LAPORAN JOBSHEET 8

Algoritma dan Struktur Data

Disusun Oleh :

DHANISA PUTRI MASHILFA

NIM. 2341720212

TI-1E/07

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No. 9 Jatimulyo, Kecamatan Lowokwaru, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 6514

8.2 Praktikum 1

8.2.1 Langkah - Langkah Praktikum 1

→ Class Queue07

```
package Praktikum1;
public class Queue07 {

    int[] data;
    int fornt;
    int rear;
    int size;
    int max;

    public Queue07(int n) {
        max = n;
        data = new int[max];
        size = 0;
        fornt = 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[fornt]);
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public void print() {
        if (!isEmpty()) {
            System.out.println("Queue masih kosong");
        } else {
            int i = fornt;
            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()) {
            fornt = rear = -1;
            size = 0;
            System.out.println("Queue berhasil dikosongkan");
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public void Enqueue (int dt) {
        if (IsFull()) {
            System.out.println("Queue sudah penuh");
        } else {
            if (isEmpty()) {
                fornt= 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 [fornt];
            size--;
            if (isEmpty()) { fornt = rear = -1;
            } else {
                if (fornt == max - 1) {
                    fornt = 0;
                } else {
                    fornt++;
                }
            }
        }
        return dt;
    }
```

```
}
```

→ Class QueueMain07

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

public class QueueMain07 {

    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("-----------------------------------------------");

    }

    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println();
        System.out.print("Masukan kapasitas queue : ");
        int n = sc.nextInt();

        Queue07 Q = new Queue07(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 dikeluarka : " +
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 );
    }
}
```

8.2.2 Hasil Verifikasi Praktikum 1

```
 ;141e80da-d83d-4992-bae7-bc68ad880649
Masukan kapasitas queue : 4
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
1
Masukkan data baru : 15
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
1
Masukkan data baru : 31
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
4
Elemen terdepan : 15
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
```

8.2.3 Pertanyaan Praktikum 1

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
   → Untuk menjalankan antrian menggunakan array, metode yang paling umum adalah mengatur nilai awal atribut depan dan belakang sebagai -1 dalam konstruktor, dan nilai atribut ukuran diatur sebagai 0. Ini adalah penjelasannya:

    1. atribut depan maupun belakang diatur menjadi -1: Nilai -1 untuk kedua atribut menunjukkan bahwa tidak ada antrian. Jika antrian kosong tidak memiliki elemen, nilainya diatur sebagai -1, jadi tidak ada elemen di depan atau di belakangnya.

2. Atribut ukuran diatur sebagai 0: Nilai 0 untuk ukuran menunjukkan bahwa tidak ada elemen dalam antrian saat inisialisasi, dan ketika antrian baru dibuat, tidak ada elemen yang dimasukkan, sehingga ukurannya diatur sebagai 0.

Dengan mengatur nilai awal atribut depan dan belakang menjadi -1 dan ukurannya diatur sebagai 0, kita dapat dengan mudah memeriksa apakah antrian kosong atau tidak, dan memulai operasi penambahan dan penghapusan elemen dengan benar. Misalnya, saat melakukan penambahan dan penghapusan elemen

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

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

→ Apabila indeks **rear** mencapai batas maksimum array **data,** maka kode tersebut akan digunakan. Oleh karena itu, Anda dapat menambahkan elemen baru tanpa menunggu seluruh array terisi terlebih dahulu jika elemen sebelumnya telah dihapus dari antrian. Anda hanya perlu menggunakan ruang yang tersisa di belakang antrian untuk menambahkan elemen baru.

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

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

→ Atribut ukuran diatur sebagai 0, nilai 0 untuk ukuran menunjukkan bahwa tidak ada elemen dalam antrian saat inisialisasi, dan ketika antrian baru dibuat, tidak ada elemen yang dimasukkan, sehingga ukurannya diatur sebagai 0. Dengan mengatur nilai awal atribut depan dan belakang menjadi -1 dan ukurannya diatur sebagai 0, kita dapat dengan mudah memeriksa apakah antrian kosong atau tidak, dan memulai operasi penambahan dan penghapusan elemen dengan benar. Misalnya, saat melakukan penambahan dan penghapusan elemen

4. Pada method print, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (int i=0), melainkan int i=front?
→ Menyesuaikan dengan keadaan antrian, memulai perulangan dari depan memungkinkan kita menyesuaikan iterasi dengan keadaan aktual antrian. Misalnya, jika operasi penghapusan sebelumnya telah mengeluarkan beberapa elemen dari antrian, front mungkin berubah, dan kita ingin memulai pencetakan dari elemen yang baru menjadi depan antrian.

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

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

→ Dengan menggunakan kode ini, perulangan memiliki kemampuan untuk berputar secara circular melalui elemen-elemen antrian yang diimplementasikan menggunakan array. Nilai i ditambahkan satu setiap kali perulangan dilakukan untuk menggeser ke elemen berikutnya dalam antrian. Jika nilai i mencapai batas maksimum array, atau max, maka nilai i akan kembali ke 0, memungkinkan perulangan untuk berputar kembali ke awal array. Ini memastikan bahwa perulangan dapat bergerak melalui elemen-elemen antrian tanpa henti bahkan setelah mencapai akhir array. Ini menangani struktur queue lingkaran, di mana elemen-elemen antrian dapat berputar kembali ke awal array.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

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

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!
   → Modifikasi Program Pada Class Queue07

```
public void Enqueue (int dt) throws Exception {
        if (IsFull()) {
            throw new Exception("Queue sudah penuh! Program
Terhenti.");
        } else {
            if (isEmpty()) {
                fornt= rear = 0;
            } else {
                if (rear == max - 1) {
                    rear = 0;
                } else {
                    rear++;
                }
            }
            data [rear] = dt;
            size++;
        }
    }

    public int Dequeue() throws Exception {
        int dt = 0;
        if (isEmpty()) {
            throw new Exception("Queue masih kosong! Program
Terhenti.");
        } else {
            dt = data [fornt];
            size--;
            if (isEmpty()) { fornt = rear = -1;
            } else {
                if (fornt == max - 1) {
                    fornt = 0;
                } else {
                    fornt++;
                }
            }
        }
        return dt;
    }
```

```
}
```

→ Modifikasi Program Pada Class QueueMain07

```
do {
            menu();
            pilih = sc.nextInt();
            try {
                switch (pilih) {
                    case 1:
                        System.out.print("Masukkan data baru : ");
                        int dataMasuk = sc.nextInt();
                        Q.Enqueue(dataMasuk);
                        break;

                    case 2:
                        int dataKeluar = Q.Dequeue();
                            System.out.println("Data yang dikeluarka
: " + dataKeluar);
                            break;


                    case 3:
                        Q.print();
                        break;

                    case 4:
                        Q.peek();
                        break;

                    case 5:
                        Q.clear();
                        break;
                }

                } catch (Exception e) {
                    System.out.println(e.getMessage());
                    System.out.println("Program Dihentikan.");
                    break;
            }

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

→ **Hasil dari Modifikasi**

```
Masukan kapasitas queue : 2
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
1
Masukkan data baru : 13
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
1
Masukkan data baru : 14
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
1
Masukkan data baru : 15
Queue sudah penuh! Program Terhenti.
Program Dihentikan.
```

```
Masukan kapasitas queue : 2
Masukkan operasi yang diinginkan :
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
---------------------------------------------
2
Queue masih kosong! Program Terhenti.
Program Dihentikan.
```

8.2 Praktikum 2

8.2.1 Langkah - Langkah Praktikum 2

→ Class Nasabah07

```
package Praktikum2;

import Praktikum1.Queue07;

public class Nasabah07 {

    String norek;
    String nama;
    String alamat;
    int umur;
    double saldo;

    Nasabah07 (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;
    }

}
```

→ Class Queue07

```
package Praktikum2;

public class Queue07 {

    Nasabah07[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue07(int n) {
        max = n;
        data = new Nasabah07 [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 clear () {
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue berhasil dikosongkan");
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public void Enqueue (Nasabah07 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 Nasabah07 Dequeue() {
        Nasabah07 dt = new Nasabah07(null, null, null, max, front);
        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 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);
        }
    }

}
```

→ Class QueueMain07

```
package Praktikum2;

import java.util.Scanner;

public class QueueMain07 {

    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("------------------------------");

    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int max;
```

```
        System.out.println();
        System.out.print("Masukan kapasitas queue : ");

        int jumlah = sc.nextInt();
        Queue07 antri = new Queue07(jumlah);

        int pilih;

        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();


                    Nasabah07 nb = new Nasabah07(norek, nama, alamat,
umur, saldo);
                    sc.nextLine();
                    antri.Enqueue(nb);
                    break;

                case 2:
                    Nasabah07 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.alamat + " "  + data.umur + " " + data.nama + " "
+ data.saldo);
                break;
                }

                case 3:
                    antri.peek();
                break;

                case 4:
                    antri.print();
                break;
            }

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

```
    }
}
```

8.2.2 Hasil Verifikasi Praktikum 2

```
Masukan kapasitas queue : 8
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
------------------------------
1
No Rekening   : 12345
Nama          : dewi
Alamat        : malang
Umur          : 23
Saldo         : 1300000
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
------------------------------
1
No Rekening   : 32940
Nama          : susan
Alamat        : surabay
Umur          : 39
Saldo         : 42000000
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
------------------------------
4
12345 dewi malang 23 1300000.0
32940 susan  surabay 39 4.2E7
Jumlah elemen = 2
Pilih menu :
```

8.2.3 Pertanyaan Praktikum 2

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;
}
```

→ Potongan kode program yang Anda berikan adalah bagian dari logika. Fungsi blok if adalah untuk memeriksa apakah data yang diambil dari antrian dengan metode Dequeue memiliki nilai yang sah sebelum mencetaknya.

" ".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat) && data.umur != 0 && data.saldo != 0: Dalam bagian ini, setiap atribut objek data (mungkin dari kelas Nasabah07) diperiksa untuk memastikan bahwa mereka tidak kosong atau memiliki nilai default. "Equals(data.norek) memeriksa apakah atribut nama objek data tidak kosong, "Equals(data.alamat) memeriksa apakah atribut alamat objek data tidak kosong, dan data.umur!= 0 dan data.saldo!= 0.

Jika semua kondisi dalam blok if terpenuhi, yang berarti data yang diambil dari antrian memiliki nilai yang valid, maka blok kode berikut akan dieksekusi: System.out.println("Antrian yang keluar: " + data.norek + " " + data.alamat + " " + data.umur + " " + data.nama + " " + data.saldo); Kode ini digunakan untuk mencetak nomor rekening, alamat, umur, nama, dan saldo objek data.

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!

→ Modifikasi Pada Class Queue07

```
public Nasabah07 peekRear() throws Exception {
        if (isEmpty()) {
            throw new Exception("Queue underflow! Program
stopped.");
        } else {
            return data[rear];
        }
    }
```

→ Modifikasi Pada Class QueueMain07

```
do {
            menu();
            pilih = sc.nextInt();
            sc.nextLine();
            try {
                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();


                        Nasabah07 nb = new Nasabah07(norek, nama,
alamat, umur, saldo);
```

```
                        sc.nextLine();
                        antri.Enqueue(nb);
                        break;

                    case 2:
                        Nasabah07 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.alamat + " "  + data.umur + " " +
data.nama + " " + data.saldo);
                    break;
                    }

                    case 3:
                        antri.peek();
                    break;

                    case 4:
                        antri.print();
                    break;

                    case 5:
                    try {
                        Nasabah07 rearData = antri.peekRear();
                        System.out.println("Data di posisi belakang
antrian: " + rearData.norek + " " + rearData.nama + " " +
rearData.alamat + " " + rearData.umur + " " + rearData.saldo);
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                }

            } catch (Exception e) {
                System.out.println(e.getMessage());
                System.out.println("Program dihentikan.");
                break;
            }

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

→ Hasil dari Modifikasi

```
Masukan kapasitas queue : 2
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Belakang
------------------------------
1
No Rekening   : 12345
Nama          : dewi
Alamat        : malang
Umur          : 23
Saldo         : 1300000
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Belakang
------------------------------
1
No Rekening   : 32940
Nama          : susi
Alamat        : surabaya
Umur          : 39
Saldo         : 42000000
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Belakang
------------------------------
5
Data di posisi belakang antrian: 32940 susi surabaya 39 4.2E7
```

8.4 Tugas

→ Class Pembeli07

```
package TugasPraktikum;

public class Pembeli07 {

    String nama;
    int noHP;

    Pembeli07(String nama, int noHP) {
        this.nama = nama;
        this.noHP = noHP;
    }

}
```

→ Class QueuePembeli07

```
package TugasPraktikum;

import Praktikum2.Nasabah07;

public class QueuePembeli07 {

    Pembeli07[] antrian;
    int front;
    int rear;
    int size;
    int max;

    public QueuePembeli07(int n) {
        max = n;
        antrian = new Pembeli07 [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 clear () {
        if (!isEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue berhasil dikosongkan");
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public void Enqueue (Pembeli07 an) {
        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] = an;
            size++;
        }
    }

    public Pembeli07 Dequeue() {
        Pembeli07 an= new Pembeli07(null, front);
        if (isEmpty()) {
            System.out.println("Queue masih kosong");
        } else {
            an = antrian [front];
            size--;
            if (isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max - 1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return an;
    }

    public void peek() {
        if (!isEmpty()) {
```

```
            System.out.println("Elemen terdepan: " + antrian[front].nama +
" " + antrian[front].noHP + " ");
        } 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[front].nama + " " +
antrian[front].noHP + " ");
            i = (i + 1) % max;
            }

            System.out.println(antrian[front].nama + " " +
antrian[front].noHP + " ");
            System.out.println("Jumlah elemen = " + size);
        }
    }

    public Pembeli07 peekRear() throws Exception {
        if (isEmpty()) {
            throw new Exception("Queue underflow! Program stopped.");
        } else {
            return antrian[rear];
        }
    }

    public void peekPosition(String nama) {
        if (isEmpty()) {
            System.out.println("Antrian masih kosong");
            return;
        }

        int currentPosition = front;
        int position = 1;
        boolean found = false;

        while (currentPosition != rear) {
            if (antrian[currentPosition].nama.equalsIgnoreCase(nama)) {
                System.out.println("Nama " + nama + " ditemukan pada
posisi : " + position);
                found = true;
                break;
            }
            currentPosition = (currentPosition + 1) % max;
            position++;
        }
```

```
        if (!found &&
antrian[currentPosition].nama.equalsIgnoreCase(nama)) {
            System.out.println("Nama " + nama + " ditemukan pada posisi: "
+ position);
            found = true;
        }

        if (!found) {
            System.out.println("Nama " + nama + " tidak ditemukan dalam
antrian");
        }
    }

    public void daftarPembeli() {
        if (isEmpty()) {
            System.out.println("Antrian masih kosong");
            return;
        }

        int currentPosition = front;

        System.out.println("Daftar Pembeli:");
        while (currentPosition != rear) {
            System.out.println(antrian[currentPosition].nama);
            currentPosition = (currentPosition + 1) % max;
        }

        System.out.println(antrian[currentPosition].nama);
    }

}
```

→ Class mainPembeli07

```
package TugasPraktikum;

import java.util.Scanner;

import Praktikum2.Nasabah07;

public class mainPembeli07 {

    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("5. Cek Antrian Paling Belakang");
        System.out.println("6. Cari Posisi Pembeli");
        System.out.println("7. Tampilkan Daftar Pembeli");
        System.out.println("------------------------------");
```

```
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int max;
        System.out.println();
        System.out.print("Masukan kapasitas queue : ");

        int jumlah = sc.nextInt();
        QueuePembeli07 antri = new QueuePembeli07(jumlah);

        int pilih;

        do {
            menu();
            pilih = sc.nextInt();
            sc.nextLine();
            try {
                switch (pilih) {

                    case 1:
                        System.out.print("Nama Pembeli  : ");
                        String nama = sc.nextLine();
                        System.out.print("Nomor Ponsel  : ");
                        int noHP= sc.nextInt();

                        Pembeli07 ant = new Pembeli07(nama, noHP);
                        sc.nextLine();
                        antri.Enqueue(ant);
                        break;

                    case 2:
                        Pembeli07 an = antri.Dequeue();
                        if (!" ".equals(an.nama) && an.noHP != 0) {
                            System.out.println("Antrian yang keluar: " +
an.nama + " " + an.noHP);
                    break;
                    }

                    case 3:
                        antri.peek();
                    break;

                    case 4:
                        antri.print();
                    break;

                    case 5:
                    try {
                        Pembeli07 rearData = antri.peekRear();
```

```
                        System.out.println("Data di posisi belakang
antrian: " + rearData.nama + " " + rearData.noHP + " ");
                    } catch (Exception e) {
                        System.out.println(e.getMessage());
                    }
                    break;

                    case 6:
                        System.out.print("Nama pembeli yang ingin dicari
posisinya: ");
                        String namaPembeli = sc.nextLine();
                        antri.peekPosition(namaPembeli);
                    break;

                    case 7:
                        antri.daftarPembeli();
                }

            } catch (Exception e) {
                System.out.println(e.getMessage());
                System.out.println("Program dihentikan.");
                break;
            }

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

}
```

→ Hasil Tugas Praktikum

```
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Belakang
6. Cari Posisi Pembeli
7. Tampilkan Daftar Pembeli
-----------------------------
6
Nama pembeli yang ingin dicari posisinya: ipin
Nama ipin ditemukan pada posisi : 2
Pilih menu :
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Belakang
6. Cari Posisi Pembeli
7. Tampilkan Daftar Pembeli
-----------------------------
7
Daftar Pembeli:
upin
ipin
meimei
susanti
```