

VERSION 1.2

February 25, 2023



# [STRUKTUR DATA]

MODUL 3, STACK & QUEUE

DISUSUN OLEH:

MUHAMMAD SYAUQI AMIQ AMRULLAH  
GILANG DWI DARMAWAN

DIAUDIT OLEH:

DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM.

PRESENTED BY: TIM

LAB-IT UNIVERSITAS MUHAMMADIYAH MALANG

## [STRUKTUR DATA]

### PETUNJUK Pengerjaan Modul

Perhatikan petunjuk praktikum dibawah ini:

1. Wajib membaca materi modul, sewaktu – waktu dapat direview oleh asisten saat demo
2. Gunakan referensi yang disediakan modul dan referensi lain pada google (yang kredibel)
3. Latihan praktikum wajib dikerjakan pada praktikum minggu pertama secara bersama – sama di laboratorium dan tidak boleh dijadikan pekerjaan rumah
4. Tugas praktikum boleh dijadikan pekerjaan rumah dan di demokan kepada asisten pada praktikum minggu kedua
5. Memperhatikan kerapian *source code* termasuk aturan penamaan Class, Method, Variable, File, dan lain - lainnya.
6. Segera lapor kepada asisten jika ada kesalahan pada modul praktikum.
7. Boleh mengerjakan menggunakan bahasa pemrograman Java atau Python
8. Mengerjakan Python akan mendapatkan nilai tambah

### PERSIAPAN MATERI

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

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

### TUJUAN

Mahasiswa mampu menguasai & menjelaskan konsep dari struktur data *stack* & *queue*.

### TARGET MODUL

Mahasiswa mampu memahami :

1. Contoh penggunaan *stack*
2. Contoh penggunaan *queue*
3. Cara pengoperasian *stack*
4. Cara pengoperasian *queue*

### PERSIAPAN SOFTWARE/APLIKASI

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

### REFERENSI MATERI

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/>

Youtube

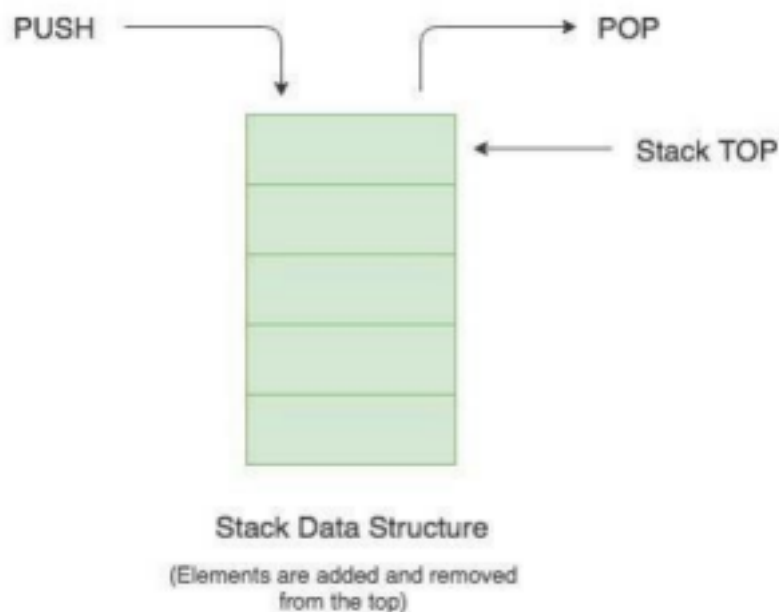
[https://www.youtube.com/results?search\\_query=stack+dan+queue+java](https://www.youtube.com/results?search_query=stack+dan+queue+java)  
[https://www.youtube.com/watch?v=hj6E7LyZKNI&ab\\_channel=KuliahInformatika](https://www.youtube.com/watch?v=hj6E7LyZKNI&ab_channel=KuliahInformatika)  
[https://www.youtube.com/watch?v=3F5kq-4jxPI&ab\\_channel=CodeDecode](https://www.youtube.com/watch?v=3F5kq-4jxPI&ab_channel=CodeDecode)

## MATERI POKOK

### A. Stack

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



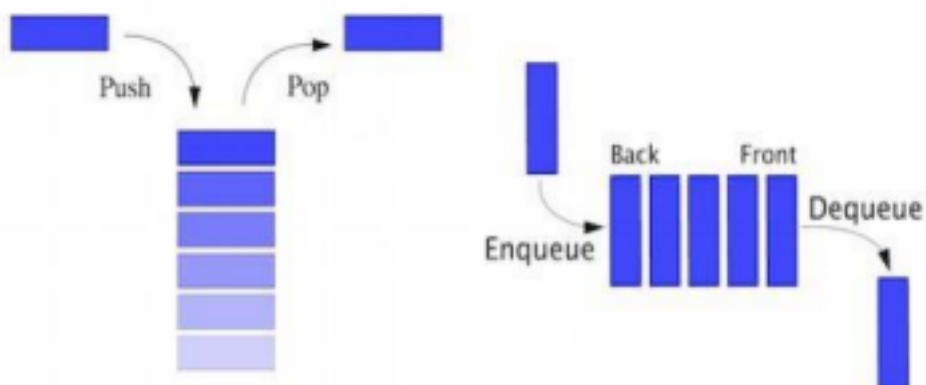
· Method yang terdapat pada kelas Stack

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

## B. Queue

Queue adalah kumpulan data yang mana penambahan elemen nya hanya dapat dilakukan di satu sisi yang disebut sisi belakang (tail) , dan penghapusan elemen nya 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).

## · Method pada Interface Queue

Method	Deskripsi
<code>add(element)</code>	This method is used to add elements at the tail of queue. More specifically, at the last of linked-list if it is used, or according to the priority in case of priority queue implementation.
<code>element()</code>	This method is similar to <code>peek()</code> . It throws <code>NoSuchElementException</code> when the queue is empty.
<code>offer(element)</code>	This method is used to insert an element in the queue. This method is preferable to <code>add()</code> method since this method does not throws an exception when the capacity of the container is full since it returns false.
<code>peek()</code>	This method is used to view the head of queue without removing it. It returns Null if the queue is empty.
<code>poll()</code>	This method removes and returns the head of the queue. It returns null if the queue is empty.
<code>remove()</code>	This method removes and returns the head of the queue. It throws <code>NoSuchElementException</code> when the queue is empty.

## C. Contoh Implementasi Program Stack

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

```
package modul4contoh;

import java.util.Stack; //LIBRARY STACK

public class Modul4 {

    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"));
    }
}
```

## 2. Membuat Program Struktur Data Stack Menggunakan Array tanpa Library

### a) Membuat Class Stack

```
package modul4contoh2;

public class Stack {
    private int maxSize;
    private long[] stackArray;
    private int top; //Top dari stack

    public Stack(int s){
        maxSize = s; //Set ukuran array
        stackArray = new long[maxSize]; //Membuat array
        top = -1; //Item masih kosong
    }

    public void push(long j){
        stackArray[++top] = j; //Increment top, Insert item
    }

    public long pop(){
        return stackArray[top--]; //Akses item, Decrement top
    }

    public long peek(){
        return stackArray[top];
    }

    public boolean isEmpty(){
        return (top == -1); //Bernilai TRUE jika stack empty
    }

    public boolean isFull(){
        return (top == maxSize-1); //Bernilai TRUE jika stack full
    }
}
```

### b) Membuat Class Main

```
package modul4contoh2;

public class Main {
    public static void main(String[] args) {
        Stack theStack = new Stack(10); //Membuat stack baru
        theStack.push(9); //Push item
        theStack.push(30);
        theStack.push(10);
        theStack.push(100);

        while (!theStack.isEmpty()){ //Until it's empty
            long value = theStack.pop(); //Hapus item dari stack
            System.out.print(value);
            System.out.print(" ");
        }
        System.out.println("");
    }
}
```

## D. Contoh Implementasi Program Queue

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

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

public class Modul4Queue {

    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);
    }
}
```

### 2. Membuat Program Struktur Data Queue dengan LinkedList tanpa Library a) Membuat Class Link

```
package modul4contoh4;

public class Link {

    public int dData; //data item
    public Link next; //next link pada list

    public Link(int d){
        dData = d;
    }

    public void displayLink(){
        System.out.print(dData + " ");
    }
}
```

## b) Membuat Class FirstLastList

```
public class FirstLastList {

    public Link first; //Ref to first item
    public Link last; //Ref to last item

    public FirstLastList(){
        first = null; //No items on list yet
        last = null;
    }

    public boolean isEmpty(){
        return first == null; //True if no links
    }

    public void insertLast(int dd){
        Link newLink = new Link(dd); //Make new link
        if(isEmpty()){
            first = newLink; //First --> newLink else
        } else {
            last.next = newLink; //Old last --> newLink
        }
        last = newLink; //newLink <-- last
    }

    public int deleteFirst(){
        int temp = (int) first.dData;
        if(first.next == null){ //if only one item
            last = null; //null <-- last
        }
        first = first.next; //first --> old next return temp
        return temp;
    }

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



### c) Membuat Class LinkQueue

```
public class LinkQueue {  
  
    public FirstLastList theList;  
  
    public LinkQueue() {  
        theList = new FirstLastList(); //Make a 2-ended list  
    }  
  
    public boolean isEmpty() {  
        return theList.isEmpty(); //true jika queue empty  
    }  
  
    public void enqueue(int j) {  
        theList.insertLast(j); //Insert, tail of queue  
    }  
  
    public long dequeue() {  
        return theList.deleteFirst(); //Hapus, Head of queue  
    }  
  
    public void displayQueue() {  
        System.out.print("Queue (Head-->Tail) : ");  
        theList.displayList();  
    }  
}
```

### d) Membuat Class Main

```
public class Main {  
    public static void main(String[] args) {  
        LinkQueue queue = new LinkQueue();  
  
        queue.enqueue(10); //Insert item  
        queue.enqueue(40);  
        queue.displayQueue(); //Display queue  
  
        queue.enqueue(50); //Insert item  
        queue.enqueue(5);  
        queue.displayQueue(); //Display item  
  
        queue.dequeue(); // Hapus item  
        queue.displayQueue();  
    }  
}
```

## TAMBAHAN MATERI

Mari kita coba untuk membuat code versi bahasa pemrograman Python untuk menambah pengetahuan kita.

### A. Stack

```
stack = []

stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)

print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)
```

Bisa dilihat untuk prosesnya adalah kita membuat sebuah array kosong dengan nama “stack” lalu kita isi menggunakan method “append()” setelah itu kita akan mengeluarkan isi dari stack yang kita buat dengan method “pop()”, dan sifat dari struktur data stack ini adalah LIFO (Last in First Out) yaitu data yang terakhir kita insert adalah yang pertama untuk keluar ketika kita keluarkan dengan method “pop()”.

#### Contoh Output:

```
Initial stack
['a', 'b', 'c']

Elements popped from stack:
c
b
a

Stack after elements are popped:
[]
sh-3.2$
```

### B. Queue

```
queue = []

# Adding elements to the queue
queue.append('a')
queue.append('b')
queue.append('c')

print("Initial queue")
print(queue)

# Removing elements from the queue
print("\nElements dequeued from queue")
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

print("\nQueue after removing elements")
print(queue)
```

Hampir sama dengan *source code* untuk struktur data Stack namun disini Queue bersifat FIFO (First in First Out) yaitu data yang pertama kita insert akan di keluarkan terlebih dahulu ketika kita keluarkan dengan method “pop()”.

## Contoh Output:

```
Initial queue
['a', 'b', 'c']

Elements dequeued from queue
a
b
c

Queue after removing elements
[]
sh-3.2$
```

## LATIHAN PRAKTIKUM

### LATIHAN 1

#### Menggunakan *Stack Collection* pada *java.util.stack*

Tulislah kembali program dibawah ini dengan baik dan benar untuk mengetahui output dari program dan menambah pemahaman.

```
package modul4;
import java.util.Stack;

public class ExampleStack {

    public static void main(String[] args) {
        Stack st = new Stack();

        st.push("Aku");
        st.push("Anak");
        st.push("Indonesia");

        System.out.println("Next : " + st.peek());
        st.push("Raya");
        System.out.println(st.pop());
        st.push("!");

        int count = st.search("Aku");
        while (count != -1 && count > 1){
            st.pop();
            count--;
        }
        System.out.println(st.pop());
        System.out.println(st.empty());
    }
}
```

Jika sudah selesai menulis dan menjalankan program diatas, untuk lebih menambah pemahaman Anda silahkan program diatas dikembangkan tanpa menggunakan library *java.util.Stack* dan buatlah menggunakan bahasa pemrograman Python untuk menambah pengetahuan kalian. Dan yang terakhir silahkan untuk push project nya ke akun Github kalian dengan visibility Public.

## LATIHAN 2

### LinkedList menerapkan interface Queue

Tulislah kembali program dibawah ini dengan baik dan benar untuk mengetahui output dari program dan menambah pemahaman.

```
package mod4kegl;

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

public class MainQueue {

    public void queueExample() {
        Queue queue = new LinkedList();
        queue.add("Java");
        queue.add("DotNet");
        queue.offer("PHP");
        queue.offer("HTML");
        System.out.println("remove: " + queue.remove());
        System.out.println("element: " + queue.element());
        System.out.println("poll: " + queue.poll());
        System.out.println("peek: " + queue.peek());
    }

    public static void main(String[] args) {
        new MainQueue().queueExample();
    }
}
```

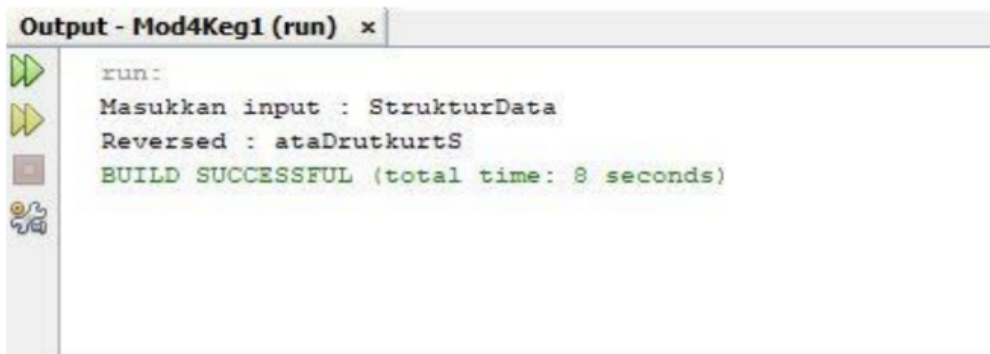
Jika sudah selesai menulis dan menjalankan program diatas, untuk lebih menambah pemahaman Anda silahkan program diatas dikembangkan tanpa menggunakan library dan buatlah menggunakan bahasa pemrograman Python untuk menambah pengetahuan kalian. Dan yang terakhir silahkan untuk push project nya ke akun Github kalian dengan visibility Public.

## TUGAS

### KEGIATAN 1

Buatlah program menggunakan struktur data stack dimana setiap kita memasukkan sebuah String maka program akan menghasilkan output berupa String yang mempunyai urutan terbalik dari inputan kita. Program dibuat secara manual **tidak diperkenankan menggunakan library**. **Kerjakan menggunakan Python untuk mendapatkan nilai tambah.**

Contoh output :



```
Output - Mod4Keg1 (run) x
run:
Masukkan input : StrukturData
Reversed : ataDrutkurtS
BUILD SUCCESSFUL (total time: 8 seconds)
```

## KEGIATAN 2

Buatlah program yang mengimplementasikan Struktur data queue menggunakan linkedlist secara manual **tanpa menggunakan library** dengan ketentuan membuat dan menggunakan method sebagai berikut :

- 1) Method enqueue()
- 2) Method dequeue()
- 3) Method peek()
- 4) Method isEmpty()
- 5) Method size()

Diizinkan untuk mengkreasikan program dan inputan dengan tetap beracuan pada ketentuan yang sudah ditetapkan diatas. **Kerjakan menggunakan Python untuk mendapatkan nilai tambah.**

**Catatan : Studi kasus setiap praktikan untuk kegiatan 2 tidak boleh sama, H-1 minggu praktikum akan dishare file spreadsheet oleh CO Asisten di kelas masing-masing untuk mengisi tema/topic Studi Kasus yang akan dibuat dari masing-masing praktikan.**

## CATATAN

Kegiatan 1 : Tidak diperkenankan menggunakan Library

Kegiatan 2 : Tidak diperkenankan menggunakan Library

**Aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:**

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakanmenggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: Kursi , JalanRaya, ParkiranGedung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal kata dan menggunakan huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, harga, setNamaJalan, dan lain seterusnya.
3. Jika menggunakan IDE IntelliJ jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok. Silahkan dikerjakan tanpa **copy – paste**.

## DETAIL PENILAIAN TUGAS

Kriteria	Nilai
Semua Ketentuan Pada Tugas Praktikum Terpenuhi Saat Demo dan Kerapihan Code Serta Tidak Ada Plagiasi	40
Mengerjakan Latihan Praktikum	40
Presensi Kehadiran	20

