LAPORAN PRAKTIKUM STRUKTUR DATA

PEKAN 4

Oleh :

SEPTIAN RIYANDA PUTRA

NIM 2411532016

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU : Dr. Wahyudi, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI
DEPARTEMEN INFORMATIKA UNIVERSITAS ANDALAS

PADANG 2025

A. Pendahuluan

Queue (antrian) adalah salah satu struktur data yang menerapkan prinsip FIFO (First In, First Out), yaitu elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan antrian pada kehidupan sehari-hari, seperti antre di loket atau kasir, di mana orang yang datang lebih dulu akan dilayani lebih dulu.

Dalam bahasa pemrograman Java, struktur data queue dapat diimplementasikan menggunakan berbagai kelas yang disediakan dalam pustaka java.util. Masing-masing kelas memiliki karakteristik tersendiri—**LinkedList** memungkinkan operasi penambahan dan penghapusan dari kedua ujung, **PriorityQueue** menyusun elemen berdasarkan prioritas tertentu, sedangkan **ArrayDeque** menawarkan performa yang lebih efisien dibandingkan LinkedList dalam kebanyakan kasus.

Queue banyak digunakan dalam berbagai skenario pemrograman, seperti pemrosesan tugas secara berurutan, manajemen antrian dalam sistem operasi, serta komunikasi antarproses dan thread.

Praktikum ini bertujuan untuk memahami konsep dasar dan operasi penting dalam queue, seperti menambahkan (enqueue) dan menghapus elemen (dequeue), serta menerapkannya dalam penyelesaian masalah menggunakan bahasa Java secara efisien dan terstruktur.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

- Menerapkan konsep dasar struktur data Queue (FIFO) di Java.
- Melakukan analisis ekspresi menggunakan Queue.

C. Program yang dibuat

a. ContohQueue2

```
1  package Pekan4;
2
3  import java.util.LinkedList;
5
6  public class ContohQueue2 {
7
8      public static void main(String[] args) {
9          Queue<Integer> q = new LinkedList<>();
10         for (int i = 0; i < 6; i++)
11             q.add(i);
12         System.out.println("elemen Antrian "+ q);
13         int hapus = q.remove();
14         System.out.println("Hapus elemen = "+ hapus);
15         System.out.println(q);
16         int depan = q.peek();
17         System.out.println("Kepala Antrian = "+ depan);
18
19         int banyak = q.size();
20         System.out.println("size antrian = "+ banyak);
21     }
22
23 }
24
```

- Queue<Integer> q = new LinkedList<>();

Program ini berfungsi untuk membuat antrian dengan tipe data Integer.

- for (int i = 0; i < 6; i++) q.add(i);

Program ini berfungsi untuk menambahkan angka 0–5 ke dalam antrian dengan menggunakan perulangan for.

- q.remove();

Program ini berfungsi untuk menghapus elemen pertama dari antrian.

- q.peek();

Program ini berfungsi untuk mengambil (mengintip) elemen pertama dari antrian tanpa menghapusnya.

- q.size();

Program ini berfungsi untuk menghitung jumlah elemen yang terdapat dalam antrian.

b. inputQueue

```java
package Pekan4;

public class inputQueue {
int front, rear, size;
int capacity;
int array[];
public inputQueue(int capacity)
{
    this.capacity = capacity;
    front = this.size=0;
    rear= capacity -1;
    array= new int[this.capacity];
}
boolean isFull(inputQueue queue)
{
    return (queue.size==queue.capacity);
}
boolean isEmpty(inputQueue queue)
{
    return (queue.size == 0);
}
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.size = (this.front + 1) %
        this.capacity;
        this.size = this.size -1;
        return item;
    }
    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];
}
}
```

- **Deklarasi Variabel Utama:**
  - front: indeks terdepan antrian.
  - rear: indeks paling belakang antrian.
  - size: jumlah elemen dalam antrian.
  - capacity: kapasitas maksimal antrian.
  - array[]: array penyimpanan elemen queue.

- **Konstruktor inputQueue(int capacity)**

  Program ini berfungsi untuk menginisialisasi kapasitas, front, rear, dan array penyimpanan.

- **isFull()**

  Program ini berfungsi untuk memeriksa apakah antrian sudah dalam keadaan penuh (size == capacity).

- **isEmpty**()

  Program ini berfungsi untuk memeriksa apakah antrian kosong (size == 0).

- **enqueue(int item)**
  - Menambahkan elemen ke antrian.
  - rear digeser secara modular (circular queue).
  - Size antrian ditambah.
  - Cetak item yang masuk.

- **dequeue**()
  - Menghapus elemen dari depan antrian.
  - front digeser secara modular.
  - size dikurangi.
  - Mengembalikan elemen yang dihapus.

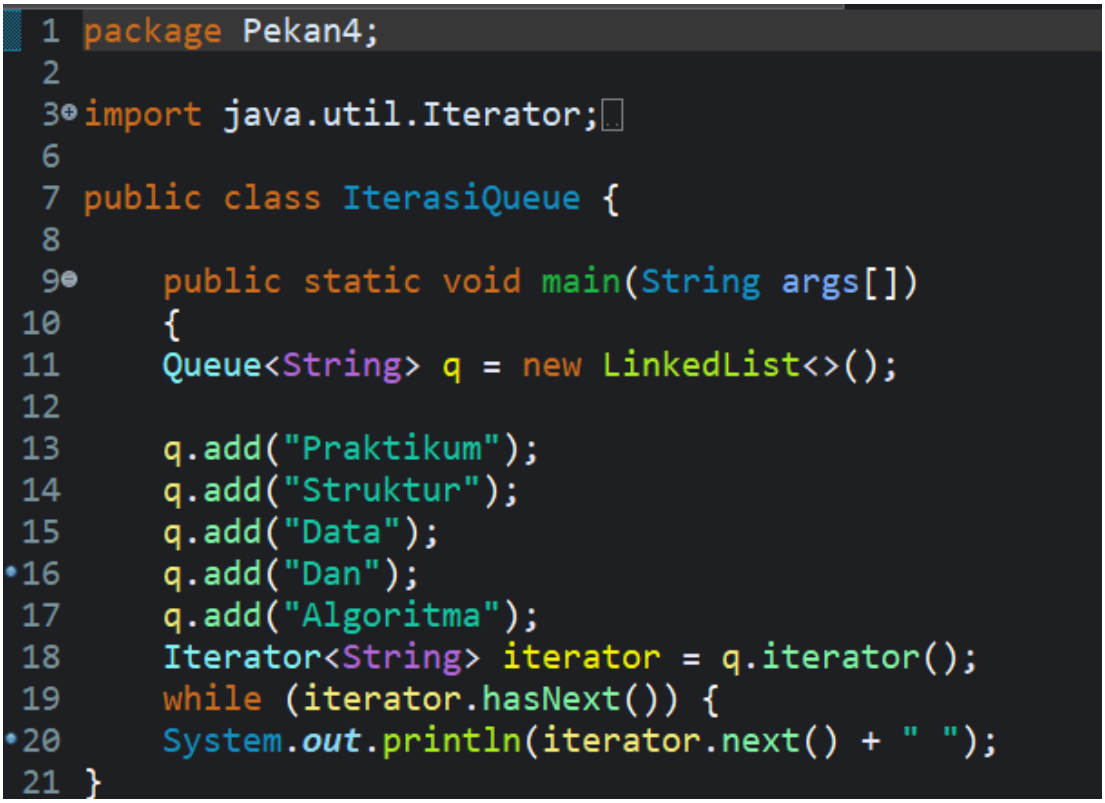
- **front()**

  Program ini berfungsi untuk mengembalikan elemen di depan antrian (tanpa menghapus).Lalu, jika kosong, kembalikan Integer.MIN_VALUE.

- **rear()**

  Program ini berfungsi untuk mengembalikan elemen di belakang antrian. Lalu, Jika kosong, kembalikan Integer.MIN_VALUE.

c. IterasiQueue



```
1  package Pekan4;
2
3  import java.util.Iterator;
6
7  public class IterasiQueue {
8
9      public static void main(String args[])
10     {
11     Queue<String> q = new LinkedList<>();
12
13     q.add("Praktikum");
14     q.add("Struktur");
15     q.add("Data");
16     q.add("Dan");
17     q.add("Algoritma");
18     Iterator<String> iterator = q.iterator();
19     while (iterator.hasNext()) {
20     System.out.println(iterator.next() + " ");
21 }
```

- Queue<String> q = new LinkedList<>();

  Program ini berfungsi untuk membuat antrian q untuk menyimpan elemen dengan tipe data string.

- q.add(...)

  Program ini berfungsi untuk menambahkan elemen "Praktikum", "Struktur", "Data", "Dan", "Algoritma" ke dalam antrian.

- Iterator<String> iterator = q.iterator();

  Program ini berfungsi untuk membuat iterator untuk membaca isi antrian satu per satu.

- while (iterator.hasNext())

  Perulangan ini memastikan bahwa selama masih ada elemen, terus lakukan iterasi.

- System.out.println(iterator.next() + " ");

  Menampilkan setiap elemen dari antrian.

d. ReverseData

```
1  package Pekan4;
2
3  import java.util.LinkedList;
6
7  public class ReverseData {
8
9      public static void main(String[] args) {
10         Queue<Integer> q = new LinkedList<Integer>();
11         q.add(1);
12         q.add(2);
13         q.add(3);
14         System.out.println("sebelum reverse" + q);
15         Stack<Integer> s = new Stack<Integer>();
16         while (!q.isEmpty()) {
17             s.push(q.remove());
18         }
19         while (!q.isEmpty()) {
20         q.add(s.pop());
21         }
22         System.out.println("sesudah reverse= " + q);
23     }
```

- Membuat Queue dengan tipe data Integer lalu mengisinya dengan nilai [1, 2, 3.]
- Menampilkan isi Queue sebelum dibalik.
- Membuat Stack.
- Memindahkan semua elemen dari Queue ke Stack, proses ini menghasilkan elemen dengan urutan yang terbalik.
- Memindahkan elemen dari Stack kembali ke Queue, hasilnya Queue sekarang berisi elemen dalam urutan terbalik.
- Menampilkan Queue setelah dibalik.

e. TestQueue

```java
package Pekan4;

public class TestQueue {
    public static void main(String[] args) {
        inputQueue queue = new inputQueue(1000);
        queue.enqueue(10);
        queue.enqueue(20);
        queue.enqueue(30);
        queue.enqueue(40);

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

- Membuat objek queue baru dari kelas inputQueue dengan kapasitas 1000.
- Menambahkan elemen ke dalam queue menggunakan enqueue(10), enqueue(20), enqueue(30), dan enqueue(40).
- Menampilkan elemen paling depan menggunakan front() dan paling belakang menggunakan rear().
- Menghapus satu elemen dari depan queue menggunakan dequeue() dan mencetaknya.
- Menampilkan kembali elemen depan dan belakang setelah satu elemen di-dequeue.
- menunjukkan cara kerja dasar struktur data queue (enqueue, dequeue, front, rear).

D. Kesimpulan

Dalam sesi praktikum ini, kami mempelajari sekaligus menerapkan konsep **struktur data queue (antrian)** dengan menggunakan bahasa pemrograman **Java**. Queue merupakan struktur data linear yang bekerja berdasarkan prinsip **FIFO (First In, First Out)**, di mana elemen yang pertama kali dimasukkan ke dalam antrian akan menjadi yang pertama dikeluarkan. Untuk mengimplementasikannya, kami memanfaatkan kelas-kelas dari pustaka Java seperti **Queue** dan **LinkedList**, serta menggunakan beberapa metode dasar seperti add(), remove(), dan peek() guna memanipulasi data di dalam antrian.

Melalui praktikum ini, kami memperoleh pemahaman yang lebih mendalam tentang bagaimana struktur queue dapat digunakan untuk menyelesaikan berbagai permasalahan nyata, seperti **penjadwalan proses**, **antrian pelanggan**, hingga **pengolahan data secara berurutan**. Pemahaman ini memperkaya wawasan kami dalam merancang solusi pemrograman yang menuntut pengelolaan data secara sistematis dan efisien. Dengan menguasai implementasi queue di Java, kami merasa lebih siap dalam menghadapi tantangan pemrograman yang berkaitan dengan pengaturan aliran data secara logis dan terstruktur.