PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET PERTEMUAN KE-10

RIO TRI PRAYOGO

TI 1A

26

2341720236

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

Queue

Praktikum 1 : Implementasi Penggunaan Class Queue

Percobaan :

Queue

```
package minggu10.Praktikum1;


public class Queue26 {
    int[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue26(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");
```

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

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

QueueMain

```
package minggu10.Praktikum1;

import java.util.Scanner;

public class QueueMain26 {
    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 scan = new Scanner(System.in);

        System.out.print("Masukkan kapasitas queue: ");
        int n = scan.nextInt();

        Queue26 Q = new Queue26(n);
        int pilih;

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

Output :

```
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
```

Pertanyaan :

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
   - Nilai awal atribut front dan rear pada konstruktor bernilai -1 karena kedua atribut tersebut tidak menunjuk ke data manapun. Sementara atribut size bernilai 0 karena array dari size masih kosong dan nanti akan diisikan melalui inputan scanner.
2. Pada method **Enqueue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

   - Potongan kode diatas bermaksud dan berguna untuk mengecek apakah data paling belakang dari queue atau rear berada di indeks terakhir array, jika iya maka **Enqueue** atau data baru yang dimasukkan ke dalam indeks 0 yang berarti posisi `rear = 0;`

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

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

   - Potongan kode diatas memiliki maksud dan kegunaan untuk mengecek apakah data paling depan dari queue atau front berada di indeks terakhir array, jika iya maka **Dequeue** atau ketika pengambilan data posisi front akan bergeser ke indeks 0 atau `front = 0;`

4. Pada method **print**, mengapa pada proses perulangan variabel i tidak dimulai dari `0 (int i=0)`, melainkan `int i=front`?
   - Karena method **print** menampilkan semua data dengan cara melakukan loop/perulangan di semua isi array. Looping tersebut tidak selalu mulai dari indeks ke-0 karena front tidak selalu berada di indeks ke-0, dan looping tersebut selalu mulai dari front.

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

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

   - Potongan kode dalam method **print** diatas memiliki maksud untuk menggeser data yang akan di print. `i` merupakan data yang di tunjuk pada saat itu. `i + 1` berfungsi untuk menggeser data yang ditunjuk. `% max` berfungsi untuk melihat sisa dari data yang belum ditunjuk. Hal ini berarti data akan ditunjuk satu persatu dan digeser sampai tidak ada sisa.

6. Tunjukkan potongan kode program yang merupakan queue overflow!
   - Queue overflow merupakan kondisi dimana data dicoba ditambahkan kedalam queue yang sudah penuh. Potongan kode yang menggambarkan queue overflow adalah:

```
...
    public void Enqueue(int dt) {
        if (IsFull()) {
            System.out.println("Queue sudah penuh");
        } else {
...
```

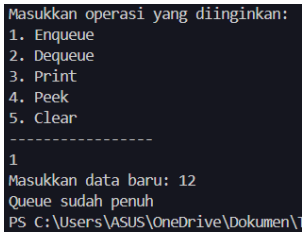
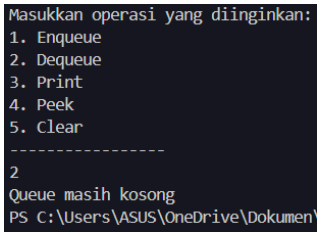
   Dimana IsFull() merupakan method yang berisi:

```
...
    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!
   - Untuk membuat program berhenti pada saat overflow dan underflow bisa ditambahkan kode `System.exit(0);` setelah if/else seperti dibawah ini:

Queue Overflow	Queue Underflow

...
 	
public void Enqueue(int dt) {
if (IsFull()) {
System.out.println("Queue sudah penuh");
System.exit(0);
} else {
...
	...
	
	public int Dequeue() {
	int dt = 0;
	if (IsEmpty()) {
	System.out.println("Queue masih kosong");
	System.exit(0);
	} else {
	...

Praktikum 2 : Ilustrasikan Teller di Bank Dalam Melayani Nasabah.

Percobaan :

Nasabah

```
package minggu10.Praktikum2;


public class Nasabah26 {

    String norek, nama, alamat;

    int umur;

    double saldo;


    Nasabah26() {


    }


    Nasabah26(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;

    }

}
```

Queue

```
package minggu10.Praktikum2;

public class Queue26 {
    Nasabah26[] data;
    int front;
    int rear;
    int size;
    int max;

    public Queue26(int n) {
        max = n;
        data = new Nasabah26[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].norek + "
" + data[front].nama + " "
                    + data[front].alamat + " " + data[front].umur + " " +
data[front].saldo);
        } else {
            System.out.println("Queue masih kosong");
        }
    }

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

    public void Enqueue(Nasabah26 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 Nasabah26 Dequeue() {
        Nasabah26 dt = new Nasabah26();
        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;
    }
}
```

QueueMain

```
package minggu10.Praktikum2;

import java.util.Scanner;

public class QueueMain26 {
    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 scan = new Scanner(System.in);

        System.out.print("Masukkan kapasitas queue: ");
        int jumlah = scan.nextInt();

        Queue26 antri = new Queue26(jumlah);
        int pilih;

        do {
            menu();
            pilih = scan.nextInt();
            scan.nextLine();
            switch (pilih) {
                case 1:
                    System.out.print("No Rekening: ");
                    String norek = scan.nextLine();
                    System.out.print("Nama: ");
                    String nama = scan.nextLine();
                    System.out.print("Alamat: ");
                    String alamat = scan.nextLine();
                    System.out.print("Umur: ");
                    int umur = scan.nextInt();
                    System.out.print("Saldo: ");
                    double saldo = scan.nextDouble();
                    Nasabah26 nb = new Nasabah26(norek, nama, alamat, umur,
saldo);
                    scan.nextLine();
                    antri.Enqueue(nb);
```

```
                    break;
                case 2:
                    Nasabah26 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;
                default:
                    break;
            }
        } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4);
    }
}
```

Output :

```
Masukkan 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: 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
```

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;
}
```

   - Potongan kode diatas berfungsi untuk mengecek apakah data dalam antrian kosong atau tidak, jika tidak kosong maka akan muncul antrian mana yang dikeluarkan.

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!
   - Untuk menambahkan method untuk mengecek antrian paling belakang bisa menambah baris kode berikut:
     Queue

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

     QueueMain

```
...
    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("-------------------------");
    }
...
...
                case 5:
                    antri.peekRear();
                    break;
...
```

Output:

```
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 4.2E8
Jumlah elemen = 2
Pilih menu:
1. Antrian baru
2. Antrian keluar
3. Cek Antrian terdepan
4. Cek Semua Antrian
5. Cek Antrian paling belakang
------------------------
Elemen paling belakang: 32940 Susan Surabaya 39 4.2E8
```

Tugas

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

Kode :

Pembeli

```
package minggu10.Tugas;

public class Pembeli26 {
    String nama;
    int noHP;

    Pembeli26() {

    }

    Pembeli26(String nama, int noHp) {
        this.nama = nama;
        this.noHP = noHp;
    }
}
```

Queue

```
package minggu10.Tugas;

public class Queue26 {
    Pembeli26[] data;
    int front;
    int rear;
    int size;
    int max;
    Pembeli26[] dataDeq;
    int dequeuedIndex;

    public Queue26(int n) {
        max = n;
        data = new Pembeli26[max];
        size = 0;
        front = rear = dequeuedIndex = -1;
        dataDeq = new Pembeli26[max];
```

```
}

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

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

public void Enqueue(Pembeli26 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 Pembeli26 Dequeue() {
        Pembeli26 dt = null;
        if (IsEmpty()) {
            System.out.println("Queue masih kosong");
            System.exit(0);
        } else {
            dt = data[front];
            size--;
            if (dequeuedIndex == max - 1) {
                dequeuedIndex = 0;
            } else {
                dequeuedIndex++;
            }
            dataDeq[dequeuedIndex] = dt;
            if (IsEmpty()) {
                front = rear = -1;
            } else {
                if (front == max - 1) {
                    front = 0;
                } else {
                    front++;
                }
            }
        }
        return dt;
    }

    public void peek() {
        if (!IsEmpty()) {
            System.out.println("Antrian terdepan: ");
            System.out.println("Nama: " + data[front].nama + ", No. HP: " +
data[front].noHP);
        } else {
            System.out.println("Queue masih kosong");
        }
```

```
    }

    public void peekRear() {
        if (!IsEmpty()) {
            System.out.println("Antrian paling belakang: ");
            System.out.println("Nama: " + data[rear].nama + ", No. HP: " +
data[rear].noHP);
        } else {
            System.out.println("Queue masih kosong");
        }
    }

    public int cariAntrian(String cari) {
        int posisi = -1;
        for (int j = 0; j < data.length; j++) {
            if (data[j].nama.equalsIgnoreCase(cari)) {
                posisi = j;
            }
        }
        return posisi;
    }

    public void peekPosition(String nama) {
        int i = front;
        while (i != rear) {
            if (data[i].nama.equalsIgnoreCase(nama)) {
                System.out.println(nama + " berada di antrian ke-" + (i -
front + 1));
                return;
            }
            i = (i + 1) % max;
        }
        System.out.println(nama + " tidak ada di antrian");
    }

    public void print() {
        if (IsEmpty()) {
```

```
            System.out.println("Queue masih kosong");
        } else {
            int i = front;
            while (i != rear) {
                System.out.println("Nama: " + data[i].nama + ", No. HP: " +
data[i].noHP);
                i = (i + 1) % max;
            }
            System.out.println("Nama: " + data[i].nama + ", No. HP: " +
data[i].noHP);
            System.out.println("Jumlah antrian = " + size);
        }
    }

    public Pembeli26[] cekKeluar() {
        Pembeli26[] dataKeluar = new Pembeli26[dequeuedIndex + 1];
        int index = 0;
        for (int i = 0; i <= dequeuedIndex; i++) {
            dataKeluar[index++] = dataDeq[i];
        }

        return dataKeluar;
    }

    public void dataPembeli() {
        Pembeli26[] dataKeluar = cekKeluar();
        if (dataKeluar.length == 0) {
            System.out.println("Belum ada pembeli yang di-keluar.");
        } else {
            System.out.println("Data pembeli yang sudah tidak antri:");
            for (Pembeli26 orang : dataKeluar) {
                System.out.println("Nama: " + orang.nama + ", No. HP: " +
orang.noHP);
            }
        }
    }
}
```

QueueMain

```
package minggu10.Tugas;

import java.util.Scanner;

public class QueueMain26 {
    public static void menu() {
        System.out.println("\nMasukkan operasi yang diinginkan:");
        System.out.println("1. Tambah Antrian");
        System.out.println("2. Kurangi Antrian");
        System.out.println("3. Lihat Antrian");
        System.out.println("4. Antrian Terdepan");
        System.out.println("5. Antrian Paling Belakang");
        System.out.println("6. Cari Antrian");
        System.out.println("7. Daftar Pembeli");
        System.out.println("-----------------");
    }

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

        System.out.print("Masukkan kapasitas antrian: ");
        int n = scan.nextInt();

        Queue26 Q = new Queue26(n);
        int pilih;

        do {
            menu();
            pilih = scan.nextInt();
            switch (pilih) {
                case 1:
                    System.out.println("Masukkan antrian baru!");
                    System.out.print("Masukkan Nama: ");
                    scan.nextLine();
                    String nama = scan.nextLine();
```

```
                    System.out.print("Masukkan Nomor HP: ");
                    int noHP = scan.nextInt();
                    Pembeli26 pb = new Pembeli26(nama, noHP);
                    scan.nextLine();
                    Q.Enqueue(pb);
                    break;
                case 2:
                    Pembeli26 keluar = Q.Dequeue();
                    if (!"".equals(keluar.nama) && keluar.noHP != 0) {
                        System.out.println("Antrian yang keluar: ");
                        System.out.println("Nama: " + keluar.nama + ", No.
HP: " + keluar.noHP);
                    }
                    break;
                case 3:
                    Q.print();
                    break;
                case 4:
                    Q.peek();
                    break;
                case 5:
                    Q.peekRear();
                    break;
                case 6:
                    Q.print();
                    System.out.print("Cari antrian berdasarkan nama: ");
                    scan.nextLine();
                    String cari = scan.nextLine();
                    Q.peekPosition(cari);
                    break;
                case 7:
                    System.out.println("Pembeli dalam antrian: ");
                    Q.print();
                    Q.dataPembeli();
                    break;
                default:
```

```
                    break;

            }

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

    }

}
```

Output :