

JOBSHEET VIII

QUEUE

8.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

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

8.2 Praktikum 1

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Queue.

8.2.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

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

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

2. Buat project baru dengan nama **Jobsheet8**, buat package dengan nama **Praktikum1**, kemudian buat class baru dengan nama **Queue**.
3. Tambahkan atribut max, size, front, rear, dan Q sesuai diagram class di atas.
4. Tambahkan pula konstruktor berparameter dan method **Create** seperti gambar berikut ini.

```

public Queue(int n) {
    max = n;
    Create();
}

```

Di dalam konstruktor, terdapat kode program yang digunakan untuk memanggil method **Create**.

```

public void Create() {
    Q = new int[max];
    size = 0;
    front = rear = -1;
}

```

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

```

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

```

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

```

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

```

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

```

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

```

8. 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(Q[i] + " ");
            i = (i + 1) % max;
        }
        System.out.println(Q[i] + " ");
        System.out.println("Jumlah elemen = " + size);
    }
}
}

```

9. 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");
    }
}
}

```

10. Buat method **Enqueue** bertipe void untuk menambahkan isi queue dengan parameter **data** yang bertipe integer

```

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

```

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

```

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

```

12. 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("-----");
}

```

13. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.
14. 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();

```

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

```

Queue Q = new Queue(n);

```

16. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.

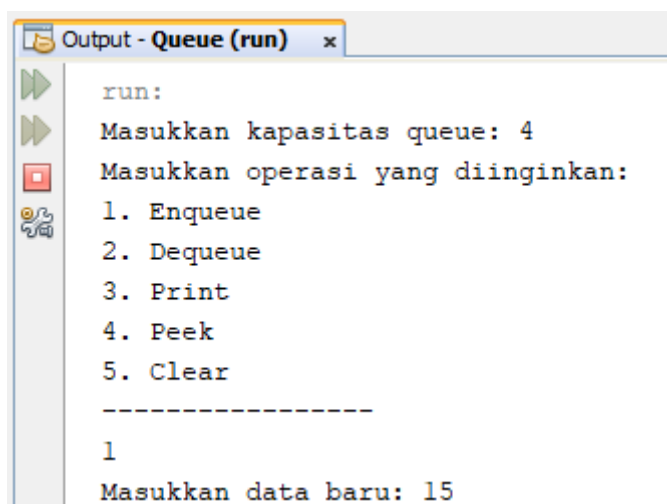
17. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case untuk menjalankan operasi queue sesuai dengan masukan pengguna.

```
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);
```

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

8.2.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.



```
Output - Queue (run) x
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 method **Create**, mengapa atribut front dan rear diinisialisasi dengan nilai -1, tidak 0?
2. Pada method **Enqueue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

3. Perhatikan kembali method **Enqueue**, baris kode program manakah yang menunjukkan bahwa data baru disimpan pada posisi terakhir di dalam queue?
4. Perhatikan kembali method **Dequeue**, baris kode program manakah yang menunjukkan bahwa data yang dikeluarkan adalah data pada posisi paling depan di dalam queue?
5. Pada method **Dequeue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

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

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

8.3 Praktikum 2

Pada percobaan ini, kita akan membuat program yang mengilustrasikan pembelian tiket kereta api yang dilakukan oleh penumpang di stasiun kereta api.

8.3.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

Penumpang
nama: String kotaAsal: String kotaTujuan: String jumlahTiket: int harga: int
Penumpang(nama: String, kotaAsal: String, kotaTujuan: String, jml: int, harga: int)

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

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

```
String nama, kotaAsal, kotaTujuan;  
int jumlahTiket, harga;  
  
Penumpang(String nama, String kotaAsal, String kotaTujuan, int jml, int harga) {  
    this.nama = nama;  
    this.kotaAsal = kotaAsal;  
    this.kotaTujuan = kotaTujuan;  
    jumlahTiket = jml;  
    this.harga = harga;  
}
```

4. Salin kode program class **Queue** pada **Praktikum1** untuk digunakan kembali pada **Praktikum2** ini. Karena pada **Praktikum1**, data yang disimpan pada queue hanya berupa array bertipe integer, sedangkan pada **Praktikum2** data yang digunakan adalah object, maka perlu dilakukan modifikasi pada class **Queue** tersebut.
5. Lakukan modifikasi pada class **Queue** dengan mengubah tipe **int[] Q** menjadi **Penumpang[] Q** karena pada kasus ini data yang akan disimpan pada queue berupa object Penumpang. Modifikasi perlu dilakukan pada **atribut**, method **Create**, method **Enqueue**, dan method **Dequeue**.

```
int front;  
int rear;  
int size;  
int max;  
Penumpang[] Q;
```

```

public void Create() {
    Q = new Penumpang[max];
    size = 0;
    front = rear = -1;
}

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

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

```

6. Karena satu elemen queue terdiri dari beberapa informasi (nama, kota asal, kota tujuan, jumlah tiket, dan harga), 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: " + Q[front].nama + " " + Q[front].kotaAsal
            + " " + Q[front].kotaTujuan + " " + Q[front].jumlahTiket + " " + Q[front].harga);
    } 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(Q[i].nama + " " + Q[i].kotaAsal + " " + Q[i].kotaTujuan
                + " " + Q[i].jumlahTiket + " " + Q[i].harga);
            i = (i + 1) % max;
            System.out.println("");
        }
        System.out.println(Q[i] + " ");
        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();
}

```

```

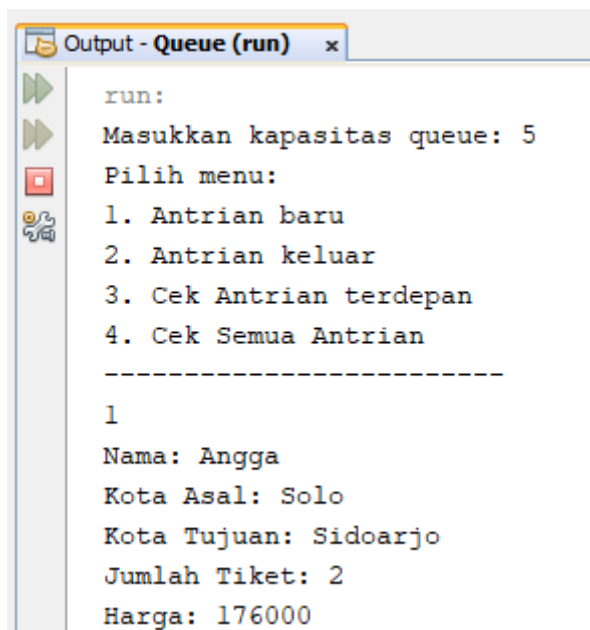
switch (pilih) {
    case 1:
        System.out.print("Nama: ");
        String nama = sc.nextLine();
        System.out.print("Kota Asal: ");
        String asal = sc.nextLine();
        System.out.print("Kota Tujuan: ");
        String tujuan = sc.nextLine();
        System.out.print("Jumlah Tiket: ");
        int jml = sc.nextInt();
        System.out.print("Harga: ");
        int hrg = sc.nextInt();
        Penumpang p = new Penumpang(nama, asal, tujuan, jml, hrg);
        sc.nextLine();
        antri.Enqueue(p);
        break;
    case 2:
        Penumpang data = antri.Dequeue();
        if (!"".equals(data.nama) && !"".equals(data.kotaAsal) && !"".equals(data.kotaTujuan)
            && data.jumlahTiket != 0 && data.harga != 0) {
            System.out.println("Antrian yang keluar: " + data.nama + " " + data.kotaAsal + " "
                + data.kotaTujuan + " " + data.jumlahTiket + " " + data.harga);
            break;
        }
    case 3:
        antri.peak();
        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.



```

run:
Masukkan kapasitas queue: 5
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
1
Nama: Angga
Kota Asal: Solo
Kota Tujuan: Sidoarjo
Jumlah Tiket: 2
Harga: 176000

```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
1
Nama: Fadin
Kota Asal: Banyuwangi
Kota Tujuan: Bandung
Jumlah Tiket: 1
Harga: 65000
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
3
Elemen terdepan: Angga Solo Sidoarjo 2 176000
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
-----
2
```

8.3.3 Pertanyaan

1. Perhatikan class Queue, apa fungsi kode program berikut pada method Dequeue?

```
Penumpang data = new Penumpang("", "", "", 0, 0);
```

2. Pada soal nomor 1, apabila kode program tersebut diganti dengan kode berikut:

```
Penumpang data = new Penumpang()
```

Apakah yang terjadi? Mengapa demikian?

3. Tunjukkan kode program yang digunakan untuk menampilkan data yang dikeluarkan dari queue!
4. 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. Tambahkan dua method berikut ke dalam class Queue pada **Praktikum1**:

- a. Method **peekPosition(data: int) : void**

Untuk menampilkan posisi dari sebuah data di dalam queue, misalnya dengan mengirimkan data tertentu, akan diketahui posisi (indeks) data tersebut berada di urutan keberapa

- b. Method **peekAt(position: int) : void**

Untuk menampilkan data yang berada pada posisi (indeks) tertentu

Sesuaikan daftar menu yang terdapat pada class **QueueMain** sehingga kedua method tersebut dapat dipanggil!

2. Buatlah program antrian untuk mengilustrasikan mahasiswa yang sedang meminta tanda tangan KRS pada dosen DPA di kampus. Ketika seorang mahasiswa akan mengantri, maka dia harus menuliskan terlebih dulu NIM, nama, absen, dan IPK seperti yang digambarkan pada Class diagram berikut:

Mahasiswa
nim: String nama: String absen: int ipk: double
Mahasiswa(nim: String, nama: String, absen: int, ipk: double)

Class diagram Queue digambarkan sebagai berikut:

Queue
max: int front: int rear: int size: int antrian: Mahasiswa[]
Queue(max: int) create(): void isEmpty(): boolean isFull(): boolean enqueue(antrian: Mahasiswa): void dequeue(): int print(): void peek(): void peekRear(): void peekPosition(nim: String): void printMahasiswa(posisi: int): void

Catatan:

- Method `create()`, `isEmpty()`, `isFull()`, `enqueue()`, `dequeue()` dan `print()`, kegunaannya sama seperti yang telah dibuat pada Praktikum
- Method `peek()`: digunakan untuk menampilkan data Mahasiswa yang berada di posisi antrian paling depan
- Method `peekRear()`: digunakan untuk menampilkan data Mahasiswa yang berada di posisi antrian paling belakang
- Method `peekPosition()`: digunakan untuk menampilkan posisi antrian ke berapa, seorang Mahasiswa berada. Pengecekan dilakukan berdasarkan NIM
- Method `printMahasiswa()`: digunakan untuk menampilkan data mahasiswa pada suatu posisi tertentu dalam antrian