

**LAPORAN HASIL PRAKTIKUM**  
**ALGORITMA DAN STRUKTUR DATA**  
**JOBSCHEET 10**



**NAMA :MAULIDYAAFRIANI**

**NIM: 2441070200559**

**KELAS : 1E**

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

**2025**

## 1.1 PERCOBAAN 1

1. Buat folder baru bernama **P1Jobsheet10** di dalam repository **Praktikum ASD**, kemudian buat class baru dengan nama **Queue**.

 Queue15.java

2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti berikut ini :

```
public class Queue15 {  
    int[] data;  
    int front;  
    int rear;  
    int size;  
    int max;  
    public Queue15(int n) {  
        max = n;  
        data = new int[max];  
        size = 0;  
        front = rear = -1;  
    }
```

3. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah queue kosong.

```
public boolean IsEmpty() {  
    if (size == 0) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

4. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.

```
public boolean IsFull () {  
    if (size == max) {  
        return true;  
    }else {  
        return false;  
    }  
}
```

5. Buat method **peek** bertipe void untuk menampilkan elemen queue pada posisi paling depan.

```
public void peek() {  
    if (!IsEmpty ()) {  
        System.out.println("Elemen terdepan: " + data[front]);  
    } else {  
        System.out.println("Queue masih kosong: ");  
    }  
}
```

6. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.

```
} else {
    int i = front;
    while (i != rear) {
        System.out.print(data[i] + " ");
        i = (i + 1) % max;
    }
    System.out.println(data[i] + " ");
    System.out.println("Jumlah elemen = " + size);
}
```

7. Buat method **clear** bertipe void untuk menghapus semua elemen pada queue

```
public void clear () {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println("Queue berhasil dikosongkan");
    } else {
        System.out.println("Queue masih kosong");
    }
}
```

8. Buat method **Enqueue** bertipe void untuk menambahkan isi queue dengan parameter **dt** yang bertipe integer

```
public void Enqueue(int dt) {
    if (IsFull()) {
        System.out.println("Queue sudah penuh");
    } else{
        if (IsEmpty()) {
            front = rear = 0;
        } else{
            if (rear == max -1) {
                rear = 0;
            }else {
                rear++;
            }
            data[rear] = dt;
            size++;
        }
    }
}
```

9. Buat method **Dequeue** bertipe int untuk mengeluarkan data pada queue di 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 == max - 1) {
        front = 0;
    } else {
        front++;
    }
}
return dt;
}
}

```

10. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum1**. Buat method **menu** bertipe void untuk memilih menu program pada saat dijalankan.

 QueueMain1... 1

```

import java.util.Scanner;

public class QueueMain15 {
    public static void menu() {
        System.out.println("Masukkan operasi yang diinginkan: ");
        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");
        System.out.println("-----");
    }
}

```

11. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.

```

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
}

```

12. Buat variabel **n** untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.

```
System.out.print("Masukkan kapasitas queue: ");
int n = sc.nextInt();
```

13. Lakukan instansiasi objek Queue dengan nama **Q** dengan mengirimkan parameter **n** sebagai kapasitas elemen queue

```
Queue15 Q = new Queue15(n);
```

14. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.

```
int pilih;
```

15. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case untuk menjalankan operasi queue sesuai dengan masukan pengguna.

```
do {
    menu();
    pilih = sc.nextInt();
    switch (pilih) {
        case 1:
            System.out.print("Masukkan 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;
    }
} while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 || pilih == 5);
}
}

```

16. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

```

1
Masukkan data baru: 44
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
-----
1
Masukkan data baru: 13
Queue sudah penuh
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
-----
4
Elemen terdepan: 44
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
-----
```

#### Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?

Jawab: Nilai front dan rear diatur ke -1 karena menandakan bahwa **queue masih kosong** — belum ada elemen masuk. Sementara size bernilai 0 karena **belum ada data yang ditambahkan** ke dalam queue. Ini jadi tanda awal kondisi queue sebelum dipakai.

2. Pada method **Enqueue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

```

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

Jawab: Kode ini digunakan karena queue-nya **bersifat melingkar (circular queue)**. Kalau rear sudah ada di posisi terakhir array (`max - 1`), maka di-set kembali ke 0 (awal array), supaya bisa lanjut mengisi dari depan jika ada ruang kosong. Kalau belum sampai akhir, cukup tambahkan 1 (`rear++`).

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

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

Jawab: Kode ini juga bagian dari konsep **queue melingkar**. Saat menghapus data, kita harus geser posisi front. Kalau front sudah di posisi akhir array, kita balikin ke 0. Kalau belum, cukup tambah 1. Ini menjaga agar queue tetap bisa dipakai berputar.

4. Pada method **print**, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (**int i=0**), melainkan **int i=front**?

Jawab: Karena **elemen pertama dalam queue belum tentu ada di indeks 0**. Posisi elemen paling depan bisa di mana saja tergantung proses enqueue dan dequeue sebelumnya. Jadi, kita mulai dari front supaya urutannya benar.

5. Perhatikan kembali method **print**, jelaskan maksud dari potongan kode berikut!

```
i = (i + 1) % max;
```

Jawab: Kode ini untuk **menampilkan semua data dalam queue dari depan sampai belakang**. Perulangan jalan dari front ke rear. Rumus  $(i + 1) \% \text{max}$  dipakai supaya saat indeks sampai akhir array, bisa lanjut ke depan lagi (karena circular).

6. Tunjukkan potongan kode program yang merupakan queue overflow!

Jawab:

```
if (IsFull()) {  
    System.out.println("Queue sudah penuh");  
}
```

7. 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!

Jawab: Untuk menghentikan program saat terjadi queue overflow atau underflow, cukup tambahkan perintah `System.exit(1);` setelah pesan error di dalam method Enqueue dan Dequeue agar program langsung berhenti ketika queue penuh atau kosong.

## 1.2 PERCOBAAN 2

1. Buat folder baru bernama **P2Jobsheet10** di dalam repository **Praktikum ASD**, kemudian buat class baru dengan nama **Mahasiswa**.

 **Mahasiswa15.java**

2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

```
public class Mahasiswa15 {  
    String nim;  
    String nama;  
    String prodi;  
    String kelas;  
  
    public Mahasiswa15(String nim, String nama, String prodi, String kelas) {  
        this.nim = nim;  
        this.nama = nama;  
        this.prodi = prodi;  
        this.kelas = kelas;  
    }  
}
```

Dan tambahkan method `tampilkanData` berikut :

```
public void tampilkanData() {  
    System.out.println("NIM : " + nim);  
    System.out.println("Nama : " + nama);  
    System.out.println("Prodi : " + prodi);  
    System.out.println("Kelas : " + kelas);  
    System.out.println("-----");  
}  
}
```

3. Salin kode program class **Queue** pada **Praktikum 1** untuk digunakan kembali pada **Praktikum 2** ini, ganti nama class-nya dengan **AntrianLayanan**. Karena pada **Praktikum 1**, data yang disimpan pada queue hanya berupa array bertipe integer, sedangkan pada **Praktikum 2** data yang digunakan adalah object, maka perlu dilakukan modifikasi pada class **AntrianLayanan** tersebut.

```
Mahasiswa[] data;  
int front;  
int rear;  
int size;  
int max;  
  
public AntrianLayanan(int max) {  
    this.max = max;  
    this.data = new Mahasiswa[max];  
    this.front = 0;  
    this.rear = -1;  
    this.size = 0;  
}
```

4. Lakukan modifikasi pada class **AntrianLayanan** dengan mengubah tipe **int[] data** menjadi **Mahasiswa[] data** karena pada kasus ini data yang akan disimpan berupa object Mahasiswa.

Modifikasi perlu dilakukan pada **atribut**, method **Enqueue**, dan method **Dequeue**. Berikutnya method peek dan print yaitu untuk menampilkan data antrian layanan paling depan dan menampilkan semua data antrian layanan. Ditambahkan dengan method getJumlahAntrian yaitu menampilkan nilai size

```
public class AntrianLayanan15 {  
    Mahasiswa15[] data;  
    int front;  
    int rear;  
    int size;  
    int max;  
  
    public AntrianLayanan15(int n) {  
        max = n;  
        data = new Mahasiswa15[max];  
        size = 0;  
        front = rear = -1;  
    }  
  
    public boolean isEmpty() {  
        return size == 0;  
    }  
  
    public boolean isFull() {  
        return size == max;  
    }  
  
    public void enqueue(Mahasiswa15 m) {  
        if (isFull()) {  
            System.out.println("Antrian penuh!");  
            return;  
        }  
        if (isEmpty()) {  
            front = rear = 0;  
        } else {  
            rear = (rear + 1) % max;  
        }  
        data[rear] = m;  
        size++;
```

```
}
```

```
public Mahasiswa15 dequeue() {
    if (isEmpty()) {
        System.out.println("Antrian kosong!");
        return null;
    }
    Mahasiswa15 temp = data[front];
    size--;
    if (isEmpty()) {
        front = rear = -1;
    } else {
        front = (front + 1) % max;
    }
    return temp;
}
```

```
public void peek() {
    if (!isEmpty()) {
        System.out.println("Antrian terdepan:");
        data[front].tampilkanData();
    } else {
        System.out.println("Antrian kosong!");
    }
}
```

```
public void print() {
    if (isEmpty()) {
        System.out.println("Antrian kosong!");
    } else {
        System.out.println("Isi Antrian:");
        int i = front;
        while (i != rear) {
            data[i].tampilkanData();
            i = (i + 1) % max;
        }
        data[i].tampilkanData();
        System.out.println("Jumlah antrian = " + size);
    }
}
```

```

    }

}

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

public int getJumlahAntrian() {
    return size;
}

public void lihatAkhir() {
    if (!isEmpty()) {
        System.out.println("Antrian paling belakang:");
        data[rear].tampilkanData();
    } else {
        System.out.println("Antrian kosong!");
    }
}
}

```

5. Selanjutnya, buat class baru dengan nama **LayananAkademikSIAKAD** tetap pada package yang sama. Buat fungsi **main**, deklarasikan Scanner dengan nama **sc**.
6. Kemudian lakukan instansiasi objek **AntrianLayanan** dengan nama **antrian** dan nilai parameternya adalah nilai maksimal antrian yang ditentukan (misal sama dengan 5).
7. Deklarasikan variabel dengan nama **pilihan** bertipe integer untuk menampung pilih menu dari pengguna.

```

import java.util.Scanner;

public class LayananAkademikSIAKAD15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianLayanan15 antrian = new AntrianLayanan15(5);
        int pilihan;
    }
}

```

8. Tambahkan kode berikut untuk melakukan perulangan menu sesuai dengan masukan yang diberikan oleh pengguna.

```
import java.util.Scanner;

public class LayananAkademikSIAKAD15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianLayanan15 antrian = new AntrianLayanan15(5);
        int pilihan;

        do {
            System.out.println("\n===== Menu Antrian Layanan Akademik =====");
            System.out.println("1. Tambah Antrian");
            System.out.println("2. Panggil Antrian");
            System.out.println("3. Lihat Antrian Terdepan");
            System.out.println("4. Lihat Semua Antrian");
            System.out.println("5. Kosongkan Antrian");
            System.out.println("6. Cek Antrian Paling Belakang");
            System.out.println("7. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = sc.nextInt();
            sc.nextLine(); // flush newline

            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();
                    Mahasiswa15 mhs = new Mahasiswa15(nim, nama, prodi, kelas);
                    antrian.enqueue(mhs);
                    break;
                case 2:
                    Mahasiswa15 keluar = antrian.dequeue();
                    if (keluar != null) {
                        System.out.println("Mahasiswa yang dipanggil:");
                        keluar.tampilkanData();
                    }
                    break;
                case 3:
                    antrian.peek();
                    break;
                case 4:
                    antrian.print();
                    break;
                case 5:
                    antrian.clear();
                    break;
                case 6:
                    antrian.lihatAkhir();
                    break;
                case 7:
                    System.out.println("Terima kasih telah menggunakan layanan ini.");
            }
        } while (pilihan != 7);
    }
}
```

```

        break;
    default:
        System.out.println("Pilihan tidak tersedia.");
    }
} while (pilihan != 7);

sc.close();
}
}
}

```

9. Compile dan jalankan class **LayananAkademikSIAKAD**, kemudian amati hasilnya.

```

===== Menu Antrian Layanan Akademik =====
===
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Semua Antrian
5. Kosongkan Antrian
6. Cek Antrian Paling Belakang
7. Keluar
Pilih menu: 1
NIM : lidya
Nama : 929
Prodi : TI
Kelas : 1e

===== Menu Antrian Layanan Akademik =====
===
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Semua Antrian
5. Kosongkan Antrian
6. Cek Antrian Paling Belakang
7. Keluar
Pilih menu: 1
NIM : 929
Nama : luqman
Prodi : TE
Kelas : 3d

```

```

===== Menu Antrian Layanan Akademik =====
===
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Semua Antrian
5. Kosongkan Antrian
6. Cek Antrian Paling Belakang
7. Keluar
Pilih menu: 4
Isi Antrian:
NIM : lidya
Nama : 929
Prodi : TI
Kelas : 1e
-----
NIM : 929
Nama : luqman
Prodi : TE
Kelas : 3d
-----
Jumlah antrian = 2

```

```

===== Menu Antrian Layanan Akademik =====
===
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Semua Antrian
5. Kosongkan Antrian
6. Cek Antrian Paling Belakang
7. Keluar
Pilih menu: 6
Antrian paling belakang:
NIM : 929
Nama : luqman
Prodi : TE
Kelas : 3d
-----
===== Menu Antrian Layanan Akademik =====
===
1. Tambah Antrian
2. Panggil Antrian
3. Lihat Antrian Terdepan
4. Lihat Semua Antrian
5. Kosongkan Antrian
6. Cek Antrian Paling Belakang
7. Keluar
Pilih menu: 7
Terima kasih telah menggunakan layanan ini.

```

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!

AntrianLayanan15.java

```
public class AntrianLayanan15 {  
    Mahasiswa15[] data;  
    int front;  
    int rear;  
    int size;  
    int max;  
  
    public AntrianLayanan15(int n) {  
        max = n;  
        data = new Mahasiswa15[max];  
        size = 0;  
        front = rear = -1;  
    }  
  
    public boolean isEmpty() {  
        return size == 0;  
    }  
  
    public boolean isFull() {  
        return size == max;  
    }  
  
    public void enqueue(Mahasiswa15 m) {  
        if (isFull()) {  
            System.out.println("Antrian penuh!");  
            return;  
        }  
        if (isEmpty()) {  
            front = rear = 0;  
        } else {  
            data[rear] = m;  
            rear++;  
        }  
        size++;  
    }  
}  
}
```

```
rear = (rear + 1) % max;
}

data[rear] = m;
size++;
}

public Mahasiswa15 dequeue() {
    if (isEmpty()) {
        System.out.println("Antrian kosong!");
        return null;
    }

    Mahasiswa15 temp = data[front];
    size--;
    if (isEmpty()) {
        front = rear = -1;
    } else {
        front = (front + 1) % max;
    }
    return temp;
}

public void peek() {
    if (!isEmpty()) {
        System.out.println("Antrian terdepan:");
        data[front].tampilkanData();
    } else {
        System.out.println("Antrian kosong!");
    }
}

public void print() {
    if (isEmpty()) {
        System.out.println("Antrian kosong!");
    } else {
        System.out.println("Isi Antrian:");
        int i = front;
        while (i != rear) {
```

```

        data[i].tampilkanData();
        i = (i + 1) % max;
    }
    data[i].tampilkanData(); // cetak rear
    System.out.println("Jumlah antrian = " + size);
}
}

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

public int getJumlahAntrian() {
    return size;
}

public void lihatAkhir() {
    if (!isEmpty()) {
        System.out.println("Antrian paling belakang:");
        data[rear].tampilkanData();
    } else {
        System.out.println("Antrian kosong!");
    }
}
}

```

### LayananAkademikSIAKAD15.java

```

import java.util.Scanner;

public class LayananAkademikSIAKAD15 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianLayanan15 antrian = new AntrianLayanan15(5);
        int pilihan;
    }
}

```

```
do {  
    System.out.println("\n===== Menu Antrian Layanan Akademik =====");  
    System.out.println("1. Tambah Antrian");  
    System.out.println("2. Panggil Antrian");  
    System.out.println("3. Lihat Antrian Terdepan");  
    System.out.println("4. Lihat Semua Antrian");  
    System.out.println("5. Kosongkan Antrian");  
    System.out.println("6. Cek Antrian Paling Belakang"); //  TAMBAHAN MENU  
    System.out.println("7. Keluar");  
    System.out.print("Pilih menu: ");  
    pilihan = sc.nextInt();  
    sc.nextLine(); // flush newline  
  
    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();  
            Mahasiswa15 mhs = new Mahasiswa15(nim, nama, prodi, kelas);  
            antrian.enqueue(mhs);  
            break;  
        case 2:  
            Mahasiswa15 keluar = antrian.dequeue();  
            if (keluar != null) {  
                System.out.println("Mahasiswa yang dipanggil:");  
                keluar.tampilkanData();  
            }  
            break;  
        case 3:  
            antrian.peek();  
            break;  
    }  
}
```

```

case 4:
    antrian.print();
    break;

case 5:
    antrian.clear();
    break;

case 6: //  MENU BARU UNTUK METHOD lihatAkhir()
    antrian.lihatAkhir();
    break;

case 7:
    System.out.println("Terima kasih telah menggunakan layanan ini.");
    break;

default:
    System.out.println("Pilihan tidak tersedia.");
}

} while (pilihan != 7);

sc.close();
}
}

```

### Mahasiswa15.java

```

public class Mahasiswa15 {

    String nim;
    String nama;
    String prodi;
    String kelas;

    public Mahasiswa15(String nim, String nama, String prodi, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }

    public void tampilanData() {

```

```

        System.out.println("NIM : " + nim);
        System.out.println("Nama : " + nama);
        System.out.println("Prodi : " + prodi);
        System.out.println("Kelas : " + kelas);
        System.out.println("-----");
    }
}

```

#### HASIL RUN:

<pre>===== Menu Antrian Layanan Akademik ====== 1. Tambah Antrian 2. Panggil Antrian 3. Lihat Antrian Terdepan 4. Lihat Semua Antrian 5. Kosongkan Antrian 6. Cek Antrian Paling Belakang 7. Keluar Pilih menu: 1 NIM : 992 Nama : lidya Prodi : TI Kelas : 1E lidya berhasil masuk ke antrian.</pre>	<pre>===== Menu Antrian Layanan Akademik ====== 1. Tambah Antrian 2. Panggil Antrian 3. Lihat Antrian Terdepan 4. Lihat Semua Antrian 5. Kosongkan Antrian 6. Cek Antrian Paling Belakang 7. Keluar Pilih menu: 6 Mahasiswa dalam antrian terakhir Antrian paling belakang: NIM : 9292 Nama : LUQMAN Prodi : TE Kelas : 1E -----</pre>
<pre>===== Menu Antrian Layanan Akademik ====== 1. Tambah Antrian 2. Panggil Antrian 3. Lihat Antrian Terdepan 4. Lihat Semua Antrian 5. Kosongkan Antrian 6. Cek Antrian Paling Belakang 7. Keluar Pilih menu: 1 NIM : 9292 Nama : LUQMAN Prodi : TE Kelas : 1E LUQMAN berhasil masuk ke antrian.</pre>	<pre>===== Menu Antrian Layanan Akademik ====== 1. Tambah Antrian 2. Panggil Antrian 3. Lihat Antrian Terdepan 4. Lihat Semua Antrian 5. Kosongkan Antrian 6. Cek Antrian Paling Belakang 7. Keluar Pilih menu: 7 Terima kasih telah menggunakan layanan ini.</pre>

#### Tugas

Waktu : 120 Menit

Buatlah program antrian untuk mengilustrasikan antrian persetujuan Kartu Rencana Studi (KRS) Mahasiswa oleh Dosen Pembina Akademik (DPA). Ketika seorang mahasiswa akan mengantri, maka dia harus mendaftarkan datanya (data mahasiswa seperti pada praktikum 2). Gunakan class untuk antrian seperti pada Praktikum 1 dan 2, dengan method-method yang berfungsi :

- Cek antrian kosong, Cek antrian penuh, Mengosongkan antrian.
- Menambahkan antrian, Memanggil antrian untuk proses KRS – setiap 1x panggilan terdiri dari 2 mahasiswa (pada antrian no 1 dan 2)

- Menampilkan semua antrian, Menampilkan 2 antrian terdepan, Menampilkan antrian paling akhir. • Cetak jumlah antrian, Cetak jumlah yang sudah melakukan proses KRS
- Jumlah antrian maximal 10, jumlah yang ditangani masing-masing DPA 30 mahasiswa, cetak jumlah mahasiswa yang belum melakukan proses KRS.

Gambarkan **Diagram Class** untuk antriannya. Implementasikan semua method menggunakan menu pilihan pada fungsi **main**.

Mahasiswa.java

```
public class Mahasiswa15 {
    String nim;
    String nama;
    String prodi;
    String kelas;

    public Mahasiswa15(String nim, String nama, String prodi, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }

    public void tampilanData() {
        System.out.println("NIM : " + nim);
        System.out.println("Nama : " + nama);
        System.out.println("Prodi : " + prodi);
        System.out.println("Kelas : " + kelas);
        System.out.println("-----");
    }
}
```

AntrianKRS.java

```
public class AntrianKRS15 {
    Mahasiswa15[] data;
    int front, rear, size, max, jumlahDiproses;

    public AntrianKRS15(int n) {
```

```
max = n;
data = new Mahasiswa15[max];
front = rear = -1;
size = 0;
jumlahDiproses = 0;
}

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

public boolean isFull() {
    return size == max;
}

public void enqueue(Mahasiswa15 m) {
    if (isFull()) {
        System.out.println("Antrian penuh!");
        return;
    }
    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear = (rear + 1) % max;
    }
    data[rear] = m;
    size++;
    System.out.println(m.nama + " berhasil masuk ke antrian.");
}

public void prosesKRS() {
    if (size < 2) {
        System.out.println("Antrian kurang dari 2, tidak dapat diproses!");
        return;
    }
    System.out.println("Mahasiswa yang diproses:");
}
```

```
for (int i = 0; i < 2; i++) {  
    Mahasiswa15 m = data[front];  
    m.tampilkanData();  
    size--;  
    jumlahDiproses++;  
    if (isEmpty()) {  
        front = rear = -1;  
    } else {  
        front = (front + 1) % max;  
    }  
}  
  
public void printAll() {  
    if (isEmpty()) {  
        System.out.println("Antrian kosong!");  
        return;  
    }  
    System.out.println("Daftar Mahasiswa dalam Antrian:");  
    int i = front;  
    while (i != rear) {  
        data[i].tampilkanData();  
        i = (i + 1) % max;  
    }  
    data[i].tampilkanData();  
}  
  
public void peek2() {  
    if (size < 2) {  
        System.out.println("Tidak cukup mahasiswa dalam antrian untuk ditampilkan (minimal 2).");  
        return;  
    }  
    System.out.println("2 Mahasiswa Terdepan:");  
    int i = front;  
    for (int j = 0; j < 2; j++) {  
        data[i].tampilkanData();  
        i = (i + 1) % max;  
    }  
}
```

```
    }

}

public void peekRear() {
    if (isEmpty()) {
        System.out.println("Antrian kosong!");
    } else {
        System.out.println("Mahasiswa paling belakang:");
        data[rear].tampilkanData();
    }
}

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

public int getJumlahAntrian() {
    return size;
}

public int getJumlahDiproses() {
    return jumlahDiproses;
}

public int getSisaMahasiswa() {
    return 30 - jumlahDiproses;
}
}
```

## KRSmain.java

```
import java.util.Scanner;

public class KRSMain15 {
    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
AntrianKRS15 antrian = new AntrianKRS15(10);
int pilih;

do {
    System.out.println("\n==== Menu Persetujuan KRS ====");
    System.out.println("1. Tambah Mahasiswa ke Antrian");
    System.out.println("2. Proses KRS (2 Mahasiswa sekaligus)");
    System.out.println("3. Lihat Semua Antrian");
    System.out.println("4. Lihat 2 Mahasiswa Terdepan");
    System.out.println("5. Lihat Mahasiswa Paling Belakang");
    System.out.println("6. Kosongkan Antrian");
    System.out.println("7. Cek Jumlah Antrian");
    System.out.println("8. Cek Jumlah yang Sudah Diproses");
    System.out.println("9. Cek Sisa Mahasiswa Belum Proses (max 30)");
    System.out.println("0. Keluar");
    System.out.print("Pilih menu: ");
    pilih = sc.nextInt();
    sc.nextLine();

    switch (pilih) {
        case 1:
            if (antrian.getJumlahDiproses() >= 30) {
                System.out.println("Kuota proses KRS oleh DPA sudah terpenuhi!");
                break;
            }
            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();
            Mahasiswa15 m = new Mahasiswa15(nim, nama, prodi, kelas);
            antrian.insert(m);
    }
}
```

```
antrian.enqueue(m);
break;
case 2:
antrian.prosesKRS();
break;
case 3:
antrian.printAll();
break;
case 4:
antrian.peek2();
break;
case 5:
antrian.peekRear();
break;
case 6:
antrian.clear();
break;
case 7:
System.out.println("Jumlah mahasiswa dalam antrian: " + antrian.getJumlahAntrian());
break;
case 8:
System.out.println("Jumlah mahasiswa yang sudah diproses: " + antrian.getJumlahDiproses());
break;
case 9:
System.out.println("Mahasiswa yang belum proses KRS: " + antrian.getSisaMahasiswa());
break;
case 0:
System.out.println("Terima kasih.");
break;
default:
System.out.println("Pilihan tidak valid.");
}
} while (pilih != 0);

sc.close();
```

```
}
```

hasil run:

```
3. Lihat Semua Antrian
4. Lihat 2 Mahasiswa Terdepan
5. Lihat Mahasiswa Paling Belakang
6. Kosongkan Antrian
7. Cek Jumlah Antrian
8. Cek Jumlah yang Sudah Diproses
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 1
NIM   : 029
Nama  : lidya
Prodi : ti
Kelas : 1e
lidya berhasil masuk ke antrian.

==== Menu Persetujuan KRS ====
1. Tambah Mahasiswa ke Antrian
2. Proses KRS (2 Mahasiswa sekaligus)
3. Lihat Semua Antrian
4. Lihat 2 Mahasiswa Terdepan
5. Lihat Mahasiswa Paling Belakang
6. Kosongkan Antrian
7. Cek Jumlah Antrian
8. Cek Jumlah yang Sudah Diproses
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 1
NIM   : 029
Nama  : lidya
Prodi : ti
Kelas : 1e
lidya berhasil masuk ke antrian.
```

```
==== Menu Persetujuan KRS ====
1. Tambah Mahasiswa ke Antrian
2. Proses KRS (2 Mahasiswa sekaligus)
3. Lihat Semua Antrian
4. Lihat 2 Mahasiswa Terdepan
5. Lihat Mahasiswa Paling Belakang
6. Kosongkan Antrian
7. Cek Jumlah Antrian
8. Cek Jumlah yang Sudah Diproses
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 2
Antrian kurang dari 2, tidak dapat diproses!

==== Menu Persetujuan KRS ====
1. Tambah Mahasiswa ke Antrian
2. Proses KRS (2 Mahasiswa sekaligus)
3. Lihat Semua Antrian
4. Lihat 2 Mahasiswa Terdepan
5. Lihat Mahasiswa Paling Belakang
6. Kosongkan Antrian
7. Cek Jumlah Antrian
8. Cek Jumlah yang Sudah Diproses
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 3
Antrian kosong!
```

```
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 1
NIM   : 929
Nama  : luqman
Prodi : te
Kelas : 3d
luqman berhasil masuk ke antrian.

==== Menu Persetujuan KRS ====
1. Tambah Mahasiswa ke Antrian
2. Proses KRS (2 Mahasiswa sekaligus)
3. Lihat Semua Antrian
4. Lihat 2 Mahasiswa Terdepan
5. Lihat Mahasiswa Paling Belakang
6. Kosongkan Antrian
7. Cek Jumlah Antrian
8. Cek Jumlah yang Sudah Diproses
9. Cek Sisa Mahasiswa Belum Proses (max 30)
0. Keluar

Pilih menu: 2
Mahasiswa yang diproses:
NIM   : 029
Nama  : lidya
Prodi : ti
Kelas : 1e
-----
NIM   : 929
Nama  : luqman
Prodi : te
Kelas : 3d
-----
```

