Laporan Jobsheet 10 Algoritma dan Struktur Data

Queue

244107020151

Ghazwan Ababil

Teknik Informatika / TI – E (11)

Politeknik Negeri Malang

Jurusan Teknologi Informasi

2025

1. Praktikum

1.1. Percobaan 1: Operasi Dasar Queue

a. Class Queue11

1. Membuat folder baru Jobsheet10 di dalam repository Praktikum-ASD. Kemudian di dalam folder Jobsheet10 membuat foder baru dengan nama P1Jobsheet10 dan membuat class baru dengan nama Queue11
2. Menambahkan atribut, method, dan konstruktor pada class Queue11 sesuai dengan class diagram

```
package P1Jobsheet10;

public class Queue11 {
    int data[];
    int front, rear, size, max;

    public Queue11(int n) {
        this.max = n;
        data = new int[max];
        size = 0;
        front = rear = -1;
    }

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

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

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

    public void print(){
        if (isEmpty()) System.out.println("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);
        }
    }
```

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

    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++;
        };
    }

    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;
    }
}
```

b. Class QueueMain11

3. Membuat class baru dengan nama QueueMain11.java
4. Menambahkan fungsi main, dan menu kemudian melakukan instansiasi object queue berserta scanner didalam main dan menambahkan menu untuk pemilihan dalam mengoperasikan queue dalam mengelola data dengan struktur do-while.

```
package P1Jobsheet10;
import java.util.Scanner;

public class QueueMain11 {
    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("-----------------");
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

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

        Queue11 Q = new Queue11(n);
        int pilih;
        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;
                default:
                    System.out.println("Pilihan tidak valid");
            }
        } while (pilih >0 && pilih < 6);
    }
}
```

c. Commit dan push kode program ke github

```
PS D:\Java\Praktikum-ASD> git add .
PS D:\Java\Praktikum-ASD> git commit -m "Jobsheet 10 Percobaan 1"
[main a27fd2f] Jobsheet 10 Percobaan 1
 2 files changed, 124 insertions(+)
 create mode 100644 Jobsheet10/P1Jobsheet10/Queue11.java
 create mode 100644 Jobsheet10/P1Jobsheet10/QueueMain11.java
PS D:\Java\Praktikum-ASD> git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.36 KiB | 465.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ghazwanz/Praktikum-ASD.git
   2a00c1e..a27fd2f  main -> main
branch 'main' set up to track 'origin/main'.
```

d. Output Kode program

```
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
```

e. Pertanyaan

1. Pada konstuktor class Queue11, nilai awal atribut front dan rear bernilai -1 karena atribut front dan rear digunakan sebagai pointer indeks pada data di dalam Queue pada indeks sebuah array, data ke-1 terletak pada indeks ke-0, sehingga jika diberikan nilai awal 0 maka atribut front dan rear akan menunjuk data ke-1 sebagai atribut front dan rear. Sedangkan atribut size bernilai 0 karena digunakan untuk menghitung berapa banyak data pada queue sehingga diberikan nilai awal 0.
2. Baris kode method enqueue berikut

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

Pada method **enqueue** berfungsi apabila kondisi indeks rear telah mencapai indeks maksimal kapasitas yaitu max -1 dan size dari queue belum penuh saat melakukan penambahan data (enqueue), maka indeks rear akan diberikan nilai indeks 0, untuk kembali lagi ke indeks pertama.

3. Baris kode Pada method **dequeue** berikut

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

berfungsi apabila kondisi indeks front telah mencapai indeks maksimal kapasitas yaitu max -1 dan size dari queue belum penuh saat melakukan penghapusan data (dequeue), maka indeks rear akan diberikan nilai indeks 0, untuk kembali lagi ke indeks pertama.

4. Karena dalam konsep queue data pertama terletak pada indeks front dan data terakhir terletak pada indeks front. Oleh karena itu, inisialisasi variabel i untuk perulangan dimulai dari front.
5. Pada method print memiliki baris kode

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

Baris kode tersebut merupakan bagian perubahan kondisi (increment / decrement), baris kode memerlukan modulo (%) max untuk mereset indeks perulangan (i) jika telah mencapai max, hak tersebut dikarenakan indeks front tidak selalu berada di awal queue, dan indeks rear tidak selalu berada pada indeks akhir (max-1), oleh karena itu diperlukan modulo (%) max untuk mereset indeks perulangan (i) ke 0 ketika mencapai indeks terakhir (max-1) hingga indeks rear ditemukan.

6. Potongan kode program yang merupakan queue overflow terletak pada method enqueue (method penambahan data) pada baris kode

```
if (isFull()) System.out.println("Queue sudah penuh");
```

Baris kode tersebut berfungsi untuk melakukan pengecekan terlebih dahulu apakah queue telah penuh dengan method isFull, jika bernilai benar maka method tersebut tidak jadi menambahkan data dan mengembalikan pesan queue sudah penuh untuk mencegah queue overflow (data melebihi maksimal queue), jika baris kode tersebut dihilangkan akan menyebabkan queue overflow pada method enqueue.

7. Melakukan modifikasi program sehingga pada saat terjadi queue overflow dan queue underflow, program dihentikan.

   Memodifikasi bagian kode pada case ke-1 (menu enqueue) dan case ke-2 (menu dequeue)

Melakukan modifikasi pada case ke-1 (menu enqueue) dengan menambahkan variabel yang menyimpan nilai boolean apakah queue sudah penuh sebelum dilakukan enqueue, setelah melakukan penambahkan data, akan dilakukan pengecekan pada variabel sebelumnya, jika queue sudah penuh menampilkan pesan queue sudah penuh dan menghentikan perulangan

```
case 1:
    boolean fullStatus = Q.isFull();
    System.out.print("Masukkan data baru: ");
    int dataMasuk = sc.nextInt();
    Q.enqueue(dataMasuk);
    if(fullStatus) return;
    break;
```

Melakukan modifikasi pada case ke-2 (menu dequeue) dengan menambahkan variabel yang menyimpan nilai boolean apakah queue masih kosong sebelum dilakukan dequeue, setelah melakukan pengurangan data, akan dilakukan pengecekan pada variabel sebelumnya, jika queue sebelumnya kosong maka akan menampilkan pesan queue masih kosong dan menghentikan perulangan

```
case 2:
    boolean emptyStatus = !Q.isEmpty();
    int dataKeluar = Q.dequeue();
    if (emptyStatus) System.out.println("Data yang dikeluarkan: " +
dataKeluar);
    else return;
    break;
```

Output kode program

Queue Overflow

```
Masukkan kapasitas queue: 1
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
1
Masukkan data baru: 123
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
1
Masukkan data baru: 12312
Queue sudah penuh
PS D:\Java\Praktikum-ASD>
```

Queue Underflow

```
Masukkan kapasitas queue: 2
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
2
Queue masih kosong
PS D:\Java\Praktikum-ASD>
```

1.2. Percobaan 2 : Antrian Layanan Akademik

a. Membuat folder baru dengan nama P2Jobsheet10 di dalam folder Jobsheet10, kemudian menambahkan class baru dengan nama Mahasiswa11.

b. Menambahkan atribut, method, dan konstruktor pada class Mahasiswa11 sesuai dengan class diagram

```
package P2Jobsheet10;

public class Mahasiswa11 {
    String nim,nama,prodi,kelas;

    Mahasiswa11(String nim, String nama, String prodi, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }

    void tampilkanData() {
        System.out.printf("%s - %s - %s - %s\n", nim, nama, prodi, kelas);
    }
}
```

c. Menyalin kode program class Queue pada percobaan 1 untuk digunakan kembali, mengganti nama class-nya dengan AntrianLayanan. Serta melakukan modifikasi tipe data array ke tipe object array pada class AntrianLayanan tersebut. Dan menyesuaikan method-method pada class tersebut agar sesuai dengan tipe data dari array-nya. Serta menambahkan method getJumlahAntrian untuk menampilkan nilai size

```
package P2Jobsheet10;

public class AntrianLayanan11 {
    Mahasiswa11 data[];
    int front, rear, size, max;

    public AntrianLayanan11(int max) {
        this.max = max;
        data = new Mahasiswa11[max];
        size = front = 0;
        rear = -1;
    }

    public void tambahAntrian(Mahasiswa11 mhs){
        if (isFull()) System.out.println("Antrian penuh, tidak dapat
menambah mahasiswa.");
        else {
            rear = (rear + 1) % max;
            data[rear] = mhs;
            size++;
            System.out.println(mhs.nama + " Berhasil masuk ke antrian.");
        };
    }

    public Mahasiswa11 layaniMahasiswa(){
        Mahasiswa11 mhs = null;
        if (isEmpty()) System.out.println("Queue masih kosong");
        else {
            mhs = data[front];
            front = (front + 1) % max;
            size--;
        }
        return mhs;
    }
```

```
    public void lihatTerdepan(){
        if (isEmpty()) System.out.println("Antrian kosong");
        else {
            System.out.println("Mahasiswa terdepan: ");
            System.out.println("NIM - Nama - Prodi - Kelas");
            data[front].tampilkanData();
        };
    }

    public void tampilkanSemua(){
        if (isEmpty()) System.out.println("Queue masih kosong");
        else {
            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) % max;
                System.out.print((i+1) + ". ");
                data[index].tampilkanData();
            }
        }
    }

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

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

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

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

d. Membuat class baru dengan nama LayananAkademikSIAKAD11, kemudian menambahkan fungsi main, kemudian melakukan instansiasi object AntrianLayanan11 berserta scanner didalam main dan menambahkan menu untuk pemilihan dalam mengoperasikan queue dalam mengelola data mahasiswa dengan struktur do-while.

```
package P2Jobsheet10;

import java.util.Scanner;

public class LayananAkademikSIAKAD11 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianLayanan11 antrian = new AntrianLayanan11(5);
        int pilihan;

        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();
                    Mahasiswa11 mhs = new Mahasiswa11(nim, nama, prodi,
kelas);
                    antrian.tambahAntrian(mhs);
                    break;
                case 2:
                    Mahasiswa11 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);
    }
}
```

e. Output kode program

```
=== 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: 0
Terima kasih.
PS D:\Java\Praktikum-ASD>
```

f. Push dan commit kode program ke github

```
PS D:\Java\Praktikum-ASD> git add .
PS D:\Java\Praktikum-ASD> git commit -m "Jobsheet 10 Percobaan 2 Antrian Layanan Akademik dan Modifikasi Pertanyaan Percobaan 2"
[main 6b8c0c6] Jobsheet 10 Percobaan 2 Antrian Layanan Akademik dan Modifikasi Pertanyaan Percobaan 2
 3 files changed, 165 insertions(+)
 create mode 100644 Jobsheet10/P2Jobsheet10/AntrianLayanan11.java
 create mode 100644 Jobsheet10/P2Jobsheet10/LayananAkademikSIAKAD11.java
 create mode 100644 Jobsheet10/P2Jobsheet10/Mahasiswa11.java
PS D:\Java\Praktikum-ASD> git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.99 KiB | 291.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ghazwanz/Praktikum-ASD.git
   1ca4e48..6b8c0c6  main -> main
branch 'main' set up to track 'origin/main'.
PS D:\Java\Praktikum-ASD>
```

g. Pertanyaan

1. memodifikasi program dengan menambahkan method baru bernama LihatAkhir pada class AntrianLayanan11 yang digunakan untuk mengecek antrian yang berada di posisi belakang. Menambahkan daftar menu 6. Cek Antrian paling belakang pada class LayananAkademikSIAKAD11 agar method LihatAkhir dapat dipanggil!

   Menambahkan method LihatAkhir pada class AntrianLayanan11

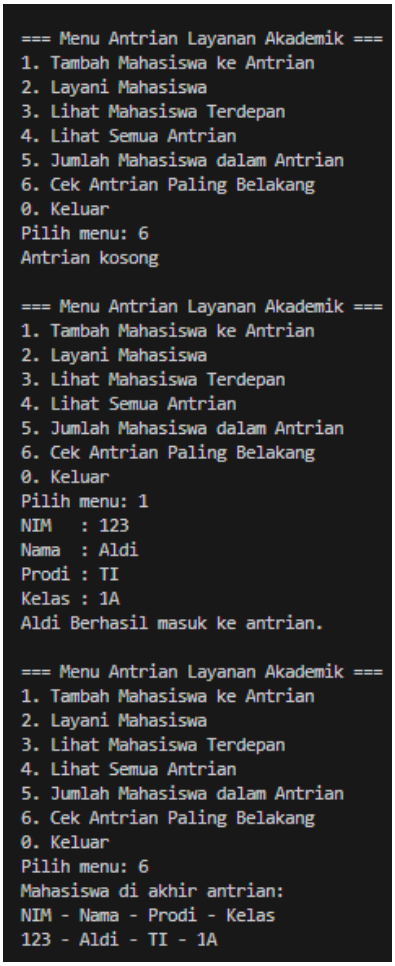
```
public void lihatAkhir(){
    if (isEmpty()) System.out.println("Antrian kosong");
    else {
        System.out.println("Mahasiswa di akhir antrian: ");
        System.out.println("NIM - Nama - Prodi - Kelas");
        data[rear].tampilkanData();
    };
}
```

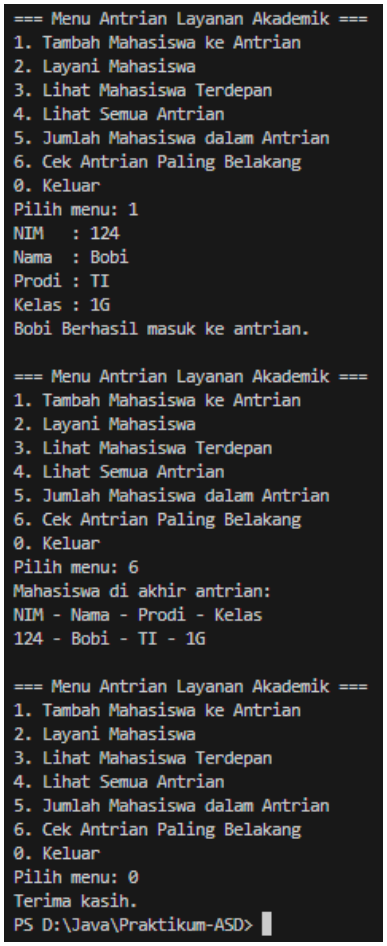
   Menambahkan menu untuk melihat mahasiswa pada antrian terakhir pada file LayananAkademikSIAKAD11

```
do {
    //...
    System.out.println("6. Cek Antrian Paling Belakang");

    switch (pilihan) {
        //...
        case 6:
            antrian.lihatAkhir();
            break;
    }
} while (pilihan != 0);
```

Hasil Output Kode Program





2. Tugas

Membuat 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. Menggunakan class untuk antrian seperti pada praktikum 1 dan 2.

Class Mahasiswa11.java

```
package TugasJobsheet10;

public class Mahasiswa11 {
    String nim,nama,prodi,kelas;

    Mahasiswa11(String nim, String nama, String prodi, String kelas) {
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.kelas = kelas;
    }

    void tampilkanData() {
        System.out.printf("%s - %s - %s - %s\n", nim, nama, prodi, kelas);
    }
}
```

Class AntrianKRS.java

```
package TugasJobsheet10;

public class AntrianKRS {
    Mahasiswa11 dataMhs[];
    int front, rear, size, max, counter, maxCounter;

    public AntrianKRS(int n) {
        this.max = n;
        dataMhs = new Mahasiswa11[max];
        size = counter = 0;
        front = rear = -1;
        maxCounter = 30;
    }
    public boolean isEmpty() {
        return size == 0;
    }
    public boolean isFull() {
        return size == max;
    }

    public void lihat2AntrianTerdepan(){
        if (isEmpty()) System.out.println("Antrian masih kosong");
        else if (size > 1) {
            int prosesMhs = 2;
            System.out.println("Antrian Terdepan: ");
            System.out.println("NIM - Nama - Prodi - Kelas");
            for (int i = 0; i < prosesMhs; i++) {
                int index = (front + i) % max;
                System.out.print((i+1) + ". ");
                dataMhs[index].tampilkanData();
            }
        }
        else System.out.println("Antrian hanya ada 1 mahasiswa");
    }

    public void lihatAntrian(){
        if (isEmpty()) System.out.println("Queue masih kosong");
        else {
            System.out.println("Daftar Antrian: ");
            System.out.println("NIM - Nama - Prodi - Kelas");
            int i = front;
            while (i != rear) {
                System.out.print((i+1) + ". ");
                dataMhs[i].tampilkanData();
                i = (i + 1) % max;
            }
            System.out.print((i+1) + ". ");
            dataMhs[i].tampilkanData();
        }
    }

    public void lihatAntrianAkhir(){
        if (isEmpty()) System.out.println("Antrian masih kosong");
        else {
            System.out.println("Antrian Paling Belakang: ");
            System.out.println("NIM - Nama - Prodi - Kelas");
            dataMhs[rear].tampilkanData();
        };
    }
```

```
    public void tambahAntrian(Mahasiswa11 mhs){
        if (isFull()) System.out.println("Antrian penuh, tidak dapat menambah
mahasiswa.");
        else {
            if(isEmpty()) front = rear = 0;
            else rear = (rear + 1) % max;
            dataMhs[rear] = mhs;
            size++;
            System.out.println(mhs.nama + " Berhasil ditambahkan ke
antrian.");
        };
    }

    public void clear(){
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Antrian Berhasil dikosongkan");
        } else System.out.println("Antrian masih kosong");
    }

    public Mahasiswa11[] prosesKRSMhs(){
        Mahasiswa11 mhs[] = new Mahasiswa11[2];
        int prosesMhs = 2;
        if (isEmpty()) System.out.println("Queue masih kosong");
        if(size < prosesMhs) System.out.println("Antrian Mahasiswa Kurang
dari 2");
        else if(counter >= maxCounter) System.out.println("DPA hanya dapat
melayani maksimal 30 mahasiswa");
        else {
            for (int i = 0; i < prosesMhs; i++) {
                mhs[i] = dataMhs[front];
                size--;
                front = (front + 1) % max;
                counter++;
            }
            if (isEmpty()) front = rear = -1;
        }
        return mhs;
    }

    public int getJumlahAntrian(){
        return size;
    }
    public int getMhsSudahKRS(){
        return counter;
    }
    public int getMhsBelumKRS(){
        return maxCounter-counter;
    }
}
```

Class LayananKRS.java

```
package TugasJobsheet10;
import java.util.Scanner;

public class LayananKRS {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int maxAntrian = 10;
        AntrianKRS antrian = new AntrianKRS(maxAntrian);
        int menu;

        do {
            System.out.println("\n=== Menu Antrian KRS ===");
            System.out.println("1. Cek Antrian Kosong");
            System.out.println("2. Cek Antrian Penuh");
            System.out.println("3. Kosongkan Antrian");
            System.out.println("4. Tambah Mahasiswa ke Antrian");
            System.out.println("5. Layani Mahasiswa");
            System.out.println("6. Tampilkan Semua Antrian");
            System.out.println("7. Lihat 2 Antrian Terdepan");
            System.out.println("8. Lihat Antrian Paling Belakang");
            System.out.println("9. Cetak Jumlah Antrian");
            System.out.println("10. Cetak Jumlah Yang Sudah Melakukan Proses
KRS");
            System.out.println("11. Cetak Jumlah Yang Belum Melakukan Proses
KRS");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            menu = sc.nextInt();
            sc.nextLine();

            switch (menu) {
                case 1:
                    if (antrian.isEmpty()) System.out.println("Antrian
kosong.");
                    else System.out.println("Antrian tidak kosong.");
                    break;
                case 2:
                    if (antrian.isFull()) System.out.println("Antrian penuh.");
                    else System.out.println("Antrian tidak penuh.");
                    break;
                case 3:
                    antrian.clear();
                    break;
                case 4:
                    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();
                    Mahasiswa11 mhs = new Mahasiswa11(nim, nama, prodi, kelas);
                    antrian.tambahAntrian(mhs);
                    break;
```

```
                case 5:
                    Mahasiswa11[] dilayani = antrian.prosesKRSMhs();
                    if (dilayani[0] != null) {
                        System.out.print("Melayani mahasiswa: ");
                        System.out.println("NIM - Nama - Prodi - Kelas");
                        for (int i = 0; i < dilayani.length; i++) {
                            System.out.print((i + 1) + ". ");
                            dilayani[i].tampilkanData();
                        }
                    }
                    break;
                case 6:
                    antrian.lihatAntrian();
                    break;
                case 7:
                    antrian.lihat2AntrianTerdepan();
                    break;
                case 8:
                    antrian.lihatAntrianAkhir();
                    break;
                case 9:
                    System.out.println("Jumlah dalam antrian: " +
antrian.getJumlahAntrian());
                    break;
                case 10:
                    System.out.println("Jumlah yang sudah melakukan proses
KRS: " + antrian.getMhsSudahKRS());
                    break;
                case 11:
                    System.out.println("Jumlah yang belum melakukan proses
KRS: " + antrian.getMhsBelumKRS());
                    break;
                case 0:
                    System.out.println("Terima kasih.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
            }
        } while (menu != 0);
    }
}
```

Hasil output

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 1
Antrian kosong.

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 2
Antrian tidak penuh.
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 3
Antrian masih kosong
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 4
NIM   : 1001
Nama  : Dila
Prodi : TI
Kelas : 1A
Dila Berhasil ditambahkan ke antrian.
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 5
Antrian Mahasiswa Kurang dari 2

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 9
Jumlah dalam antrian: 1
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 6
Daftar Antrian:
NIM - Nama - Prodi - Kelas
1. 1001 - Dila - TI - 1A

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 7
Antrian hanya ada 1 mahasiswa
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 8
Antrian Paling Belakang:
NIM - Nama - Prodi - Kelas
1001 - Dila - TI - 1A
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 10
Jumlah yang sudah melakukan proses KRS: 0

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 11
Jumlah yang belum melakukan proses KRS: 30
```

Menambah Data Mahasiswa Baru

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 4
NIM   : 1002
Nama  : Erik
Prodi : TI
Kelas : 1B
Erik Berhasil ditambahkan ke antrian.

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 8
Antrian Paling Belakang:
NIM - Nama - Prodi - Kelas
1002 - Erik - TI - 1B
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 1
Antrian tidak kosong.
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 9
Jumlah dalam antrian: 2
```

Setelah terdapat 2 data pada antrian

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 7
Antrian Terdepan:
NIM - Nama - Prodi - Kelas
1. 1001 - Dila - TI - 1A
2. 1002 - Erik - TI - 1B
```

Melayani mahasiswa ketika terdapat 2 data pada antrian

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 5
Melayani mahasiswa: NIM - Nama - Prodi - Kelas
1. 1001 - Dila - TI - 1A
2. 1002 - Erik - TI - 1B

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 1
Antrian kosong.
```

Kondisi antrian setelah melayani mahasiswa

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 1
Antrian kosong.
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 10
Jumlah yang sudah melakukan proses KRS: 2
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 11
Jumlah yang belum melakukan proses KRS: 28
```

Kondisi ketika antrian telah penuh

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 9
Jumlah dalam antrian: 10
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 6
Daftar Antrian:
NIM - Nama - Prodi - Kelas
1. 1003 - Dani - TI - 1A
2. 1004 - Zaki - TI - 1B
3. 1005 - Rian - Ti - 1A
4. 1006 - Cooper - TI - 1B
5. 1007 - Ryo - TI - 1A
6. 1008 - Nate - TI - 1B
7. 1009 - Denver - TI - 1A
8. 1010 - Karol - TI - 1B
9. 1011 - Monroe - TI - 1A
10. 1012 - Ray - TI - 1B
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 2
Antrian penuh.
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 7
Antrian Terdepan:
NIM - Nama - Prodi - Kelas
1. 1003 - Dani - TI - 1A
2. 1004 - Zaki - TI - 1B
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 8
Antrian Paling Belakang:
NIM - Nama - Prodi - Kelas
1012 - Ray - TI - 1B
```

Mencoba menambahkan mahasiswa ketika antrian penuh

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 4
NIM   : 1013
Nama  : Rigen
Prodi : TI
Kelas : 1A
Antrian penuh, tidak dapat menambah mahasiswa.
```

Kondisi ketika DPA telah memproses KRS seluruh mahasiswa

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 10
Jumlah yang sudah melakukan proses KRS: 30
```

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 11
Jumlah yang belum melakukan proses KRS: 0
```

Mencoba memproses data mahasiswa lagi ketika seluruh KRS Mahasiswa telah diproses (maksimal 30 mahasiswa)

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 5
DPA hanya dapat melayani maksimal 30 mahasiswa
```

Mengosongkan antrian

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 9
Jumlah dalam antrian: 2

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 3
Antrian Berhasil dikosongkan
```

Mengecek Antrian setelah dikosongkan

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 9
Jumlah dalam antrian: 0

=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 1
Antrian kosong.
```

Keluar dari program

```
=== Menu Antrian KRS ===
1. Cek Antrian Kosong
2. Cek Antrian Penuh
3. Kosongkan Antrian
4. Tambah Mahasiswa ke Antrian
5. Layani Mahasiswa
6. Tampilkan Semua Antrian
7. Lihat 2 Antrian Terdepan
8. Lihat Antrian Paling Belakang
9. Cetak Jumlah Antrian
10. Cetak Jumlah Yang Sudah Melakukan Proses KRS
11. Cetak Jumlah Yang Belum Melakukan Proses KRS
0. Keluar
Pilih menu: 0
Terima kasih.
```

Class Diagram

Mahasiswa11

Mahasiswa11
nim : String nama : String prodi : String kelas : String
Mahasiswa11(nim: String, nama: String, prodi: String, kelas: String) tampilkanData() : void

AntrianKRS

AntrianKRS
dataMhs : Mahasiswa11[] front : int rear : int size : int max : int counter : int maxCounter : int
AntrianKRS(n: int) isEmpty() : boolean isFull() : boolean lihat2AntrianTerdepan() : void lihatAntrian() : void lihatAntrianAkhir() : void tambahAntrian(mhs: Mahasiswa11) : void clear() : void prosesKRSMhs() : Mahasiswa11[] getJumlahAntrian() : int getMhsSudahKRS() : int getMhsBelumKRS() : int

Commit dan Push kode program ke github

```
PS D:\Java\Praktikum-ASD> git add .
PS D:\Java\Praktikum-ASD> git commit -m "Jobsheet 10 Tugas Ilustrasi Persetujuan KRS"
[main e233a3e] Jobsheet 10 Tugas Ilustrasi Persetujuan KRS
 3 files changed, 215 insertions(+)
 create mode 100644 Jobsheet10/TugasJobsheet10/AntrianKRS.java
 create mode 100644 Jobsheet10/TugasJobsheet10/LayananKRS.java
 create mode 100644 Jobsheet10/TugasJobsheet10/Mahasiswa11.java
PS D:\Java\Praktikum-ASD> git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 2.42 KiB | 496.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ghazwanz/Praktikum-ASD.git
   6b8c0c6..e233a3e  main -> main
branch 'main' set up to track 'origin/main'.
```