LAPORAN PRAKTIKUM STRUKTUR DATA

Modul ke : 3
Judul Praktikum : QUEUE
Hari dan Tanggal Pelaksanaan : Kamis, 21 April 2022
Tempat Pelaksanaan : Lab Desain
Dosen Pengampu Praktikum : Khoirul Umam, S.Pd, M.Kom

Nama Mahasiswa Pelaksana : Andri Wijaksono
NIM Pelaksana : 362155401206

A. Tugas Pendahuluan

Tuliskan hasil pengerjaan Tugas Pendahuluan pada bagian ini.

1. Jelaskan konsep queue!

   Jawab: Queue atau antrian merupakan barisan elemen yang apabila elemen ditambah maka penambahannya berada diposisi belakang (rear) dan jika dilakukan pengambilan elemen dilakukan di elemen paling depan (fornt). Oleh karena itu, queue bersifat FIFO (First In First Out). Seperti konsep antrian pada dunia nyata, orang yang pertama kali mengantri dan berada di urutan paling depan akan menjadi orang pertama yang dilayani. Sedangkan orang yang berada pada urutan paling belakang maka akan menjadi rang terakhir yang akan dilayani. Begitu pula pada sistem komputer, lain data yang pertama kali dimasukkan akan menjadi data yang pertama kali dikeluarkan/diproses dan data yang terakhir kali dimasukkan, maka akan menjadi data yang terakhir dikeluarkan/diproses.

2. Sebutkan dan jelaskan dua operasi dasar pada queue!

Jawab:

1. **Enqueue**

Proses Enqueue merupakan proses untuk menambah di posisi Rear. Jika sudah mempunyai elemen, maka nilai rear harus bertambah 1. Kemudian data barudisimpan di queue pada posisi rear.

2. **Dequeue**,

dequeue, operasi untuk mengeluarkan/menghapus data dari dalam queue. Karena konsep queue menerapkan aturan FIFO, maka kedua operasi utama ini dianggap memiliki “pintu” yang berbeda.

B. Kegiatan Praktikum

Cantumkan apa saja yang dilakukan pada latihan-latihan praktikum, *source code* yang dipakai, *screen shot* hasil eksekusi kode, dan jawaban dari pertanyaan-pertanyaan yang muncul pada tiap kegiatan latihan.

Latihan 1: Implementasi bounded queue

1. Deklarasikan interface Queue di dalam file Queue.java seperti berikut:

```
public interface Queue<T> {
abstract boolean isEmpty();
abstract T head();
abstract T dequeue();
abstract void enqueue(T item);
```

```
abstract int size();
}
```

2. Pada folder yang sama buat file bernama **BoundedQueue.java** dan tuliskan kode berikut:

```
public class BoundedQueue<T> implements Queue<T> {
private T queue[];
private int front = -1, rear = -1;
public BoundedQueue(int size) {
queue = (T[]) new Object[size];
}
public boolean isEmpty() {
return rear == -1;
}
public boolean isFull() {
return rear == queue.length - 1;
}
public void enqueue(T item) {
if (isFull()) {
System.out.println("Queue penuh, tidak dapat memasukkan item baru");
} else {
if (isEmpty()) {
front = 0;
rear = 0;
} else {
rear++;
}
queue[rear] = item;
System.out.println("Item baru disimpan pada posisi " + rear);
}
}
public T dequeue() {
if (isEmpty()) {
System.out.println("Queue kosong, tidak ada item yang dapat diambil");
return null;
} else {
T data = queue[front];
if (front == rear) {
front = -1;
rear = -1;
} else {
front++;
}
return data;
}
}
public T head() {
return isEmpty() ? null : queue[front];
}
public int size() {
```

```
return isEmpty() ? 0 : rear - front + 1;
}
public String printQueue() {
String queueStr = "";
if (isEmpty())
queueStr = "[Antrian kosong]";
else {
for (int i = front; i <= rear; i++) {
queueStr += queue[i];
if (i < rear)
queueStr += " | ";
}
}
return queueStr;
}
}
```

3. Untuk menguji implementasi class **BoundedQueue**, tuliskan kode berikut ke dalam file bernama **TestBoundedQueue.java**:

```
public class TestBoundedQueue {
public static void main(String[] args) {
BoundedQueue<String> q = new BoundedQueue<String>(5);
System.out.println("Jumlah antrian saat ini : " + q.size());
q.enqueue("Sinta");
q.enqueue("Agus");
q.enqueue("Ali");
System.out.println("Jumlah antrian saat ini : " + q.size());
System.out.println("Antrian : " + q.printQueue());
System.out.println("Item yang diambil : " + q.dequeue());
System.out.println("Jumlah antrian saat ini : " + q.size());
System.out.println("Antrian : " + q.printQueue());
q.enqueue("Yuli");
q.enqueue("Rahman");
q.enqueue("Nia");
System.out.println("Jumlah antrian saat ini : " + q.size());
System.out.println("Antrian : " + q.printQueue());
}
}
```

```
Jumlah antrian saat ini : 0
Item baru disimpan pada posisi 0
Item baru disimpan pada posisi 1
Item baru disimpan pada posisi 2
Jumlah antrian saat ini : 3
Antrian : Sinta | Agus | Ali
Item yang diambil : Sinta
Jumlah antrian saat ini : 2
Antrian : Agus | Ali
Item baru disimpan pada posisi 3
Item baru disimpan pada posisi 4
Queue penuh, tidak dapat memasukkan item baru
Jumlah antrian saat ini : 4
Antrian : Agus | Ali | Yuli | Rahman
PS C:\Users\ASUS\Documents\Struktur Data\modul3>
```

5. Apa yang terjadi pada item “Nia”? Jelaskan!
Jawab: Antrian dengan nama Nia tidak bisa dimasukan karena pada index Queue terdeteksi penuh dan tidak bisa menambahkan item baru atau antrian baru. Posisi hanyalah terbatas berjumlah 4

6. Jika pada testing code di atas dilakukan dequeue kembali, item mana yang akan didapatkan? Jelaskan!
Jawab: Item yang akan keluar merupakan antrian yang mempunyai nama Agus. Kemudian antrian hanya menyisakan atas nama Ali, Yuli, dan Rahman.

7. Kapan item baru dapat kembali mengisi antrian untuk kasus ini?
Jawab: Pada kasus ini, item baru akan kembali ke antrian jika ke 5 antrian tersebut sudah kosong. Contohnya Sinta, Agus, Ali, Yuli, dan Rahman telah selesai mengantri secara bergantian, maka antrian tersebut akan diisikan oleh 5 orang lagi. Walaupun Sinta telah selesai mengantri atau keluar dan keempat orang lainnya belum selesai, antrian tersebut tidak boleh diisi oleh orang lain. Sistemnya seperti kloter ataupun kelompok. Dalam antrian, antrian tersebut akan berkurang tapi tidak akan bisa menambah.

Latihan 2: Modifikasi bounded queue

1. Masih di folder yang sama dengan latihan sebelumnya, deklarasikan class baru di dalam file bernama **ModifiedBoundedQueue.java** seperti berikut:

```
public class ModifiedBoundedQueue<T> implements Queue<T> {
    private T queue[];
    private int rear = -1;
    public ModifiedBoundedQueue(int size) {
    queue = (T[]) new Object[size];
    }
    public boolean isEmpty() {
```

```
    return rear == -1;
    }
    public boolean isFull() {
    return rear == queue.length - 1;
    }
    public void enqueue(T item) {
    if (isFull()) {
    System.out.println("Queue penuh, tidak dapat memasukkan item baru");
    } else {
    queue[++rear] = item;
    System.out.println("Item baru disimpan pada posisi " + rear);
    }
    }
    public T dequeue() {
    if (isEmpty()) {
    System.out.println("Queue kosong, tidak ada item yang dapat diambil");
    return null;
    } else {
    T data = queue[0];
    // Majukan antrian ke depan
    for (int i = 0; i < rear; i++)
    queue[i] = queue[i + 1];
    rear--;
    return data;
    }
    }
    public T head() {
    return isEmpty() ? null : queue[0];
    }
    public int size() {
        return isEmpty() ? 0 : rear + 1;
        }
        public String printQueue() {
        String queueStr = "";
        if (isEmpty())
        queueStr = "[Antrian kosong]";
        else {
        for (int i = 0; i <= rear; i++) {
        queueStr += queue[i];
        if (i < rear)
        queueStr += " | ";
        }
        }
        return queueStr;
     }
}
```

2. Buat testing code-nya di file **TestModifiedBoundedQueue.java** seperti berikut:

```
public class TestModifiedBoundedQueue {
```

```
    public static void main(String[] args) {
    ModifiedBoundedQueue<String> q = new ModifiedBoundedQueue<String>(5);
    System.out.println("Jumlah antrian saat ini : " + q.size());
    q.enqueue("Sinta");
    q.enqueue("Agus");
    q.enqueue("Ali");
    System.out.println("Jumlah antrian saat ini : " + q.size());
    System.out.println("Antrian : " + q.printQueue());
    System.out.println("Item yang diambil : " + q.dequeue());
    System.out.println("Jumlah antrian saat ini : " + q.size());
    System.out.println("Antrian : " + q.printQueue());
    q.enqueue("Yuli");
    q.enqueue("Rahman");
    q.enqueue("Nia");
    System.out.println("Jumlah antrian saat ini : " + q.size());
    System.out.println("Antrian : " + q.printQueue());
    }
    }
```

```
PS C:\Users\ASUS\Documents\Struktur Data\modul3> javac TestModifiedBoundedQueue.java
Note: .\ModifiedBoundedQueue.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Users\ASUS\Documents\Struktur Data\modul3> java TestModifiedBoundedQueue
Jumlah antrian saat ini : 0
Item baru disimpan pada posisi 0
Item baru disimpan pada posisi 1
Item baru disimpan pada posisi 2
Jumlah antrian saat ini : 3
Antrian : Sinta | Agus | Ali
Item yang diambil : Sinta
Jumlah antrian saat ini : 2
Antrian : Agus | Ali
Item baru disimpan pada posisi 2
Item baru disimpan pada posisi 3
Item baru disimpan pada posisi 4
Jumlah antrian saat ini : 5
Antrian : Agus | Ali | Yuli | Rahman | Nia
PS C:\Users\ASUS\Documents\Struktur Data\modul3>
```

4. Apa yang terjadi pada item “Nia”? Jelaskan!

Jawab: Nia masuk ke dalam antrian jika sinta keluar dari barisan antrian. Yang dimana awalnya sinta menempati posisi ke . kemudian sinta keluar dari barisan antrian, posisi 0 akan ditempati oleh Agus. Posisi 1 yang awalnya ditempatin oleh Agus sekarang ditempati oleh Ali. Begitu seterusnya. Jadi setiap kali item bergantian menempati posisi yang baru jika ada item lain yang keluar atau dihapus.

5. Apa yang membedakan contoh kasus ini dengan contoh kasus pada latihan sebelumnya?

Jawab: Yang membedakan cara item-item tersebut keluar dan masuk. Pada kasus yang pertama, item baru akan bisa masuk jika seluruh posisinya sedang kosong. Contohnya, ada 5 posisi yang tersedia. Maka 5 orang pertama akan menempati posisi yang kosong tersebut. Walaupun ada salah satu ornag keluar, posisi tersebut tidak diperbolehkan untuk ditempati oleh orang yang baru. Karena terdapat 4 orang yang belum keluar. Jika 4 orang tersebut, maka 5 orang selanjutnya akan mengisi 5 posisi tersebut. Sistemnya sperti kelompok atau kloter.

Pada kasus kedua, sistem keluar masuk secara bergantian. Satu item baru akan masuk jika item yang pertama tadi telah keluar. Contohnya ada 5 posisi kemudian ada 5 orang menempati ke 5 posisi tersebut. 5 orang tersebut A, B, C, D, E jika orang A dikeluarkan ataupun dihapus, maka orang B akan menempati posisi orang A. orang C akan menempati posisi orang B begitu juga selanjutnya. Jadi, kelima posisi tersebut tidak boleh dalam keadaan kosong. Wajib terisi semua.

Latihan 3: Implementasi circular bounded queue

1. Masih di folder yang sama dengan latihan sebelumnya, tuliskan kode berikut ke dalam file baru dengan nama **CircularBoundedQueue.java**:4

```
public class CircularBoundedQueue<T> implements Queue<T> {
    private T queue[];
    private int front = -1, rear = -1;
    public CircularBoundedQueue(int size) {
    queue = (T[]) new Object[size];
    }
    public boolean isEmpty() {
    return rear == -1;
    }
    public boolean isFull() {
    return size() == queue.length;
    }
    public void enqueue(T item) {
    if (isFull()) {
    System.out.println("Queue penuh, tidak dapat memasukkan item baru");
    } else {
    if (isEmpty()) {
    front = 0;
    rear = 0;
    } else {
    rear = (rear + 1) % queue.length;
    }
    queue[rear] = item;
    System.out.println("Item baru disimpan pada posisi " + rear);
    }
    }
    public T dequeue() {
        if (isEmpty()) {
            System.out.println("Queue kosong, tidak ada item yang dapat
diambil");
            return null;
            } else {
            T data = queue[front];
            if (front == rear) {
            front = -1;
            rear = -1;
            } else {
```

```
            front = (front + 1) % queue.length;
            }
            return data;
            }
            }
            public T head() {
            return isEmpty() ? null : queue[front];
            }
            public int size() {
            if (isEmpty())
            return 0;
            else if (front <= rear)
            return rear - front + 1;
            else
            return rear + 1 + queue.length - front;
            }
            public String printQueue() {
            String queueStr = "";
            if (isEmpty())
            queueStr = "[Antrian kosong]";
            else if (front <= rear) {
            for (int i = front; i <= rear; i++) {
            queueStr += queue[i];
            if (i < rear)
            queueStr += " | ";
            }
            } else {
            for (int i = front; i < queue.length; i++)
            queueStr += queue[i] + " | ";
            for (int i = 0; i <= rear; i++) {
                queueStr += queue[i];
             if (i < rear)
             queueStr += " | ";
            }
        }
      return queueStr;
    }
}
```

2. Buat juga testing code-nya di dalam file **TestCircularBoundedQueue.java** seperti berikut:

```
public class TestCircularBoundedQueue {
public static void main(String[] args) {
CircularBoundedQueue<String> q = new CircularBoundedQueue<String>(5);
System.out.println("Jumlah antrian saat ini : " + q.size());
q.enqueue("Sinta");
q.enqueue("Agus");
q.enqueue("Ali");
System.out.println("Jumlah antrian saat ini : " + q.size());
```

```
System.out.println("Antrian : " + q.printQueue());
System.out.println("Item yang diambil : " + q.dequeue());
System.out.println("Jumlah antrian saat ini : " + q.size());
System.out.println("Antrian : " + q.printQueue());
q.enqueue("Yuli");
q.enqueue("Rahman");
q.enqueue("Nia");
System.out.println("Jumlah antrian saat ini : " + q.size());
System.out.println("Antrian : " + q.printQueue());
}
}
```

```
Jumlah antrian saat ini : 0
Item baru disimpan pada posisi 0
Item baru disimpan pada posisi 1
Item baru disimpan pada posisi 2
Jumlah antrian saat ini : 3
Antrian : Sinta | Agus | Ali
Item yang diambil : Sinta
Jumlah antrian saat ini : 2
Antrian : Agus | Ali
Item baru disimpan pada posisi 3
Item baru disimpan pada posisi 4
Item baru disimpan pada posisi 0
Jumlah antrian saat ini : 5
Antrian : Agus | Ali | Yuli | Rahman | Nia
PS C:\Users\ASUS\Documents\Struktur Data\modul3>
```

4. Apa yang terjadi pada item “Nia”? Jelaskan!
Jawab: Item Nia akan menempati posisi 0, dikarenakan item Sinta sudah keluar atau dihapus. Pada kasus ini jika ada satu item yang dikeluarkan dari satu posisi, tersebut akan diisi oleh item lainnya atau item yang baru.

5. Item apa yang berada di urutan paling depan untuk kasus ini?
Jawab: Item yang mempunyai nama Nia.

C. Tugas Praktikum

Tuliskan dan jabarkan hasil pengerjaan Tugas Praktikum yang tertera di dalam modul lengkap dengan *source code* yang digunakan.

4. Tugas Praktikum Buatlah sebuah aplikasi antrian yang dapat menerima input berupa nama pengantri. Selanjutnya aplikasi tersebut juga dapat memanggil nama pengantri sesuai dengan urutan antriannya.

```
import java.util.Scanner;

public class tugasQueue {
    int id;
    String nama;
    tugasQueue next;
```

```
    static Scanner in = new Scanner(System.in);
    static Scanner str = new Scanner(System.in);

    public void input(){
        System.out.print("Masukkan nama : ");
        nama = str.nextLine();
        System.out.println("Pengantri baru didaftarkan");
        next = null;
    }

    public void read(){
        System.out.println("Pengantri a.n. " + nama + " silahkan memasuki
ruangan");
        next = null;
    }

    public static void main(String[] args){
        int menu = 3;
        linked que = new linked();
        while(menu!=0){
            System.out.println("Pilih menu: ");
            System.out.println("1. Tambah antrian");
            System.out.println("2. Panggil antrian");
            System.out.println("0. Keluar");

            System.out.print("Menu pilihan : ");
            menu = in.nextInt();

            if(menu == 1)que.enqueue();
            else if(menu==2)que.view();
            else if(menu==0)System.exit(0);
            else System.out.println("Salah");
        }
    }
}

class linked{
    tugasQueue head, tail;

    public linked(){
        head = null;
        tail = null;
    }

    public void enqueue(){
        tugasQueue baru = new tugasQueue();
        baru.input();
```

```
        if(head == null)head=baru;
        else tail.next=baru;
        tail=baru;
    }

    public void view(){
        if(head==null)System.out.println("- Kosong -");
        else{
            for(tugasQueue a=head; a!=null; a=a.next) a.read();
        }
    }
}
```

```
Pilih menu:
1. Tambah antrian
2. Panggil antrian
0. Keluar
Menu pilihan : 1
Masukkan nama : khoirul umam
Pengantri baru didaftarkan
Pilih menu:
1. Tambah antrian
2. Panggil antrian
0. Keluar
Menu pilihan : 1
Masukkan nama : dwi febrianti
Pengantri baru didaftarkan
Pilih menu:
1. Tambah antrian
2. Panggil antrian
0. Keluar
Menu pilihan : 2
Pengantri a.n. khoirul umam silahkan memasuki ruangan
Pilih menu:
1. Tambah antrian
2. Panggil antrian
0. Keluar
Menu pilihan : 0
PS C:\Users\ASUS\Documents\Struktur Data>
```