Name : Azaria Cindy Sahasika
Number Id : 2341760169 / 06
Class : 1G – Business Information System
Lesson : Algorithm and Data Structure
Material : Jobsheet 10
Github Link : https://github.com/azariacindy/algorithm-ds

# JOBSHEET X
# QUEUE

## 8.1 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

## 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 behasil 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 paling depan

```
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.

```
Masukkan kapasitas queue : 6
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: 23

Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------
3
15
23
Jumlah elemen = 2

Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------
4
Elemen terdepan : 15
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------
2
Data yang dikeluarkan: 15

Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------
3
23
Jumlah elemen = 1
```

```
Enter the Queue capacity: 6
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
1
Masukkan data baru: 15
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
1
Masukkan data baru: 23
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
3
15 23
Number of elements = 2
```

```
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
4
Leading elements: 15
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
2
Data yang dikeluarkan15
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
3
23
Number of elements = 1
Enter the desired operation:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------
5
Queue successfully emptied
```

### 8.2.3 Pertanyaan

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

   ➔ Because initially the queue is empty, so there are no elements in front / behind. While the initial value of the size attribute is set to 0 at the beginning of an empty queue, there are no elements in the queue.

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

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

   ➔ To handle the situation when the 'rear' has reached the 'max-1' queue limit and needs to be repeated from the beginning of queue 0 if there is still free space in front.

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

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

➔ To handle situations when the 'front' has reached the 'max-1' queue limit and needs to be restarted from 0 if there are still elements behind.

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

   ➔ Since the first index of the queue contains the first element printed, the loop starts from 'front' since the queue structure is cyclic (last element), the first element is the next element. So, it starts from 'front' and goes around until it reaches 'rear'.

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

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

   ➔ To handle cyclic looping in the queue structure. If index 'i' returns to 0 after reaching 'max-1', thus creating a cyclic effect that allows to continue traversing the entire queue.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

```
public void Enqueue(int dt) {
    if (IsFull()) { // check if the queue is full
        System.out.println(x:"Queue is full!");
    } else {
        if (IsEmpty()) {
            front = rear = 0;
        } else {
            if (rear == max - 1) { // check if rear has reached the maximum limit
                rear = 0; // if yes, return rear to the beginning of the queue
            } else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}
```

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!

```
jobsheet8_Queue > pract106 > Queue06.java > Queue06 > Enqueue(int)
  3  public class Queue06 {
 52      public void clear() {
 62      public void Enqueue(int dt) {
 63          if (IsFull()) { // check if the queue is full
 64              System.out.println(x:"Queue is full! Program will be terminated."); // queue overflow
 65              System.exit(status:1);
 66          } else {
 67              if (IsEmpty()) {
 68                  front = rear = 0;
 69              } else {
 70                  if (rear == max - 1) { // check if rear has reached the maximum limit
 71                      rear = 0; // if yes, return rear to the beginning of the queue
 72                  } else {
 73                      rear++;
 74                  }
 75              }
 76              data[rear] = dt;
 77              size++;
 78          }
 79      }
 80
 81      public int Dequeue() {
 82          int dt = 0;
 83          if (IsEmpty()) {
 84              System.out.println(x:"Queue is still empty! Program will be terminated."); // queue underflow
 85              System.exit(status:1);
 86          } else {
 87              dt = data[front];
 88              size--;
 89              if (IsEmpty()) {
 90                  front = rear = -1;
 91              } else {
 92                  if (front == max - 1) {
 93                      front = 0;
 94                  } else {
 95                      front++;
 96                  }
 97              }
 98          }
 99          return dt;
100      }
```

## 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 meodifikasi 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.

```
Masukkan kapasitas queue : 4
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
---------------------------------
1
No rekening: 1200046675
Nama: Arif
Alamat: Sukun, Malang
Umur: 25
Saldo: 12000000
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
1
No rekening: 1200198733
Nama: Dewi
Alamat: Rungkut, Surabaya
Umur: 30
Saldo: 8600000
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
4
1200046675 Arif Sukun, Malang 25 1.2E7
1200198733 Dewi Rungkut, Surabaya 30 8600000.0
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
3
Elemen terdepan : 1200046675 Arif Sukun, Malang 25 1.2E7
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
2
Antrian yang keluar : 1200046675 Arif Sukun, Malang 25 1.2E7
```

```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
4
1200198733 Dewi Rungkut, Surabaya 30 8600000.0
Jumlah elemen = 1
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
-----------------------------------
```

```
Enter the Queue capacity: 4
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 1
----------------------------
No Rekening: 1200046675
Name: Arif
Address: Sukun, Malang
Age: 25
Saldo: 12000000
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 1
----------------------------
No Rekening: 1200198733
Name: Dewi
Address: Rungkut, Surabaya
Age: 30
Saldo: 8600000
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 4
1200046675 Arif Sukun, Malang 25 1.2E7
1200198733 Dewi Rungkut, Surabaya 30 8600000.0
Number of elements = 2
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 3
Leading elements: 1200046675 Arif Sukun, Malang 25 1.2E7
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 2
The outgoing queue: 1200046675 Arif Sukun, Malang 25 1.2E7
----------------------------
1. New queue
2. Outgoing queue
3. Front queue check
4. Check all queues
Select menu: 4
1200198733 Dewi Rungkut, Surabaya 30 8600000.0
Number of elements = 1
```

### 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;
}
```

➔ To check if the 'customer' data to be output from the queue has valid attribute values. The conditions ensure that all customer attributes (norek, name, address) are not empty, the age and balance of the customer are not the default value of 0. If all conditions are met, then the customer data will be printed as output from the queue.

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!

```
public void peekRear() {
    if(!IsEmpty()) {
        System.out.println("Trailing elements: " + data[rear].norek + " " + data[rear].nama + " " +
                           data[rear].alamat + " " + data[rear].umur + " " + data[rear].saldo);
    } else {
        System.out.println(x:"Queue is still empty! No trailing element available.");
    }
}
```

## 8.4 Tugas

1. Buatlah program antrian untuk mengilustrasikan antrian pasien di sebuah klinik. Ketika seorang pasien akan mengantri, maka dia harus mendaftarkan nama, nomor identitas, jenis kelamin dan umur seperti yang digambarkan pada Class diagram berikut:

| Pembeli |
|---|
| nama: String<br>noID: int<br>jenisKelamin: char<br>umur: int |
| Pasien (nama: String, noID: int, jenisKelamin: char, umur: int) |

Class diagram Queue digambarkan sebagai berikut:

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

Keterangan method:

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

jobsheet8_Queue > tugasQueue > Patient06.java > ...

Click here to ask Blackbox to help you code faster

```java
package tugasQueue;

public class Patient06 {
    String name;
    int idNumber;
    char gender;
    int age;

    public Patient06(String name, int idNumber, char gender, int age) {
        this.name = name;
        this.idNumber = idNumber;
        this.gender = gender;
        this.age = age;
    }

    public String toString() {
        return "Name: " + name + ", ID Number: " + idNumber + ", Gender: " + gender + ", Age: " + age;
    }
}
```

jobsheet8_Queue > tugasQueue > Queue06.java > ...

```java
public class Queue06 {
    int front, rear, size, max;

    public Queue06(int n) {
        max = n;
        queue = new Patient06[max];
        size = 0;
        front = rear = -1;
    }

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

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

    public void enqueue(Patient06 patient) {
        if (!isFull()) {
            if (isEmpty()) {
                front = rear = 0;
            } else {
                rear = (rear + 1) % max;
            }
            queue[rear] = patient;
            size++;
        } else {
            System.out.println(x:"Queue is full!");
        }
    }

    public int dequeue() {
        if (!isEmpty()) {
            int position = front;
            if (front == rear) {
                front = rear = -1;
            } else {
                front = (front + 1) % max;
            }
            size--;
            return position;
        } else {
            System.out.println(x:"Queue is empty!");
            return -1;
        }
    }
```

```java
public void listPatients() {
    if (!isEmpty()) {
        System.out.println(x:"List of Patients:");
        int i = front;
        while (i != rear) {
            System.out.println(queue[i].toString());
            i = (i + 1) % max;
        }
        System.out.println(queue[i].toString());
    } else {
        System.out.println(x:"Queue is empty!");
    }
}
```

jobsheet8_Queue > tugasQueue > Queue06.java > ...

```
3   public class Queue06 {
52      public void print() {
53          if (!isEmpty()) {
54              int i = front;
55              while (i != rear) {
56                  System.out.println(queue[i].toString());
57                  i = (i + 1) % max;
58              }
59              System.out.println(queue[i].toString());
60              System.out.println("Number of Patients: " + size);
61          } else {
62              System.out.println(x:"Queue is empty!");
63          }
64      }
65
66      public void peek() {
67          if (!isEmpty()) {
68              System.out.println("Patient at the front of the queue: " + queue[front].toString());
69          } else {
70              System.out.println(x:"Queue is empty!");
71          }
72      }
73
74      public void peekRear() {
75          if (!isEmpty()) {
76              System.out.println("Patient at the rear of the queue: " + queue[rear].toString());
77          } else {
78              System.out.println(x:"Queue is empty!");
79          }
80      }
81
82      public void peekPosition(String name) {
83          if (!isEmpty()) {
84              for (int i = front; i != rear; i = (i + 1) % max) {
85                  if (queue[i].name.equals(name)) {
86                      System.out.println("Patient " + name + " is at position " + (i - front + 1) + " in the queue");
87                      return;
88                  }
89              }
90              if (queue[rear].name.equals(name)) {
91                  System.out.println("Patient " + name + " is at position " + (rear - front + 1) + " in the queue");
92              } else {
93                  System.out.println("Patient " + name + " is not found in the queue!");
94              }
95          } else {
96              System.out.println(x:"Queue is empty!");
97          }
98      }
```

jobsheet8_Queue > tugasQueue > Clinic06.java > ...

Click here to ask Blackbox to help you code faster

```java
package tugasQueue;
import java.util.Scanner;

public class Clinic06 {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter the maximum capacity of the queue: ");
        int capacity = scanner.nextInt();
        scanner.nextLine();

        Queue06 clinicQueue = new Queue06(capacity);

        int choice;
        do {
            System.out.println(x:"\nMenu:");
            System.out.println(x:"1. Enqueue");
            System.out.println(x:"2. Dequeue");
            System.out.println(x:"3. Print queue");
            System.out.println(x:"4. Peek");
            System.out.println(x:"5. Peek Rear");
            System.out.println(x:"6. Peek Position");
            System.out.println(x:"7. List Patients");
            System.out.println(x:"0. Exit");
            System.out.print(s:"Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();
```

jobsheet8_Queue > tugasQueue > Clinic06.java > ...

```
4   public class Clinic06 {
5       public static void main(String[] args) {
29              switch (choice) {
30                  case 1:
31                      System.out.print(s:"Enter patient's name: ");
32                      String name = scanner.nextLine();
33                      System.out.print(s:"Enter patient's ID number: ");
34                      int idNumber = scanner.nextInt();
35                      scanner.nextLine();
36                      System.out.print(s:"Enter patient's gender (M/F): ");
37                      char gender = scanner.nextLine().charAt(index:0);
38                      System.out.print(s:"Enter patient's age: ");
39                      int age = scanner.nextInt();
40                      scanner.nextLine();
41                      clinicQueue.enqueue(new Patient06 (name, idNumber, gender, age));
42                      break;
43                  case 2:
44                      int dequeuedPosition = clinicQueue.dequeue();
45                      if (dequeuedPosition != -1) {
46                          System.out.println("Patient dequeued from position " + (dequeuedPosition + 1));
47                      }
48                      break;
49                  case 3:
50                      clinicQueue.print();
51                      break;
52                  case 4:
53                      clinicQueue.peek();
54                      break;
55                  case 5:
56                      clinicQueue.peekRear();
57                      break;
58                  case 6:
59                      System.out.print(s:"Enter patient's name: ");
60                      String patientName = scanner.nextLine();
61                      clinicQueue.peekPosition(patientName);
62                      break;
63                  case 7:
64                      clinicQueue.listPatients();
65                      break;
66                  case 0:
67                      System.out.println(x:"Exiting program...");
68                      break;
69                  default:
70                      System.out.println(x:"Invalid choice! Please enter a valid option.");
71                      break;
72              }
73          } while (choice != 0);
74          scanner.close();
```

```
Enter the maximum capacity of the queue: 4

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 1
Enter patient's name: Azaria
Enter patient's ID number: 12345
Enter patient's gender (M/F): F
Enter patient's age: 16

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 1
Enter patient's name: Cindy
Enter patient's ID number: 12346
Enter patient's gender (M/F): F
Enter patient's age: 17

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 1
Enter patient's name: Sahasika
Enter patient's ID number: 12347
Enter patient's gender (M/F): M
Enter patient's age: 19

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 1
Enter patient's name: Budi
Enter patient's ID number: 11242
Enter patient's gender (M/F): M
Enter patient's age: 20

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 3
Name: Azaria, ID Number: 12345, Gender: F, Age: 16
Name: Cindy, ID Number: 12346, Gender: F, Age: 17
Name: Sahasika, ID Number: 12347, Gender: M, Age: 19
Name: Budi, ID Number: 11242, Gender: M, Age: 20
Number of Patients: 4

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 4
Patient at the front of the queue: Name: Azaria, ID Number: 12345, Gender: F, Age: 16

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 5
Patient at the rear of the queue: Name: Budi, ID Number: 11242, Gender: M, Age: 20
```

```
Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 6
Enter patient's name: Cindy
Patient Cindy is at position 2 in the queue

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 7
List of Patients:
Name: Azaria, ID Number: 12345, Gender: F, Age: 16
Name: Cindy, ID Number: 12346, Gender: F, Age: 17
Name: Sahasika, ID Number: 12347, Gender: M, Age: 19
Name: Budi, ID Number: 11242, Gender: M, Age: 20

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 2
Patient dequeued from position 1
```

```
Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 2
Patient dequeued from position 2

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 3
Name: Sahasika, ID Number: 12347, Gender: M, Age: 19
Name: Budi, ID Number: 11242, Gender: M, Age: 20
Number of Patients: 2

Menu:
1. Enqueue
2. Dequeue
3. Print queue
4. Peek
5. Peek Rear
6. Peek Position
7. List Patients
0. Exit
Enter your choice: 0
Exiting program...
```