk mengecek apakah queue kosong
    public boolean IsEmpty () {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }
```

```
// method untuk mengecek apakah queue penuh
public boolean IsFull () {
    if (size == maxSize) {
        return true;
    } else {
        return false;
    }
}

// method untuk menmapilkan elemen dari paling depan
public void peek () {
    if (!IsEmpty()) {
        System.out.println("Elemen terdepan adalah " + data[front]);
    } else {
        System.out.println("Queue masih kosong");
    }
}

//method untuk menampilkan seluruh elemen Queue
public void print (){
    if (IsEmpty()) {
        System.out.println("Queue masih kososng");
    } else {
        int i = front;
        while (i != rear) {
            System.out.println(data[i] + " ");
            i = (i + 1) % maxSize;
        }
        System.out.println(data[i] + " ");
        System.out.println("jumlah elemen : " + size);
    }
}

// method untuk menghapus semua elemen pada Queue
public void clear () {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println("Queue sudah dikosongkan");
```

```
    } else {
        System.out.println("Queue masih kosong");
    }
}

// method enqueue untuk menambahkan isi queue dengan parameter
public void enQueue (int dt) {
    if (IsFull()) {
        System.out.println("Queue sudah penuh");
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == maxSize - 1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}

// method deuqueue untuk mengeluarkan data dari queue dari posisi belakang
public int deQueue () {
    int dt = 0;
    if (IsEmpty()) {
        System.out.println("Queue masih kosong");
    } else {
        dt = data[ front];
        size--;
        if (IsEmpty()) {
            front = rear = -1;
        } else {
            if (front == maxSize - 1) {
                front = 0;
            } else {
                front++;
            }
```

```
            }
        }
        return dt;
    }
}
```




1. Pada konstruktor, mengapa nilai awala atribut front dan rear bernilai -1, semmentara atribut size bernilai 0?
   - Nilai awal atribut front dan rear diatur ke -1 dalam konstruktor karena ini menandakan bahwa queue masih kosong. Sedangkan atribut size diatur ke 0 untuk menunjukkan bahwa tidak ada elemen dalam queue.
2. Pada method enqueue, jelaskan maksud dan kegunaan dari potongan berikut!

```
if (rear == max - 1) {
    rear = 0;
```

   - Kode ini mengecek apakah posisi rear (penunjuk elemen terakhir dalam antrian) sudah mencapai indeks terakhir array (max - 1, di mana max adalah ukuran maksimum array).
     Jika kondisi rear == max - 1 terpenuhi, artinya antrian sudah penuh hingga ujung array, dan nilai rear di-set kembali ke 0 untuk memanfaatkan ruang kosong di awal array (jika ada).

3. Pada method Dequeue, jelaskan maksud dan kegunaan dari potongan kode berikut!

```
if (front == max - 1) {
    front = 0;
```

   - Kode ini mengecek apakah posisi `rear` (penunjuk elemen terakhir dalam antrian) sudah mencapai indeks terakhir array (`max - 1`, di mana `max` adalah ukuran maksimum array). Jika kondisi `rear == max - 1` terpenuhi, artinya antrian sudah penuh hingga ujung array, dan nilai `rear` di-set kembali ke `0` untuk memanfaatkan ruang kosong di awal array (jika ada).

4. Pada method print, jelaskan jelaskan maksud dari potongan kode berikut!

   - Digunakan untuk menelusuri elemen antrian dari `front` ke `rear` tanpa keluar dari batas array (`0` hingga `max - 1`).Contoh: Jika `max = 5` dan `i = 4`, maka `(4 + 1) % 5 = 0` → `i` berpindah ke awal array.

5. Tunjukkan potongan kode program yang merupakan queue overflow!

   - 
```
// method enqueue untuk menambahkan isi queue dengan
public void enQueue (int dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == maxSize - 1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}
```

6. Pada saat terjadi queue overflow dan queue underflow, program tersebut tetap dapat berjalan dan hanya menampilkan teks informasi. Lakukan modifikasi program sehingga pada saat terjadi queue overflow dan queue underflow, program dihentikan!

   - 
```
public int deQueue () {
    int dt = 0;
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
        System.exit(status:1); // Menghentikan program
```
   - 
```
public void enQueue (int dt) {
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
        System.exit(status:1); // Menghentikan program
```

# Percobaan 2

## Classs Mahasiswa

```
package Pertemuan_11;

public class Mahasiswa {
    String nim;
    String nama;
    String prodi;
    String kelas;

    Mahasiswa (String nim, String nama, String prodi, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }

    public void tampilkanData () {
        System.out.println(nim + " - " + nama + " - " + prodi + " - " + kelas);
    }

}
```

## Class AntrianLayanan

```
package Pertemuan_11;

public class AntrianLayanan {
    Mahasiswa [] data;
    int front;
    int rear;
    int maxSize;
    int size;

    // konstruktor Queue
    public AntrianLayanan (int max) {
        maxSize = max;
        this.data = new Mahasiswa [maxSize];
        this.size = 0;
        this.front = 0;
```

```
    this.rear = -1;

}

// method untuk mengecek apakah queue kosong
public boolean IsEmpty () {
    if (size == 0) {
        return true;
    } else {
        return false;
    }
}

// method untuk mengecek apakah queue penuh
public boolean IsFull () {
    if (size == maxSize) {
        return true;
    } else {
        return false;
    }
}

// method untuk menmapilkan elemen dari paling depan
// method untuk melihat barisan terdepan
public void peek () {
    if (IsEmpty()) {
        System.out.println("Antrian kosong.");
    } else {
        System.out.println("Mahasiswa terdepan : ");
        System.out.println("NIM - NAMA - PRODI - KELAS");
        data[front].tampilkanData();
    }
}

//method untuk menampilkan seluruh elemen Queue
//method untuk menampilkan seluruh antrian
public void print (){
    if (IsEmpty()) {
        System.out.println("Queue masih kososng");
        return;
```

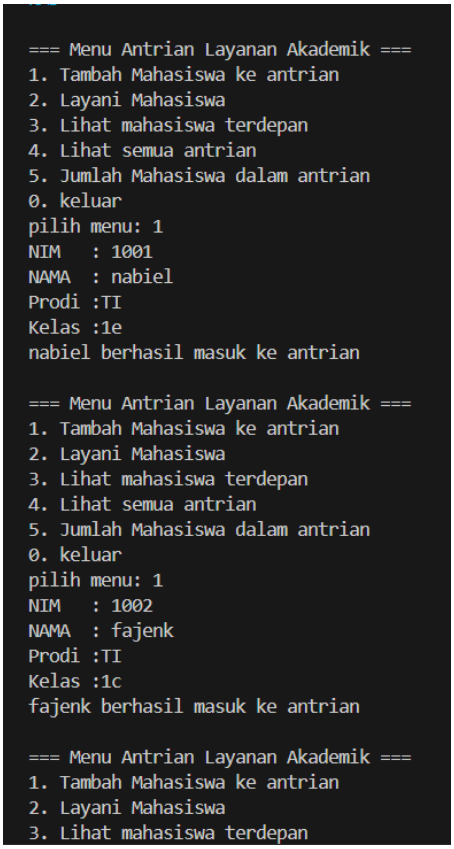
```
    }

    System.out.println("Daftar Mahasiswa dalam Antrian: ");
    System.out.println("NIM - NAMA - PRODI - KELAS");
    for (int i = 0; i < size; i++) {
        int index = (front + i) % maxSize;
        System.out.print((i + 1) + ". ");
        data[index].tampilkanData();
    }
}

// method untuk menghapus semua elemen pada Queue
public void clear () {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println("Queue sudah dikosongkan");
    } else {
        System.out.println("Queue masih kosong");
    }
}

// method enqueue untuk menambahkan isi queue dengan parameter
// method tambah antrian
public void enQueue (Mahasiswa mhs) {
    if (IsFull()) {
        System.out.println("Antrian sudah penuh, tidak dapat menambah mahasiswa");
        return;
    }
    rear = (rear + 1) % maxSize;
    data[rear] = mhs;
    size++;
    System.out.println(mhs.nama +" berhasil masuk ke antrian");
}

// method deuqueue untuk mengeluarkan data dari queue dari posisi belakang
// method melayani mahasiswa
public Mahasiswa deQueue () {
    if (IsEmpty()) {
        System.out.println("Antrian masih kosong");
```

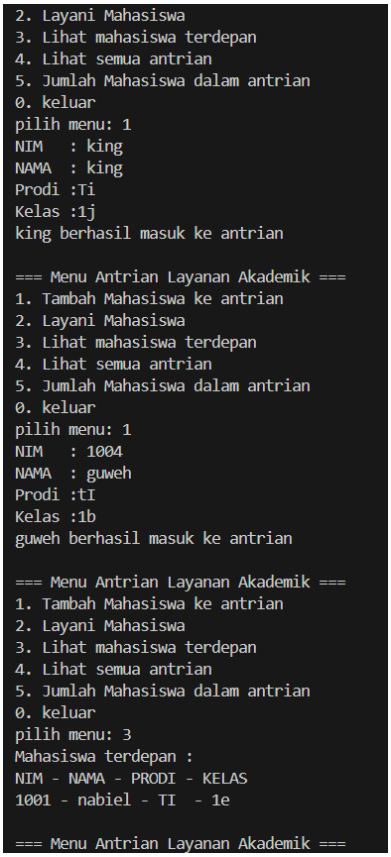
```
            return null;
        }
        Mahasiswa mhs = data[front];
        front = (front + 1) % maxSize;
        size--;
        return mhs;
    }

    //method menampilkan jumlah antrian
    public int jumlahantrian () {
        return size;
    }
}
```

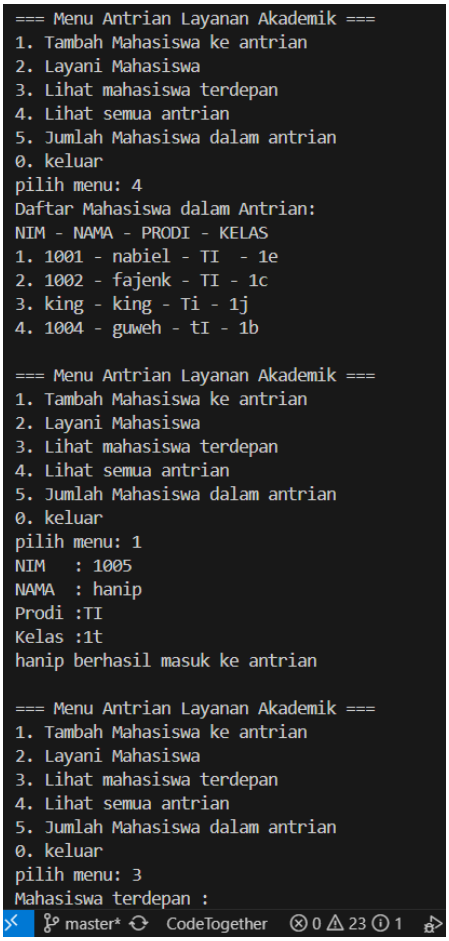
## Class LayananAkademik

```
package Pertemuan_11;
import java.util.Scanner;

public class LayananAkademikSIAKAD {
    public static void main (String[]args) {
        Scanner sc = new Scanner (System.in);
        AntrianLayanan antrian = new AntrianLayanan(5);
        int pilihan = 0;

        do {
            System.out.println("\n=== Menu Antrian Layanan Akademik ===");
            System.out.println("1. Tambah Mahasiswa ke antrian");
            System.out.println("2. Layani Mahasiswa");
            System.out.println("3. Lihat mahasiswa terdepan");
            System.out.println("4. Lihat semua antrian");
            System.out.println("5. Jumlah Mahasiswa dalam antrian");
            System.out.println("0. keluar");
            System.out.print("pilih menu: ");
            pilihan = sc.nextInt(); sc.nextLine();

            switch (pilihan) {
                case 1:
                    System.out.print("NIM   : ");
                    String nim = sc.nextLine();
                    System.out.print("NAMA  : ");
                    String nama = sc.nextLine();
```

```
                System.out.print("Prodi :");
                String prodi = sc.nextLine();
                System.out.print("Kelas :");
                String kelas = sc.nextLine();
                Mahasiswa mhs = new Mahasiswa(nim, nama, prodi, kelas);
                antrian.enQueue(mhs);
                break;
            case 2:
                Mahasiswa dilayani = antrian.deQueue();
                if (dilayani != null) {
                    System.out.println("Melayani mahasiswa: ");
                    dilayani.tampilkanData();
                }
                break;
            case 3:
                antrian.peek();
                break;
            case 4:
                antrian.print();
                break;
            case 5:
                System.out.println("jumlah dalam antrian: " + antrian.jumlahantrian());
                break;
            case 0:
                System.out.println("Terima Kasih.");
                break;
            default:
                System.out.println("Pilihan tidak valid");
                break;
        }
    } while (pilihan != 0);
    sc.close();
  }
}
```




# Pertanyaan

Lakukan modifikasi program dengan menambahkan method baru bernama LihatAkhir pada class AntrianLayanan yang digunakan untuk mengecek antrian yang berada di posisi belakang. Tambahkan pula daftar menu 6. Cek Antrian paling belakang pada class LayananAkademikSIAKAD sehingga method LihatAkhir dapat dipanggil!

```
    // method untuk menmapilkan elemen terakhir
    // method untuk melihat barisan terakhir
    public void last () {
        if (IsEmpty()) {
            System.out.println(x:"Antrian kosong.");
        } else {
            System.out.println(x:"Mahasiswa terdepan : ");
            System.out.println(x:"NIM - NAMA - PRODI - KELAS");
            data[rear].tampilkanData();
        }
    }
}
```

# TUGAS

```
package Pertemuan_11;

import java.util.Scanner;


class Mahasiswa {

    private String nim;

    private String nama;

    private String prodi;

    private int semester;


    public Mahasiswa(String nim, String nama, String prodi, int semester) {

        this.nim = nim;

        this.nama = nama;

        this.prodi = prodi;

        this.semester = semester;

    }


    public String getNim() {

        return nim;

    }


    public String getNama() {

        return nama;

    }


    public String getProdi() {

        return prodi;

    }


    public int getSemester() {

        return semester;

    }


    @Override

    public String toString() {

        return "NIM: " + nim + ", Nama: " + nama + ", Prodi: " + prodi + ", Semester: " +
semester;

    }

}
```

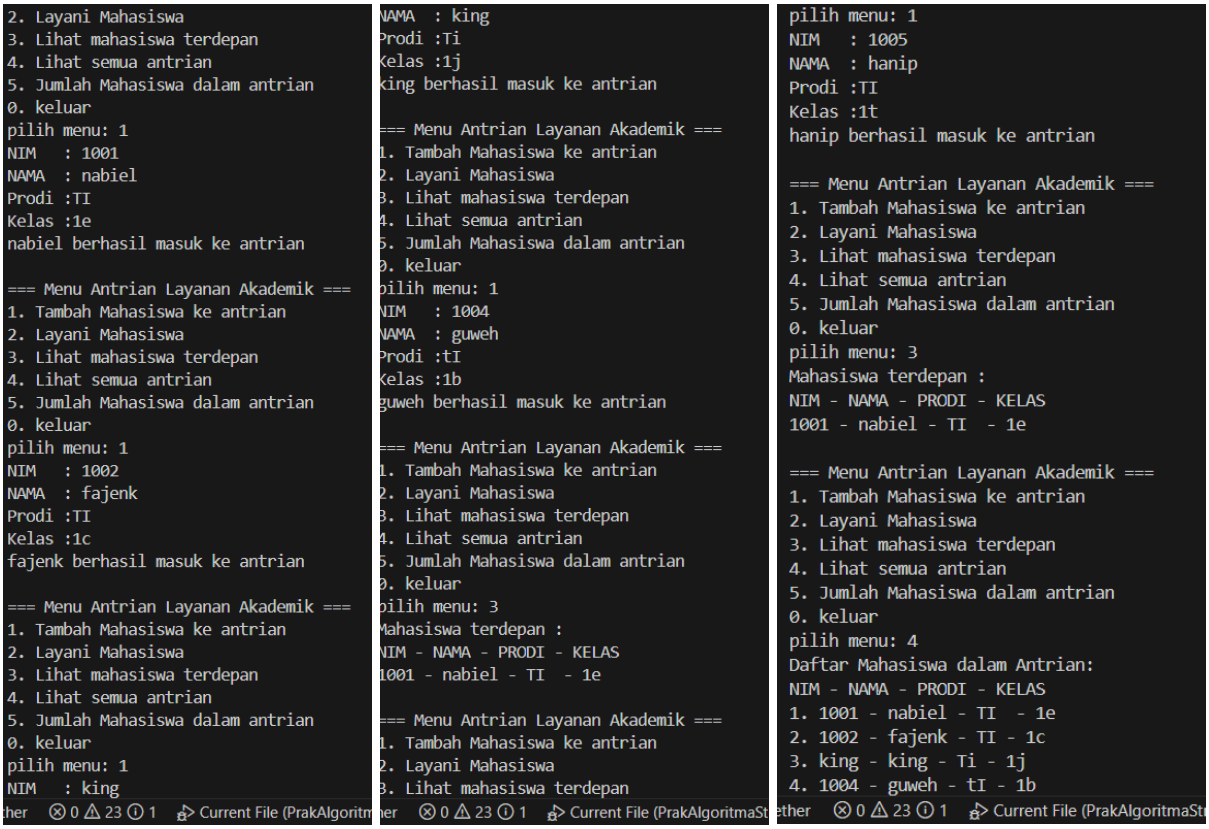
```
class AntrianKRS {
    private final int maxSize = 10;
    private Mahasiswa[] queue;
    private int front;
    private int rear;
    private int count;
    private int processed;

    public AntrianKRS() {
        queue = new Mahasiswa[maxSize];
        front = 0;
        rear = -1;
        count = 0;
        processed = 0;
    }

    public boolean isEmpty() {
        return count == 0;
    }

    public boolean isFull() {
        return count == maxSize;
    }

    public void clear() {
        front = 0;
        rear = -1;
        count = 0;
        System.out.println("Antrian telah dikosongkan");
    }

    public void enqueue(Mahasiswa mhs) {
        if (isFull()) {
            System.out.println("Antrian penuh! Tidak bisa menambahkan mahasiswa.");
            return;
        }
        rear = (rear + 1) % maxSize;
        queue[rear] = mhs;
        count++;
        System.out.println("Mahasiswa " + mhs.getNama() + " berhasil ditambahkan ke antrian");
```

```
}

public Mahasiswa dequeue() {
    if (isEmpty()) {
        System.out.println("Antrian kosong! Tidak ada mahasiswa untuk diproses.");
        return null;
    }
    Mahasiswa temp = queue[front];
    front = (front + 1) % maxSize;
    count--;
    processed++;
    return temp;
}

public void displayAll() {
    if (isEmpty()) {
        System.out.println("Antrian kosong");
        return;
    }
    System.out.println("Daftar seluruh mahasiswa dalam antrian:");
    int i = front;
    for (int c = 0; c < count; c++) {
        System.out.println((c + 1) + ". " + queue[i]);
        i = (i + 1) % maxSize;
    }
}

public void displayFront2() {
    if (count < 1) {
        System.out.println("Tidak ada cukup mahasiswa dalam antrian");
        return;
    }
    System.out.println("2 mahasiswa terdepan dalam antrian:");
    System.out.println("1. " + queue[front]);
    if (count >= 2) {
        System.out.println("2. " + queue[(front + 1) % maxSize]);
    }
}

public void displayLast() {
```

```
        if (isEmpty()) {
            System.out.println("Antrian kosong");
            return;
        }
        System.out.println("Mahasiswa terakhir dalam antrian:");
        System.out.println(queue[rear]);
    }

    public int getQueueCount() {
        return count;
    }

    public int getProcessedCount() {
        return processed;
    }

    public int getRemaining() {
        return Math.max(0, 30 - processed);
    }
}

 class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        AntrianKRS antrian = new AntrianKRS();
        int choice;

        do {
            System.out.println("\n=== Sistem Antrian Persetujuan KRS ===");
            System.out.println("1. Tambah Mahasiswa ke Antrian");
            System.out.println("2. Proses KRS (2 mahasiswa)");
            System.out.println("3. Tampilkan Semua Antrian");
            System.out.println("4. Tampilkan 2 Antrian Terdepan");
            System.out.println("5. Tampilkan Antrian Terakhir");
            System.out.println("6. Cek Jumlah Antrian");
            System.out.println("7. Cek Jumlah yang Sudah Diproses");
            System.out.println("8. Cek Jumlah yang Belum Diproses");
            System.out.println("9. Kosongkan Antrian");
            System.out.println("0. Keluar");
            System.out.print("Pilihan Anda: ");
```

```
choice = scanner.nextInt();
scanner.nextLine(); // consume newline

switch (choice) {
    case 1:
        System.out.print("Masukkan NIM: ");
        String nim = scanner.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = scanner.nextLine();
        System.out.print("Masukkan Prodi: ");
        String prodi = scanner.nextLine();
        System.out.print("Masukkan Semester: ");
        int semester = scanner.nextInt();
        scanner.nextLine(); // consume newline

        Mahasiswa mhs = new Mahasiswa(nim, nama, prodi, semester);
        antrian.enqueue(mhs);
        break;

    case 2:
        System.out.println("Memproses KRS untuk 2 mahasiswa terdepan:");
        Mahasiswa mhs1 = antrian.dequeue();
        Mahasiswa mhs2 = antrian.dequeue();

        if (mhs1 != null) {
            System.out.println("1. " + mhs1 + " - KRS telah disetujui");
        }
        if (mhs2 != null) {
            System.out.println("2. " + mhs2 + " - KRS telah disetujui");
        }
        break;

    case 3:
        antrian.displayAll();
        break;

    case 4:
        antrian.displayFront2();
        break;
```

```
                case 5:
                    antrian.displayLast();
                    break;

                case 6:
                    System.out.println("Jumlah mahasiswa dalam antrian: " +
antrian.getQueueCount());
                    break;

                case 7:
                    System.out.println("Jumlah mahasiswa yang sudah diproses: " +
antrian.getProcessedCount());
                    break;

                case 8:
                    System.out.println("Jumlah mahasiswa yang belum diproses: " +
antrian.getRemaining());
                    break;

                case 9:
                    antrian.clear();
                    break;

                case 0:
                    System.out.println("Terima kasih telah menggunakan sistem ini");
                    break;

                default:
                    System.out.println("Pilihan tidak valid!");
            }
        } while (choice != 0);

        scanner.close();
    }
}
```




https://github.com/AlfredaDhaifullah04/Semester-2/tree/master/Pertemuan_11