LAPORAN PRAKTIKUM 4

STRUKTUR DATA

Oleh:
Ibrahim Mousa Dhani
2411532010

Dosen Pengampu : Dr. Wahyudi S.T, M.T.

Mata Kuliah : Struktur Data

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG

A. Pendahuluan

Struktur data adalah bagian penting dalam pengembangan algoritma dan program yang efisien. Salah satu struktur data linear yang banyak digunakan adalah **queue** atau antrian. Queue bekerja berdasarkan prinsip **First In, First Out (FIFO)**, artinya elemen yang pertama kali dimasukkan ke dalam antrian akan menjadi elemen pertama yang keluar.

Konsep queue sangat relevan dengan situasi dunia nyata, seperti antrian pembelian tiket, pemrosesan data dalam sistem operasi, atau penjadwalan tugas dalam manajemen proses. Dalam pemrograman, queue digunakan untuk mengelola data dalam urutan tertentu agar dapat diproses secara efisien.

B. Tujuan Praktikum

1. Memahami konsep dasar Queue pada Java yaitu First In, First Out (FIFO)
2. Mempelajari dan mengimplemetasikan operasi dasar pada Queue yaitu enqueue, dequeue, serta operasi tambahan
3. Memahami implementasi Queue dalam berbagai bentuk kode pada program Java
4. Melatih kemampuan logika dan algoritma dalam menyelesaikan permasalahan yang melibatkan Queue

C. Langkah Langkah

a. inputQueue

1. Pertama buat package pekan4, lalu buat class baru dan beri nama inputQueue
2. Tambahkan nama kelas, dan buat variabel-variabel yang dibutuhkan untuk queue. yaitu front untuk menunjukkan indeks elemen depan antrian (tempat keluar data). rear uuntuk menunjukkan indeks elemen terakhir (tempat masuk data).size untuk menunjukkan jumlah elemen saat ini di antrian. capacity merupakan kapasitas maksimum array, dan array merupakan penyimpanan data antrian.

```
package pekan4;

public class inputQueue {
    int front, rear, size;
    int capacity;
    int array[];
```

3. Setelah itu buat konstruktor yang akan digunakan untuk inisialisasi saat objek dibuat

```
public inputQueue(int capacity) {
    this.capacity = capacity;
    front = this.size = 0;
    rear = capacity - 1;
    array = new int[this.capacity];
}
```

4. Lalu buat fungsi isFull() dan isEmpty() untuk mengecek apakah antrian penuh atau kosong

```
boolean isFull(inputQueue queue) {
    return (queue.size == queue.capacity);
}

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

5. Buat fungsi enqueue(int item) untuk menambahkan data ke dalam antrian
6. Buat fungsi dequeue() untuk mengambil/menghapus data dari depan antrian

```
void enqueue(int item) {
    if (isFull(this))
        return;
    this.rear = (this.rear + 1) % this.capacity;
    this.array[this.rear] = item;
    this.size = this.size + 1;
    System.out.println(item + " enqueued to queue");
}

int dequeue() {
    if (isEmpty(this))
        return Integer.MIN_VALUE;
    int item = this.array[this.front];
    this.front = (this.front + 1) % this.capacity;
    this.size = this.size - 1;
    return item;
}
```

7. Buat fungsi ntuk melihat data yang berada di depan dan data yang berada di belakang tanpa menghapusnya yaitu int front() dan int rear()

```
int front() {
    if (isEmpty(this))
        return Integer.MIN_VALUE;
    return this.array[this.front];
}

int rear() {
    if (isEmpty(this))
        return Integer.MIN_VALUE;
    return this.array[this.rear];
}
```

b. TestQueue

1. Buat class baru pada package pekan4, dan beri nama TestQueue
2. Sebelum membuat kelas ini pastikan anda sudah membuat class inputQueue terlebih dahulu, karena kode pada kelas ini bergantung pada class inputQueue

```
package pekan4;

public class TestQueue {
```

3. Setelah itu inisialisasi queue, dan buat objek queue dengan kapasitas 1000

```
public static void main(String[] args) {
    inputQueue queue = new inputQueue(1000);
```

4. Lalu tambahkan elemen ke dalam queue satu persatu seperti gambar berikut

```
queue.enqueue(10);
System.out.println("Front item is " + queue.front());
System.out.println("Rear item is " + queue.rear());
```

5. Untuk menampilkan elemen pertama (front) dan terakhir (rear) dalam queue.
6. Menghapus (dequeue) elemen pertama dari queue dan menampilkannya.
7. Menampilkan kembali elemen front dan rear setelah satu elemen dihapus.
8. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan

```
System.out.println(queue.dequeue() + " dequeued from queue");
System.out.println("Rear item is " + queue.rear());
```

menghasilkan output sebagai berikut

```
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
Front item is 10
Rear item is 40
10 dequeued from queue
Front item is 20
Rear item is 40
```

9. Berikut kode program versi lengkapnya

```
1  package pekan4;
2
3  public class TestQueue {
4
5      public static void main(String[] args) {
6          inputQueue queue = new inputQueue(1000);
7          queue.enqueue(10);
8          queue.enqueue(20);
9          queue.enqueue(30);
10         queue.enqueue(40);
11         System.out.println("Front item is " + queue.front());
12         System.out.println("Rear item is " + queue.rear());
13         System.out.println(queue.dequeue() + " dequeued from queue");
14         System.out.println("Front item is " + queue.front());
15         System.out.println("Rear item is " + queue.rear());
16     }
```

c. ReverseData

1. Buat class baru pada package pekan4, dan beri nama ReverseData
2. Setelah itu Import library Queue, LinkedList, dan Stack atau import semua library dari Java Collection Framework dengan cara seperti berikut

```
package pekan4;

import java.util.*;

public class ReverseData {
```

3. Lalu inisialisasi Queue dengan implementasi LinkedList dengan nama ‘q’ yang menyimpan data bertipe Integer dan tambahkan elemen ke dalam queue, lalu tampilkam isi queue sebelum dibalik

```
public static void main(String[] args) {
    Queue<Integer> q = new LinkedList<Integer>();
    q.add(1);
    q.add(2);
    q.add(3); // [1, 2, 3]
    System.out.println("Sebelum reverse = " + q);
```

4. Lalu inisialisasi Stack, kita akan membuat stack kosong untuk menyimpan elemen dari queue sementara. Dan pindahkan dari queue ke stack. Selama queue tidak kosong, ambil elemen dari depan queue (remove()). Masukkan ke dalam stack (push()). Akibatnya, stack berisi [3, 2, 1], karena stack bersifat **LIFO** (Last In, First Out).

```
Stack<Integer> s = new Stack<Integer>();
while (!q.isEmpty()) { // Q -> S
s.push(q.remove());
}
```

5. Selanjutnya pindahkan kembali dari stack ke queue. Selama stack tidak kosong, ambil elemen dari atas stack (pop()). Masukkan kembali ke dalam queue (add()). Sekarang queue menjadi [3, 2, 1] — urutannya sudah dibalik. Lalu cetak hasilnya.

```
while (!s.isEmpty()) { // S -> Q
q.add(s.pop());
}
System.out.println("Sesudah reverse = " + q);
```

6. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Sebelum reverse = [1, 2, 3]
Sesudah reverse = [3, 2, 1]
```

7. Berikut kode program versi lengkapnya

```
1  package pekan4;
2
3  import java.util.*;
4
5  public class ReverseData {
6      public static void main(String[] args) {
7          Queue<Integer> q = new LinkedList<Integer>();
8          q.add(1);
9          q.add(2);
10         q.add(3); // [1, 2, 3]
11         System.out.println("Sebelum reverse = " + q);
12
13         Stack<Integer> s = new Stack<Integer>();
14         while (!q.isEmpty()) { // Q -> S
15         s.push(q.remove());
16         }
17         while (!s.isEmpty()) { // S -> Q
18         q.add(s.pop());
19         }
20         System.out.println("Sesudah reverse = " + q);
21     }
```

d. iterasiQueue

1. Buat class baru pada package pekan4, dan beri nama iterasiQueue
2. Setelah itu import library yang dperlukan dari Java Collection Framework yaitu Queue, Iterator, dan LinkedList

```
package pekan4;

import java.util.LinkedList;
import java.util.Iterator;
import java.util.Queue;

public class IterasiQueue {

    public static void main(String[] args) {
```

3. Lalu inisialisasi/ buat objek Queue yang menyimpan tipe data String, dan menggunakan LinkedList sebagai implementasi konkret dari Queue.
4. Selanjutnya tambahkan elemen string ke dalam queue secara berurutan.

```
Queue<String> q = new LinkedList<> ();
q.add("Praktikum");
q.add("Struktur");
q.add("Data");
q.add("Dan");
q.add("Algoritma");
```

5. Buat objek iterator dari queue. Iterator memungkinkan kita untuk melintasi elemen satu per satu tanpa menghapusnya. iterator.hasNext() untuk mengecek apakah masih ada elemen berikutnya. iterator.next() untuk mengambil elemen berikutnya.
6. Dan terakhir tampilkan elemen pada queue satu per satu

```
Iterator<String> iterator = q.iterator();
while (iterator.hasNext()) {
    System.out.println(iterator.next()+ " ");
}
```

7. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Praktikum
Struktur
Data
Dan
Algoritma
```

8. Berikut kode program versi lengkapnya

```
1  package pekan4;
2
3  import java.util.LinkedList;
4  import java.util.Iterator;
5  import java.util.Queue;
6
7  public class IterasiQueue {
8
9      public static void main(String[] args) {
10         Queue<String> q = new LinkedList<> ();
11         q.add("Praktikum");
12         q.add("Struktur");
13         q.add("Data");
14         q.add("Dan");
15         q.add("Algoritma");
16         Iterator<String> iterator = q.iterator();
17         while (iterator.hasNext()) {
18             System.out.println(iterator.next()+ " ");
19         }
```

e. ContohQueue2

1. Buat class baru pada package pekan4, dan beri nama ContohQueue2
2. Setelah itu import library yang dperlukan dari Java Collection Framework yaitu Queue, dan LinkedList

```
package pekan4;

import java.util.LinkedList;
import java.util.Queue;

public class ContohQueue2 {
    public static void main(String[] args) {
```

3. Lalu inisialisasi/ buat objek Queue bertipe data String dengan nama ‘q’ dan menggunakan LinkedList sebagai implementasi konkret dari Queue.
4. Tambahaan elemen yaitu angkan 0 sampai 5 ke dalam Queue menggunakan perulangan For

```
Queue<Integer> q = new LinkedList<>();
// Tambahkan elemen{0, 1, 2, 3, 4, 5} k
for (int i = 0; i < 6; i++)
    q.add(i);
```

5. Menampilkan semua elemen di dalam queue.

```
// Menampilkan isi antrian.
System.out.println("Elemen Antrian = " + q);
```

6. Menghapus elemen pertama (paling depan) dari queue menggunakan remove()

```
int hapus = q.remove();
System.out.println("Hapus elemen = " + hapus);
System.out.println(q);
```

7. Untuk melihat elemen terdepan tanpa menghapusnya gunakan peek()

```
int depan = q.peek();
System.out.println("Kepala Antrian = " + depan);
```

8. Menghitung berapa banyak elemen saat ini di dalam queue menggunakan size()

```
int banyak = q.size();
System.out.println("Size Antrian = " + banyak);
```

9. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
Elemen Antrian = [0, 1, 2, 3, 4, 5]
Hapus elemen = 0
[1, 2, 3, 4, 5]
Kepala Antrian = 1
Size Antrian = 5
```

10. Berikut kode program versi lengkapnya

```java
package pekan4;

import java.util.LinkedList;
import java.util.Queue;

public class ContohQueue2 {
    public static void main(String[] args) {
        Queue<Integer> q = new LinkedList<>();
        // Tambahkan elemen{0, 1, 2, 3, 4, 5} ke antrian
        for (int i = 0; i < 6; i++)
            q.add(i);

        // Menampilkan isi antrian.
        System.out.println("Elemen Antrian = " + q);

        // Menghapus kepala antrian
        int hapus = q.remove();
        System.out.println("Hapus elemen = " + hapus);
        System.out.println(q);

        // Untuk melihat antrian terdepan
        int depan = q.peek();
        System.out.println("Kepala Antrian = " + depan);

        // Ukuran antrian
        int banyak = q.size();
        System.out.println("Size Antrian = " + banyak);
```

D. Kesimpulan

Berdasarkan praktikum yang telah dilakukan kita dapat memahami bahwa Queue (Antrian) adalah struktur data linear yang mengikuti prinsip FIFO (First In, First Out), di mana elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan.

Queue dapat diimplementasikan menggunakan kelas bawaan Java seperti LinkedList, yang mendukung semua operasi queue dengan efisien. Pada pratikum kali ini kita juga dapat memahami operasi dasar Queue seperti enqueue, dequeue, size, peek dll. Terakhir Queue juga dapat dimodifikasi atau dikombinasikan dengan struktur data lain seperti Stack dan Iterator