Nama : Romy Mahardika Pangestu Lazuardi
No : 25
Kelas : 1H/D4TI

JOBSHEET X

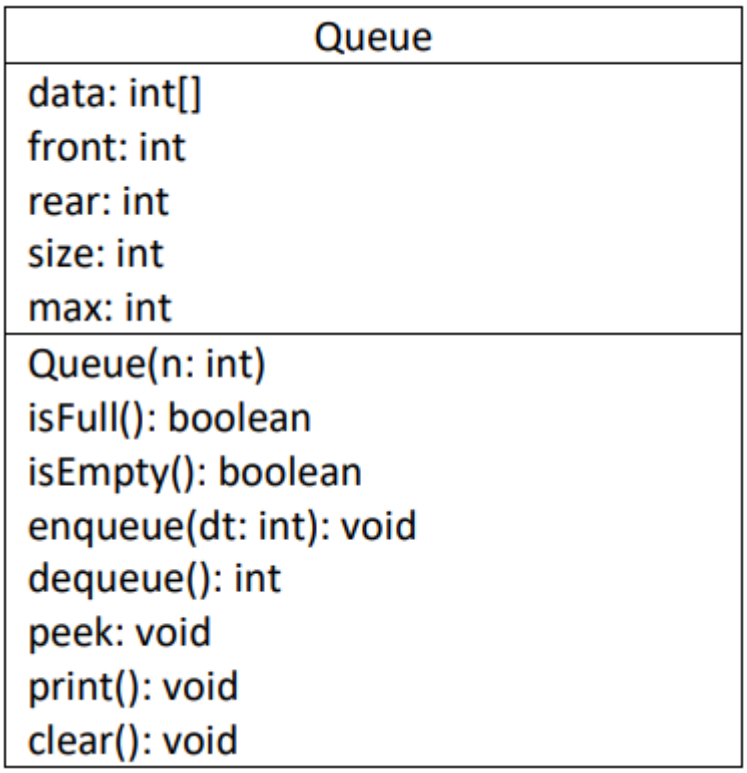
QUEUE

10.2 Praktikum 1

Waktu percobaan : 45 menit
Pada percobaan ini, kita akan mengimplementasikan penggunaan class Queue.

10.2.1 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

```
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 belakang

```
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.

```
public class Queue {
    int[] data;
    int front, rear, size, 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("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] + " ");
                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("Queue berhasil dikosongkan");
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public void Enqueue(int dt){
        if (isFull()){
            System.out.println("Queue sudah penuh");
            System.exit(0);
        } else {
            if (isEmpty()){
                front = rear = 0;
            } else {
```

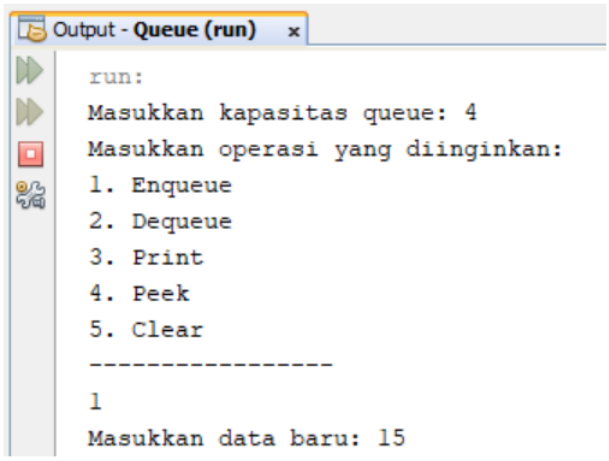
```
                if (rear == max -1){
                    rear = 0;
                } else {
                    rear++;
                }
            }
            data[rear] = dt;
            size++;
        }
    }

    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;
    }
}
```

```
import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        Scanner sc = new Scanner(System.in);
        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) {
```

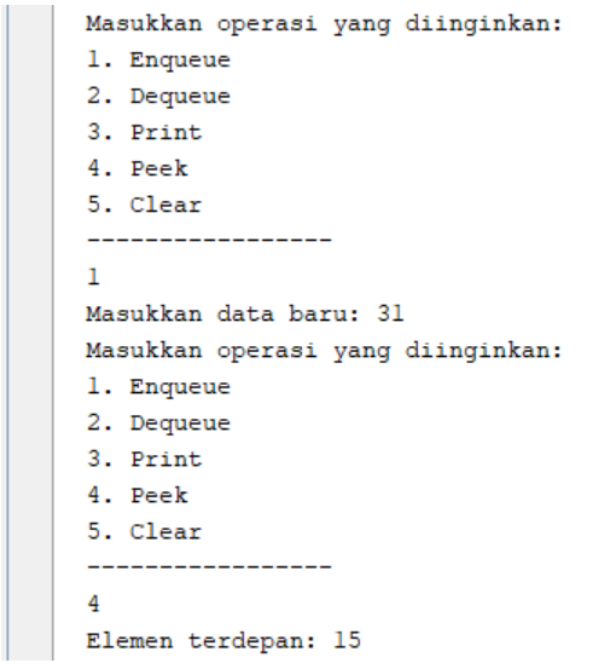
```
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan kapasitas queue: ");
        int n = sc.nextInt();
        Queue Q = new Queue(n);
        int pilih;

        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);

    }
}
```

10.2.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.


Output - Queue (run)
run:
Masukkan kapasitas queue: 4
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear

1
Masukkan data baru: 15


Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear

1
Masukkan data baru: 31
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear

4
Elemen terdepan: 15

```
Masukkan kapasitas queue: 4
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------------
1
Masukkan data baru: 15
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------------
1
Masukkan data baru: 31
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
----------------------
4
Elemen terdepan: 15
```

10.2.3 Pertanyaan

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

Karena pada atribut front dan rear menunjukkan index dari data, dimana index pertama dimulai dari 0 dan untuk menampilkan data yang masih kosong dengan -1.

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

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

Potongan kode diatas digunakan untuk melihat apakah pointer rear sudah mencapai batas panjang array, jika sudah mencapai maka pointer rear akan berulang ke index 0.

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

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

Kode diatas digunakan untuk melihat apakah pointer front sudah mencapai batas panjang array, jika sudah mencapai batas maka pointer front akan berulang ke index 0.

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

Karena data yang berada di posisi depan belum tentu pada index ke 0, bisa saja berada pada index belakang karena terjadinya proses enqueue dan dequeue.

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

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

Kode program diatas digunakan untuk melakukan increment untuk variable i, dan akan mereset ke 0 jika i sudah melampaui batas panjang array.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

```
public void Enqueue(int dt){

    if (isFull()){
        System.out.println("Queue sudah penuh");
        System.exit(0);
    } else {
```

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!

```
import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        Scanner sc = new Scanner(System.in);
        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.print("Masukkan kapasitas queue: ");
        int n = sc.nextInt();
        Queue Q = new Queue(n);
        int pilih;

        while (true) { // Menggunakan loop tak terbatas
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    if (Q.isFull()) {
                        System.out.println("Queue sudah penuh");
                        return; // Menghentikan program saat terjadi queue
overflow
                    }
```

```
                    System.out.print("Masukkan data baru: ");
                    int dataMasuk = sc.nextInt();
                    Q.Enqueue(dataMasuk);
                    break;

                case 2:
                    if (Q.isEmpty()) {
                        System.out.println("Queue masih kosong");
                        return; // Menghentikan program saat terjadi queue
underflow
                    }
                    int dataKeluar = Q.Dequeue();
                    System.out.println("Data yang dikeluarkan: "+ dataKeluar);
                    break;

                case 3:
                    Q.print();
                    break;

                case 4:
                    Q.peek();
                    break;

                case 5:
                    Q.clear();
                    break;

                default:
                    return; // Menghentikan program jika input tidak valid
            }
        }
    }
}
```

10.3 Praktikum 2

Waktu percobaan : 45 menit

Pada percobaan ini, kita akan membuat program yang mengilustrasikan teller di bank dalam melayani nasabah.

10.3.1 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.

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

    public 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(){
```

```
    }
}
```

```
public class Queue {
    Nasabah[] data;
    int front, rear, size, max;

    public Queue(int n){
        max = n;
        data = new Nasabah[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("Nasabah 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 peekRear(){
        if(!isEmpty()){
            System.out.println("Nasabah paling belakang: "+data[rear].norek +
" "+ data[rear].nama + " " + data[rear].alamat + " "+data[rear].umur + " " +
data[rear].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 kosong");
        }
    }

    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();
```

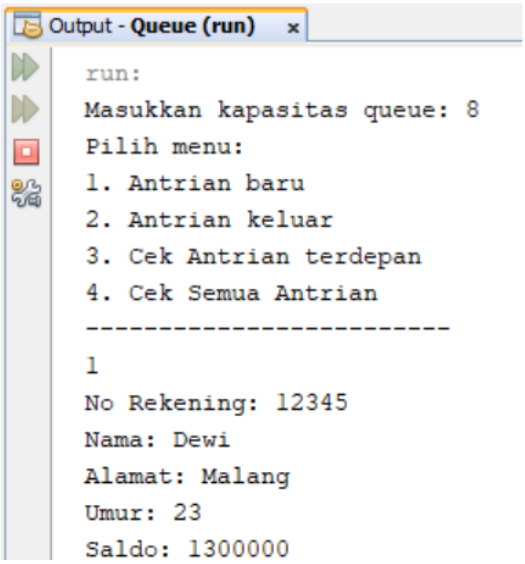
```
        if(isEmpty()){
            System.out.println("Queue masih kosong");
        } else {
            dt = data[front];
            size--;
            if(isEmpty()) {
                front = rear = -1;
            } else {
                if (front == max -1){
                    front = 0;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

```
import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        Scanner sc = new Scanner(System.in);
        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("----------------------");
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan kapasitas queue: ");
        int jumlah = sc.nextInt();
        Queue antri = new Queue(jumlah);
        int pilih;

        do{
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    System.out.print("No Rekening   : ");
                    String norek = input.nextLine();
```

```
                    System.out.print("Nama          : ");
                    String nama = input.nextLine();
                    System.out.print("Alamat        : ");
                    String alamat = input.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);
    }
}
```

10.3.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.




```
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
---
1
No Rekening: 32940
Nama: Susan
Alamat: Surabaya
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 Surabaya 39 4.2E7
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
---
```

```
Masukkan kapasitas queue: 8
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          : Susan
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
----------------------
4
12345 Dewi Malang 23 1300000.0
32940 Susan Surabaya 39 4.2E7
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek antrian terdepan
4. Cek semua antrian
5. Cek antrian paling belakang
----------------------
```

10.3.3 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;
}
```

Pada proses dequeue akan mengembalikan 1 return data berupa nasabah, fungsi IF diatas digunakan untuk memastikan bahwa data nasabah yang diambil bukan merupakan data kosong, melainkan data yang sudah diisi.

2. Lakukan modifikasi program dengan menambahkan method baru bernama peekRear pada class Queue yang digunakan untuk mengecek antrian yang berada di posisi belakang! Tambahkan pula

daftar menu 5. Cek Antrian paling belakang pada class QueueMain sehingga method peekRear dapat dipanggil!

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

```
import java.util.Scanner;

public class QueueMain {
    public static void menu() {
        Scanner sc = new Scanner(System.in);
        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"); // Menambahkan
pilihan menu 5
        System.out.println("----------------------");
    }
```

10.4 Tugas

1. Buatlah program antrian untuk mengilustasikan pesanan disebuah warung. Ketika seorang pembeli akan mengantri, maka dia harus mendaftarkan nama, dan nomor HP seperti yang digambarkan pada Class diagram berikut:

Pembeli
nama: String noHP: int
Pembeli(nama: String, noHP: int)

Class diagram Queue digambarkan sebagai berikut:

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

Keterangan:

· Method create(), isEmpty(), isFull(), enqueue(), dequeue() dan print(), kegunaannya sama seperti yang telah dibuat pada Praktikum

· Method peek(): digunakan untuk menampilkan data Pembeli yang berada di posisi antrian paling depan

· Method peekRear(): digunakan untuk menampilkan data Pembeli yang berada di posisi antrian paling belakang

· Method peekPosition(): digunakan untuk menampilkan seorang pembeli (berdasarkan nama)posisi antrian ke berapa

· Method daftarPembeli(): digunakan untuk menampilkan data seluruh pembeli

```
class Pembeli {
    String nama;
    int noHP;

    Pembeli(String nama, int noHP) {
        this.nama = nama;
        this.noHP = noHP;
    }
}
class Queue {
    Pembeli[] antrian;
    int front, rear, size, max;

    Queue(int n) {
        max = n;
        antrian = new Pembeli[max];
        front = size = 0;
        rear = -1;
    }
```

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

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

    void enqueue(Pembeli antri) {
        if (isFull()) {
            System.out.println("Antrian penuh, mohon tunggu sebentar.");
            return;
        }
        rear = (rear + 1) % max;
        antrian[rear] = antri;
        size++;
    }

    int dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong, tidak ada yang bisa
dikeluarkan.");
            return -1;
        }
        Pembeli temp = antrian[front];
        front = (front + 1) % max;
        size--;
        return temp.noHP;
    }

    void print() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        System.out.println("Antrian pembeli:");
        for (int i = front; i <= rear; i = (i + 1) % max) {
            System.out.println("Nama: " + antrian[i].nama + ", No HP: " +
antrian[i].noHP);
        }
    }

    void peek() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
```

```
        System.out.println("Pembeli paling depan: " + antrian[front].nama);
    }

    void peekRear() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        System.out.println("Pembeli paling belakang: " + antrian[rear].nama);
    }

    void peekPosition(String nama) {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        for (int i = front; i <= rear; i = (i + 1) % max) {
            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.");
    }

    void daftarPembeli() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        System.out.println("Daftar seluruh pembeli:");
        for (int i = front; i <= rear; i = (i + 1) % max) {
            System.out.println("Nama: " + antrian[i].nama + ", No HP: " +
antrian[i].noHP);
        }
    }
}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Queue antrian = new Queue(5);
```

```
        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Tambah Pembeli");
            System.out.println("2. Hapus Pembeli");
            System.out.println("3. Lihat Pembeli Terdepan");
            System.out.println("4. Lihat Pembeli Terbelakang");
            System.out.println("5. Cari Posisi Pembeli");
            System.out.println("6. Daftar Seluruh Pembeli");
            System.out.println("7. Keluar");
            System.out.print("Pilih menu: ");
            int menu = scanner.nextInt();

            switch (menu) {
                case 1:
                    System.out.print("Masukkan nama pembeli: ");
                    String nama = scanner.next();
                    System.out.print("Masukkan nomor HP pembeli: ");
                    int noHP = scanner.nextInt();
                    antrian.enqueue(new Pembeli(nama, noHP));
                    break;
                case 2:
                    int deletedNoHP = antrian.dequeue();
                    if (deletedNoHP != -1)
                        System.out.println("Pembeli dengan nomor HP " +
deletedNoHP + " telah dihapus dari antrian.");
                    break;
                case 3:
                    antrian.peek();
                    break;
                case 4:
                    antrian.peekRear();
                    break;
                case 5:
                    System.out.print("Masukkan nama pembeli yang ingin dicari:
");
                    String searchNama = scanner.next();
                    antrian.peekPosition(searchNama);
                    break;
                case 6:
                    antrian.daftarPembeli();
                    break;
                case 7:
                    System.out.println("Terima kasih!");
                    System.exit(0);
                default:
                    System.out.println("Menu tidak valid.");
                    break;
            }
```

```
        }
    }
}
```