Nama : Muhammad Nazril Nur Rahman
Kelas : SIB –1G

NIM : 2341760174

https://github.com/nazrilnr/JobSMT2

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

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

2. 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
```

```java
package Praktikum1;
import java.util.Scanner;

public class QueueMain {
    public static void menu(){
        System.out.println (x:"Masukkan operasi yang diinginkan:");
        System.out.println (x:"1. Enqueue") ;
        System.out.println (x:"2. Dequeue") ;
        System.out.println (x:"3. Print");
        System.out.println (x:"4. Peek");
        System.out.println (x:"5. Clear");
        System.out.println(x:"---------------------------------");
    }

    Run | Debug
    public static void main(String[] args) {
        Scanner naz = new Scanner (System.in);
        System.out.print(s:"Masukkan kapasitas queue: ");
        int n = naz.nextInt();
        Queue Q = new Queue(n);
        int pilih;

        do {
            menu();
            pilih = naz.nextInt();
            switch (pilih){
                case 1 :
                System.out.println(x:"Masukkan data baru: ");
                int dataMasuk = naz.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();
            }
        }
        while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 || pilih == 5);
    }
}
```

```java
package Praktikum1;

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 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(x:"Queue masih kosong");
        }
    }

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

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

    public void Enqueue(int dt){
        if(IsFull()){
            System.out.println(x:"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(x:"Queue maasih kosong");
        } else {
            dt = data[front];
            size--;
            if (IsEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return dt;
    }
}
```

```
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
Masukkan operasi yang 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?

Tentu, inisialisasi atribut front dan rear dengan nilai -1 menunjukkan bahwa antrian kosong pada awalnya, sedangkan inisialisasi atribut size dengan nilai 0 menunjukkan bahwa belum ada elemen yang ada dalam antrian pada saat penciptaan objek.

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

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

Potongan kode tersebut bertujuan untuk menangani situasi di mana rear telah mencapai indeks maksimum dari array yang menyimpan elemen antrian. Jika ini terjadi, rear diatur kembali ke indeks awal array (0) sehingga antrian dapat melingkari array dan terus menerima elemen baru. Dengan

cara ini, array bisa digunakan secara efektif tanpa harus membuang elemen yang sudah ada ketika mencapai kapasitas maksimum.

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

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

ketika front telah mencapai indeks maksimum dari array yang menyimpan elemen antrian. Ketika ini terjadi, front diatur kembali ke indeks awal array (0) sehingga elemen pertama dari antrian dapat dihapus, dan antrian dapat melanjutkan operasinya dengan melingkari array dan tetap kontinu.

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

perulangan dari variabel front memungkinkan kita untuk mencetak elemen antrian secara berurutan, mengikuti urutan di mana mereka dimasukkan. Ini memastikan bahwa kita mencetak elemen-elemen antrian dalam urutan yang benar, dimulai dari elemen pertama, tanpa perlu memperhatikan elemen-elemen yang mungkin telah dihapus sebelumnya.

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

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

Potongan kode i = (i + 1) % max; digunakan untuk memperbarui nilai variabel i dalam perulangan while sehingga kita dapat melintasi array antrian secara melingkar. Ini memastikan bahwa setelah mencapai akhir array, perulangan akan kembali ke awal array untuk melanjutkan pencetakan elemen antrian dengan urutan yang benar.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

```
public void Enqueue(int dt){
    if(IsFull()){
        System.out.println(x:"Queue sudah penuh");
    } else {
        if (IsEmpty()){
            front = rear = 0;
        } else {
            if (rear == max - 1){
                rear = 0;
            } 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!

```
public void Enqueue(int dt){
    if(IsFull()){
        System.out.println(x:"Queue sudah penuh. Program dihentikan.");
        System.exit(status: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(x:"Queue masih kosong. Program dihentikan.");
        System.exit(status: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.

```
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
-----------------------------------
```

```java
package Praktikum2;

import java.util.Scanner;

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

        System.out.print(s:"Masukan kapasitas queue: ");
        int Jumlah = sc.nextInt();
        QueueN antri = new QueueN(Jumlah);
        sc.nextLine();

        int pilih;

        do {
            menu();
            pilih = sc.nextInt();
            sc.nextLine();
            switch (pilih) {
                case 1:
                    System.out.print(s:"No Rekening: ");
                    String norek = sc.nextLine();
                    System.out.print(s:"Nama: ");
                    String nama = sc.nextLine();
                    System.out.print(s:"Alamat: ");
                    String alamat = sc.nextLine();
                    System.out.print(s:"Umur: ");
                    int umur = sc.nextInt();
                    System.out.print(s:"saldo: ");
                    double saldo = sc.nextDouble();
                    Nasabah nb = new Nasabah(nama, norek, alamat, umur, saldo);
                    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 != 6); // Perubahan di sini

        sc.close();
    }

    public static void menu() {
        System.out.println(x:"Pilih menu: ");
        System.out.println(x:"1. Antrian baru");
        System.out.println(x:"2. Antrian Keluar");
        System.out.println(x:"3. Cek Antrian terdepan");
        System.out.println(x:"4. Cek Semua antrian");
        System.out.println(x:"5. Cek Antrian paling belakang");
        System.out.println(x:"-------------------------");
    }
}
```

```java
package Praktikum2;

public class Nasabah {
    String norek;
    String nama;
    String alamat;
    int umur;
    double saldo;

    Nasabah(){

    }

    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;
    }
}
```

```java
package Praktikum2;

public class QueueN {
    Nasabah[] data;
    int front;
    int rear;
    int size;
    int max;

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

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

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

    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(x:"Queue masih kosong");
        }
    }

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

    public void print() {
        if (isEmpty()) {
            System.out.println(x:"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);
        }
    }
```

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

    public void enqueue(Nasabah dt) {
        if (isFull()) {
            System.out.println(x:"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(x:"Queue masih kosong");
        } else {
            dt = data[front];
            size--;
            if (isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1){
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return dt;
    }
}
```

```
Masukan kapasitas queue: 4
Pilih menu:
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian terdepan
4. Cek Semua antrian
5. Cek Antrian paling belakang
-------------------------
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
5. Cek Antrian paling belakang
-------------------------
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
5. Cek Antrian paling belakang
-------------------------
4
Arif 1200046675 Sukun, Malang 25 1.2E7
Dewi 1200198733 Rungkut, Surabaya 30 8600000.0
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian terdepan
4. Cek Semua antrian
5. Cek Antrian paling belakang
-------------------------
3
Elemen terdepan: Arif 1200046675 Sukun, Malang 25 1.2E7
Pilih menu:
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian terdepan
4. Cek Semua antrian
5. Cek Antrian paling belakang
-------------------------
2
Antrian yang keluar: Arif 1200046675 Sukun, Malang 25 1.2E7
Pilih menu:
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian terdepan
4. Cek Semua antrian
5. Cek Antrian paling belakang
-------------------------
4
Dewi 1200198733 Rungkut, Surabaya 30 8600000.0
Jumlah elemen = 1
Pilih menu:
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian terdepan
4. Cek Semua antrian
5. Cek Antrian paling belakang
-------------------------
```

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;
}
```

Pernyataan if digunakan untuk memeriksa apakah data yang akan diproses memenuhi semua kondisi yang telah ditetapkan sebelumnya. Jika semua kondisi terpenuhi, maka informasi tentang data tersebut akan ditampilkan, dan iterasi loop akan dihentikan dengan menggunakan 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!

```
case 5:
    antri.peekRear();
    break;
```

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

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 noID: int jenisKelamin: char umur: int
Pasien (nama: String, noID: int, jenisKelamin: char, umur: int)

Class diagram Queue digambarkan sebagai berikut:

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

```
package Praktikum3;

public class Costumer16 {
    String nama;
    int noHP;

    Costumer16(String nama, int noHP) {
        this.nama = nama;
        this.noHP = noHP;
    }
}
```

```
package Praktikum3;

public class Queue16 {
    Costumer16[] antrian;
    int front;
    int rear;
    int size;
    int max;

    public Queue16(int n) {
        max = n;
        antrian = new Costumer16[max];
        size = 0;
        front = rear = -1;
    }

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

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

    public void peek() {
        if (!isEmpty()) {
            System.out.println("Pembeli paling depan: " + antrian[front].nama + ", No HP: " + antrian[front].noHP);
        } else {
            System.out.println(x:"Antrian masih kosong");
        }
    }

    public void peekRear() {
        if (!isEmpty()) {
            System.out.println("Pembeli paling belakang: " + antrian[rear].nama + ", No HP: " + antrian[rear].noHP);
        } else {
            System.out.println(x:"Antrian masih kosong");
        }
    }

    public void peekPosition(String nama) {
        if (!isEmpty()) {
            for (int i = front; i <= rear; i++) {
                if (antrian[i].nama.equals(nama)) {
                    System.out.println("Pembeli " + nama + " berada di posisi antrian ke-" + (i - front + 1));
                    return;
                }
            }
            System.out.println("Pembeli " + nama + " tidak ditemukan dalam antrian");
        } else {
            System.out.println(x:"Antrian masih kosong");
        }
    }

    public void enqueue(Costumer16 p) {
        if (isFull()) {
            System.out.println(x:"Antrian sudah penuh");
        } else {
            if (isEmpty()) {
                front = rear = 0;
            } else {
                rear = (rear + 1) % max;
            }
            antrian[rear] = p;
            size++;
        }
    }
```

```
public Costumer16 dequeue() {
    Costumer16 p = null;
    if (isEmpty()) {
        System.out.println(x:"Antrian masih kosong");
    } else {
        p = antrian[front];
        size--;
        if (isEmpty()) {
            front = rear = -1;
        } else {
            front = (front + 1) % max;
        }
    }
    return p;
}

public void print() {
    if (isEmpty()) {
        System.out.println(x:"Antrian masih kosong");
    } else {
        int i = front;
        while (i != rear) {
            System.out.println("Pembeli: " + antrian[i].nama + ", No HP: " + antrian[i].noHP);
            i = (i + 1) % max;
        }
        System.out.println("Pembeli: " + antrian[i].nama + ", No HP: " + antrian[i].noHP);
    }
}

public void daftarPembeli() {
    if (isEmpty()) {
        System.out.println(x:"Antrian masih kosong");
    } else {
        System.out.println(x:"Daftar Pembeli:");
        int i = front;
        while (i != rear) {
            System.out.println("Pembeli: " + antrian[i].nama + ", No HP: " + antrian[i].noHP);
            i = (i + 1) % max;
        }
        System.out.println("Pembeli: " + antrian[i].nama + ", No HP: " + antrian[i].noHP);
    }
}
```

```java
package Praktikum3;
import java.util.Scanner;

public class QueueMain16 {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print(s:"Masukan kapasitas queue: ");
        int Jumlah = sc.nextInt();
        Queue16 antri = new Queue16(Jumlah);
        sc.nextLine(); // Konsumsi newline

        int pilih;

        do {
            menu();
            pilih = sc.nextInt();
            sc.nextLine(); // Konsumsi newline setelah memilih menu
            switch (pilih) {
                case 1:
                    System.out.print(s:"Nama Pembeli: ");
                    String nama = sc.nextLine();
                    System.out.print(s:"Nomor HP: ");
                    int noHP = sc.nextInt();
                    Costumer16 pembeli = new Costumer16(nama, noHP);
                    antri.enqueue(pembeli);
                    break;
                case 2:
                    if (!antri.isEmpty()) {
                        Costumer16 data = antri.dequeue();
                        System.out.println("Antrian yang keluar: " + data.nama + ", No HP: " + data.noHP);
                    } else {
                        System.out.println(x:"Antrian masih kosong");
                    }
                    break;
                case 3:
                    antri.peek();
                    break;
                case 4:
                    antri.peekRear();
                    break;
                case 5:
                    System.out.println(x:"Masukkan nama pembeli yang ingin dicari: ");
                    String namaCari = sc.nextLine();
                    antri.peekPosition(namaCari);
                    break;
                case 6:
                    antri.print();
                    break;
            }
        } while (pilih != 7); // Perubahan di sini

        sc.close();
    }

    public static void menu() {
        System.out.println(x:"Pilih menu: ");
        System.out.println(x:"1. Antrian baru");
        System.out.println(x:"2. Antrian Keluar");
        System.out.println(x:"3. Cek Antrian terdepan");
        System.out.println(x:"4. Cek Antrian paling belakang");
        System.out.println(x:"5. Cek posisi antrian berdasarkan nama");
        System.out.println(x:"6. Cek Semua antrian");
        System.out.println(x:"-------------------------");
    }
}
```