

VERSION 1.1  
JANUARI 25, 2024



# [STRUKTUR DATA]

MODUL 3, STACK & QUEUE

DISUSUN OLEH:  
IZZA IHSAN FATHONY  
MOCH IQBAL ARIZKI WIDYANSYAKH

DIAUDIT OLEH:  
MUHAMMAD ILHAM PERDANA. S.TR.T., M.T.

LAB. INFORMATIKA  
UNIVERSITAS MUHAMMADIYAH MALANG

## [STRUKTUR DATA]

---

### PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi praktikum dengan baik, sesuai dengan materi yang diberikan oleh dosen pengajar dikelas. Terutama dalam penerapan materi OOP JAVA:

1. Array
2. LinkedList
3. Stack
4. Queue

---

### TUJUAN

Mahasiswa mampu menguasai dan menjelaskan konsep dari Struktur Data Stack & Queue.

---

### TARGET MODUL

Mahasiswa mampu memahami:

1. Contoh penggunaan *Stack*
2. Contoh penggunaan *Queue*
3. Contoh pengoperasian *Stack*
4. Contoh pengoperasian *Queue*

---

### PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (IntelliJ IDEA, Eclipse, Netbeans, dll)

---

### REFERENSI MATERI

Youtube:

[https://www.youtube.com/results?search\\_query=stack+dan+queue+java](https://www.youtube.com/results?search_query=stack+dan+queue+java)

<https://youtu.be/hj6E7LyZKNI?si=SpnDNBP1LGgFppsP>

<https://www.youtube.com/watch?v=3F5kq-4jxPI>

Artikel:

<https://www.scaler.com/topics/java/stack-and-queue-in-java/>

<https://www.educative.io/blog/data-structures-stack-queue-java-tutorial>

<https://www.geeksforgeeks.org/queue-using-stacks/>

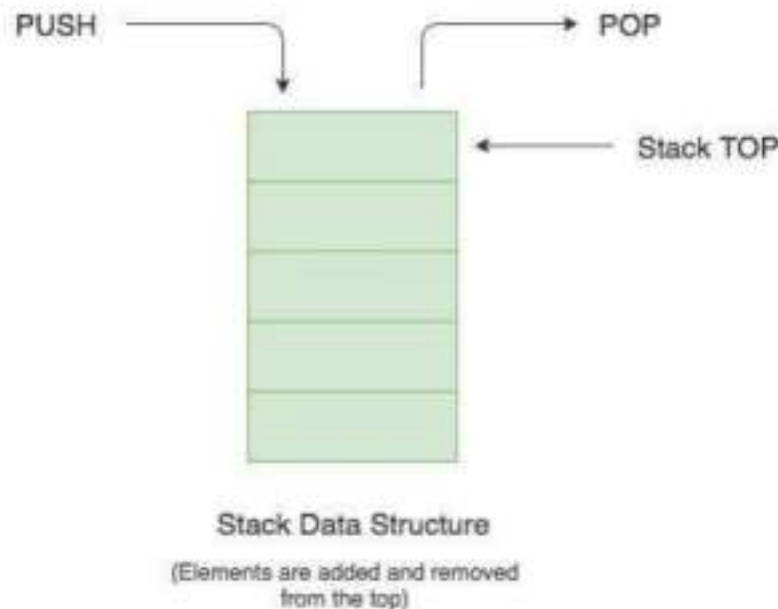
Note: Dari referensi tersebut mungkin terdapat sedikit perbedaan satu sama yang lain, cukup pahami konsepnya dan terapkan pada kasus di modul ini.

## MATERI POKOK

### 1. Stack

Sebuah stack dapat dianalogikan dengan suatu tumpukan benda, sekumpulan data yang diletakkan diatas data yang lain. Elemennya dapat di ambil dan di tambahkan pada posisi akhir/puncak (top) saja. Data yang terletak ditengah atau berada paling bawah dapat di abil apabila data yang terletak di atas nya sudah diambil terlebih dahulu.

Operasi stack dapat dilakukan pada elemen pada top dari stack. Yaitu Push() menambah item pada top, Pop() menghapus elemen dari top, Peek() mengakses nilai pada top. Stack memiliki urutan LIFO (Last-In-First-Out).



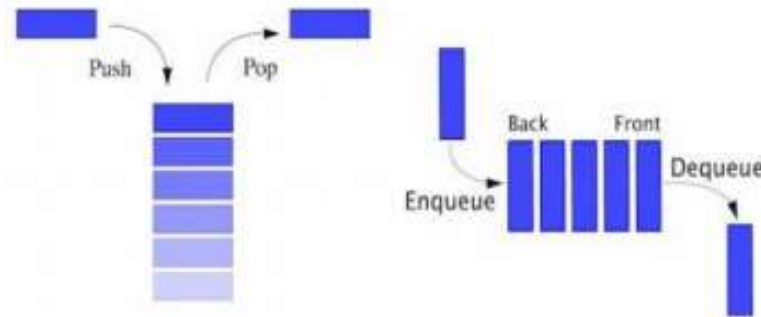
#### a. Method yang terdapat pada kelas Stack

Method	Description
boolean empty()	Menghasilkan nilai <b>True</b> jika stack kosong, dan nilai <b>False</b> jika stack berisi elemen.
object peek()	Menghasilkan elemen pada top stack, tetapi tidak me <i>remove</i> .
object pop()	Menghasilkan elemen pada top stack, dan mengambil/menghapus ( <i>remove</i> ) elemen tersebut.
object push()	Menambahkan elemen pada stack.
search (object element)	Mencari elemen dalam stack. Jika ditemukan, menghasilkan offset dari top stack. Sebaliknya jika tidak menghasilkan nilai -1.

## 2. Queue

Queue adalah kumpulan data yang mana penambahan elemennya hanya dapat dilakukan di satu sisi yang disebut belakang (tail), dan penghapusan elemennya dilakukan pada sisi lain atau bisa disebut sisi depan (head).

Operasi queue bekerja pada ujung list, head, dan tail. Berbeda dengan stack, queue memiliki urutan FIFO (First-In-First-Out). Enqueue() menambah item pada tail dan Dequeue() menghapus item pada head.



Dalam kehidupan sehari-hari queue dapat dianalogikan seperti antrian pada penjualan tiket kereta api, Dimana orang yang pertama datang adalah orang yang pertama kali dilayani untuk membeli tiket. Jika ada orang baru yang datang akan membeli tiket, maka posisinya berada pada urutan paling belakang dalam antrian tersebut. Orang yang berada pada posisi terakhir dalam antrian adalah yang terakhir kali dapat dilayani dan memperoleh tiket kereta api (kalau kurang beruntung, maka akan kehabisan tiket).

### A. Method pada Interface Queue

Method	Description
add (element)	Metode ini digunakan untuk menambahkan elemen di akhir dari sebuah queue. Secara khusus, elemen ditambahkan di bagian terakhir dari linked-list jika digunakan, atau sesuai dengan prioritas dalam implementasi priority queue.
peek()	Metode ini digunakan melihat elemen yang berada di kepala antrian tanpa menghapusnya. Metode ini akan mengembalikan <b>Null</b> jika antrian kosong.
offer(element)	Metode ini digunakan untuk menyisipkan suatu elemen dalam antrian. Metode ini lebih disukai dibandingkan metode add() karena metode ini tidak melemparkan pengecualian ketika kapasitas wadah penuh, melainkan mengembalikan <b>False</b> .
element()	Metode ini mirip dengan peek(). Metode ini melemparkan <b>NoSuchElementException</b> ketika antrian kosong.
poll()	Metode ini menghapus dan mengembalikan elemen yang berada di kepala antrian. Metode ini mengembalikan <b>Null</b> jika antrian kosong.
remove()	Metode ini menghapus dan mengembalikan elemen yang berada di kepala antrian. Metode ini melemparkan <b>NoSuchElementException</b> ketika antrian kosong.

### 3. Contoh Implementasi Program Stack

#### a. Membuat Program Struktur Data Stack Menggunakan Library

```
import java.util.Stack;

public class StackLibrary {
    public static void main(String[] args) {
        Stack s = new Stack();

        System.out.println(s.empty());

        s.push("Bebek");
        s.push("Angsa");
        s.push("Kuda");
        s.push("Buaya");
        s.push("Tikus");

        System.out.println(s.empty());

        System.out.println("Peek: " + s.peek());
        System.out.println("Animals: " + s);

        s.pop();
        s.pop();

        System.out.println("Animals: " + s);
        System.out.println("Position of Kuda: " + s.search("Kuda"));
    }
}
```

- b. Membuat Program Struktur Data Stack Menggunakan Array Tanpa Library
- i. Membuat Class Stack

```
public class Stack {  
    private int maxSize;  
    private long[] stackArray;  
    private int top;  
  
    Stack(int s) {  
        maxSize = s;  
        stackArray = new long[maxSize];  
        top = -1;  
    }  
    public void push(long j) {  
        stackArray[++top] = j;  
    }  
    public long pop() {  
        return stackArray[top--];  
    }  
    public long peek() {  
        return stackArray[top];  
    }  
    public boolean isEmpty() {  
        return (top == -1);  
    }  
    public boolean isFull() {  
        return (top == maxSize-1);  
    }  
}
```

## ii. Membuat Class Main

```
public class Main {
    public static void main(String[] args) {
        Stack theStack = new Stack(10);
        theStack.push(9);
        theStack.push(30);
        theStack.push(10);
        theStack.push(100);

        while (!theStack.isEmpty()) {
            long value = theStack.pop();
            System.out.print(value);
            System.out.print(" ");
        }
        System.out.println("");
    }
}
```

## 4. Contoh Implementasi Program Queue

### a. Membuat Program Struktur Data Queue Menggunakan Library

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

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

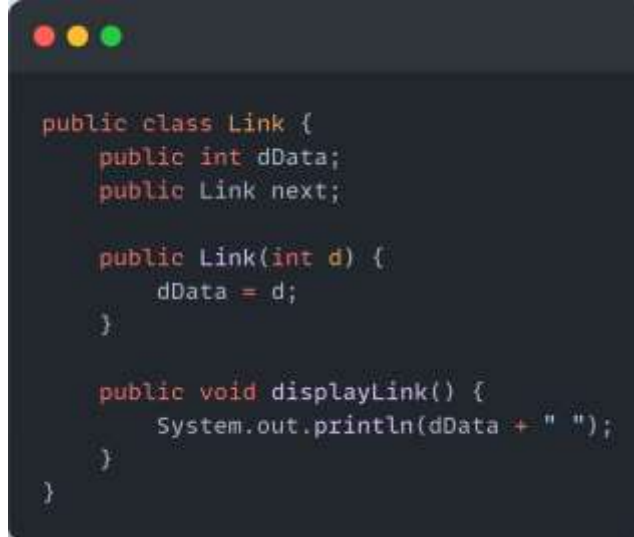
        q.add("Bebek");
        q.add("Angsa");
        q.add("Kuda");
        q.add("Buaya");
        q.add("Tikus");

        System.out.println("Peek: " + q.peek());
        System.out.println("Animals: " + q);

        q.poll();
        q.poll();

        System.out.println("Animals: " + q);
    }
}
```

- b. Membuat Program Struktur Data Queue Dengan LinkedList Tanpa Library
  - i. Membuat Class Link

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in Java and defines a class named 'Link'. It includes two public attributes: 'dData' of type 'int' and 'next' of type 'Link'. There are two public methods: a constructor 'Link(int d)' that initializes 'dData' with the parameter 'd', and a method 'displayLink()' that prints the value of 'dData' followed by a space character.

```
public class Link {  
    public int dData;  
    public Link next;  
  
    public Link(int d) {  
        dData = d;  
    }  
  
    public void displayLink() {  
        System.out.println(dData + " ");  
    }  
}
```



## ii. Membuat Class FirstLastList

```
public class FirstLastList {
    public Link first;
    public Link last;

    public FirstLastList() {
        first = null;
        last = null;
    }

    public boolean isEmpty() {
        return first == null;
    }

    public void insertLast(int dd) {
        Link newLink = new Link(dd);
        if (isEmpty()) {
            first = newLink;
        } else {
            last.next = newLink;
        }
        last = newLink;
    }

    public int deleteFirst() {
        int temp = (int) first.dData;
        if (first.next == null) {
            last = null;
        }
        first = first.next;
        return temp;
    }

    public void displayList() {
        Link current = first;
        while (current != null) {
            current.displayLink();
            current = current.next;
        }
        System.out.println("");
    }
}
```

## iii. Membuat Class LinkQueue

```
public class LinkQueue {
    public FirstLastList theList;

    public LinkQueue() {
        theList = new FirstLastList();
    }

    public boolean isEmpty() {
        return theList.isEmpty();
    }

    public void enqueue(int j) {
        theList.insertLast(j);
    }

    public long dequeue() {
        return theList.deleteFirst();
    }

    public void displayQueue() {
        System.out.println("Queue (Head→Tail): ");
        theList.displayList();
    }
}
```

## iv. Membuat Class Main

```
public class Main {
    public static void main(String[] args) {
        LinkQueue queue = new LinkQueue();

        queue.enqueue(10);
        queue.enqueue(40);
        queue.displayQueue();

        queue.enqueue(50);
        queue.enqueue(5);
        queue.displayQueue();

        queue.dequeue();
        queue.displayQueue();
    }
}
```

---

## CODELAB

### LATIHAN 1

Tulislah kembali program nomor 3A (Stack menggunakan library) dengan jelas dan benar agar memahami hasil output dari program dan menambah pemahaman.

### LATIHAN 2

Tulislah Kembali program nomor 4A (Queue menggunakan library) dengan jelas dan benar agar memahami hasil output dari program dan menambah pemahaman.

---

## TUGAS

### TUGAS 1

Buatlah program sederhana untuk mengelola riwayat navigasi pada sebuah browser. Berikut adalah ketentuan tugas:

1. Method `visitURL()`, program harus dapat mengunjungi URL baru yang dimasukkan pengguna dan menyimpannya dalam riwayat navigasi.
2. Method `back()`, program harus dapat mengembalikan pengguna ke URL sebelumnya dalam riwayat navigasi.
3. Method `forward()`, program harus dapat mengarahkan pengguna ke URL berikutnya dalam riwayat navigasi.
4. Method `getCurrentURL()`, program harus menampilkan URL yang sedang diakses pengguna.

### TUGAS 2

Buatlah program sederhana untuk pemesanan tiket menggunakan Queue. Berikut adalah ketentuan tugas:

1. Program harus meminta input pengguna untuk jumlah tiket yang ingin dipesan dan setiap tiket harus mencakup informasi seperti nama pemesan dan jumlah tiket.
2. Setiap pemesanan harus dimasukkan ke dalam Queue.
3. Program harus dapat menambahkan, menampilkan, dan menghapus dari Queue.
4. Setiap pemesanan harus diberi nomor unik untuk identifikasi.

Mahasiswa diperbolehkan melakukan improvisasi dari tugas yang diberikan dengan syarat tidak mengurangi ketentuan yang ada.

Catatan:

1. Tugas 1: Buat Stack secara manual tidak diperkenankan menggunakan library.
2. Tugas 2: Buat Queue secara manual tidak diperkenankan menggunakan library.

---

**KRITERIA & DETAIL PENILAIAN**

Kriteria	Nilai
Codelab 1	10
Codelab 2	10
Tugas 1	30
Tugas 2	30
Pemahaman	20
<b>Total</b>	<b>100</b>