LAPORAN HASIL PRAKTIKUM

Algoritma dan Struktur Data

Jobsheet 10 Queue

Alexsa Fitria Ayu Siswoyo.

244107020020

1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

PRAKTIKUM 1
PERCOBAA 1 : OPERASI DASAR QUEUE

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 :

```
public class Queue02 {
    int [] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue02 (int n) {
        max = n;
        data = new int[max];
        size =0;
        front = rear = -1;
    }
}
```

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

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

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

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

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

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

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

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

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

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

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

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

9. Buat method Dequeue bertipe int untuk mengeluarkan data pada queue di posisi belakang.

```
public int Dequeue () {
    int dt = 0;
    if (IsEmpty()) {
        System.out.println("Queue masih kosong");
    } else {
        dt = data[front];
        size--;
        if (IsEmpty()) {
            front = rear = -1;
        } else {
            if (front == max - 1) {
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
```

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

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

11. Buat fungsi main, kemudian deklarasikan Scanner dengan nama sc.
12. Buat variabel n untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.
13. Lakukan instansiasi objek Queue dengan nama Q dengan mengirimkan parameter n sebagai kapasitas elemen queue
14. Deklarasikan variabel dengan nama pilih bertipe integer untuk menampung pilih menu dari pengguna.

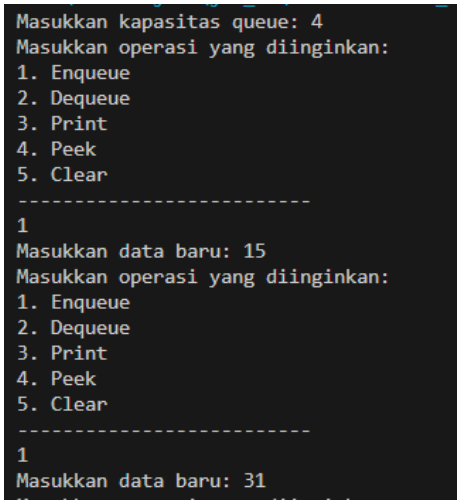
```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Masukkan kapasitas queue: ");
    int n = sc.nextInt();
    Queue02 Q = new Queue02(n);
    int pilih;
}
```

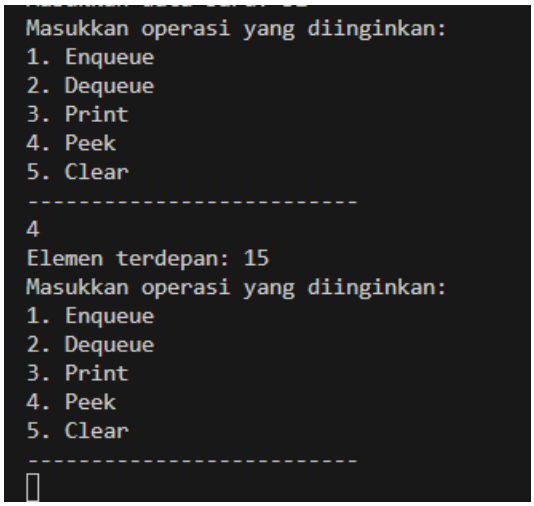
15. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case untuk menjalankan operasi queue sesuai dengan masukan pengguna.

```
do {
    menu();
    pilih = sc.nextInt();
    switch (pilih) {
        case 1:
            System.out.print("Masukkan data baru: ");
            int dataMasuk = sc.nextInt();
            Q.Enqueue(dataMasuk);
            break;
```

```
            case 2:
                int dataKeluar = Q.Dequeue();
                if (dataKeluar != 0) {
                    System.out.println("Data yang dikeluarkan: "
+ dataKeluar);
                }
                break;
            case 3:
                Q.Print();
                break;
            case 4:
                Q.peek();
                break;
            case 5:
                Q.clear();
                break;
        }
    } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih ==
4 || pilih == 5);
}
```

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





PERTANYAAN

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
   - **Inisialisasi front dan rear dengan -1:** Nilai awal -1 pada variabel front (depan) dan rear (belakang) mengindikasikan bahwa antrian (queue) saat ini tidak berisi elemen atau dalam keadaan kosong. Ketika elemen pertama dimasukkan ke dalam antrian, kedua variabel ini akan diperbarui menjadi 0, yang menunjukkan indeks elemen pertama. Penggunaan -1 sebagai nilai awal memberikan cara yang mudah untuk memverifikasi apakah antrian kosong.
   - **Inisialisasi size dengan 0:** Variabel size (ukuran) diatur ke 0 pada saat objek antrian pertama kali dibuat. Hal ini karena pada awalnya, belum ada elemen yang tersimpan di dalam antrian.

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

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

   Kode ini bertujuan untuk menerapkan struktur data antrian melingkar (circular queue). Mekanisme utamanya adalah menangani kondisi ketika indeks rear (belakang) telah mencapai batas akhir dari array penyimpanan (indeks max - 1). Dalam situasi ini, alih-alih menyebabkan overflow, kode akan secara otomatis mengembalikan nilai rear menjadi 0. Tindakan ini memungkinkan penambahan elemen-elemen baru pada bagian awal array, asalkan ada ruang kosong yang tercipta karena elemen-elemen sebelumnya telah dihapus (di-dequeue). Dengan demikian, antrian dapat menggunakan kembali ruang kosong di bagian depannya, menciptakan perilaku "melingkar" dan memanfaatkan kapasitas penyimpanan secara lebih efisien.

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

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

   Maksud dan kegunaan dari potongan kode ini juga berkaitan dengan implementasi *circular queue*. Ketika indeks front mencapai indeks terakhir dari array data (yaitu max - 1) dan queue belum kosong (dicek pada bagian lain kode), kode ini akan mengatur kembali nilai front menjadi 0. Hal ini memungkinkan pengambilan elemen berikutnya dari awal array.

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

   Pada method print(), variabel i pada perulangan while dimulai dari int i = front; karena kita ingin mencetak elemen-elemen queue sesuai dengan urutan masuknya. Elemen pertama yang masuk ke dalam queue berada pada indeks front. Perulangan dimulai dari front dan bergerak secara sirkular hingga semua elemen dalam queue tercetak.

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

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

Kode ini untuk melakukan pergerakan indeks secara sirkular di dalam array data. Operator modulo (%) digunakan untuk memastikan bahwa ketika indeks i + 1 melebihi batas atas array (max), indeks akan kembali ke 0. Ini memungkinkan method print() untuk mencetak semua elemen dalam queue, bahkan jika elemen-elemen tersebut tersebar di awal dan akhir array karena operasi enqueue dan dequeue pada *circular queue*.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

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

Kondisi IsFull() bernilai true menandakan bahwa queue telah mencapai kapasitas maksimumnya (size == max), sehingga tidak dapat lagi menerima elemen baru.

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!

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

System.exit(1); di dalam kondisi if (IsFull()) pada Enqueue() dan if (IsEmpty()) pada Dequeue(), program akan berhenti secara paksa dan memberikan kode error 1 ketika terjadi overflow atau underflow.

PERCOBAAN 2 : ANTRIAN LAYANAN AKADEMIK

Langkah-langkah Percobaan

1. Buat folder baru bernama P2Jobsheet10 di dalam repository Praktikum ASD, kemudian buat class baru dengan nama Mahasiswa.
2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya.
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.
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.
5. Berikutnya method peek dan print yaitu untuk menampilkan data antrian layanan paling depan dan menampilkan semua data antrian layanan

```
    public Mahasiswa02(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);
    }

    Mahasiswa02[] data;
    int front;
    int rear;
    int size;
    int max;

    public AntrianLayanan(int max) {
        this.max = max;
        this.data = new Mahasiswa02[max];
        this.front = 0;
        this.rear = -1;
        this.size = 0;
    }
```

```
    public Mahasiswa02 layaniMahasiswa() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return null;
        }
        Mahasiswa02 mhs = data[front];
        front = (front + 1) % max;
        size--;
        return mhs;
    }

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

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

    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.
7. Kemudian lakukan instansiasi objek AntrianLayanan dengan nama antrian dan nilai parameternya adalah nilai maksimal antrian yang ditentukan (misal sama dengan 5).
8. Deklarasikan variabel dengan nama pilihan bertipe integer untuk menampung pilih menu dari pengguna.

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

```
class LayananAkademikSIAKAD {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        AntrianLayanan antrian = new AntrianLayanan(5);

        int pilihan;


        do {

            System.out.println("--- 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();
```

```
                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();
                    Mahasiswa02 mhs = new Mahasiswa02(nim,
nama, prodi, kelas);
                    antrian.tambahAntrian(mhs);
                    break;
                case 2:
                    Mahasiswa02 mahasiswaDilayani =
antrian.layaniMahasiswa();
                    if (mahasiswaDilayani != null) {
                        System.out.println("Melayani
mahasiswa: ");
                        mahasiswaDilayani.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.

```
--- 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
123 - Aldi - TI - 1A
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
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.
```

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!

- Penambahan Method LihatAkhir() di Class AntrianLayanan

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

- Di dalam *loop* do-while pada method main(), ditambahkan opsi menu baru

```
System.out.println("6. Cek Antrian paling belakang");
```

- Di dalam blok switch (pilihan), ditambahkan case baru untuk menangani pilihan 6

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

Output :

```
--- 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 paling belakang: NIM - NAMA - PRODI - KELAS
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
6. Cek Antrian paling belakang
0. Keluar
Pilih menu: 3
Mahasiswa terdepan: NIM - NAMA - PRODI - KELAS
124 - Bobi - TI - 1G
```

TUGAS

KODE PROGRAM

1. StudentKRSQueue

```
package P2jobsheet10;

public class StudentKRSQueue {
    StudentKRS[] data;
    int front, rear, size, max;
    int approvedCount = 0;
    int max_students = 30;

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

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

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

    public void enqueue(StudentKRS dt) {
        if (isFull()) {
            System.out.println("Antrian penuh!!");
            return;
        }
        rear = (rear + 1) % max;
        data[rear] = dt;
        size++;
        System.out.printf("%s berhasil ditambahkan pada
indeks %d\n", dt.name, rear);
    }
```

```
    void dequeue2() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
        } else {
            for (int i = 0; i < 2; i++) {
                if (!isEmpty()) {
                    System.out.println("Memproses mahasiswa:");
                    data[front].print();
                    front = (front + 1) % max;
                    size--;
                    approvedCount++;
                } else {
                    System.out.println("Tidak ada lagi
mahasiswa dalam antrian.");
                    break;
                }
            }
        }
    }

    void printApprovedCount() {
        System.out.println("Total mahasiswa yang disetujui: " +
approvedCount);
    }

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

    void peek2() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
        } else {
            if (size == 1) {
                System.out.println("Data terdepan: ");
                data[front].print();
            } else {
                int temp = front;
                System.out.println("Data terdepan: ");
                data[front].print();

                temp = (temp + 1) % max;
                System.out.println("Data terdepan ke-2: ");
                data[temp].print();
            }
        }
    }
```

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

    public void clear() {
        if (isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Semua data berhasil dihapus");
        } else {
            System.out.println("Antrian sudah kosong!!");
        }
    }

    public void viewRear() {
        if (!isEmpty()) {
            System.out.println("Data terakhir: ");
            data[rear].print();
        } else {
            System.out.println("Antrian kosong!!");

        }
    }

    public void printRemainingStudent() {
        int remaining = max_students - approvedCount;
        System.out.println("Mahasiswa yang belum disetujui: "
+ remaining);
    }
}
```

2. StudentKRS

```
package P2jobsheet10;
public class StudentKRS {
    String nim, name, studyProgram, className;
    public StudentKRS(String nim, String name, String
studyProgram, String className){
        this.nim = nim;
        this.name = name;
        this.studyProgram = studyProgram;
        this.className = className;
    }
    void print(){
        System.out.println(nim+ " - " + name+ " - " +
studyProgram+ " - " +className );
    }
}
```

3. StudentKRSQueueMain

```
public class StudentKRSQueueMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        StudentKRSQueue queue = new StudentKRSQueue(10);
        int choice;
        do {
            System.out.println("\n=== Menu Layanan Akademik
===");
            System.out.println("1. Tambah Mahasiswa");
            System.out.println("2. Proses 2 Mahasiswa
(Persetujuan KRS)");
            System.out.println("3. Tampilkan Semua
Mahasiswa");
            System.out.println("4. Tampilkan Dua Mahasiswa
Pertama");
            System.out.println("5. Tampilkan Mahasiswa
Terakhir");
            System.out.println("6. Tampilkan Jumlah Total
Mahasiswa");
            System.out.println("7. Tampilkan jumlah
mahasiswa yang telah menyelesaikan proses persetujuan KRS");
            System.out.println("8. Tampilkan jumlah
mahasiswa yang belum menyelesaikan proses persetujuan
KRS.");
            System.out.println("0. Keluar");
            System.out.print("Pilih Menu: ");
            choice = sc.nextInt();
            sc.nextLine();
```

```
            switch (choice) {
                case 1:
                    System.out.println("NIM: ");
                    String nim = sc.nextLine();
                    System.out.println("Nama: ");
                    String name = sc.nextLine();
                    System.out.println("Program Studi: ");
                    String studyProgram = sc.nextLine();
                    System.out.println("Kelas: ");
                    String className = sc.nextLine();
                    StudentKRS std = new StudentKRS(nim,
name, studyProgram, className);
                    queue.enqueue(std);
                    break;
                case 2:
                    // Proses 2 mahasiswa (sesi persetujuan
pertama)

                    queue.dequeue2();

                    // Proses 2 mahasiswa lagi (sesi
persetujuan kedua)


                    break; // Keluar dari loop jika antrian
kosong
                case 3:
                    queue.print();
                    break;
                case 4:
                    queue.peek2();
                    break;
                case 5:
                    queue.viewRear();
                    break;
                case 6:
                    System.out.println("Ukuran antrian: " +
queue.size);
                    break;
                case 7:
                    queue.printApprovedCount();
                    break;
                case 8:
                    queue.printRemainingStudent();
                    break;
                case 0:
                    System.out.println("Terima kasih!!");
                    break;
                default:
                    System.out.println("Menu Tidak
Valid!!");
            }
        } while (choice != 0);
    }
}
```