LAPORAN HASIL PRAKTIKUM
ALGORITMA STRUKTUR DATA
JOBSHEET 10

NAMA : JIRO AMMAR WAFI

NIM : 244107020190

KELAS : 1-E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLINEMA

2025

|| JOBSHEET X - Queue

1. Percobaan 1: Operasi Dasar Queue

A. Class Queue

1. Pada package P1Jobsheet10. Menambahkan atribut yang sesuai pada diagram class, dan membuat konstruktor paremeter untuk mengubah panjang queue.

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

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

2. Method isEmpty(): cek apakah queue saat ini kosong.

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

3. Method isFull(): cek apakah queue telah penuh.

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

4. Method peek(): method untuk menampilkan data dari queue dengan urutan paling depan (indeks front).

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

5. Method clear(): mengosongkan isi dari queue dengan set nilai rear & front & size menjadi 0.

```
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");
    }
}
```

6. Method Enqueue(): Membawa parameter data untuk di isi ke dalam queue.

```
public boolean Enqueue (int dt) {
    if (isFull()) {
        System.out.println(x:"Queue sudah penuh");
        System.out.println(x:"Queue Overflow -> Program dihentikan !")
        return false;
    } else {
        if (isEmpty()) {
            front = rear = 0;
        } else {
            if (rear == max -1) {
                rear = 0;
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
        return true;
    }
}
```

7. Method Dequeue(): Mengeluarkan data pada indeks front dari queue.

```
public int Dequeue () {
    int dt = 0;
    if (isEmpty()) {
        System.out.println(x:"Qeueue masih kosong");
        System.out.println(x:"Queue Underflow -> Program dihentikan !");
    } else {
        dt = data[front];
        size--;
        if (isEmpty()) {
            front = rear = -1;
        } else {
            if (front == max - 1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
```

B. Class QueueMain

1. Masih dengan package yang sama, pada class ini dibuat method menu untuk menampilkan operasi yang dapat dijalankan pada queue.

```
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:"--------------------------------");
    }
```

2. Membuat method main, mendeklarasikan variabel sc (scanner) & pilih (int). Memberikan opsi kepada pengguna untuk mengatur kapasitas dari queue dan melakukan instansiasi objek dari class Queue menggunakan konstruktor ber-parameter.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);    Resource leak: 'sc' is never closed
    int pilih;
    boolean status = true;

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

3. Membuat perulangan do-while dengan syarat pilihan menu, didalam perulangan terdapat switch-case untuk memanggil method yang sesuai:

```
do {
    menu();
    pilih = sc.nextInt();
    System.out.println(x:"--------------------------------");


    switch (pilih) {
        case 1:
            System.out.print(s:"Masukkan data baru: ");
            int dataMasuk = sc.nextInt();
            status = Q.Enqueue(dataMasuk);
            break;
        case 2:
            int dataKeluar = Q.Dequeue();
            if (dataKeluar != 0) {
                System.out.println("Data yang dikeluarkan: " + dataKeluar);
            } else {
                status = false;
            }
            break;
        case 3:
            Q.print();
            break;
        case 4:
            Q.peek();
            break;
        case 5:
            Q.clear();
            break;
    }
} while (pilih>=1 && pilih <= 5 && status);
```

Verifikasi Percobaan

✔️ Hasil sesuai dengan yang ada di jobsheet

```
Masukkan kapasitas queue: 3
Masukkan operasi yang diinginkan
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
--------------------------------
1
--------------------------------
Masukkan data baru: 25
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
--------------------------------
4
--------------------------------
Elemen terdepan: 25
```

|| Pertanyaan Percobaan 1

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

Jawaban:

```
size = 0;
front = rear = -1;
```

Nilai front & rear = -1 ditujukan dengan maksud bahwa isi queue saat ini masih dalam keadaan kosong. Sedangkan, size = 0 juga memiliki alasan serupa, queue sedang kosong. Front, rear dan size akan terus berubah seiring data dimasukkan ke dalam queue.

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

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

Jawaban:

Dalam potongan kode tersebut, terdapat pemilihan if dengan kondisi (rear == max -1), kondisi ini menyeleksi apakah rear saat ini sedang berada di baris paling belakang pada queue. -1 digunakan karena menggunakan indeks, sehingga susunan queue dihitung dari indeks 0. Apabila kondisi tersebut sesuai, maka rear akan di set menjadi 0 yang berarti rear akan kembali ke indeks 0 yaitu baris paling depan untuk digunakan untuk menambahkan data.

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

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

Jawaban:

Kode ini, menyeleksi kondisi apakah front berada pada indeks bagian paling kanan/belakang dalam queue. Jika true, maka front akan di set 0 yang berarti front akan kembali ke indeks 0 atau barisan paling kiri/depan

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

Jawaban:

Posisi front tidak selalu pada indeks 0, demi menghindari kemungkinan error saat menjalankan metode print() dalam kasus indeks pada queue tak memiliki data atau null.

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

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

Jawaban:

Potongan kode tersebut digunakan untuk melakukan traverse queue secara melingkar, apabila i saat ini berada pada indeks paling belakang(max-1), dengan kode ini i akan kembali ke indeks paling depan (0) untuk menampilkan data sesuai indeks pada queue tersebut. Sehingga, data pada queue pada indeks i akan terus dicetak hingga menyentuh rear.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

Jawaban:

```
if (isFull()) {
    System.out.println(x:"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!

Jawaban:

Modifikasi Class Queue:

1. Enqueue(): Menambahkan baris teks untuk konfirmasi problem dan mengubah nilai return (void menjadi boolean).

```
public boolean Enqueue (int dt) {
    if (isFull()) {
        System.out.println(x:"Queue sudah penuh");
        System.out.println(x:"Queue Overflow -> Program dihentikan !");
        return false;
```

2. Dequeue(): Menambahkan baris teks untuk konfirmasi problem.

```
public int Dequeue () {
    int dt = 0;
    if (isEmpty()) {
        System.out.println(x:"Qeueue masih kosong");
        System.out.println(x:"Queue Underflow -> Program dihentikan !");
```

Modifikasi Class QueueMain:

1. Menambahkan variabel status

```
boolean status = true;
```

2. Case 1 & 2: Memperbarui update berdasarkan return masing2 method

```
case 1:
    System.out.print(s:"Masukkan data baru: ");
    int dataMasuk = sc.nextInt();
    status = Q.Enqueue(dataMasuk);
    break;
case 2:
    int dataKeluar = Q.Dequeue();
    if (dataKeluar != 0) {
        System.out.println("Data yang dikeluarkan: " + dataKeluar);
    } else {
        status = false;
    }
    break;
```

3. Kondisi do-while: menambahkan status sebagai syarat untuk loop terus berlanjut.

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

2. Percobaan 2: Antrian Layanan Akademik

1. Class Mahasiswa, berisi atribut, konstruktor parameter dan method tampilkanData() untuk menampilkan informasi Mahasiswa.

```java
package Jobsheet10.P2Jobsheet10;

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

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

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

2. Class AntrianLayanan, mengisi atribut Queue serta konstruktor parameter untuk menginisialisasi atribut tersebut.

```java
package Jobsheet10.P2Jobsheet10;

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

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

3. tambahAntrian() & layaniMahasiswa(), merupakan method Enqueue & Dequeue untuk mengelola object yang berisi informasi Mahasiswa dari sebuah class ke dalam Antrian.

```java
public void tambahAntrian(Mahasiswa mhs) { // Enqueue
    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() { // Dequeue
    if(isEmpty()) {
        System.out.println(x:"Antrian kosong.");
        return null;
    }
    Mahasiswa mhs = data[front];
    front = (front + 1) % max;
    size--;
    return mhs;
}
```

4. Method yang serupa Peek & Print untuk class LayaniMahasiswa, digunakan untuk menampilkan informasi dari suatu atau seluruh object pada Mahasiswa. Method getJumlahAntrian() untuk menginformasikan banyaknya object Mahasiswa terkini yang disimpan pada Queue Antrian.

```
public void lihatTerdepan() { // Peek
    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();
    }
}

public void tampilkanSemua() { // Print
    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++) {···
}

public void clear() {···

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

5. Class LayananAkademikSiakad, mengisi class dengan menu yang akan digunakan sebagai tempat untuk mengoperasikan queue, dan memilih method yang sesuai dengan menu menggunakan switch-case.

```
do {
    System.out.println(x:"\n=== Menu Antrian Layanan Akademik ===");
    System.out.println(x:"1. Tambah Mahasiswa ke Antrian");
    System.out.println(x:"2. Layani Mahasiswa");
    System.out.println(x:"3. Lihat Mahasiswa terdepan");
    System.out.println(x:"4. Lihat semua antrian");
    System.out.println(x:"5. Jumlah mahasiswa dalam antrian");
    System.out.println(x:"6. Cek Antrian paling belakang");
    System.out.println(x:"0. Keluar");
    System.out.print(s:"Pilih Menu: "); pilihan = sc.nextInt(); sc.nextLine();

    switch (pilihan) {
        case 1:···
        case 2:···
        case 3:
            antrian.lihatTerdepan();
            break;
        case 4:
            antrian.tampilkanSemua();
            break;
        case 5:
            System.out.println("Jumlah dalam antrian: " + antrian.getJumlahAntrian());
            break;
        case 6:
            antrian.LihatAkhir();
            break;
        case 0:
            System.out.println(x:"Terima kasih.");
            break;
        default:
            System.out.println(x:"Pilihan tidak valid.");
    }
} while (pilihan != 0);
sc.close();
```

Verifikasi Percobaan

✔️ Hasil sesuai dengan yang ada di jobsheet

```
=== 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: 5
Jumlah dalam antrian: 2
```

```
=== 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 C:\Users\luvma\Downloads\KULIAH\Semester 2
```

|| Pertanyaan Percobaan 2

Modifikasi: Menambahkan method LihatAkhir(), untuk menampilkan informasi Mahasiswa yang berada pada antrian belakang.

```
public void LihatAkhir() {
    if(isEmpty()) {
        System.out.println(x:"Antrian kosong.");
    } else {
        System.out.println(x:"Mahasiswa terakhir: ");
        System.out.println(x:"NIM - NAMA - PRODI - KELAS");
        data[rear].tampilkanData();
    }
}
```

Verifikasi Hasil:

```
Pilih Menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 124 - 124 - 124 - 124
2. 125 - 125 - 125 - 125
3. 126 - 126 - 126 - 126

=== 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. Cek Antrian paling belakang
0. Keluar
Pilih Menu: 3
Mahasiswa terdepan:
NIM - NAMA - PRODI - KELAS
124 - 124 - 124 - 124

=== 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. Cek Antrian paling belakang
0. Keluar
Pilih Menu: 6
Mahasiswa terakhir:
NIM - NAMA - PRODI - KELAS
126 - 126 - 126 - 126
```

3. Latihan Praktikum

Demonstrasi Praktikum:

A. Input informasi KRS

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 1
NIM: A
Nama: A
Prodi: A
Kelas: A
A berhasil masuk ke antrian.
```

B. Memproses KRS (2 kondisi)

Kondisi apabila terdapat 2 atau lebih antrian:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 2
Melayani KRS milik mahasiswa:
A - A - A - A
B - B - B - B
```

Kondisi apabila hanya ada satu antrian:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 2
Melayani KRS milik mahasiswa:
C - C - C - C
Antrian ke 2 kosong.
```

C. Cek 2 Antrian paling depan, dengan kondisi;

Kondisi apabila terdapat 2 atau lebih antrian:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 3
KRS dengan antrian terdepan:
NIM - NAMA - PRODI - KELAS
A - A - A - A
B - B - B - B
```

Kondisi apabila hanya ada satu antrian:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 3
KRS dengan antrian terdepan:
NIM - NAMA - PRODI - KELAS
C - C - C - C
Hanya terdapat satu antrian saat ini
```

D. Menampilkan semua antrian yang ada dalam antrian:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 4
Daftar KRS dalam Antrian:
NO - NIM - NAMA - PRODI - KELAS
1. A - A - A - A
2. B - B - B - B
3. C - C - C - C
```

E. Cek informasi KRS yang berada pada antrian terakhir:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 5
KRS dengan antrian terakhir:
NIM - NAMA - PRODI - KELAS
C - C - C - C
```

F. Cek jumlah antrian saat ini:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 6
Jumlah KRS dalam antrian: 3
```

G. Jumlah KRS yang telah diproses oleh DPA:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 7
KRS telah diproses sebanyak: 3 kali.
```

H. Jumlah Mahasiswa yang belum mengajukan KRS untuk diproses:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 8
Mahasiswa yang belum memproses KRS sebanyak: 27
```

I. Penutup dari layanan ini:

```
=== Menu Antrian KRS ===
1. Tambah Permohonan KRS
2. Proses KRS
3. Cek antrian paling depan
4. Lihat semua antrian
5. Cek Antrian paling belakang
6. Jumlah Antrian saat ini
7. Jumlah KRS yang telah diproses
8. Jumlah Mahasiswa yang belum melakukan proses KRS
0. Keluar
Pilih Menu: 0
Terima kasih.
Seluruh KRS telah diproses, Terima Kasih !
```

Diagram Class

QueueKRS	
data: InfoKRS[]	size: int
front: int	max: int
rear: int	count: int
QueueKRS()	void tampilkanSemua()
boolean AntrianKosong()	void lihatTerdepan()
boolean AntrianPenuh()	void lihatAkhir()
void KosongkanAntrian()	int getJumlahAntrian()
void tambahAntrian()	int ApprovedCount()
InfoKRS layaniKRS()	boolean LimitApproval()