

LAPORAN PRAKTIKUM STRUKTUR DATA  
IMPLEMENTASI QUEUE PADA PEMROGRAMAN JAVA



Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

ASISTEN PRAKTIKUM : RAHMAT DWIRIZKI OLDERS

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

## **A. Pendahuluan**

Praktikum ini dilakukan untuk membuat program yang dapat menyimpan/mengelola data dengan implementasi struktur data pada java yaitu Queue. Sebagai salah satu struktur data yang umum digunakan adalah Queue (Antrian), banyak digunakan dalam berbagai aplikasi seperti pembelian tiket, pemrosesan data dalam jaringan, pencetakan dokumen dan lainnya. Mengimplementasikan dengan program java yang sudah dipelajari sebelumnya seperti if statement, looping, scanner class, OOP dan lainnya dalam pemrograman Java.

### **1. Queue (Tumpukan)**

Struktur data FIFO (First-In-First-Out) untuk pemrosesan berurutan, dimana data pertama yang ditambahkan ke dalam antrian akan dikeluarkan pertama. Visual operasinya dimana elemen ditambahkan pada satu ujung (rear) dan dihapus dari ujung lainnya (front). Dapat diimplementasikan dengan array, arraylist dan linkedlist. Jenis jenis antrian yaitu antrean masukan terbatas, output terbatas, melingkar, double ended dan antrean prioritas. Digunakan dalam algoritma dan aplikasi karena sederhana dan efisiensinya mengelola data.

Method :

- Enqueue() : Sisipkan, menambahkan elemen ke bagian belakang antrean
- Dequeue() : menghapus dan mengembalikan elemen dari bagian depan antrean
- Peek() : Mengembalikan elemen teratas pada stack, seperti get().
- isEmpty() : Memeriksa apakah tumpukan kosong
- Size() : Menemukan jumlah elemen dalam tumpukan

## **B. Tujuan**

Tujuan dari dilakukannya praktikum ini adalah :

1. Memahami dan mengaplikasikan Queue dalam program java untuk menyimpan data.
2. Implementasi method Queue pada program java.
3. Dapat mengaplikasikan statement (if, while), class (Scanner), dll. yang dipelajari sebelumnya ke dalam program.
4. Membuat class OOP(Object Oriented Programming) dengan implementasi Queue.

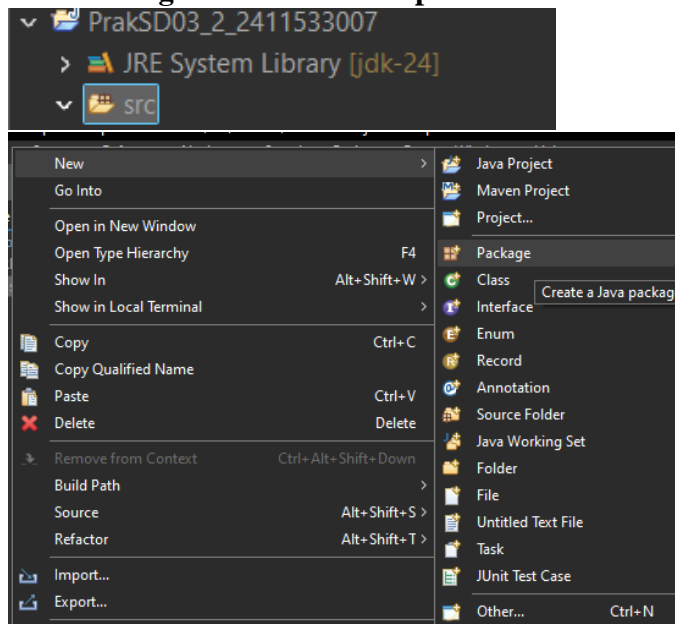
## **C. Langkah kerja praktikum**

### **a. Alat dan Bahan**

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integreted Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)

## b. Package pekan 4

1. Buat new package. **Buka java project** yang telah dibuat sebelumnya, lalu **klik kanan** pada folder 'src', setelahnya akan muncul list option, pilih 'new', lalu 'Package' dan beri nama 'pekan4'.



## c. Program ContohQueue2

1. **Buat** terlebih dahulu new **class** dengan klik kanan pada package pekan4 dan beri nama "**ContohQueue2**", centang method public static void.
2. **Import Queue** class dari paket **Java.util.Stack** dan **LinkedList** dari paket **java.util.LinkedList** untuk diimplementasikan dengan Queue. Lakukan import untuk menggunakan fungsi/method dari classnya.

```
1 package pekan4;
2 import java.util.Queue;
3 import java.util.LinkedList;
4
5 public class ContohQueue2 {
```

3. **Deklarasi object Queue** bertipe **Integer** dengan identifiernya "q".

```
7 public static void main(String[] args) {
8     Queue<Integer> q = new LinkedList<>();
```

4. Tambahkan elemen ke dalam objek Queue menggunakan for loop. For loop dengan inisialisasi awal 0, kondisi sampai kecil dari 6, serta iterasi penambahan satu untuk tiap perulangan. Buat method **add()** dalam blok kode yang akan **menambahkan nilai** ke dalam **objek queue** sesuai dengan nilai dari iterasi dari loop, yang akan dijalankan selama For loop kondisi bernilai true.

```
9         //tambah elemen {0, 1, 2, 3, 4, 5} ke antrian
10        for (int i = 0; i < 6; i++) {
11            q.add(i);
12        }
```

5. **Tampilkan data** tersimpan pada objek Queue menggunakan **sysout**.

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

6. Menghapus nilai tersimpan pada objek queue menggunakan method `remove()`, lalu disimpan pada variable integer “hapus”, sesuai konsep ‘reuse’ pada variabel. Menampilkan data yang dihapus menggunakan `sysout` dengan memanggil variabel “hapus” sebelumnya, dimana data dari method `remove()` disimpan.

```
15 //Untuk menghapus kepala antrian
16 int hapus = q.remove();
17 System.out.println("Hapus elemen = " + hapus);
18 System.out.println(q);
```

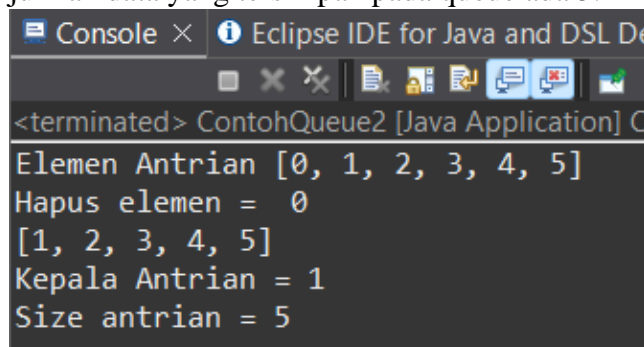
7. Melihat data pada posisi terdepan yang tersimpan pada objek queue, menggunakan method `peek()`, yang lalu disimpan pada variabel integer “depan”. Menampilkannya menggunakan `sysout` dengan memanggil variabel “depan”.

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

8. Menampilkan jumlah data element tersimpan pada object queue, menggunakan method `size()` yang lalu disimpan pada variabel integer “banyak”. Menampilkannya menggunakan `sysout` dengan memanggil variabel “banyak”.

```
22 //Untuk menampilkan jumlah element tersimpan
23 int banyak = q.size();
24 System.out.println("Size antrian = " + banyak);
25 }
26 }
```

9. Output program. Menampilkan data yang tersimpan pada Queue, yang sebelumnya diinputkan menggunakan method `add()` dengan menggunakan for loop. Data awal yang di inputkan akan berada pada posisi terdepan, dan data terakhir diinputkan akan berada di paling diakhir antrian, jadi ketika method `remove()` akan menghapus data paling depan yaitu 0, lalu method `peek` yang akan mengembalikan data paling depan antrian yaitu 1. Jadi jumlah data yang tersimpan pada queue ada 5.



```
<terminated> ContohQueue2 [Java Application] C
Elemen Antrian [0, 1, 2, 3, 4, 5]
Hapus elemen = 0
[1, 2, 3, 4, 5]
Kepala Antrian = 1
Size antrian = 5
```

#### d. Program IterasiQueue

1. **Buat new class** dengan klik kanan pada package `pekan4` dan beri nama ‘**IterasiQueue**’, centang method `public static void`.
2. **Import** Queue, LinkedList dan Iterator dari package **Java.util.** untuk menggunakan fungsi/method dari classnya.

```

1 package pekan4;
2 import java.util.Queue;
3 import java.util.LinkedList;
4 import java.util.Iterator;

```

3. Instiasi objek **Queue** bertipe String dengan identifer “q”, dengan implementasi **LinkedList**.

```

6 public class IterasiQueue {
7
8     public static void main(String[] args) {
9         Queue<String> q = new LinkedList<>();

```

4. Menambahkan data pada object “q” menggunakan **method add()**, sebagai hasil implementasi **LinkedList** pada **Queue**.

```

10        q.add("Praktikum");
11        q.add("Struktur");
12        q.add("Data");
13        q.add("Dan");
14        q.add("Algoritma");

```

5. Instiasi object **Iterator** bertipe String dengan identifer “iterator”. **Iterator** merupakan sebuah **interface** dari package **java.util**, digunakan untuk **mengakses elemen-elemen dalam koleksi (collection)** secara berurutan (berupa **LinkedList**, **ArrayList**), tanpa harus mengetahui struktur internal koleksi tersebut.

```

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

```

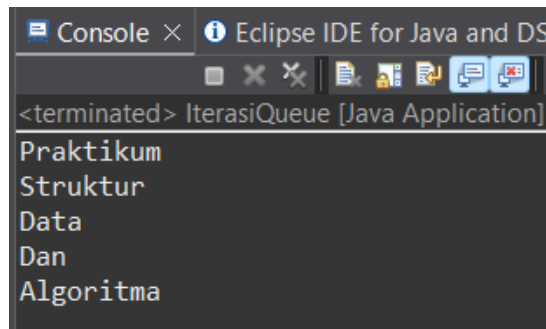
6. Buat **While** loop dengan kondisi menggunakan **method iterator**. Selama masih ada elemen selanjutnya yang belum diambil dari koleksi, jalankan blok kode di dalam **while** loop. Perulangan akan terus berjalan selama koleksi belum habis ditelusuri. Lalu tampilkan data menggunakan **sysout** dengan memanggil **method iterator**.

```

16        while (iterator.hasNext()) {
17            System.out.println(iterator.next() + " ");
18        }
19    }
20 }

```

7. Output program : Program membuat sebuah antrian (**Queue**) yang berisi lima elemen string: "Praktikum", "Struktur", "Data", "Dan", dan "Algoritma". Menggunakan **Iterator**, program menelusuri setiap elemen dalam antrian dan mencetaknya ke konsol menggunakan **System.out.println()**. Karena **println()** mencetak setiap elemen diikuti dengan baris baru, hasil output di konsol akan menampilkan setiap kata berada di baris yang terpisah. Oleh karena itu, output yang dihasilkan terdiri dari lima baris dengan masing-masing kata: "**Praktikum**", "**Struktur**", "**Data**", "**Dan**", dan "**Algoritma**", yang menunjukkan bahwa antrian ditelusuri dan ditampilkan secara berurutan sesuai urutan penambahannya.



**e. Program inputQueue**

1. **Buat new class** dengan klik kanan pada package pekan 4 dan beri nama “**inputQueue**”, centang method public static void. Program ini merupakan penerapan OOP dari Queue menggunakan array secara manual.

```
1 package pekan4;  
2  
3 public class inputQueue {
```

2. Deklarasikan variabel integer sebagai berikut, sebagai **attribut class**.

```
4     int front, rear, size;  
5     int capacity;  
6     int array[];
```

3. Buat **konstruktor** dengan parameter integer “capacity”. Menerima nilai “capacity” saat object dibuat. Inisialisasi front dan size ke 0. rear di-set ke capacity - 1 untuk mendukung circular queue. Membuat array berukuran sesuai kapasitas.

```
7●     public inputQueue(int capacity) {  
8         this.capacity = capacity;  
9         front = this.size = 0;  
10        rear = capacity - 1;  
11        array = new int [this.capacity];  
12    }
```

4. Membuat **method** “**isFull**” dengan parameter “inputQueue queue”. Mengembalikan true jika **antrian** sudah **penuh**.

```
13●    boolean isFull (inputQueue queue) {  
14        return (queue.size == queue.capacity);  
15    }
```

5. Membuat **method** “**isEmpty**” dengan parameter “inputQueue queue”. Mengembalikan true jika **antrian** masih **kosong**..

```
16●    boolean isEmpty (inputQueue queue) {  
17        return (queue.size == 0);  
18    }
```

6. Membuat **method** “**enqueue**” dengan parameter int “item”, untuk **Menambahkan data ke Antrian**. Jika queue penuh, tidak bisa menambah dan langsung keluar. Kalau belum penuh: rear digeser maju, dan pakai % capacity agar kembali ke awal jika sudah mentok (circular). Simpan item di array posisi rear. Tambahkan size. Cetak pesan ke konsol.

```

19● void enqueue (int item) {
20     if (isFull (this))
21         return;
22     this.rear = (this.rear + 1) %
23         this.capacity;
24     this.array[this.rear] = item;
25     this.size = this.size + 1;
26     System.out.println(item + " enqueued to queue");
27 }

```

7. Membuat **method “dequeue”** ,untuk **Menghapus data dari Antrian**. Jika queue kosong, kembalikan ‘Integer.MIN\_VALUE’ sebagai penanda gagal. Kalau tidak kosong : Ambil data dari front. Geser front ke depan (circular). Kurangi size. Kembalikan nilai yang dihapus.

```

28● int dequeue() {
29     if (isEmpty(this))
30         return Integer.MIN_VALUE;
31
32     int item = this.array[this.front];
33     this.front = (this.front + 1) %
34         this.capacity;
35     this.size = this.size - 1;
36     return item;
37 }

```

8. Membuat **method “front”** ,untuk **Melihat Elemen Paling Depan**. Mengembalikan nilai **elemen paling depan** tanpa menghapusnya.

```

38● int front() {
39     if (isEmpty(this))
40         return Integer.MIN_VALUE;
41
42     return this.array[this.front];
43 }

```

9. Membuat **method “rear”** ,untuk **Melihat Elemen Paling belakang**. Mengembalikan nilai elemen paling belakang tanpa menghapusnya.

```

44● int rear() {
45     if (isEmpty(this))
46         return Integer.MIN_VALUE;
47
48     return this.array[this.rear];
49 }
50 }

```

10. Penjelasan program : implementasi struktur data queue (antrian) menggunakan array secara manual dengan konsep circular queue, yang memungkinkan elemen untuk bergeser kembali ke indeks awal saat mencapai batas kapasitas. Program ini memiliki variabel utama seperti front, rear, size, dan capacity untuk mengatur posisi elemen dalam antrian, serta menyediakan berbagai method seperti enqueue() untuk menambahkan elemen ke belakang antrian, dequeue() untuk menghapus elemen dari depan, isFull() dan isEmpty() untuk mengecek kondisi antrian, serta front() dan rear() untuk melihat isi antrian tanpa mengubahnya. Metode enqueue dan dequeue menggunakan operasi modulus (%) untuk menjaga agar indeks tetap valid meskipun berputar, sehingga efisien dalam pemanfaatan ruang array. Program ini dirancang agar

mampu menangani operasi antrian secara optimal meskipun dalam keterbatasan kapasitas array.

**f. Program TestQueue**

1. **Buat new class** dengan klik kanan pada package pekan2 dan beri nama ‘TestQueue’, centang method public static void.

```
1 package pekan4;  
2  
3 public class TestQueue {
```

2. **Instiasi object InputQueue** dari class InputQueue dengan konsep OOP, dengan identifier “queue”, inisialisasi capacity nya 1000, yaitu element yang dapat ditampung.

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

3. Menambahkan data pada object queue dengan method enqueue().

```
7     queue.enqueue(10);  
8     queue.enqueue(20);  
9     queue.enqueue(30);  
10    queue.enqueue(40);
```

4. Menampilkan data yang tersimpan terdepan pada object queue menggunakan sysout dengan memanggil method front().

```
11    System.out.println("Front item is " + queue.front());
```

5. Menampilkan data yang tersimpan paling belakang antrian pada object queue menggunakan sysout dengan memanggil method rear().

```
12    System.out.println("Rear item is " + queue.rear());
```

6. Menampilkan data yang dikeluarkan dari antrian pada object queue menggunakan sysout dengan memanggil method dequeue().

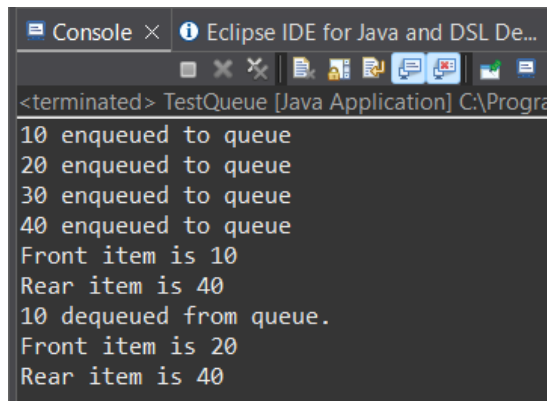
```
13    System.out.println(queue.dequeue() + " dequeued from queue.");
```

7. Menampilkan data paling depan dan belakang pada antrian saat ini pada object “queue” menggunakan sysout dengan memanggil method front() dan rear().

```
14    System.out.println("Front item is " + queue.front());  
15    System.out.println("Rear item is " + queue.rear());  
16    }  
17 }
```

8. Output program : Program ini menampilkan cara kerja antrian (queue) dengan menambahkan empat elemen yaitu 10, 20, 30, dan 40, lalu menunjukkan elemen terdepan dan terakhir sebelum dan sesudah satu elemen dihapus. Setelah semua elemen dimasukkan, elemen terdepan (front) adalah 10 dan elemen terakhir (rear) adalah 40. Ketika dequeue() dipanggil, elemen 10 dihapus dari antrian. Output selanjutnya menunjukkan bahwa elemen terdepan bergeser menjadi 20, sedangkan elemen terakhir tetap 40. Kesimpulannya, output program ini menunjukkan bahwa antrian bekerja sesuai prinsip FIFO (First In First Out), di mana elemen pertama yang masuk adalah yang pertama keluar, dan posisi front serta rear diperbarui sesuai operasi yang dilakukan.





```
<terminated> TestQueue [Java Application] C:\Progra
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
```

#### g. Program ReverseData

1. **Buat new class** dengan klik kanan pada package pekan4 dan beri nama “ReverseData”, centang method public static void
2. **Import** Queue, Stack dan LinkedList dari paket **Java.util.** terlebih dahulu untuk menggunakan fungsi/method dari classnya.

```
1 package pekan4;
2 import java.util.Queue;
3 import java.util.Stack;
4 import java.util.LinkedList;
5
```

3. **Instansi object Queue** bertiper integer dengan identifier “q”, dengan implementasi LinkedList.

```
6 public class ReverseData {
7
8     public static void main(String[] args) {
9         Queue <Integer> q = new LinkedList<Integer>();
```

4. Menambahkan data pada object “q” menggunakan method add().

```
10         q.add(1);
11         q.add(2);
12         q.add(3); // [1,2,3]
```

5. Menampilkan data tersimpan pada object “q” menggunakan sysout.

```
13         System.out.println("Sebelum reverse " + q);
```

6. Instansi objek Stack bertipe integer dengan identifier “s”.

```
14         Stack<Integer> s = new Stack<Integer>();
```

7. While loop dengan kondisi selama object element tersimpan pada object “q” tidak kosong (menggunakan method isEmpty()), maka akan dijalankan blok kodenya yaitu melakukan push()/menambahkan data pada object “s” dari hasil remove data pada object “q”.

```
15         while (!q.isEmpty()) { // Q -> S
16             s.push(q.remove());
17         }
```

8. While loop dengan kondisi selama object element tersimpan pada object “s” tidak kosong (menggunakan method isEmpty()), maka akan dijalankan blok kodenya yaitu melakukan add()/menambahkan data pada object “q” dari hasil pop()/mengeluarkan data pada object “s”.

```

18         while (!s.isEmpty()) { // S -> Q
19             q.add(s.pop());
20         }

```

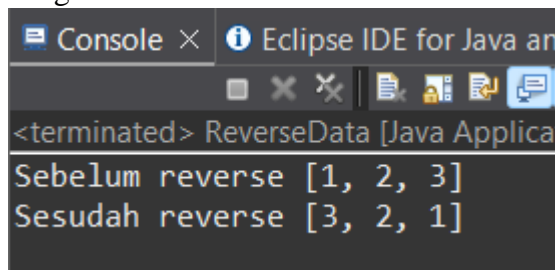
9. Menampilkan data tersimpan pada object q menggunakan sysout.

```

21         System.out.println("Sesudah reverse " + q); // {3,2,1}
22     }
23 }

```

10. Output program : Program ini bertujuan untuk membalik urutan elemen-elemen dalam sebuah **queue** menggunakan bantuan **stack**. Pertama, elemen 1, 2, dan 3 dimasukkan ke dalam queue sehingga urutannya menjadi [1, 2, 3]. Output pertama mencetak isi queue sebelum dibalik: "Sebelum reverse [1, 2, 3]". Kemudian, semua elemen queue dipindahkan satu per satu ke dalam stack menggunakan `s.push(q.remove())`, yang secara otomatis membalik urutannya karena stack bekerja dengan prinsip LIFO (Last In, First Out). Setelah itu, elemen dalam stack dipindahkan kembali ke queue menggunakan `q.add(s.pop())`, sehingga queue kini berisi elemen dalam urutan terbalik: [3, 2, 1]. Output kedua mencetak isi queue setelah proses pembalikan: "Sesudah reverse [3, 2, 1]". Jadi, program ini menunjukkan cara membalik isi queue dengan memanfaatkan struktur data stack.



```

Console x Eclipse IDE for Java an
<terminated> ReverseData [Java Applicat
Sebelum reverse [1, 2, 3]
Sesudah reverse [3, 2, 1]

```

#### D. Kesimpulan

Setelah melakukan praktikum ini dapat memahami dan mengimplementasikan Queue (Antrian) dan methodnya. Membuat program dengan konsep OOP seperti InputQueue. Menerapkan Queue (Antrian) untuk menyelesaikan permasalahan nyata, seperti antrian pembelian. Menggunakan while loop dalam method main untuk membuat program lebih terstruktur. Queue bekerja dengan prinsip FIFO (First In First Out), di mana elemen yang pertama masuk akan menjadi elemen pertama yang keluar.

Struktur data Queue (antrian) dapat diimplementasikan dalam Java baik menggunakan class bawaan seperti Queue, LinkedList, dan Iterator, maupun dengan cara manual menggunakan array biasa. Melalui program seperti IterasiQueue, mahasiswa mempelajari cara menambahkan data ke queue menggunakan `add()` dan menelusuri isi queue menggunakan Iterator dan loop while. Sedangkan pada program ReverseData, peserta memahami bagaimana stack dapat digunakan untuk membalik urutan elemen dalam queue, yang menunjukkan penerapan kombinasi dua struktur data untuk mencapai manipulasi data tertentu.

program inputQueue dan TestQueue memperkenalkan implementasi queue secara manual menggunakan array dan konsep circular queue, yang memungkinkan penggunaan ruang array secara efisien tanpa harus menggeser elemen. Program ini

memperlihatkan bagaimana operasi dasar queue seperti enqueue, dequeue, front, dan rear dapat dibangun dari nol dengan logika pemrograman yang tepat.