Nama : Siti Mutmainah

Kelas : TI-1B/21

NIM : 244107020143

Praktikum 1

Percobaan 1 : Operasi Dasar Queue

Pada percobaan ini, kita akan menerapkan operasi dasar pada algoritma Queue. Perhatikan Diagram Class Queue berikut ini:

Queue
data: int[] front: int rear: int size: int max: int
Queue(n: int) isFull(): boolean isEmpty(): boolean enqueue(dt: int): void dequeue(): int peek: void print(): void clear(): void

Berdasarkan diagram class tersebut, akan dibuat program class Queue dalam Java.

Langkah-langkah Percobaan

1. Buat folder baru bernama **P1Jobsheet10** di dalam repository **Praktikum ASD**, kemudian buat class baru dengan nama **Queue**.
2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti berikut ini :

```java
public class Queue21 {

    int[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue21(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.

```java
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.

```java
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.

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

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

```
public void print() {
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
    } 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(x:"Queue berhasil dikosongkan");
    } else {
        System.out.println(x:"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(x:"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(x:"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.

```
public class QueueMain {

    public static void menu() {
        System.out.println(x:"Masukkan operasi yang diinginkan:");
        System.out.println(x:"1. Enqueue");
        System.out.println(x:"2. Dequeue");
        System.out.println(x:"3. Print");
        System.out.println(x:"4. Peek");
        System.out.println(x:"5. Clear");
        System.out.println(x:"-----------------");
    }
```

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(s:"Masukkan kapasitas queue: ");
int n = sc.nextInt();
```

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

```
Queue21 Q = new Queue21(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(s:"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 <= 5);
```

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

Verifikasi Hasil Percobaan

```
PS C:\Semester 2\Praktikum ASD\Jobsheet\Jobsheet10>  & 'C:\Program Files\J
jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMe
es' '-cp' 'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage\e1d93f
d321c2139586dd55d91f19\redhat.java\jdt_ws\Jobsheet10_cab07d26\bin' 'QueueM

Masukkan kapasitas queue: 4
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
------------------
1
Masukkan data baru: 15
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
------------------
1
Masukkan data baru: 31
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
------------------
4
Elemen terdepan: 15
```

Pertanyaan

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

   Jawab: Bernilai -1 karena menandakan bahwa queue masih kosong dan belum ada elemen yang masuk. Sedangkan nilai 0 karena memang jumlah elemen masih kosong.

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

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

   Jawab: Rear (posisi belakang antrian, tempat untuk memasukkan elemen baru), Max (ukuran maksimum dari antrian array) Jika rear sudah mencapai posisi terakhir (max – 1) di array, artinya sudah sampai di ujung array. Kegunaan kode ini adalah jika rear sampai di ujung array, maka posisi rear akan melompat ke indeks 0 lagi (awal array).

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

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

   Jawab: Front (posisi depan antrian), Max (ukuran maksimum dari antrian array) Jika front

sudah mencapai posisi terakhir (max – 1) di array, artinya sudah sampai di ujung array. Kegunaan kode ini adalah jika front sampai di ujung array, maka dikembalikan posisinya ke indeks 0, supaya bisa lanjut mengambil data dari awal array lagi. Membuat antrian tidak penuh hanya karena elemen di awal sudah dihapus.

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

   Jawab: Dimulai dari i = front karena kita hanya ingin mencetak data yang masih ada di antrian, bukan dari awal array. Antrian bisa bergeser, jadi posisi terdepan tidak selalu di indeks 0.

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

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

   Jawab: Artinya tambah 1 ke i, lalu ambil sisa bagi modulus dengam max. Ini memastikan kalau i sudah di akhir array (max -1) akan kembali ke 0. Kode ini digunakan untuk mw=enghindari indeks luar batas dan memastikan antrian bisa berjalan dari akhir ke awal lagi.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

   Jawab:

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

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:

```
if (IsFull()) {
    System.out.println(x:"Queue sudah penuh");
    System.exit(status:0);
```

```
if (IsEmpty()) {
    System.out.println(x:"Queue masih kosong");
    System.exit(status:0);
```

Percobaan 2 : Antrian Layanan Akademik

Pada percobaan ini, kita akan membuat program yang mengilustrasikan Layanan pada Admin Akademik. Perhatikan Diagram Class berikut ini:

Mahasiswa
nim:String nama: String prodi: String kelas: String
Mahasiswa(nim: String, nama: String, prodi: String, kelas: String) void tampilkanData()

Langkah-langkah Percobaan

Berdasarkan diagram class tersebut, akan dibuat program class Mahasiswa dalam Java.

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

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

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

```
public Mahasiswa(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 + " - " + nama + " - " + prodi + " - " + kelas);
}
```

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.

```
public class AntrianLayanan {

    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**.

```
    public void tambahAntrian(Mahasiswa mhs) {
        if (IsFull()) {
            System.out.println(x:"Antrian penuh, tidak dapat menambah mahasiswa.");
            return;
        }
        rear = (rear + 1) % max;
        data[rear] = mhs;
        size++;
        System.out.println(mhs.nama + " berhasil masuk ke antrian.");
    }

    public Mahasiswa layaniMahasiswa() {
        if (IsEmpty()) {
            System.out.println(x:"Antrian kosong.");
            return null;
        }
        Mahasiswa mhs = data[front];
        front = (front + 1) % max;
        size--;
        return mhs;
    }
}
```

5. Berikutnya method peek dan print yaitu untuk menampilkan data antrian layanan paling depan dan menampilkan semua data antrian layanan.

```
public void lihatTerdepan() {
    if (IsEmpty()) {
        System.out.println(x:"Antrian kosong.");
    } else {
        System.out.print(s:"Mahasiswa terdepan: ");
        System.out.println(x:"NIM - NAMA - PRODI - KELAS");
        data[front].tampilkanData();
    }
}

public void tampilkanSemua() {
    if (IsEmpty()) {
        System.out.println(x:"Antrian kosong.");
        return;
    }
    System.out.println(x:"Daftar Mahasiswa dalam Antrian:");
    System.out.println(x:"NIM - NAMA - PRODI - KELAS");
    for (int i = 0; i < size; i++) {
        int index = (front + i) % max;
        System.out.print((i + 1) + ". ");
        data[index].tampilkanData();
    }
}
```

Ditambahkan dengan method getJumlahAntrian yaitu menampilkan nilai size

```
public int getJumlahAntrian() {
    return size;
}
```

6. Selanjutnya, buat class baru dengan nama **LayananAkademikSIAKAD** tetap pada package yang sama. Buat fungsi **main**, deklarasikan Scanner dengan nama **sc.**

```
import java.util.Scanner;

public class LayananAkademikSIAKAD {

    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

7. Kemudian lakukan instansiasi objek **AntrianLayanan** dengan nama **antrian** dan nilai parameternya adalah nilai maksimal antrian yang ditentukan (misal sama dengan 5).

```
AntrianLayanan antrian = new AntrianLayanan(max:5);
```

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

```
int pilihan;
```

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

```
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.tambahAntrian(mhs);
        break;
    case 2:
        Mahasiswa dilayani = antrian.layaniMahasiswa();
        if (dilayani != null) {
            System.out.print("Melayani mahasiswa: ");
            dilayani.tampilkanData();
        }
        break;
    case 3:
        antrian.lihatTerdepan();
        break;
            case 4:
                antrian.tampilkanSemua();
                break;
            case 5:
                System.out.println("Jumlah dalam antrian: " + antrian.getJumlahAntrian());
                break;
            case 0:
                System.out.println("Terima kasih.");
                break;
            default:
                System.out.println("Pilihan tidak valid.");
        }
    } while (pilihan != 0);
    sc.close();
  }
}
```

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

Verifikasi Hasil Percobaan

```
PS C:\Semester 2\Praktikum ASD\Jobsheet\Jobsheet10>  & 'C:
jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetai
es' '-cp' 'C:\Users\ASUS\AppData\Roaming\Code\User\workspac
d321c2139586dd55d91f19\redhat.java\jdt_ws\Jobsheet10_cab07c
demikSIAKAD'

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 1
NIM : 123
Nama : Aldi
Prodi : TI
Kelas : 1A
Aldi berhasil masuk ke antrian.

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 1
NIM : 124
Nama : Bobi
Prodi : TI
Kelas : 1G
Bobi berhasil masuk ke antrian.
```

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 123 - Aldi - TI - 1A
2. 124 - Bobi - TI - 1G

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 2
Melayani mahasiswa: 123 - Aldi - TI - 1A
```

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 124 - Bobi - TI - 1G

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 5
Jumlah dalam antrian: 1

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 3
Mahasiswa terdepan: NIM - NAMA - PRODI - KELAS
124 - Bobi - TI - 1G
```

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!

```
public Mahasiswa lihatAkhir() {
if (IsEmpty()) {
    System.out.println(x:"Antrian kosong.");
    return null;
}
System.out.print(s:"Mahasiswa terakhir: ");
System.out.println(x:"NIM - NAMA - PRODI - KELAS");
data[rear].tampilkanData();
return data[rear];
```

```
System.out.println(x:"6. Lihat Antrian Paling Belakang");
```

```
        break;
    case 6:
        antrian.lihatAkhir();
        break;
```

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
6. Lihat Antrian Paling Belakang
0. Keluar
Pilih menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 123 - ilham - TE - 2E
2. 124 - ingrid - TK - 1A

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Semua Antrian
5. Jumlah Mahasiswa dalam Antrian
6. Lihat Antrian Paling Belakang
0. Keluar
Pilih menu: 6
Mahasiswa terakhir: NIM - NAMA - PRODI - KELAS
124 - ingrid - TK - 1A
```

Tugas

Waktu : 120 Menit

Buatlah program antrian untuk mengilustasikan 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**.

Diagram Class

Antrian
data:Mahasiswa[] front: int rear: int size: int max: int totalProse: int
AntrianKRS21 (max: int) IsEmpty(): boolean IsFull(): boolean clear(): void Enqueue(dt: Mahasiswa): void tambahAntrian(mhs: Mahasiswa): void layaniKRS(): void tampilkanSemua(): void getJumlahAntrian(): int lihatTerdepan(): void lihatAkhir(): void jumlahAntrian(): int jumlahDiproses(): int jumlahBelumDiproses(): int

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

    public 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);
    }
}
```

```java
public class AntrianKRS21 {
    Mahasiswa[] data;
    int front = 0;
    int rear = 0;
    int size = 0;
    int max = 10;
    int totalProses = 30;

    public AntrianKRS21(int max) {
        data = new Mahasiswa[max];
        this.front = 0;
        this.max = max;
        this.rear = -1;
        this.size = 0;
    }

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

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

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

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

```
public void tambahAntrian(Mahasiswa mhs) {
    if (IsFull()) {
        System.out.println(x:"Antrian penuh, tidak dapat menambah mahasiswa.");
        return;
    }
    rear = (rear + 1) % max;
    data[rear] = mhs;
    size++;
    System.out.println(mhs.nama + " berhasil masuk ke antrian.");
}

public void layaniRS25() {
    if (size < 2) {
        System.out.println(x:"Antrian harus minimal 2 mahasiswa");
        return;
    }
    System.out.println(x:"Memproses 2 mahasiswa:");
    for (int i = 0; i < 2; i++) {
        data[front].tampilkanData();
        front = (front + 1) % max;
        size--;
        totalProses++;
    }
}
```

```
    public void tampilkanSemua() {
        if (IsEmpty()) {
            System.out.println(x:"Antrian kosong.");
            return;
        }
        System.out.println(x:"Daftar mahasiswa dalam antrian: ");
        System.out.println(x:" NIM - NAMA - PRODI - KELAS");
        for (int i = 0; i < size; i++) {
            int index = (front + i) % max;
            System.out.print((i + 1) + ". ");
            data[index].tampilkanData();
        }
    }

    public int getJumlahAntrian() {
        return size;
    }

    public void lihatTerdepan() {
        if (IsEmpty()) {
            System.out.println(x:"Antrian kosong.");
        } else {
            System.out.println(x:"Mahasiswa terdepan: ");
            System.out.println(x:" NIM - NAMA - PRODI - KELAS");
            data[front].tampilkanData();
        }
    }
```

```java
public void lihatAkhir() {
    if (IsEmpty()) {
        System.out.println(x:"Antrian kosong.");
    } else {
        System.out.println(x:"Mahasiswa paling belakang: ");
        System.out.println(x:"NIM - NAMA - PRODI - KELAS");
        data[rear].tampilkanData();
    }
}

public int jumlahAntrian() {
    return size;
}

public int jumlahDiproses() {
    return totalProses;
}

public int jumlahBelumDiproses() {
    return max - totalProses;
}
```

```java
import java.util.Scanner;

public class LayananKRS {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianKRS21 antrian = new AntrianKRS21(max:10);
        int pilihan;

        do {
            System.out.println(x:"\n=== MENU ANTRIAN KRS ===");
            System.out.println(x:"1. Tambah Mahasiswa ke Antrian");
            System.out.println(x:"2. Proses 2 Mahasiswa");
            System.out.println(x:"3. Tampilkan semua Antrian");
            System.out.println(x:"4. Tampilkan 2 Antrian Terdepan");
            System.out.println(x:"5. Tampilkan Antrian Terakhir");
            System.out.println(x:"6. Cek Antrian Kosong");
            System.out.println(x:"7. Cek Antrian Penuh");
            System.out.println(x:"8. Kosongkan Antrian");
            System.out.println(x:"9. Tampilkan Jumlah Antrian");
            System.out.println(x:"10. Tampilkan Jumlah Mahasiswa Sudah Diproses");
            System.out.println(x:"11. Tampilkan Jumlah Mahasiswa Belum Diproses");
            System.out.println(x:"0. Keluar");
            System.out.print(s:"Pilih menu: ");
            pilihan = sc.nextInt();
            sc.nextLine();
```

```java
switch (pilihan) {
    case 1:
        if (!antrian.IsFull()) {
            System.out.print(s:"NIM: ");
            String nim = sc.nextLine();
            System.out.print(s:"Nama: ");
            String nama = sc.nextLine();
            System.out.print(s:"Jurusan: ");
            String prodi = sc.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas = sc.nextLine();
            antrian.Enqueue(new Mahasiswa(nim, nama, prodi, kelas));
        } else {
            System.out.println(x:"Antrian sudah penuh.");
        }
        break;
    case 2:
        antrian.layaniRS25();
        break;
    case 3:
        antrian.tampilkanSemua();
        break;
    case 4:
        antrian.lihatTerdepan();
        break;
    case 5:
        antrian.lihatAkhir();
        break;
```

```
            case 6:
                System.out.println(antrian.IsEmpty() ? "Antrian kosong." : "Antrian tidak kosong.");
                break;
            case 7:
                System.out.println(antrian.IsFull() ? "Antrian penuh." : "Antrian belum penuh.");
                break;
            case 8:
                antrian.clear();
                break;
            case 9:
                System.out.println("Jumlah mahasiswa dalam antrian: " + antrian.jumlahAntrian());
                break;
            case 10:
                System.out.println("Jumlah mahasiswa yang sudah diproses: " + antrian.jumlahDiproses());
                break;
            case 11:
                System.out.println("Jumlah mahasiswa yang belum diproses: " + antrian.jumlahBelumDiproses());
                break;
            case 12:
                System.out.println(x:"Keluar dari program.");
                System.exit(status:0);
                break;
            default:
                System.out.println(x:"Pilihan tidak valid.");
        }
    } while (pilihan != 0);
}
```