LAPORAN PRATIKUM

"PEKAN 4"

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

DOSEN PENGAMPU:

Wahyudi, Dr. S.T. M.T.

DISUSUN OLEH:

Karimah Irsyadiyah (2411533018)

UNIVERSITAS ANDALAS

T.A 2024/2025

Daftar Pustaka

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pemrograman, struktur data memegang peranan penting dalam mengelola dan memanipulasi data. Salah satu struktur data yang umum digunakan adalah **queue** (antrian), di mana elemen ditambahkan dari belakang dan dihapus dari depan (FIFO: First In First Out). Praktikum ini bertujuan untuk memperkenalkan berbagai implementasi antrian menggunakan Java.

1.2 Tujuan

- Memahami konsep dasar antrian (queue) dan stack.
- Menerapkan operasi dasar pada queue: `enqueue`, `dequeue`, `peek`, dan `size`.
- Menggunakan kelas `Queue` dan `LinkedList` dari `java.util`.
- Menunjukkan bagaimana antrian dapat dibalik menggunakan `Stack`

1.3 Alat dan Bahan

- Perangkat Keras: Laptop/PC
- Perangkat komputer dengan IDE (contoh: IntelliJ IDEA / Eclipse).
- Java Development Kit (JDK).

BAB II
PEMBAHASAN

2.1 Langkah-langkah Praktikum dan Pembahasan Program

A. Class : ContohQueue2.java

- Membuat objek queue bertipe Integer menggunakan LinkedList.
- Menambahkan elemen 0 hingga 5 ke dalam antrian.
- Menampilkan isi antrian.
- Menghapus elemen pertama menggunakan remove().
- Menampilkan isi antrian setelah penghapusan.
- Melihat elemen terdepan menggunakan peek().
- Menampilkan jumlah elemen dalam antrian dengan size().

```
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());
		}
}
```

B. Class : inputQueue.java dan TestQueue.java

1. Membuat class inputQueue yang mengimplementasikan queue secara manual menggunakan array.
2. Mengatur atribut front, rear, dan size untuk pengelolaan antrian.
3. Menyediakan method:

- enqueue(): untuk menambahkan elemen.
- dequeue(): untuk menghapus elemen depan.
- front() dan rear(): untuk mengakses elemen terdepan dan terakhir.

4. Di TestQueue.java, dilakukan pengujian dengan:

   - Menambahkan elemen 10, 20, 30, 40.
   - Menampilkan elemen depan dan belakang.
   - Melakukan satu kali dequeue() lalu menampilkan kembali posisi front dan rear.

```
package pekan4;

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

    public inputQueue(int capacity) {
        this.capacity = capacity;
        this.size = 0;
        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.front = (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];
    }
}


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());
                }
}
```

C. Class : IterasiQueue.java

Langkah-langkah:

1. Membuat queue bertipe String dan menambahkan beberapa kata ke dalamnya.
2. Menggunakan Iterator untuk menelusuri seluruh elemen dalam antrian.
3. Menampilkan seluruh elemen menggunakan while (iterator.hasNext()).

```
package pekan4;

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

public class IterasiQueue {
   public static void main(String args[]) {
      Queue<String> q = new LinkedList<>();

      q.add("Praktikum");
      q.add("Struktur");
      q.add("Data");
      q.add("Dan");
      q.add("Algoritma");

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

D. Class ReverseData.java

Langkah-langkah:

1. Membuat queue q dan stack s.

2. Menambahkan angka 1, 2, 3 ke dalam queue.
3. Memindahkan semua elemen dari queue ke stack (q -> s).
4. Memindahkan kembali dari stack ke queue (s -> q) agar urutan terbalik.
5. Menampilkan hasil queue setelah reverse.

```
package pekan4;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

public class ReverseData {
    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);

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

        while (!s.isEmpty()) { // S -> Q
            q.add(s.pop());
        }

        System.out.println("sesudah reverse= " + q); // [3, 2, 1]
    }
}
```

BAB III
PENUTUP

3.1 Kesimpulan

Dari praktikum pekan ini, dapat disimpulkan bahwa struktur data **queue** (antrian) merupakan struktur yang menggunakan prinsip **FIFO (First In First Out)**, di mana elemen yang pertama masuk akan menjadi yang pertama keluar. Melalui implementasi menggunakan kelas LinkedList dari Java maupun implementasi manual dengan array, mahasiswa dapat memahami cara kerja operasi dasar queue seperti enqueue, dequeue, peek, dan size. Selain itu, penggunaan Iterator memungkinkan untuk melakukan iterasi terhadap elemen-elemen dalam antrian. Praktikum ini juga menunjukkan bagaimana struktur data **stack** dapat dimanfaatkan untuk membalik urutan elemen dalam queue, sehingga memberikan pemahaman yang lebih luas tentang bagaimana struktur data dapat saling berinteraksi. Secara keseluruhan, praktikum ini memberikan dasar yang kuat dalam memahami dan mengimplementasikan konsep antrian dan stack dalam pemrograman Java.