

## JOB SHEET VII

### STACK

#### 7.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Menjelaskan mengenai algoritma Stack
2. Membuat dan mendeklarasikan struktur algoritma Stack
3. Menerapkan dan mengimplementasikan algoritma Stack

#### 7.2 Ulasan Teori

Stack adalah sebuah kumpulan data dimana data diletakkan di atas data yang lain. Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO (Last In First Out), yaitu elemen data yang terakhir masuk ke dalam stack akan menjadi elemen data pertama yang dikeluarkan dari stack. Ilustrasi stack dapat digambarkan seperti tumpukan buku, tumpukan piring, atau tumpukan sate. Stack merupakan suatu susunan koleksi data dimana data dapat ditambahkan dan dihapus dengan 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. Sedangkan untuk mengeluarkan data dari tempat yang atas tersebut, kita melakukan pop. Ilustrasi stack ditunjukkan pada Gambar berikut.

Keadaan awal



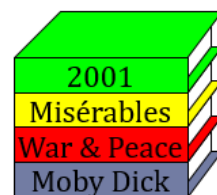
Setelah mengambil buku



Setelah menambah "Misérables":



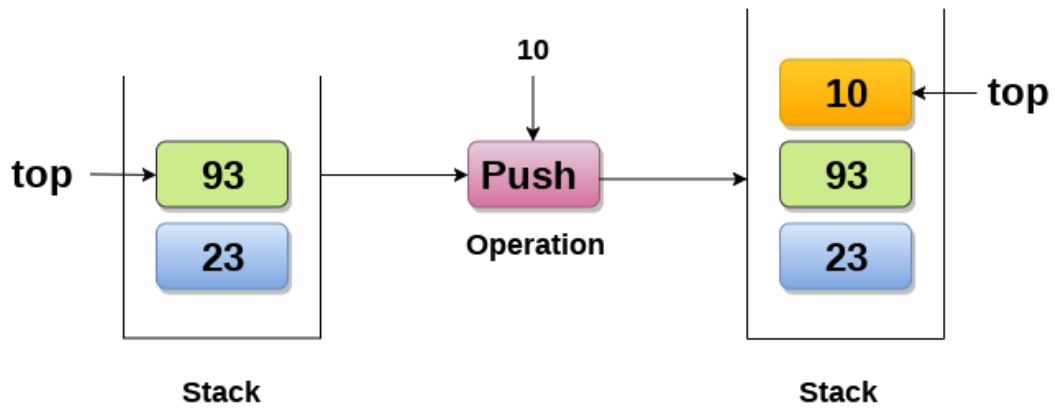
Setelah menambah "2001":



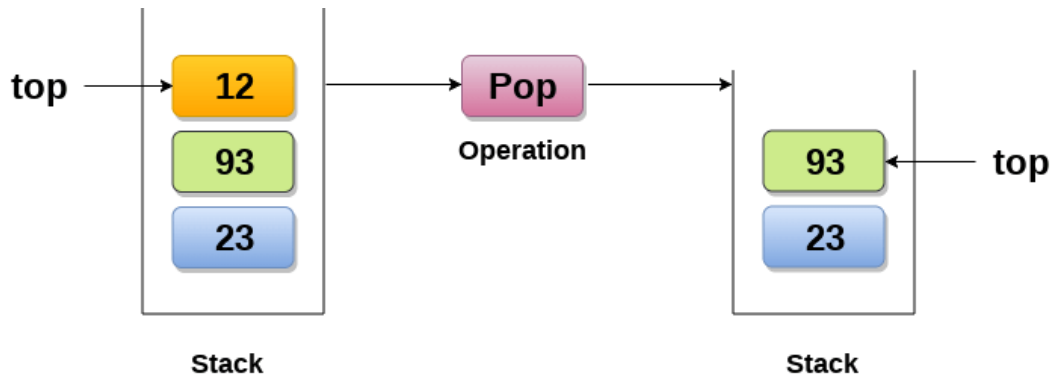
##### 7.2.1 Operasi Stack

Stack mempunyai beberapa operasi yang sering diterapkan, antara lain sebagai berikut:

1. **Push:** untuk menambah elemen pada stack pada tumpukan paling atas



2. **Pop**: untuk mengambil elemen pada stack pada tumpukan paling atas



3. **IsEmpty** untuk mengecek apakah stack sudah kosong

4. **IsFull**: untuk mengecek apakah stack sudah penuh

5. **Clear**: mengosongkan stack. Jika terdapat elemen di dalam stack, maka semua elemen dihapus

6. **Peek**: memeriksa elemen stack pada posisi paling atas (top of stack)

Dua jenis algoritma utama pada stack adalah push dan pop, sebagai berikut:

1. Algoritma Push(Stack, Item)

- 1) IF Top  $\neq$  MAX THEN
- 2) Top = Top + 1
- 3) STACK[Top] = item
- 4) Exit

2. Algoritma Pop(Stack)

- 1) IF Top  $\neq$  -1 THEN
- 2) item = Stack[Top]
- 3) Top = Top - 1
- 4) Exit

### 7.2.2 Postfix Expressions

Salah satu penerapan aplikasi stack terdapat pada bidang aritmatika adalah penulisan ekspresi matematika. Bentuk penulisan ekspresi matematika dibagi menjadi 3 notasi utama:

1. Notasi Infix  $\rightarrow A + B$  (operand, operator, operand)
2. Notasi Prefix  $\rightarrow + A B$  (operator, operand, operand)
3. Notasi Postfix  $\rightarrow A B +$  (operand, operand, operator)

Contoh :

- Infix  $\rightarrow 3 + 4 * 2$
- Prefix  $\rightarrow + 3 * 4 2$
- Postfix  $\rightarrow 3 4 + 2 *$

Notasi infix adalah notasi aritmatika yang paling sering dipakai, akan tetapi notasi postfix adalah notasi yang digunakan oleh mesin kompilasi komputer untuk mempermudah proses pengodean.

Derajat operator aritmatika yang digunakan pada ekspresi matematika antara lain:

- Pangkat (^)
- Perkalian (\*) setara dengan pembagian (/)
- Penjumlahan (+) setara dengan pengurangan (-)

Algoritma untuk mengubah notasi infix menjadi notasi postfix dijelaskan sebagai berikut:

Algoritma Postfix(Q, P) {Q=Infix, P=Postfix}

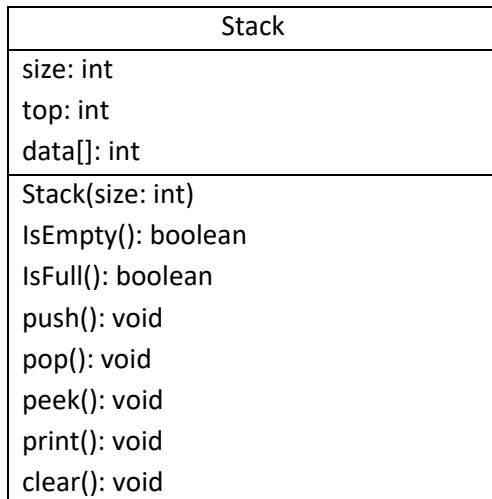
1. Telusuri Q dari elemen pertama sampai elemen yang terakhir
2. Jika menemukan operand, maka masukkan ke P
3. Jika menemukan '(', maka PUSH ke dalam Stack
4. Jika menemukan operator, maka:
  - a. Jika derajat operator lebih tinggi dari derajat operator top of stack, maka PUSH operator ke Stack
  - b. Jika derajat operator lebih rendah atau sama dengan derajat operator top of stack
    - 1) Ulangi POP(S) pada setiap elemen dan masukkan ke dalam P
    - 2) PUSH operator ke Stack
5. Jika menemukan ')', maka :
  - 1) Ulangi POP(S) pada setiap elemen sampai menemukan '(' dan masukkan ke dalam P
  - 2) Hapus '('
6. Selesai

## 7.3 Praktikum 1

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Stack.

### 7.3.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:



Berdasarkan diagram class tersebut, akan dibuat program class Stack dalam Java.

2. Pada Project **StrukturData** yang sudah dibuat pada Minggu sebelumnya, buat package dengan nama **minggu7**, kemudian buat class baru dengan nama **Stack**.
3. Tambahkan atribut size, top, dan data seperti gambar berikut ini.

```
int size;  
int top;  
int data[];
```

4. Tambahkan pula konstruktor berparameter seperti gambar berikut ini.

```
public Stack(int size) {  
    this.size = size;  
    data = new int[size];  
    top = -1;  
}
```

5. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah stack kosong.

```
public boolean IsEmpty() {  
    if (top == -1) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

6. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah stack sudah terisi penuh.

```
public boolean IsFull() {  
    if (top == size - 1) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

7. Buat method **push** bertipe void untuk menambahkan isi elemen stack dengan parameter **dt** yang bertipe integer

```
public void push(int dt) {  
    if (!IsFull()) {  
        top++;  
        data[top] = dt;  
    } else {  
        System.out.println("Isi stack penuh!");  
    }  
}
```

8. Buat method **Pop** bertipe void untuk mengeluarkan isi elemen stack

```
public void pop() {  
    if (!IsEmpty()) {  
        int x = data[top];  
        top--;  
        System.out.println("Data yang keluar: " + x);  
    } else {  
        System.out.println("Stack masih kosong");  
    }  
}
```

9. Buat method **peek** bertipe void untuk memeriksa elemen stack pada posisi paling atas.

```
public void peek() {  
    System.out.println("Elemen teratas: " + data[top]);  
}
```

10. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada stack.

```
public void print() {  
    System.out.println("Isi stack: ");  
    for (int i = top; i >= 0; i--) {  
        System.out.println(data[i] + " ");  
    }  
    System.out.println("");  
}
```

11. Buat method **clear** bertipe void untuk menghapus seluruh isi stack.

```
public void clear() {  
    if (!IsEmpty()) {  
        for (int i = top; i >= 0; i--) {  
            top--;  
        }  
        System.out.println("Stack sudah dikosongkan");  
    } else {  
        System.out.println("Gagal! Stack masih kosong");  
    }  
}
```

12. Buat class baru dengan nama **StackMain** tetap pada package **minggu7**. Buat class main, kemudian lakukan instansiasi objek dengan nama **tumpukan** dan nilai parameternya adalah 4.

```
Stack tumpukan = new Stack(4);
```

13. Lakukan pengisian data pada Stack dengan cara memanggil method **push**. Data diisi secara satu persatu.

```
tumpukan.push(15);  
tumpukan.push(31);  
tumpukan.push(9);  
tumpukan.push(12);
```

14. Tampilkan data yang sudah Anda isikan di langkah sebelumnya dengan cara memanggil method **print**.

```
tumpukan.print();
```

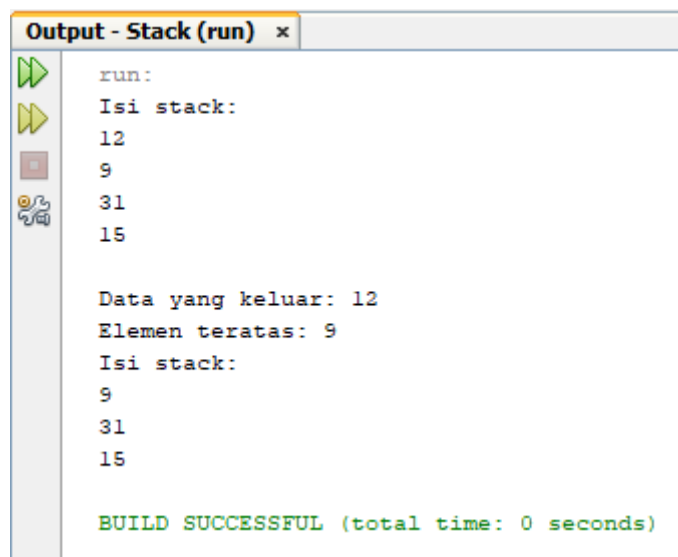
15. Lakukan pemanggilan method **pop** untuk mengeluarkan sebuah data, kemudian cek data teratas dengan method **peek**, dan tampilkan kembali seluruh data menggunakan method **print**.

```
tumpukan.pop();  
tumpukan.peek();  
tumpukan.print();
```

16. Jalankan (run) class **StackMain** dan amati hasilnya.

### 7.3.2 Verifikasi Hasil Percobaan

Samakan hasil compile kode program Anda dengan gambar berikut ini.



### 7.3.3 Pertanyaan

1. Perhatikan class **StackMain**, apakah fungsi angka 4 pada potongan kode program berikut?

```
Stack tumpukan = new Stack(4);
```

2. Lakukan penambahan data angka 17 dan 98 pada program. Tampilkan hasilnya!
3. Pada soal nomor 2, mengapa data yang bisa dimasukkan ke dalam Stack hanya angka 17, sedangkan angka 98 tidak bisa? Jelaskan!
4. Modifikasi program tersebut sehingga pengguna dapat memasukkan data melalui keyboard (menggunakan Scanner)! Catatan: Anda dapat melakukan modifikasi dengan membentuk menu program

## 7.4 Praktikum 2

Pada percobaan ini, kita akan membuat program untuk mengubah notasi infix menjadi notasi postfix.

### 7.4.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

Postfix
n: int top: int stack[]: char
Postfix(total: int) push(): void pop(): void IsOperand(char c): boolean IsOperator(char c): boolean precedence(char c): int konversi(String Q): string

Berdasarkan diagram class tersebut, akan dibuat program class Postfix dalam Java.

2. Pada package **minggu7**, buat class baru dengan nama **Postfix**. Tambahkan atribut n, top, dan stack sesuai dengan diagram class di atas.
3. Tambahkan pula konstruktor berparameter seperti gambar berikut ini.

```
public Postfix(int total) {  
    n = total;  
    top = -1;  
    stack = new char[n];  
    push('(');  
}
```



4. Buat method **push** dan **pop** bertipe void.

```
public void push(char c) {
    top++;
    stack[top] = c;
}

public char pop() {
    char item = stack[top];
    top--;
    return item;
}
```

5. Buat method **IsOperand** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operand.

```
public boolean IsOperand(char c) {
    if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == ' ' || c == '.') {
        return true;
    } else {
        return false;
    }
}
```

6. Buat method **IsOperator** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operator.

```
public boolean IsOperator(char c) {
    if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+') {
        return true;
    } else {
        return false;
    }
}
```

7. Buat method **precedence** yang mempunyai nilai kembalian integer.

```
public int presedence(char c) {
    switch (c) {
        case '^': return 3;
        case '%': return 2;
        case '/': return 2;
        case '*': return 2;
        case '-': return 1;
        case '+': return 1;
        default: return 0;
    }
}
```

8. Buat method konversi untuk melakukan konversi notasi infix menjadi notasi postfix dengan cara mengecek satu persatu elemen data di dalam stack.

```
public String konversi(String Q) {
    String P = "";
    char c;
    for (int i = 0; i < n; i++) {
        c = Q.charAt(i);
        if (IsOperand(c)) {
            P = P + c;
        }
        if (c == '(') {
            push(c);
        }
        if (c == ')') {
            while (stack[top] != '(') {
                P = P + pop();
            }
            char temp = pop();
        }
        if (IsOperator(c)) {
            while (presedence(stack[top]) >= presedence(c)) {
                P = P + pop();
            }
            push(c);
        }
    }
    return P;
}
```

9. Buat class baru dengan nama **PostfixMain** tetap pada package **minggu7**. Buat class main, kemudian buat variabel P dan Q. Variabel P digunakan untuk menyimpan hasil akhir notasi postfix setelah dikonversi, sedangkan variabel Q digunakan untuk menyimpan masukan dari pengguna berupa ekspresi matematika dengan notasi infix. Buat scanner untuk menampung data masukan dari pengguna, kemudian panggil fungsi built-in **trim** yang digunakan untuk menghapus adanya spasi di depan atau di belakang teks.

```

Scanner sc = new Scanner(System.in);
String P, Q;
System.out.println("Masukkan ekspresi matematika: ");
Q = sc.nextLine();
Q = Q.trim();
Q = Q + ")";

```

10. Buat variabel total untuk menghitung banyaknya karakter pada variabel Q.

```

int total = Q.length();

```

11. Lakukan instansiasi objek dengan nama **post** dan nilai parameternya adalah total. Kemudian panggil method **konversi** untuk melakukan konversi notasi infix Q menjadi notasi postfix P.

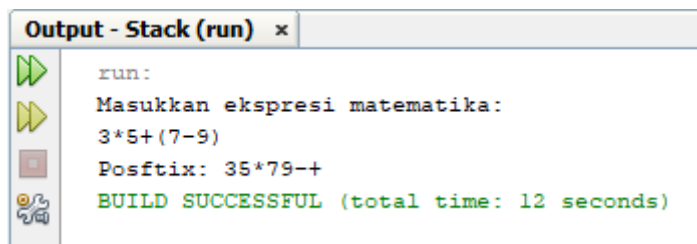
```

Postfix post = new Postfix(total);
P = post.konversi(Q);
System.out.println("Posftix: " + P);

```

12. Jalankan (run) class **PostfixMain** dan amati hasilnya.

#### 7.4.2 Verifikasi Hasil Percobaan



```

Output - Stack (run) x
run:
Masukkan ekspresi matematika:
3*5+(7-9)
Posftix: 35*79-+
BUILD SUCCESSFUL (total time: 12 seconds)

```

#### 7.4.3 Pertanyaan

1. Perhatikan class **Postfix**, jelaskan fungsi dari method **precedence**!
2. Jalankan kembali program tersebut, masukkan ekspresi  $5 \% 2 + 8 / (6 - 3)$ . Tampilkan hasilnya!
3. Pada soal nomor 2, mengapa tanda kurung tidak ditampilkan pada hasil konversi? Jelaskan!

#### 7.5 Tugas

1. Modifikasi program pada Praktikum 1 dengan mengganti data berupa angka menjadi data berupa teks (kalimat)!
2. Buat program dengan menggunakan konsep Stack untuk memasukkan sebuah kalimat, kemudian keluaran yang ditampilkan berupa kalimat terbalik!

```

run:
Masukkan Kalimat : polinema jaya
Kalimat dibalik : jaya polinema
BUILD SUCCESSFUL (total time: 7 seconds)

```