

VERSI 2.2

NOVEMBER, 2020



[STRUKTUR DATA]

MODUL 4, STACK & QUEUE

TIM PENYUSUN: - DOSEN
- DICKY PRABOWO OCTIANTO

PRESENTED BY: LAB. TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

CAPAIAN PEMBELAJARAN MATA KULIAH

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

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu memahami:

1. Contoh penggunaan stack
2. Contoh penggunaan queue
3. Cara pengoprasian stack
4. Cara pengoprasian queue

PERSYARATAN PEMAHAMAN

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

KEBUTUHAN HARDWARE & SOFTWARE

- Java Development Kit
- Java Runtime Environment
- IDE (Intellij IDEA, Eclipse, Netbeans, dll.)

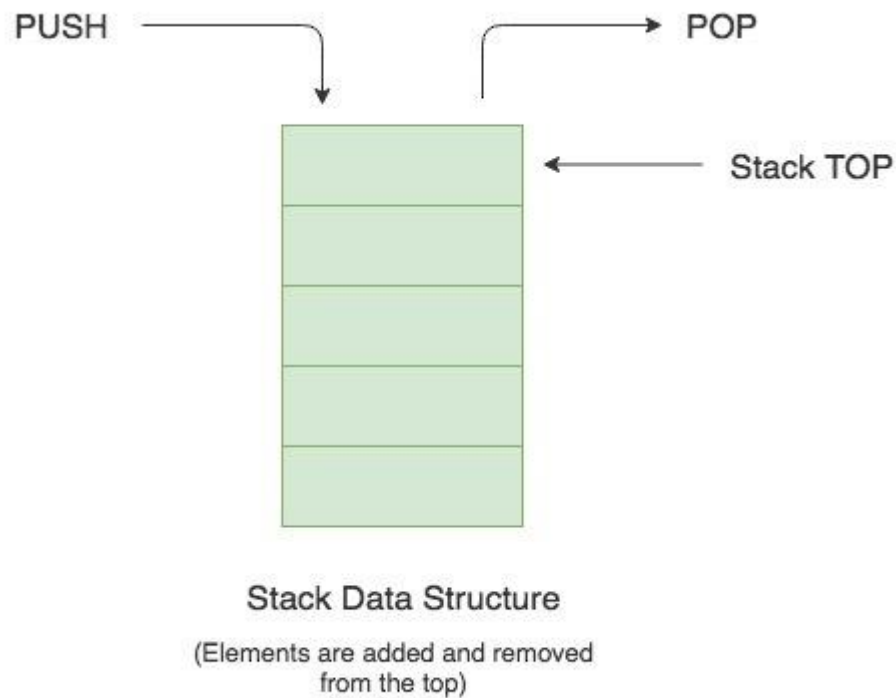
REFERENSI MATERI

Oracle iLearning Java Programming section 6-3 (2,4)

MATERI POKOK

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.



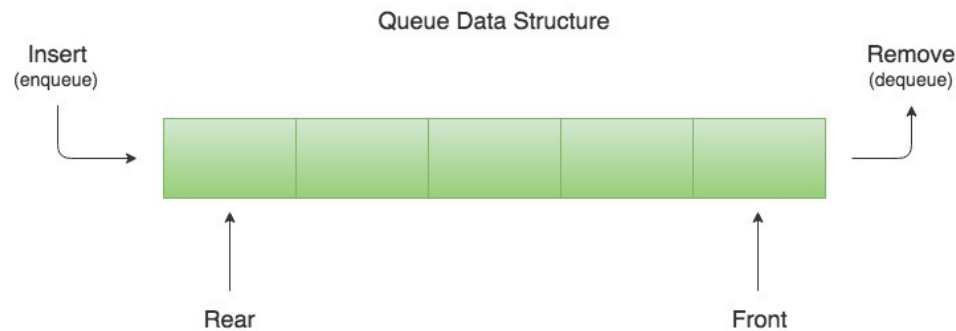
Operasi stack dapat dilakukan pada elemen pada top dari stack. Yaitu *Push()* menambah item pada top, *Pop()* menghapus elemen dari top, *Peek()* mengakses nilai pada top. Stack memiliki urutan LIFO (last-in-first-out) yang berarti data yang terakhir masuk akan dikeluarkan paling pertama.

Methods in Stack Class

METHOD	DESCRIPTION
<code>empty()</code>	It returns true if nothing is on the top of the stack. Else, returns false.
<code>peek()</code>	Returns the element on the top of the stack, but does not remove it.
<code>pop()</code>	Removes and returns the top element of the stack. An 'EmptyStackException' An exception is thrown if we call <code>pop()</code> when the invoking stack is empty.
<code>push(Object element)</code>	Pushes an element on the top of the stack.
<code>search(Object element)</code>	It determines whether an object exists in the stack. If the element is found, It returns the position of the element from the top of the stack. Else, it returns -1.

Queue

Queue adalah struktur data dimana data yang pertama kali dimasukkan adalah data yang pertama kali bisa dihapus. 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.



Queue dalam kehidupan sehari-hari 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).

Methods in Queue Interface

METHOD	DESCRIPTION
add(element)	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.
element()	This method is similar to peek(). It throws NoSuchElementException when the queue is empty.
offer(element)	This method is used to insert an element in the queue. This method is preferable to add() method since this method does not throw an exception when the capacity of the container is full since it returns false.
peek()	This method is used to view the head of queue without removing it. It returns Null if the queue is empty.
poll()	This method removes and returns the head of the queue. It returns null if the queue is empty.
remove()	This method removes and returns the head of the queue. It throws NoSuchElementException when the queue is empty.

MATERI PRAKTIKUM

Silahkan mencoba program – program dibawah ini untuk mengetahui nilai output nya.

1. Penerapan Struktur Data Stack dengan Library

```
import java.util.Stack;

public class Modul4Stack {
    public static void main(String[] args) {
        Stack stack = new Stack();

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

        stack.push("Anton");
        stack.push("Kevin");
        stack.push("David");
        stack.push("Gina");
        stack.push("Ahmad");

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

        System.out.println("Peek : "+stack.peek());
        System.out.println("Name :"+stack);

        stack.pop();
        stack.pop();

        System.out.println("Name :"+stack);
        System.out.println("Position of Anton : "+stack.search("Anton"));
    }
}
```

2. Penerapan Struktur Data Queue dengan Library

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

public class Modul4Queue {
    public static void main(String[] args) {
        Queue queue = new LinkedList();

        queue.add("Anton");
        queue.add("Kevin");
        queue.add("David");
        queue.add("Gina");
        queue.add("Ahmad");

        System.out.println("Peek : "+queue.peek());
        System.out.println("Name :"+queue);

        queue.poll();
        queue.poll();

        System.out.println("Name :"+queue);
    }
}
```

3. Penerapan Struktur Data Stack tanpa Library

a. Membuat class dengan nama stack

```
public class Stack {  
  
    private int maxSize;  
    private String[] stackArray;  
    private int top; // top dari stack  
  
    public Stack(int s) {  
        maxSize = s; // set ukuran array  
        stackArray = new String[maxSize]; // membuat array  
        top = -1; // belum ada item  
    }  
  
    public void push(String j) {  
        stackArray[++top] = j; // increment top, insert item  
    }  
  
    public String pop() {  
        return stackArray[top--]; // akses item, decrement top  
    }  
  
    public String peek() {  
        return stackArray[top];  
    }  
  
    public boolean isEmpty() {  
        return (top == -1); // true jika stack empty  
    }  
  
    public boolean isFull() {  
        return (top == maxSize - 1); // true jika stack full  
    }  
}
```

b. Membuat driver class

```
public class main {  
  
    public static void main(String[] args) {  
        Stack stack = new Stack(10);  
  
        System.out.println(stack.isEmpty());  
  
        stack.push("Anton");  
        stack.push("Kevin");  
        stack.push("David");  
        stack.push("Gina");  
        stack.push("Ahmad");  
  
        System.out.println(stack.isEmpty());  
  
        System.out.println("Peek : " + stack.peek());  
  
        while(!stack.isEmpty())  
        {  
            System.out.print(stack.pop()+" ");  
        }  
        System.out.println("");  
    }  
}
```

4. Penerapan Struktur Data Queue dengan Linked List tanpa Library**a. Membuat class Link**

```
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) { // insert at end of list
        Link newLink = new Link(dd); // make new link
        if (isEmpty()) // if empty list
        {
            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 coba = new LinkQueue();

        System.out.println(coba.isEmpty());

        coba.enqueue(12); // insert items
        coba.enqueue(99);
        coba.enqueue(18);
        coba.enqueue(19);

        System.out.println(coba.isEmpty());
        coba.displayQueue(); // display queue

        coba.enqueue(10); // insert items
        coba.enqueue(5);
        coba.enqueue(20); // display queue
        coba.displayQueue();

        coba.dequeue(); // hapus items
        coba.displayQueue(); // display queue
        coba.dequeue();
        coba.dequeue();
        coba.displayQueue();

    }

}

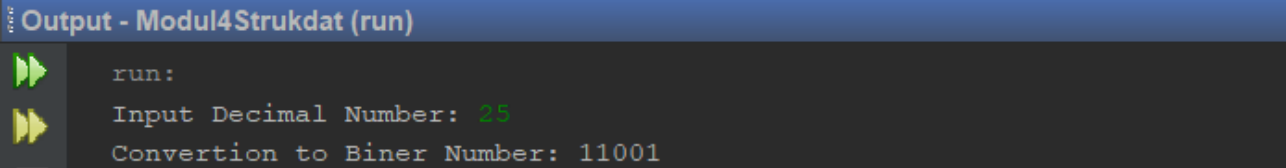
```

LEMBAR KERJA

KEGIATAN 1

Buatlah program konversi dari bilangan decimal ke bilangan biner dengan menerapkan konsep struktur data stack secara manual **tanpa menggunakan library** dengan ketentuan membuat dan menggunakan method push dan pop. Kreasikan Source Code dan Outputan sekreatif kalian dengan tetap mengikuti ketentuan yang ada.

Output:



```
run:
Input Decimal Number: 25
Conversion to Biner Number: 11001
```

KEGIATAN 2

Buatlah program yang mengimplementasikan Struktur data queue menggunakan linkedlist secara manual **tanpa menggunakan library** yang menyimpan data **String berupa nama** 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 sudah ditetapkan diatas.

CATATAN

Kerjakan **Kegiatan 1 dan Kegiatan 2** secara manual tanpa import library.

Silahkan dikerjakan **tanpa copy – paste**, jika ada indikasi program yang kembar maka akan dilakukan **pengurangan poin**.

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

1. Untuk nama kelas, interface, enum, dan yang lainnya biasakan menggunakan 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.

RUBRIK PENILAIAN

Soal	Nilai
Kegiatan 1:	50%
Kegiatan 2:	50%