

# STACK

## Tujuan

Setelah mempelajari bab ini diharapkan mahasiswa akan mampu :

1. Memahami terminologi yang terkait dengan struktur data stack.
2. Memahami operasi-operasi yang ada dalam stack.
3. Dapat mengidentifikasi permasalahan-permasalahan pemrograman yang harus diselesaikan dengan menggunakan stack, sekaligus menyelesaikannya.

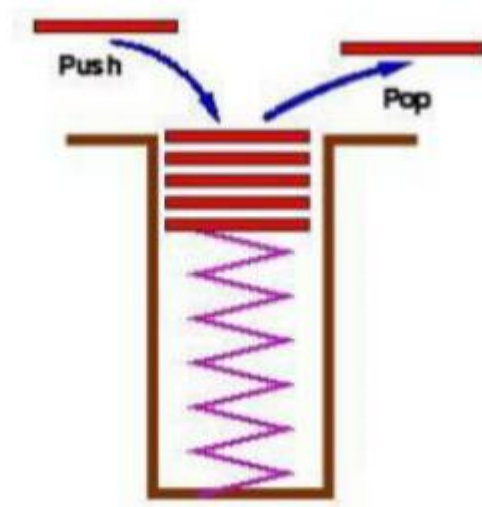
## DASAR TEORI

### STACK

#### Pengertian Stack

Stack adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO. Dengan demikian, elemen terakhir yang disimpan dalam stack menjadi elemen pertama yang diambil. Stack (Tumpukan) adalah kumpulan elemen-elemen data yang disimpan dalam satu lajur linear. Kumpulan elemen-elemen data hanya boleh diakses pada satu lokasi saja yaitu posisi ATAS (TOP) tumpukan. Tumpukan digunakan dalam algoritma pengimbas (parsing), algoritma penilaian (evaluation) dan algoritma penjajahan balik (backtrack). Elemen-elemen di dalam tumpukan dapat bertipe integer, real, record dalam bentuk sederhana atau terstruktur. Stack adalah suatu tumpukan dari benda. Konsep utamanya adalah LIFO (Last In First Out), benda yang terakhir masuk dalam stack akan menjadi benda pertama yang dikeluarkan dari stack. Tumpukan disebut juga “Push Down Stack” yaitu penambahan elemen baru (PUSH) dan penghapusan elemen dari tumpukan (POP). Contoh pada PDA (Push Down Automaton). Sistem pada pengaksesan pada tumpukan menggunakan system LIFO (Last In First Out), artinya elemen yang terakhir masuk itu yang akan pertama dikeluarkan dari tumpukan (Stack). Ilustrasi tumpukan (Stack) dapat digambarkan seperti tumpukan CD atau tumpukan sate. Stack merupakan suatu susunan koleksi data dimana dapat ditambahkan dan dihapus selalu dilakukan pada bagian akhir data, yang disebut dengan Top Of Stack. Dalam proses komputasi, untuk meletakkan sebuah elemen pada bagian atas dari stack, maka kita melakukan push. Dan untuk memindahkan dari tempat yang atas tersebut, kita melakukan pop. Ada beberapa cara untuk

menyajikan sebuah stack tergantung pada permasalahan yang akan kita selesaikan. Dalam bab ini kita akan menggunakan cara yang paling sederhana, tipe data yang sudah kita kenal, yaitu array. Kita dapat menggunakan array untuk menyajikan sebuah stack, dengan anggapan bahwa banyaknya elemen maksimum dari stack tersebut tidak akan melebihi batas maksimum banyaknya elemen dalam array. Pada saat ukuran stack, kalau kita teruskan menambah data lagi, akan terjadi overflow. Dengan demikian perlu data tambahan untuk mencatat posisi ujung stack. Dengan kebutuhan seperti ini, kita dapat menyajikan stack dengan menggunakan tipe data struktur (struct) yang terdiri dari dua field. Field pertama bertipe array untuk menyimpan elemen stack, medan kedua bertipe integer untuk mencatat posisi ujung stack.

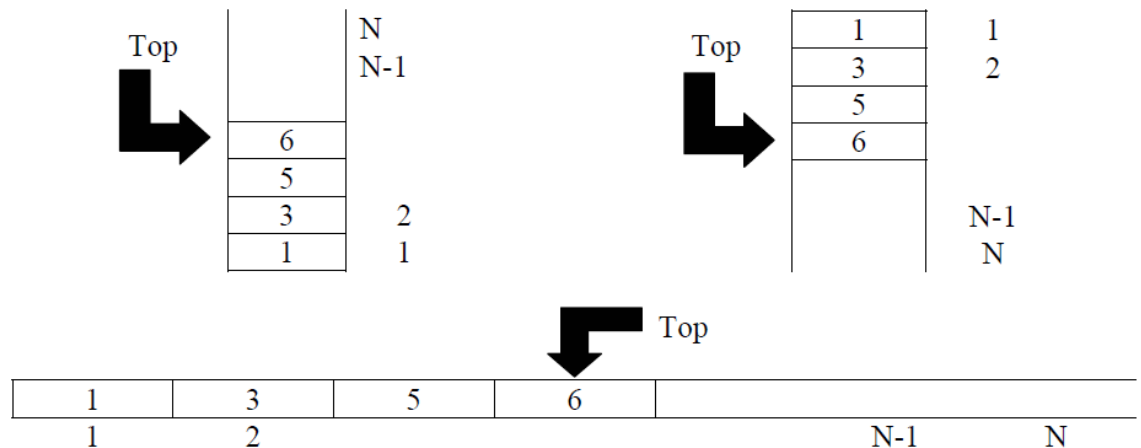


Sebelum struktur data tumpukan ini bisa digunakan, harus dideklarasikan dahulu dalam kamus data. Ada beberapa cara pendeklarasian struktur data ini, salah satunya dengan menggunakan tata susunan linear (larik) dan sebuah variable, yang dikemas dalam tipe data record. Stack (tumpukan) adalah struktur data bertipe record yang terdiri dari field elemen, bertipe larik/array dengan indek dari 1 sampai dengan MaksTum (Maksimum Tumpukan), atas, bertipe interger berkisar dari 0 (saat kosong) sampai dengan MaksTum (Maksimum Tumpukan).

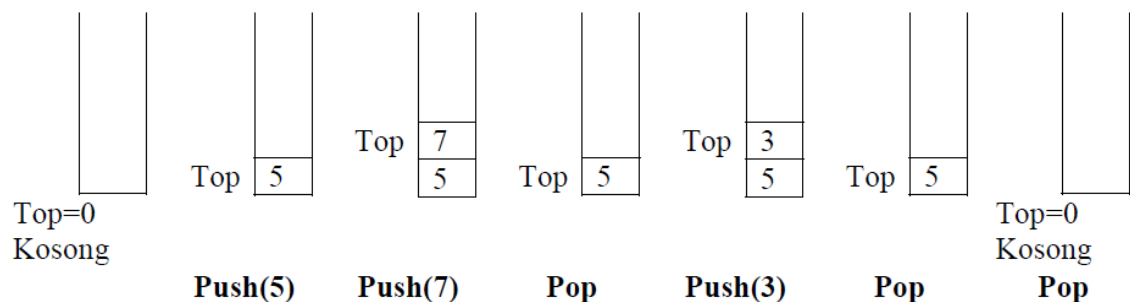
### **Operasi –operasi pada Stack (Tumpukan)**

Operasi yang sering diterapkan pada struktur data Stack (Tumpukan) adalah Push dan Pop. Operasi – operasi yang dapat diterapkan adalah sebagai berikut :

1. Push : digunakan untuk menambah item pada Stack pada Tumpukan paling atas.
  2. Pop : digunakan untuk mengambil item pada Stack pada Tumpukan paling atas.
- Sebagai contoh, misalkah ada data sebagai berikut : 1 3 5 6, maka data tersebut dapat tersimpan dalam bentuk sebagai berikut :



Contoh lain adalah ada sekumpulan perintah stack yaitu push(5), push(7), pop, push(3), pop. Jika dijalankan, maka yang akan terjadi adalah :



3. Clear : digunakan untuk mengosongkan Stack.
4. Create Stack : membuat Tumpukan baru S, dengan jumlah elemen kosong.
5. MakeNull : mengosongkan Tumpukan S, jika ada elemen maka semua elemen dihapus.
6. IsEmpty : fungsi yang digunakan untuk mengecek apakah Stack sudah kosong.
7. Isfull : fungsi yang digunakan untuk mengecek apakah Stack sudah penuh.

Pada proses Push, Tumpukan (Stack) harus diperiksa apakah jumlah elemen sudah mencapai masimum atau tidak. Jika sudah mencapai maksimum maka OVERFLOW, artinya Tumpukan penuh tidak ada elemen yang dapat dimasukkan ke dalam Tumpukan. Sedangkan pada proses Pop, Tumpukan harus diperiksa apakah

ada elemen yang hendak dikeluarkan atau tidak. Jika tidak ada maka UNDERFLOW, artinya tumpukan kosong tidak ada elemen yang dapat diambil.

## 1. Java Stack Collection

Package Java juga menyediakan class Stack pada java.util.Stack, yang merupakan subclass dari Vector yang menggunakan standar last-in first-out (LIFO). Class Stack hanya digunakan untuk menentukan default constructor, untuk membuat stack kosong. Berikut ini beberapa metode yang digunakan dalam stack seperti terlihat pada Tabel 1.

Tabel 1. Metode pada java.util.stack

Metode-metode pada kelas Stack	Deskripsi
<code>boolean empty( )</code>	Menghasilkan nilai True jika stack kosong, dan nilai False jika stack berisi elemen
<code>Object peek( )</code>	Menghasilkan elemen pada top stack, tetapi tidak me-remove.
<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 -1.

## 2. Implementasi Stack dengan Array dan ArrayList

Selain menggunakan java stack collection, kita dapat mengimplementasikan stack dengan menggunakan arraylist. Untuk mengimplementasikan stack Digunakan interface Stack yang berisi fungsi-fungsi berikut:

Tabel 2. Interface Stack

Interface Stack	
boolean	<code>isEmpty()</code> mengembalikan true jika stack kosong dan false jika stack berisi elemen.
T	<code>peek()</code> mengambil nilai dari atas stack, jika stack kosong maka melempar (throw) <code>EmptyStackException</code>
T	<code>pop()</code> menghapus elemen dari atas stack dan mengembalikan nilainya. Jika stack kosong maka melempar(throw) <code>EmptyStackException</code> .
void	<code>push(T item)</code> menambahkan item di atas stack
int	<code>size()</code> mengembalikan jumlah elemen yang terdapat di stack.

Implementasi stack dapat menggunakan array atau arraylist. Untuk mengimplementasikan stack menggunakan array seperti di bawah ini.

```

public class StackArr<T> implements Stack {
    T value[] ;
    int topOfStack ;

    public boolean isEmpty() {...}
    public T pop() {...}
    public void push(T item) {...}

    public T peek() {...}
    public int size() {...}
}

```

Sedangkan untuk mengimplementasikan stack menggunakan arraylist seperti di bawah ini.

```

public class ALStack<T> implements Stack {
    // storage structure
    private ArrayList<T> stackList = null;
    // create an empty stack by creating an empty ArrayList
    public ALStack() {
        stackList = new ArrayList<T>();
    }

    public boolean isEmpty() {...}
    public T pop() {...}
    public void push(T item) {...}
    public T peek() {...}
    public int size() {...}
}

```

### 3. Mengubah Notasi Postfix menjadi Infix dengan Stack

Salah satu penggunaan stack adalah mengubah notasi infix menjadi postfix. Berikut ini adalah algoritma untuk mengubah notasi infix menjadi notasi postfix:

1. Baca ungkapan dalam notasi infix, misalnya S, tentukan panjang ungkapan tersebut, misalnya N karakter, siapkan sebuah stack kosong dan siapkan derajat masing masing operator, misalnya: ^ berderajat 3, \* dan / berderajat 2, + dan - berderajat 1 dan ( berderajat 0.
2. Dimulai dari i = 1 sampai N kerjakan langkah-langkah sebagai berikut:
  - a.  $R = S[i]$
  - b. Test nilai R. Jika R adalah:
    - Operand : langsung ditulis
    - kurung buka : push ke dalam tumpukan
    - kurung tutup : pop dan tulis semua isi tumpukan sampai ujung tumpukan
  - c. = '('. Pop juga tanda '(' ini, tetapi tidak usah ditulis operator : jika tumpukan kosong atau derajat R lebih tinggi dibanding derajat ujung tumpukan, push operator ke dalam tumpukan. Jika tidak, pop ujung tumpukan dan tulis; kemudian ulangi perbandingan R dengan ujung tumpukan. Kemudian R di-push
  - d. Jika akhir notasi infix telah tercapai, dan tumpukan masih belum kosong, pop semua

isi tumpukan dan tulis hasilnya Untuk memahami algoritma di atas, kita coba mengubah ungkapan berikut, yang ditulis menggunakan notasi infix, menjadi notasi postfix  $(A + B) / ((C - D) * E ^ F)$  Ilustrasi pengubahan notasi infix di atas menjadi notasi postfix secara lengkap tersaji dalam tabel sebagai berikut:

Tabel 3. Proses Mengubah Notasi Infix menjadi Postfix

Karakter dibaca	Isi Tumpukan	Karakter tercetak	Hasil Notasi Postfix Yang Terbentuk
(	(		
A	( +	A	A
+	( +		
B		B	A B
)		+	A B +
/	/		
(	/(		
(	/( (		
C	/( (	C	A B + C
-	/( (-		
D	/( (-	D	A B + C D
)	/(	-	A B + C D -
*	/( *		
E	/( *	E	A B + C D - E
^	/( * ^		
F	/( * ^	F	A B + C D - F
)	/( *	^	A B + C D - F ^
	/(	*	A B + C D - F ^ *
	/		
		/	A B + C D - F ^ *

Dari ilustrasi di atas, bisa kita lihat bahwa notasi postfix dari ungkapan:

$(A + B) / ((C - D) * E ^ F)$  adalah  $A B + C D - F ^ *$

Tabel 4. Contoh Infix ke Postfix

Infix	Postfix
$(A + B) / ((C - D) * E ^ F)$	$A B + C D - F ^ *$
$a + b * c$	$abc*+$
$a * b / c + d$	$ab*c/d+$
$a * (b + c)$	$abc+*$
$a^b^c$	$abc^^$

## Operasi-Operasi *Stack*

### 1. Pendeklarasian *Stack*

Suatu *stack* memiliki beberapa bagian yaitu:

Top : yang berisi posisi data terakhir.

Elemen : yang berisi data yang ada dalam *stack* bagian inilah yang berbentuk *array*.

Maks\_elemen yaitu variabel yang menunjuk maksimal banyaknya elemen dalam *stack*.

### 2. Inisialisasi

Inisialisasi *Stack* adalah proses pembuatan suatu *stack* kosong. Adapun langkah-langkah proses inisialisasi *stack* yang menggunakan *array*. Dengan mengisi nilai *field top* dengan 0. Jika elemen pertama diawali dengan nomor 1, kalau elemen pertamanya *array* dimulai dengan 0, maka *top* diisi dengan -1.

### 3. Operasi IsEmpty

Operasi yang digunakan untuk memeriksa apakah *stack* dalam keadaan kosong. Operasi ini dilakukan dengan memeriksa *field top*, jika *top* bernilai 0 atau *top* bernilai -1, maka berarti dalam keadaan *empty*.

### 4. Operasi IsFull

Operasi ini untuk memeriksa keadaan *stack* apakah sudah penuh/belum. Operasi ini memberikan nilai true(1) jika *field top* sama dengan *field maks\_elemen*.

### 5. Operasi Push

Operasi *push* adalah operasi dasar dari *stack* yang berguna untuk menambahkan suatu elemen data baru pada *stack* dan di simpan pada posisi *top* yang akan mengakibatkan posisi *top* akan berubah.

Langkah-langkah operasi ini:

Periksa apakah *stack* penuh(IsFull). Jika berniali *false*/0(tidak penuh) maka proses *push* dilakukan dan jika pemeriksaan ini bernilai *true*/1, maka proses *push* digagalkan.

Proses *Push*-nya sendiri adalah dengan menambahkan *field top* dengan 1, kemudian elemen pada posisi *top* di isi dengan elemen data baru.

## 6. Operasi Pop

Operasi *Pop* adalah salah satu operasi paling besar dari *stack* yang berguna untuk mengambil elemen terakhir(*top*) dan kemudian menghapus elemen tersebut sehingga posisi *top* akan berpindah.

Operasi ini biasanya dibuat dalam bentuk *function* yang me-*return*-kan nilai sesuai data yang ada di *top*.

Operasi *Pop* pada *Stack* yang menggunakan array adalah terlebih dahulu memeriksa apakah *stack* sedang keadaan kosong, jika tidak kosong maka data diambil pada posisi yang ditunjuk oleh posisi *top*. Kemudian disimpan dalam variabel baru dengan nama "data".



## Tugas I

### Pemrograman Java

**Nama Program** : Stack Collection pada java.util.stack

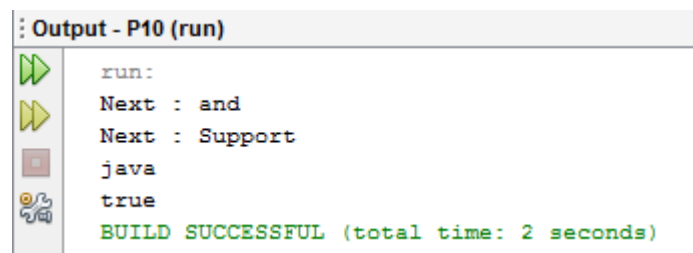
**Bahasa Pemrograman** : Java

**Compiler** : NetBeans IDE 8.2

**Script program** :

```
package p10;
import java.util.Stack;
/**
 *
 * @author toshiba
 */
public class PERCOBAAN_1 {
    public static void main(String[] args) {
        Stack s = new Stack();
        s.push("java");
        s.push("source");
        s.push("and");
        System.out.println("Next : " + s.peek());
        s.push("Support");
        System.out.println(s.pop());
        s.push(".");
        int count = s.search("java");
        while (count != -1 && count > 1)
        {
            s.pop();
            count--;
        }
        System.out.println(s.pop());
        System.out.println(s.empty());
    }
}
```

**Output Program** :



```
Output - P10 (run)
run:
Next : and
Next : Support
java
true
BUILD SUCCESSFUL (total time: 2 seconds)
```

**Penjelasan Program** :

Program diatas merupakan contoh Java Stack Collection. Digynakan beberapa metode pada java.uti.stack yang bisa diakses secara public seperti push(), empty(), search(), peek(), dan pop(). Method Push() digunakan untuk menambahkan elemen pada

stack. Method Peek() digunakan untuk menghasilkan elemen pada top stack, tetapi tidak remove. Method empty() digunakan untuk memeriksa kosong atau tidaknya stack, akan bernilai True jika stack kosong dan akan bernilai False jika stack berisi elemen. Method Search() digunakan untuk mencari elemen dalam stack, jika ditemukan akan menghasilkan offset dari top stack dan sebaliknya jika tidak menemukan elemen akan menghasilkan nilai -1. Dan yang terakhir method Pop() yang digunakan untuk menghasilkan elemen pada top stack, dan mengambil atau menghapus (remove) elemen tersebut. Pada program diatas praktikkan memasukkan elemen s.push("java"), s.push("source"), dan s.push("and"). Setelah itu akan digunakan method s.peek yang berguna untuk menampilkan elemen yang terakhir dimasukkan/top yaitu "and". Kemudian praktikkan memasukkan elemen lagi dengan nama s.push("support") dan selanjutnya digunakan method s.pop untuk menghasilkan elemen pada top stack, dan mengambil atau menghapus (remove) elemen tersebut yaitu elemen "support" yang posisinya diatas karena elemen tersebut adalah elemen yang terakhir dimasukkan. Kemudian ditambah elemen lagi s.push("."), yang kemudian akan digunakan method s.search untuk mencari elemen "java". Setelah itu akan dilakukan perulangan dengan while untuk mengosongkan stack tersebut dengan method s.pop dan langkah terakhir digunakan method s.empty untuk memeriksa apakah stack tersebut kosong atau tidak.

## Tugas II

### Pemrograman Java

**Nama Program** : Stack Collection pada java.util.stack dan iterator  
**Bahasa Pemrograman** : Java  
**Compiler** : NetBeans IDE 8.2  
**Script program** :

```
package p10;

import java.util.Iterator;
import java.util.Stack;
/**
 *
 * @author toshiba
 */
public class StackIterator {
    public static void main(String[] args)
    {
        Stack<String> sk=new Stack<String>();
        sk.push("a");
        sk.push("c");
        sk.push("e");
        sk.push("d");
        Iterator it=sk.iterator();
        System.out.println("Size before pop() :"+sk.size());
        while(it.hasNext())
        {
            String iValue=(String)it.next();
            System.out.println("Iterator value :"+iValue);
        }

        //get and remove last element from stack
        String value =(String)sk.pop();
        System.out.println("value :"+value);
        System.out.println("Size After pop() :"+sk.size());
    }
}
```

**Output Program** :

```
Output - P10 (run)

run:
Size before pop() :4
Iterator value :a
Iterator value :c
Iterator value :e
Iterator value :d
value :d
Size After pop() :3
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Penjelasan Program** :

Program diatas merupakan program yang menggunakan Stack Collection pada java.util.stack dan iterator. Pada program sebelumnya sudah dijelaskan mengenai Stack collection pada

java. Dan iterator yang digunakan untuk membuat element-element seperti collection. ListIterator adalah extend dari class Iterator, bisa memudahkan untuk mengambil element-element yang ada di collection dengan cara maju atau mundur. Pada program diatas menggunakan import java.util.Iterator dan import java.util.Stack karena program mencakup dua operasi yaitu stack dan iterator. Pertama digunakan method push untuk menambahkan elemen pada stack yaitu sk.push("a"), sk.push("c"), sk.push("e"), sk.push("d"). Dan karena stack itu bersifat LIFO (Last In First Out) maka elemen yang disebut top adalah "d". Kemudiann terdapat perintah `System.out.println("Size before pop() : "+sk.size());` script ini digunakan untuk menampilkan ukuran stack, yaitu berjumlah 4 elemen. Kemudian ada script `String value = (String)sk.pop();` yang mana merupakan script untuk menunjukkan ada operasi pop pada stack dan disimpan dalam variable yang bernama Value. Elemen yang disimpan dalam variable adalah elemen yang top yaitu "d", maka hasil size after pop adalah 3.

### Tugas III

#### Pemrograman Java

**Nama Program** : Implementasi stack dengan Array

**Bahasa Pemrograman** : Java

**Compiler** : NetBeans IDE 8.2

**Script program** :

```
public class StackArray {
    private static final int capacity = 3;
    int arr[] = new int[capacity];
    int top = 1;

    public void push(int pushedElement){
        if (top < capacity - 1){
            top++;
            arr[top] = pushedElement;
            System.out.println("Element " + pushedElement + " is pushed to Stack !");
            printElements();
        }else{
            System.out.println("Stack Overflow !");
        }
    }

    public void pop(){
        if (top >= 0){
            top--;
            System.out.println("Pop operation done !");
        }else{
            System.out.println("Stack Underflow !");
        }
    }

    public void printElements(){
        if (top > -1){
            System.out.println("Elements in stack :");
            for (int i=0; i<= top; i++){
                System.out.println(arr[i]);
            }
        }
    }
}
```

```
-    }
- }
public static void main(String[] args){
    StackArray stackDemo = new StackArray();
    stackDemo.pop();
    stackDemo.push(23);
    stackDemo.push(2);
    stackDemo.push(73);
    stackDemo.push(21);
    stackDemo.pop();
    stackDemo.pop();
    stackDemo.pop();
    stackDemo.pop();
- }
}
```

## Output Program :

```
Output - P10 (run)

run:
Pop operation done !
Element 23 is pushed to Stack !
Elements in stack :
0
23
Element 2 is pushed to Stack !
Elements in stack :
0
23
2
Stack Overflow !
Stack Overflow !
Pop operation done !
Pop operation done !
Pop operation done !
Stack Underflow !
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Penjelasan Program :

program diatas merupakan contoh program stack dengan menggunakan array. Program diatas mendefinisikan array ada 3. Digunakan beberapa interface stack atau operasi - operasi antara lain push(), pop(), dan printElements(). Operasi push() yang mengandung parameter int pushedElement digunakan menambah data dalam operasi array jika nilai top < capacity -1, jika bernilai benar maka top akan ditambah 1 (top++) dan nilai parameter akan ditambahkan kedalam elemen array dan jika nilai top sudah lebih dari capacity maka akan menampilkan pesan “Stack Overflow !”. Operasi kedua yaitu pop(), operasi ini digunakan untuk mengambil data yang ada didalam stack array jika nilai top >= 0 maka elemen array akan dikurangi (top-) sehingga elemen yang bisa diakses adalah elemen top yang ketikan awal stack penuh diambil dengan operasi pop() maka stack terdapat satu elemen kosong. Dan jika nilai top < 0 maka akan menampilkan pesan “Stack Underflow !”. Dan yang terakhir adalah operasi printElements(), operasi ini digunakan untuk menampilkan data elemen pada stack array.

## Tugas IV

### Pemrograman Java

**Nama Program** : Implementasi stack dengan Link List

**Bahasa Pemrograman** : Java

**Compiler** : NetBeans IDE 8.2

**Script program** :

Node.java

```
1 package p10no4;
2 public class Node {
3     public int data;
4     public Node next;
5
6     public Node(int data) {
7         this.data=data;
8     }
9
10    public void displayNode() {
11        System.out.print(data);
12        System.out.print(" ");
13    }
14 }
```

LinkedList.java

```
1 package p10no4;
2 public class LinkedList{
3     private Node first = null;
4     public void insertFirst(int data) {
5         Node n = new Node (data);
6         n.next = first;
7         first = n;
8     }
9     public Node deleteFirst() {
10        Node temp = first;
11        first = first.next;
12        return temp;
13    }
14    public void displayList() {
15        Node current = first;
16        while (current != null) {
17            current.displayNode();
18            current = current.next;
19        }
20    }
21    public boolean isEmpty() {
22        return (first == null);
23    }
24 }
```

LinkedListStack.java

```
1 package p10no4;
2 public class LinkedListStack {
3     LinkedList li = new LinkedList();
4     public void push(int data){
5         li.insertFirst(data);
6     }
7     public void pop(){
8         while(!li.isEmpty()){
9             li.deleteFirst();
10        }
11    }
12    public void displayStack(){
13        System.out.println(" ");
14        li.displayList();
15    }
16 }
17
```

ContohLinkedListStack.java

```
1 package p10no4;
2 public class ContohLinkedListStack{
3     public static void main(String[] args){
4         LinkedListStack st = new LinkedListStack();
5         st.push(50);
6         st.push(70);
7         st.push(190);
8         st.displayStack();
9         st.pop();
10        st.displayStack();
11    }
12 }
```

**Output Program** :

```
: Output - P10NO4 (run)
run:
190 70 50
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Penjelasan Program** :

Program diatas merupakan program implementasi stack dengan menggunakan linkedlist. Pada program diatas menggunakan beberapa class yang berbeda yang mengandung beberapa method yaitu Node.java, LinkedList.java, LinkedListStack.java dan untuk menampilkan hasil operasi-operasi pada method tersebut menggunakan class ContohLinkedListStack.java. Pada class-class tersebut terdapat beberapa method antara lain push( ) yang berfungsi menambahkan data pertama pada linked list. Method pop( ) yang didalamnya terdapat



perulangan yang digunakan untuk menghapus data pertama pada linked list. Jika data pada linked list != empty atau tidak kosong perulangan akan terus berjalan sampai data dari linked list kosong sehingga secara otomatis tidak ada elemen dalam stack. Pada class LinkListStack terdapat method pop ini terdapat perulangan while (!li.isEmpty) deleteFirst yang berarti penggunaan perulangan untuk menghapus data dari data pertama sampai linked tidak mempunyai data sama sekali. Kemudian ada method displayStack( ) yang berfungsi untuk menampilkan data dari stack yang dimulai dari data yang pertama dalam linked list.

## Tugas V

### Pemrograman Java

**Nama Program** : Balik Kalimat  
**Bahasa Pemrograman** : Java  
**Compiler** : NetBeans IDE 8.2  
**Script program** :

```
public class TUGAS_5 {  
    public static void main(String[] args) {  
        String kalimat = "algoritma dan struktur data";  
        String huruf[] = kalimat.split("");  
  
        Stack<String> call = new Stack<>();  
        System.out.print("Masukkan Kalimat : ");  
        for(int i=0; i<kalimat.length(); i++){  
            call.push(huruf[i]);  
            System.out.print(huruf[i]);  
        }  
        System.out.println();  
  
        System.out.print("Hasil : ");  
        int jumlah = call.size();  
        for (int i=0; i<jumlah; i++) {  
            System.out.print(call.pop());  
        }  
        System.out.println();  
    }  
}
```

**Output Program** :

```
Output - P10 (run)  
run:  
Masukkan Kalimat : algoritma dan struktur data  
Hasil : atad rutkurts nad amtirogla  
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Penjelasan Program** :

Program diatas merupakan program untuk membalik kata menggunakan stack. Yang pertama praktikkan memasukkan kalimat yaitu “algoritma dan struktur data”. Yang kemudian kalimat tersebut akan dipecah-pecah perhuruf dengan perintah split dan juga digunakan kalimat.length yang berfungsi untuk menghitung panjang kalimat tersebut yang kemudian akan dipush() pada array satu persatu, push() pada stack yang berfungsi untuk menambah item pada stack pada tumpukan pertama. Setelah itu terdapat perintah call.size yaitu digunakan untuk melihat ukuran stack yang akan digunakan. Dan kemudian akan di call.pop

yaitu kemudian akan di outputkan hasil balikan kalimat tersebut. Perintah `call.pop` untuk mengambil item pada stack pada tumpukan paling atas. Karena stack bersifat Last In First Out (LIFO) maka yang akan pertama keluar adalah huruf “a”, “d”, dan sampai menghasilkan “atad rutkurts nad amtirogla”.

## **KESIMPULAN**

1. Dalam belajar pemrograman terlebih dahulu harus mengerti tentang sistem operasi, algoritma dan flowchart.
2. Dibutuhkan ketelitian dalam penulisan kode program bahasa Java, karena salah penulisan sekecil apapun tetap tidak akan bisa di eksekusi oleh program (error).
3. Stack adalah sebuah kumpulan data dimana data yang diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO.
4. Benda yang terakhir masuk dalam stack akan menjadi benda pertama yang dikeluarkan dari stack.
5. Stack dapat direpresentasikan dengan menggunakan array atau juga data menggunakan linked list.
6. Operasi yang sering diterapkan pada struktur data Stack (Tumpukan) adalah Push dan Pop.

## **DAFTAR RUJUKAN**

1. Annisa Puspa Kirana, S.Kom, M.Kom. 2017. Modul Praktikum Algoritma dan Struktur Data. Malang : Universitas Negeri Malang.
2. Pandaa, niia. 2010. Stack Dalam Struktur Data, <http://niiapanpan.blogspot.co.id/2013/05/stack-dalam-struktur-data.html>, diakses pada tanggal 13 April 2017, pukul 21.00.
3. Prabowo, Rahmat. 2015. Struktur Data dan Algoritma Stack (Tumpukan). <http://rahmat-clns.blogspot.co.id/2015/04/struktur-data-dan-algoritma.html>, diakses pada tanggal 13 April 2017, pukul 21.10.