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:

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.
4. Buat method IsEmpty bertipe boolean yang digunakan untuk mengecek apakah queue kosong.
5. Buat method IsFull bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.
6. Buat method peek bertipe void untuk menampilkan elemen queue pada posisi paling depan.
7. Buat method print bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.
8. Buat method clear bertipe void untuk menghapus semua elemen pada queue
9. Buat method Enqueue bertipe void untuk menambahkan isi queue dengan parameter dt yang bertipe integer
10. Buat method Dequeue bertipe int untuk mengeluarkan data pada queue di posisi belakang
11. Selanjutnya, buat class baru dengan nama QueueMain tetap pada package Praktikum1. Buat method menu bertipe void untuk memilih menu program pada saat dijalankan.
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.
14. Lakukan instansiasi objek Queue dengan nama Q dengan mengirimkan parameter n sebagai kapasitas elemen queue
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
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.

Class Queue

```
package praktikum1;

/**
 *
 * @author Hp
 */
public class Queue {
    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;
    }

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


    public void peek(){
        if (!IsEmpty()){
            System.out.println("Elemen terdepan :" + data[front]);
        } 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] + "");
            System.out.println("Jumlah elemen = " +size);
        }
    }
    }

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


    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++;
    }
    }


    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;
    }
}
```

QueueMain

```
package praktikum1;

import java.util.Scanner;

/**
 *
 * @author Hp
 */
public class QueueMain {
    public static void menu(){
        System.out.println("Masukkan operasi yang akan 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("------------");
    }

    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner (System.in);
        System.out.print("Maukkan Kapasitas queue :");
        int n = sc.nextInt();
        Queue Q = new Queue(n);



        int pilih;
        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);
    }
}
```

Output

```
: Output - Praktikum1 (run)
run:
Maukkan Kapasitas queue :4
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
------------
1
Masukkan data baru: 15
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
------------
1
Masukkan data baru: 31
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
------------
4
Elemen terdepan :15
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
------------
```

8.2.3 Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
   Pada konstruktor, nilai awal atribut front dan rear bernilai -1, Karena jika bernilai 0 maka pada queue akan langsung terpanggil, sehingga diinisialisasikan dengan -1 agar tidak memanggil indeks manapun.

2. Pada method Enqueue, jelaskan maksud dan kegunaan dari potongan kode berikut!
   ```
   if (rear == max - 1) {
       rear = 0;
   ```
   kegunaan dari potongan kode berikut adalah sebuah kondisi dari enqueue yang mana ketika data rear berada pada indeks terakhir, maka rear selanjutnya berada di indeks 0.

3. Pada method Dequeue, jelaskan maksud dan kegunaan dari potongan kode berikut!
   ```
   if (front == max - 1) {
       front = 0;
   ```
   kegunaan dari potongan kode berikut adalah sebuah kondisi dari dequeue yang mana ketika data front berada pada indeks terakhir, maka front selanjutnya berada di indeks 0.

4. Pada method print, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (int i=0), melainkan int i=front?
   Karena indeks ke-0 tidak selalu menjadi front, dan yang baru masuk bisa jadi tidak terletak pada indeks ke-0, karena pada queue ini yang lebih dulu masuk akan lebih dulu keluar, dan mendahulukan posisi front.

5. Perhatikan kembali method print, jelaskan maksud dari potongan kode berikut!
   ```
   i = (i + 1) % max;
   ```
   Maksud dari kode diatas adalah jika i+1 tidak = max maka I akan tetap (i+1), dan jika sudah sampai angka (i+1) nilainya = max, maka nilai tersebut di% dengan nilai max yang menghasilkan 0 dan akan dikembalikan ke 0

6. Tunjukkan potongan kode program yang merupakan queue overflow!
   ```
   if (IsFull ()){
       System.out.println("Queue sudah penuh");
   ```

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(0);
}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(0);
    }else {
        dt = data[front];
        size--;
        if (IsEmpty()){
            front = rear = -1;
        }else {
            if (front == max -1){
                front = 0;
            } else {
                front++;
            }
        }
    }
    return dt;
}
}
```

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

nama: String

alamat: String

umur: int

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.
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.
   Baris program Nasabah dt = new Nasabah(); akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class 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.
7. Selanjutnya, buat class baru dengan nama QueueMain tetap pada package Praktikum2. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna
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.
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.
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.

Class Nasabah

```
package praktikum2;

/**
 *
 * @author Hp
 */
public class Nasabah {
    String norek;
    String nama;
    String alamat;
    int umur;
    double saldo;

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

```
    public Nasabah (int n){
        max = n;
        data = new Nasabah [max];
        size = 0;
        front = rear = -1;
    }
     Nasabah(){
     }
    public boolean IsEmpty(){
        if (size ==0){
            return true;
        } else {
            return false;
        }
    }

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

```
    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);
        }
    }
    }
```

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

    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;
    }

}
```

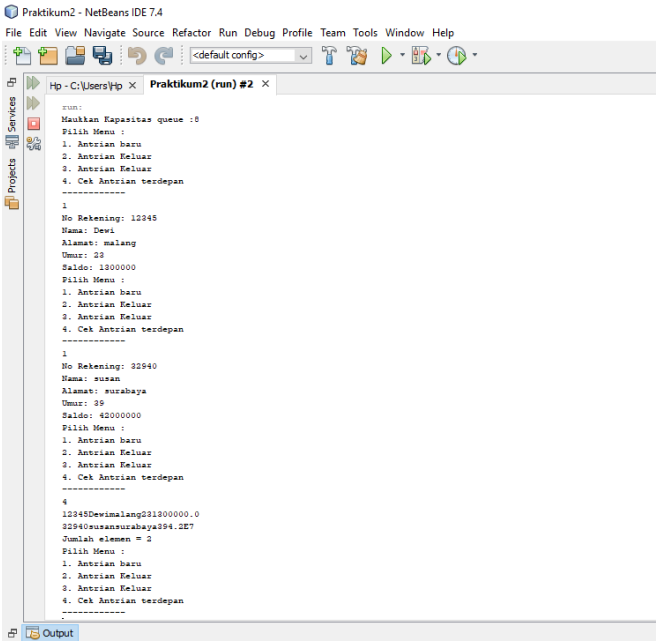
NasabahMain

```java
package praktikum2;

import java.util.Scanner;

/**
 *
 * @author Hp
 */
public class NasabahMain {
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. Antrian Keluar");
        System.out.println("4. Cek Antrian terdepan");
        System.out.println("------------");
    }
    public static void main(String[] args) {
        // TODO code application logic here
        int pilih;
        Scanner sc = new Scanner (System.in);
        System.out.print("Maukkan Kapasitas queue :");
        int jumlah = sc.nextInt();
        Nasabah antri = new Nasabah(jumlah);
```

```java
        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;
                    }
```

```java
                case 3:
                    antri.peek();
                    break;
                case 4:
                    antri.print();
                    break;

            }
        }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4|| pilih == 5);

    }

}
```

Output



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;
}
```

fungsi IF pada potongan kode program di atas berfungsi untuk mengecek apakah queue sudah terisi dan tidak kosong.

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!

Class Nasabah

```
package praktikum2;

/**
 *
 * @author Hp
 */
public class Nasabah {
    String norek;
    String nama;
    String alamat;
    int umur;
    double saldo;

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

```
    public Nasabah (int n){
        max = n;
        data = new Nasabah [max];
        size = 0;
        front = rear = -1;
    }
     Nasabah(){
     }
    public boolean IsEmpty(){
        if (size ==0){
            return true;
        } else {
            return false;
        }
    }

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

```
    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);
        }
    }
    }
```

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

    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;
    }
```

```
    public void peekRear(){
        if(!IsEmpty()){
            System.out.println("Elemen paling belakang : " + data[rear].nama + " " + data[rear].norek + " " + data[rear].nama +
                    " " + data[rear].alamat + " " + data[rear].umur + " " + data[rear].saldo);
        }else {
            System.out.println("Queue masih kosong");
        }
    }

}
```

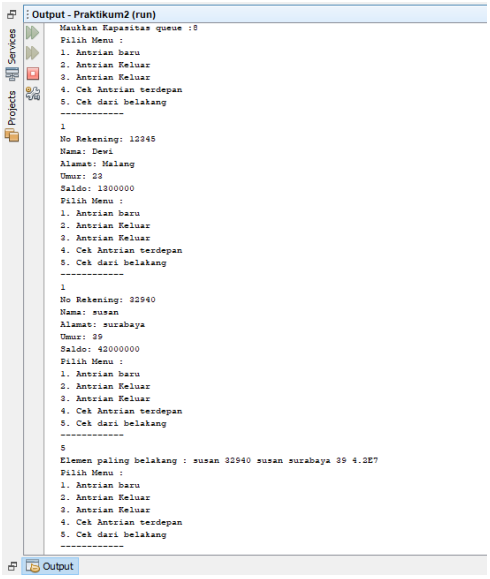
Nasabah Main

```
package praktikum2;

import java.util.Scanner;

/**
 *
 * @author Hp
 */
public class NasabahMain {
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. Antrian Keluar");
        System.out.println("4. Cek Antrian terdepan");
        System.out.println("5. Cek dari belakang");
        System.out.println("------------");
    }
    public static void main(String[] args) {
        // TODO code application logic here
        int pilih;
        Scanner sc = new Scanner (System.in);
        System.out.print("Maukkan Kapasitas queue :");
        int jumlah = sc.nextInt();
        Nasabah antri = new Nasabah(jumlah);
```

```
        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;

                case 5:
                    antri.peekRear();
                    break;

            }
        }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4|| pilih == 5);

    }

}
```

Output



8.4 Tugas

1. Tambahkan dua method berikut ke dalam class Queue pada Praktikum 1:

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 ke berapa

```
package tugas1;

/**
 *
 * @author Hp
 */
public class Queue1 {
     int[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue1 (int n){
        max = n;
        data = new int [max];
        size = 0;
        front = rear = -1;
    }

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

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

public void peek(){
    if (!IsEmpty()){
        System.out.println("Elemen terdepan :" + data[front]);
    } 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] + "");
        System.out.println("Jumlah elemen = " +size);
    }
}
}
```

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

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++;
}
}
```

```
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;
}
```

```
public void peekPosition(int n) {
           if(! IsEmpty()){
        for(int i = 0; i< data.length; i++){
            if(n == data[i]){
                System.out.println("Elemen : " + data[i] + " = indeks ke-" + i);
            }
        }
    }else{
        System.out.println("Queue masih kosong");
    }
}
```

b) Method peekAt(position: int) : void

c) Untuk menampilkan data yang berada pada posisi (indeks) tertentu

```
    public void peekAt(int n) {
        if(IsEmpty()){
            for(int i = 0; i< data.length; i++){
                if(n == i){
                System.out.println("Indeks : " + i + " = data ke-" + data[i]);
                }
            }
        }else{
            System.out.println("Queue masih kosong");
        }
    }


}
```

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

```
Output
Tugas1 (run) #3 × Tugas1 (run) #5 ×
run:
Maukkan Kapasitas queue :4
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
1
Masukkan data baru: 67
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
1
Masukkan data baru: 43
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
1
```

```
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
1
Masukkan data baru: 76
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
6
Masukkan data yang ingin dicari : 76
Elemen : 76 = indeks ke-2
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
7
Masukkan indeks ke : 4
Queue masih kosong
Masukkan operasi yang akan diinginkan :
1. Enqueue
2. Dequeue
3. print
4. peek
5. clear
6. peek Position
7. peek At
------------
```

2. Buatlah program antrian untuk mengilustasikan 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:

Keterangan:

- 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

Class Mahasiswa

```
package tugas2;

public class Mahasiswa {
    String nim;
    String nama;
    int absen;
    double ipk;

    Mahasiswa(String nim, String nama, int absen, double ipk){
        this.nim = nim;
        this.nama = nama;
        this.absen = absen;
        this.ipk = ipk;
    }
    Mahasiswa[] antrian;
    int front;
    int rear;
    int size;
    int max;


    public Mahasiswa (int n){
        max = n;
        antrian = new Mahasiswa [max];
        size = 0;
        front = rear = -1;
    }


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

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

    public void peek(){
        if (!IsEmpty()){
            System.out.println("Elemen terdepan :" + antrian[front].nim + " " + antrian[front].nama + " " +
                    antrian[front].absen + " " +antrian[front].ipk);
        } 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(antrian[i].nim + " " + antrian[i].nama + " " +
                    antrian[i].absen + " " + antrian[i].ipk);
            i = (i + 1)% max;
            System.out.println(antrian[i].nim + " " + antrian[i].nama + " " + antrian[i].absen + " " + antrian[i].ipk);
            System.out.println("Jumlah elemen = " +size);
        }
    }
    }

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


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


    public Mahasiswa Dequeue(){
        Mahasiswa dt = new Mahasiswa();
        if (IsEmpty()){
        System.out.println("Queue masih kosong");
        }else {
            dt = antrian[front];
            size--;
            if (IsEmpty()){
                front = rear = -1;
            }else {
                if (front == max -1){
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return dt;
    }
    public void peekRear(){
        if(!IsEmpty()){
            System.out.println("Elemen paling belakang : " + antrian[rear].nim + " " + antrian[rear].nama +
                    " " + antrian[rear].absen + " " + antrian[rear].ipk);
        }else {
            System.out.println("Queue masih kosong");
        }
    }


    public void peekPosition(String data){
         if(!IsEmpty()){
             for (int i = 0; i < (front+size); i++){
                 if(antrian[i].nim.equals(data)){
                     System.out.println("Mahasiswa dengan NIM " + data + "berada pada indeks ke-" + i);
                     break;
                 }
             }
         }else{
             System.out.println("Queue masih kosong");
         }
     }
    public void printMahasiswa(int posisi){
         if(!IsEmpty()){
             for(int i = 0; i < antrian.length; i++){
                 if(posisi == i){
                     System.out.println(antrian[i].nim + " " + antrian[i].nama + " " + antrian[i].absen + " " + antrian[i].ipk);
                 }
             }
         }else{
             System.out.println("Queue masih kosong");
         }
     }
}
```

Class Main QueueMahasiswa

```
package tugas2;

import java.util.Scanner;

/**
 *
 * @author Hp
 */
public class QueueMahasiswa {
    public static void menu(){
        System.out.println("Masukkan operasi yang akan diinginkan : ");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Tampilkan Data Mahasiswa");
        System.out.println("4. Tampilkan Data Mahasiswa Paling Depan");
        System.out.println("5. Tampilkan Data Mahasiswa Paling Belakang");
        System.out.println("6. Tampilkan Posisi Anntrian Data Mahasiswa");
        System.out.println("7. Tampilkan Data Mahasiswa Dalam Antrian");
        System.out.println("------------");
    }
    public static void main(String[] args) {
        // TODO code application logic here
         int pilih;
        Scanner sc = new Scanner (System.in);
        System.out.print("Maukkan Kapasitas queue :");
        int jumlah = sc.nextInt();
        Mahasiswa antri = new Mahasiswa(jumlah);


        do{
            menu();
            pilih = sc.nextInt();
            sc.nextLine();
            switch(pilih) {
                case 1:
                    System.out.print("NIM Mahasiswa: ");
                    String nim = sc.nextLine();
                    System.out.print("Nama Mahasiswa: ");
                    String nama = sc.nextLine();
                    System.out.print("Absen Mahasiswa: ");
                    int absen = sc.nextInt();
                    System.out.print("IPK Mahasiswa: ");
                    double ipk = sc.nextDouble();
                    Mahasiswa nb = new Mahasiswa (nim, nama, absen, ipk);
                    sc.nextLine();
                    antri.Enqueue(nb);
                    break;
                case 2:
                    Mahasiswa data = antri.Dequeue();
                    if(!"".equals(data.nim) && !"".equals(data.nama) && !"".equals(data.absen)
                            && data.ipk != 0) {
                        System.out.println("Antrian yang keluar: " + data.nim + " " +  data.nama
                            + " " + data.absen + " " + data.ipk);
                        break;
                    }

                case 3:
                    antri.print();
                    break;
                case 4:
                    antri.peek();
                    break;
                case 5:
                    antri.peekRear();
                    break;
                case 6:
                    System.out.print("Masukkan NIM Mahasiswa yang ingin dicari : ");
                    String Data = sc.nextLine();
                    antri.peekPosition(Data);
                    break;
                case 7:
                    System.out.print("Masukkan Posisi yang ingin dicari : ");
                    int Posisi = sc.nextInt();
                    antri.printMahasiswa(Posisi);
                    break;

            }
        }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4|| pilih == 5|| pilih == 6|| pilih == 7);
    }

}
```

Output