



## JOBSHEET VIII QUEUE

### 8.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

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

### 8.2 Praktikum 1

**Waktu percobaan : 45 menit**

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Queue.

#### 8.2.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class Queue berikut ini:

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

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

2. Buat package dengan nama **Praktikum1**, kemudian buat class baru dengan nama **Queue**.
3. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti gambar 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;
}
```

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

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

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

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

6. 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");
    }
}
```

7. 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);
    }
}
```

8. 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");
    }
}
```

9. 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++;
    }
}
```

10. 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;
}

```

11. 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("-----");
}

```

12. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.
13. 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();

```

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

```

Queue Q = new Queue(n);

```

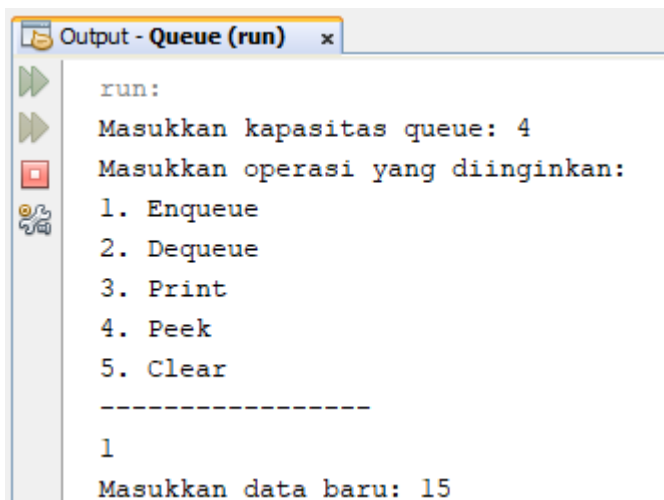
15. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
16. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case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

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

### 8.2.2 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

### 8.2.3 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!



## 8.3 Praktikum 2

**Waktu percobaan : 45 menit**

Pada percobaan ini, kita akan membuat program yang mengilustrasikan teller di bank dalam melayani nasabah.

### 8.3.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

| Nasabah  |
|--|
| norek: String<br>nama: String<br>alamat: String<br>umur: int<br>saldo: double  |
| Nasabah(norek: String, nama: String, alamat: String, umur: int, saldo: double) |

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

2. Buat package dengan nama **Praktikum2**, kemudian buat class baru dengan nama **Nasabah**.
3. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

```
Nasabah(String norek, String nama, String alamat, int umur, double saldo) {
    this.norek = norek;
    this.nama = nama;
    this.alamat = alamat;
    this.umur = umur;
    this.saldo = saldo;
}
```

4. Salin kode program class **Queue** pada **Praktikum 1** untuk digunakan kembali pada **Praktikum 2** ini. 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 **Queue** tersebut.
5. Lakukan modifikasi pada class **Queue** dengan mengubah tipe **int[] data** menjadi **Nasabah[] data** karena pada kasus ini data yang akan disimpan pada queue berupa object Nasabah. Modifikasi perlu dilakukan pada **atribut**, method **Enqueue**, dan method **Dequeue**.

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

```

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

public void Enqueue(Nasabah 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 Nasabah Dequeue() {
    Nasabah dt = new Nasabah();
    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;
}

```

Baris program **Nasabah dt = new Nasabah();** akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class Nasabah.





```
Nasabah () {  
  
}
```

6. Karena satu elemen queue terdiri dari beberapa informasi (norek, nama, alamat, umur, dan saldo), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga modifikasi perlu dilakukan pada method **peek** dan method **print**.

```
public void peek() {  
    if (!IsEmpty()) {  
        System.out.println("Elemen terdepan: " + data[front].norek + " " + data[front].nama  
            + " " + data[front].alamat + " " + data[front].umur + " " + data[front].saldo);  
    } 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.println(data[i].norek + " " + data[i].nama  
                + " " + data[i].alamat + " " + data[i].umur + " " + data[i].saldo);  
            i = (i + 1) % max;  
        }  
        System.out.println(data[i].norek + " " + data[i].nama  
            + " " + data[i].alamat + " " + data[i].umur + " " + data[i].saldo);  
        System.out.println("Jumlah elemen = " + size);  
    }  
}
```

7. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum2**. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna

```
public static void menu() {  
    System.out.println("Pilih menu: ");  
    System.out.println("1. Antrian baru");  
    System.out.println("2. Antrian keluar");  
    System.out.println("3. Cek Antrian terdepan");  
    System.out.println("4. Cek Semua Antrian");  
    System.out.println("-----");  
}
```

8. Buat fungsi **main**, deklarasikan Scanner dengan nama **sc**
9. Buat variabel **max** untuk menampung kapasitas elemen pada queue. Kemudian lakukan instansiasi objek queue dengan nama **antri** dan nilai parameternya adalah variabel **jumlah**.

```
System.out.print("Masukkan kapasitas queue: ");  
int jumlah = sc.nextInt();  
Queue antri = new Queue(jumlah);
```

10. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
11. Tambahkan kode berikut untuk melakukan perulangan menu sesuai dengan masukan yang diberikan oleh pengguna.

```
do {
    menu();
    pilih = sc.nextInt();
    sc.nextLine();
    switch (pilih) {
        case 1:
            System.out.print("No Rekening: ");
            String norek = sc.nextLine();
            System.out.print("Nama: ");
            String nama = sc.nextLine();
            System.out.print("Alamat: ");
            String alamat = sc.nextLine();
            System.out.print("Umur: ");
            int umur = sc.nextInt();
            System.out.print("Saldo: ");
            double saldo = sc.nextDouble();
            Nasabah nb = new Nasabah(norek, nama, alamat, umur, saldo);
            sc.nextLine();
            antri.Enqueue(nb);
            break;

        case 2:
            Nasabah data = antri.Dequeue();
            if (!"".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat)
                && data.umur != 0 && data.saldo != 0) {
                System.out.println("Antrian yang keluar: " + data.norek + " " + data.nama + " "
                    + data.alamat + " " + data.umur + " " + data.saldo);
                break;
            }

        case 3:
            antri.peek();
            break;

        case 4:
            antri.print();
            break;
    }
} while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4);
```

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

### 8.3.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.

```

Output - Queue (run) x
run:
Masukkan kapasitas queue: 8
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
1
No Rekening: 12345
Nama: Dewi
Alamat: Malang
Umur: 23
Saldo: 1300000
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
1
No Rekening: 32940
Nama: Susan
Alamat: Surabaya
Umur: 39
Saldo: 42000000
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
4
12345 Dewi Malang 23 1300000.0
32940 Susan Surabaya 39 4.2E7
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----

```



### 8.3.3 Pertanyaan

1. Pada class QueueMain, jelaskan fungsi IF pada potongan kode program berikut!

```
if (!"".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat)
    && data.umur != 0 && data.saldo != 0) {
    System.out.println("Antrian yang keluar: " + data.norek + " " + data.nama + " "
        + data.alamat + " " + data.umur + " " + data.saldo);
    break;
}
```

2. Lakukan modifikasi program dengan menambahkan method baru bernama **peekRear** pada class Queue yang digunakan untuk mengecek antrian yang berada di posisi belakang! Tambahkan pula daftar menu **5. Cek Antrian paling belakang** pada class **QueueMain** sehingga method **peekRear** dapat dipanggil!

### 8.4 Tugas

1. Buatlah program antrian untuk mengilustrasikan pesanan disebuah warung. Ketika seorang pembeli akan mengantri, maka dia harus mendaftarkan nama, dan nomor HP seperti yang digambarkan pada Class diagram berikut:

| Pembeli                          |
|----------------------------------|
| nama: String<br>noHP: int        |
| Pembeli(nama: String, noHP: int) |

Class diagram Queue digambarkan sebagai berikut:

| Queue   |
|---|
| antrian: Pembeli[]<br>front: int<br>rear: int<br>size: int<br>max: int  |
| Queue(n: int)<br>isEmpty(): boolean<br>isFull(): boolean<br>enqueue(antri: Pembeli): void<br>dequeue(): int<br>print(): void<br>peek(): void<br>peekRear(): void<br>peekPosition(nama: String): void<br>daftarPembeli(): void |

Keterangan:



- 
- Method `create()`, `isEmpty()`, `isFull()`, `enqueue()`, `dequeue()` dan `print()`, kegunaannya sama seperti yang telah dibuat pada Praktikum
  - Method `peek()`: digunakan untuk menampilkan data Pembeli yang berada di posisi antrian paling depan
  - Method `peekRear()`: digunakan untuk menampilkan data Pembeli yang berada di posisi antrian paling belakang
  - Method `peekPosition()`: digunakan untuk menampilkan seorang pembeli (berdasarkan nama) posisi antrian ke berapa
  - Method `daftarPembeli()`: digunakan untuk menampilkan data seluruh pembeli