



JOBSHEET X QUEUE

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mengenal struktur data Queue
2. Membuat dan mendeklarasikan struktur data Queue
3. Menerapkan algoritma Queue dengan menggunakan array

2. Praktikum 1

2.1 Percobaan 1 : Operasi Dasar Queue

Waktu percobaan : 90 menit

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

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

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

2.1.1. 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 :



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

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

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

```
public void clear() {
    if (!IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println("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.

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

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

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

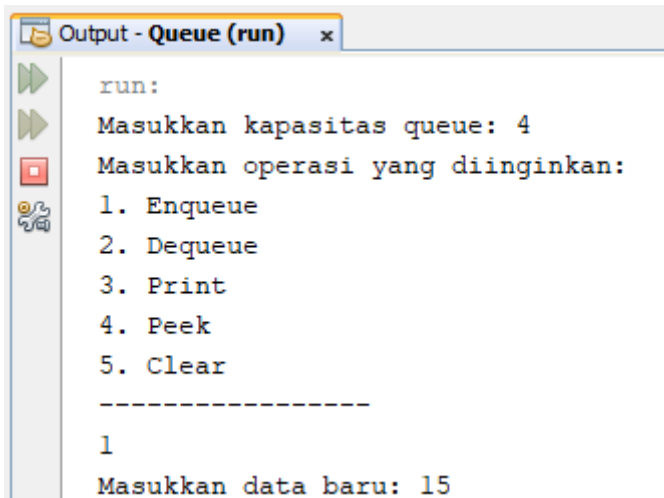
14. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
15. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case 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.

2.1.2. Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.



```
run:
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

2.1.3. Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
2. Pada method **Enqueue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

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

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

4. Pada method **print**, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (**int i=0**), melainkan **int i=front**?
5. Perhatikan kembali method **print**, jelaskan maksud dari potongan kode berikut!

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

6. Tunjukkan potongan kode program yang merupakan queue overflow!
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!

2.2. Percobaan 2 : Antrian Layanan Akademik

Waktu percobaan : 90 menit

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

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

2.2.1. Langkah-langkah Percobaan

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

1. Buat folder baru bernama **P2Jobsheet10** di dalam repository **Praktikum ASD**, kemudian buat class baru dengan nama **Mahasiswa**.
2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

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

Dan tambahkan method tampilkanData berikut :

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

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

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

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

4. Lakukan modifikasi pada class **AntrianLayanan** dengan mengubah tipe **int[] data** menjadi **Mahasiswa[] data** karena pada kasus ini data yang akan disimpan berupa object Mahasiswa. Modifikasi perlu dilakukan pada **atribut**, method **Enqueue**, dan method **Dequeue**.

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

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

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

```
public void lihatTerdepan() {
    if (isEmpty()) {
        System.out.println("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;
        System.out.print((i + 1) + ". ");
        data[index].tampilkanData();
    }
}
```

Ditambahkan dengan method `getJumlahAntrian` yaitu menampilkan nilai `size`

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

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

```
public class LayananAkademikSIKAD {

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

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

```
do {

    System.out.println("\n=== Menu Antrian Layanan Akademik ===");
    System.out.println("1. Tambah Mahasiswa ke Antrian");
    System.out.println("2. Layani Mahasiswa");
    System.out.println("3. Lihat Mahasiswa Terdepan");
    System.out.println("4. Lihat Semua Antrian");
    System.out.println("5. Jumlah Mahasiswa dalam Antrian");
    System.out.println("0. Keluar");
    System.out.print("Pilih menu: ");
    pilihan = sc.nextInt(); sc.nextLine();
```

```

        switch (pilihan) {
            case 1:
                System.out.print("NIM : ");
                String nim = sc.nextLine();
                System.out.print("Nama : ");
                String nama = sc.nextLine();
                System.out.print("Prodi : ");
                String prodi = sc.nextLine();
                System.out.print("Kelas : ");
                String kelas = sc.nextLine();
                Mahasiswa mhs = new Mahasiswa(nim, nama, prodi, kelas);
                antrian.tambahAntrian(mhs);
                break;
            case 2:
                Mahasiswa dilayani = antrian.layaniMahasiswa();
                if (dilayani != null) {
                    System.out.print("Melayani mahasiswa: ");
                    dilayani.tampilkanData();
                }
                break;
            case 3:
                antrian.lihatTerdepan();
                break;
            case 4:
                antrian.tampilkanSemua();
                break;
            case 5:
                System.out.println("Jumlah dalam antrian: " + antrian.getJumlahAntrian());
                break;
            case 0:
                System.out.println("Terima kasih.");
                break;
            default:
                System.out.println("Pilihan tidak valid.");
        }
    } while (pilihan != 0);

    sc.close();
}

```

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

2.2.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.

```

=== 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.
    
```

2.2.3 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 **LayananAkademikSIKAD** sehingga method **LihatAkhir** dapat dipanggil!

2.3 Tugas

Waktu : 120 Menit

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

- Cek antrian kosong, Cek antrian penuh, Mengosongkan antrian.
- Menambahkan antrian, Memanggil antrian untuk proses KRS – setiap 1x panggilan terdiri dari 2 mahasiswa (pada antrian no 1 dan 2)
- Menampilkan semua antrian, Menampilkan 2 antrian terdepan, Menampilkan antrian paling akhir.
- Cetak jumlah antrian, Cetak jumlah yang sudah melakukan proses KRS
- Jumlah antrian maksimal 10, jumlah yang ditangani masing-masing DPA 30 mahasiswa, cetak jumlah mahasiswa yang belum melakukan proses KRS.

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