Nama : Deanissa Sherly Sabilla

Kelas / Absen : 06

JOBSHEET VII
STACK

7.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mengenal struktur data Stack
2. Membuat dan mendeklarasikan struktur data Stack
3. Menerapkan algoritma Stack dengan menggunakan array

7.2. Praktikum 1

Waktu percobaan: 30 menit

Pada percobaan ini, kita akan membuat program yang mengimplementasikan struktur data Stack dan operasi-operasi dasar pada struktur data Stack menggunakan array.

7.2.1 Langkah-langkah Percobaan

1. Buat folder dengan nama Praktikum07. Buat file Stack.java.
2. Tulis kode untuk membuat atribut dan konstruktor pada class Stack sebagai berikut:

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

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

3. Lalu tambahkan method isFull() dan isEmpty() pada class Stack sebagai berikut:

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

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

4. Tambahkan method push(int data) dan pop() sebagai berikut:

```
public void push(int dt){
    if (!isFull()){
        top++;
        data[top] = dt;
    } else {
        System.out.println(x:"Stack penuh");
    }
}

public void pop(){
    if (!isEmpty()){
        int x = data[top];
        top--;
        System.out.println("Data yang dikeluarkan dari stack: "+x);
    } else {
        System.out.println(x:"Stock masih kosong");
    }
}
```

5. Tambahkan method peek() sebagai berikut:

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

6. Tambahkan method print() dan clear() sebagai berikut:

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

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

7. Buat file **StackDemo.java** untuk mengimplementasikan class StackDemo yang berisi fungsi main untuk membuat objek Stack dan mengoperasikan method-method pada class Stack.

```
public class StackDemo {
    Run | Debug
    public static void main(String[] args) {
        Stack stack = new Stack(size:10);
        stack.push(dt:8);
        stack.push(dt:12);
        stack.push(dt:18);
        stack.print();
        stack.pop();
        stack.peek();
        stack.pop();
        stack.push(-5);
        stack.print();

    }
}
```

8. Compile dan run class StackDemo.

7.2.2 Verifikasi Hasil Percobaan

```
Isi stack:
18
12
8

Data yang dikeluarkan dari stack: 18
Elemen teratas stack: 12
Data yang dikeluarkan dari stack: 12
Isi stack:
-5
8
```

HASIL INPUT :

CLASS STACK

```java
public class Stack06 {

    int data[];
    int size;
    int top;

    public Stack06 (int size){
        this.size = size;
        data = new int [size];
        top = -1;
    }
    public boolean isFull(){
        if (top == size -1) {
            return true;
        } else {
            return false;
        }
    }
    public boolean isEmpty() {
        if (top == -1){
            return true;
        } else {
            return false;
        }
    }
```

```java
public class Stack06 {
    public void push (int dt) {
        if(!isFull()){
            top++;
            data[top] = dt;
        } else {
            System.out.println(x:"Stack Penuh");
        }
    }
    public void pop (){
        if(!isEmpty()){
            int x = data[top];
            top--;
            System.out.println("Data yang dikeluarkan dari stack : " +x);
        } else {
            System.out.println(x:"Stack masih kosong");
        }
    }
    public void peek(){
        System.out.println("Elemen teratas stack : " +data[top]);
    }
    public void print(){
        System.out.println(x:"Isi Stack : ");
        for (int i = top; i >= 0; i--) {
            System.out.println(data[i]+" ");
```

```java
        } else {
            System.out.println(x:"Stack masih kosong");
        }
    }
    public void peek(){
        System.out.println("Elemen teratas stack : " +data[top]);
    }
    public void print(){
        System.out.println(x:"Isi Stack : ");
        for (int i = top; i >= 0; i--) {
            System.out.println(data[i]+" ");
        }
        System.out.println(x:" ");
    }
    public void clear(){
        if (!isEmpty()){
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println(x:"Stack sudah dikosongkan");
        } else {
            System.out.println(x:"Stack masih kosong");
        }
```

STACK DEMO STACK

```java
public class StackDemo06 {
    Run | Debug
    public static void main(String[] args) {
        Stack06 stack = new Stack06 (size:10);
        stack.push (dt:8);
        stack.push(dt:12);
        stack.push(dt:18);
        stack.print();
        stack.pop();
        stack.peek();
        stack.pop();
        stack.push(-5);
        stack.print();
    }
}
```

HASIL OUTPUT :

```
PS C:\Users\TOSHIBA\Semester 2\Pratikum07>  c:; c
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C
\redhat.java\jdt_ws\Pratikum07_7e2865\bin' 'Stackl
Isi Stack :
18
12
8

Data yang dikeluarkan dari stack : 18
Elemen teratas stack : 12
Data yang dikeluarkan dari stack : 12
Isi Stack :
-5
8
```

7.2.3 Pertanyaan

1. Pada method pop(), mengapa diperlukan pemanggilan method isEmpty()? Apa yang terjadi jika tidak ada pemanggilan isEmpty()?
   - Karena, digunakan untuk memeriksa apakah stack kosong sebelum mencoba untuk mengeluarkan data
   - Jika tidak ada pemanggilan isEmpty() dalam method pop(), maka akan terjadi kesalahan saat mencoba untuk mengeluarkan data dari stack ketika stack sedang kosong, akan menyebabkan error yang tidak diinginkan yang dapat mempengaruhi kestabilan program
2. Jelaskan perbedaan antara method peek() dengan method pop() pada class Stack.
   - Method **pop()** pada class Stack digunakan untuk mengeluarkan dan menghapus elemen teratas dari stack, sedangkan method **peek()** digunakan untuk hanya melihat elemen teratas dari stack tanpa menghapusnya.

7.3. Praktikum 2

Waktu percobaan : 45 menit

Pada percobaan ini, kita akan membuat program yang mengilustrasikan tumpukan pakaian yang disimpan ke dalam stack. Karena sebuah pakaian mempunyai beberapa informasi, maka implementasi Stack dilakukan dengan menggunakan array of object untuk mewakili setiap elemennya.

7.3.1. Langkah-langkah Percobaan

1. Perhatikan Diagram Class Pakaian berikut ini:

Pakaian
jenis: String warna: String merk: String ukuran: String harga: double
Pakaian(jenis: String, warna: String, merk: String, ukuran: String, harga: double)

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

2. Buat class baru dengan nama **Pakaian**.
3. Tambahkan atribut-atribut Pakaian seperti pada Class Diagram Pakaian, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

```
String jenis, warna, merk, ukuran;
double harga;

Pakaian(String jenis, String warna, String merk, String ukuran, double harga) {
    this.jenis = jenis;
    this.warna = warna;
    this.merk = merk;
    this.ukuran = ukuran;
    this.harga = harga;
}
```

4. Setelah membuat class Pakaian, selanjutnya perlu dibuat class **Stack** yang berisi atribut dan method sesuai diagram Class Stack berikut ini:

Stack
size: int top: int data[]: Pakaian
Stack(size: int) IsEmpty(): boolean IsFull(): boolean push(): void pop(): void peek(): void

print(): void clear(): void

Keterangan**: Tipe data pada variabell **data** menyesuaikan dengan data yang akan akan disimpan di dalam Stack. Pada praktikum ini, data yang akan disimpan merupakan array of object dari Pakaian, sehingga tipe data yang digunakan adalah **Pakaian

5. Buat class baru dengan nama Stack. Kemudian tambahkan atribut dan konstruktor seperti gambar berikut ini.

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

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

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

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

7. 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;
    }
}
```

8. Buat method **push** bertipe void untuk menambahkan isi elemen stack dengan parameter **pkn** yang berupa object **Pakaian**

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

9. Buat method **Pop** bertipe void untuk mengeluarkan isi elemen stack. Karena satu elemen stack terdiri dari beberapa informasi (jenis, warna, merk, ukuran, dan harga), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut

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

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

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

11. 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].jenis + " " + data[i].warna + " " +
                data[i].merk + " " + data[i].ukuran + " " + data[i].harga + " ");
    }
    System.out.println("");
}
```

12. 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("Stack masih kosong");
    }
}
```

13. Selanjutnya, buat class baru dengan nama **StackMain**. Buat fungsi main, kemudian lakukan instansiasi objek dari class **Stack** dengan nama **stk** dan nilai parameternya adalah 5.

```
Stack stk = new Stack(5);
```

14. Deklarasikan Scanner dengan nama **sc**

15. Tambahkan kode berikut ini untuk menerima input data Pakaian, kemudian semua informasi tersebut dimasukkan ke dalam stack

```
char pilih;
do {
    System.out.print("Jenis: ");
    String jenis = sc.nextLine();
    System.out.print("Warna: ");
    String warna = sc.nextLine();
    System.out.print("Merk: ");
    String merk = sc.nextLine();
    System.out.print("Ukuran: ");
    String ukuran = sc.nextLine();
    System.out.print("Harga: ");
    double harga = sc.nextDouble();

    Pakaian p = new Pakaian(jenis, warna, merk, ukuran, harga);
    System.out.print("Apakah Anda akan menambahkan data baru ke stack (y/n)? ");
    pilih = sc.next().charAt(0);
    sc.nextLine();
    stk.push(p);
} while (pilih == 'y');
```

***Catatan**: sintaks sc.nextLine() sebelum sintaks st.push(p) digunakan untuk mengabaikan karakter new line*

16. Lakukan pemanggilan method print, method pop, dan method peek dengan urutan sebagai berikut.

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

17. Compile dan jalankan class **StackMain**, kemudian amati hasilnya.

7.3.2. Verifikasi Hasil Percobaan

```
Jenis: Kaos
Warna: Hitam
Merk: Nevada
Ukuran: M
Harga: 85000
Apakah Anda akan menambahkan data baru ke stack (y/n)? y
Jenis: Kemeja
Warna: Putih
Merk: Styves
Ukuran: XL
Harga: 127000
Apakah Anda akan menambahkan data baru ke stack (y/n)? y
Jenis: Celana
Warna: Biru
Merk: Levis
Ukuran: L
Harga: 189500
Apakah Anda akan menambahkan data baru ke stack (y/n)? n
Isi stack:
Celana Biru Levis L 189500.0
Kemeja Putih Styves XL 127000.0
Kaos Hitam Nevada M 85000.0

Data yang keluar: Celana Biru Levis L 189500.0
Elemen teratas: Kemeja Putih Styves XL 127000.0
Isi stack:
Kemeja Putih Styves XL 127000.0
Kaos Hitam Nevada M 85000.0
```

HASIL INPUT :

CLASS PAKAIAN

Pakaian06.java > Pakaian06 > Pakaian06(String, String, String, String, double)

```java
public class Pakaian06 {
    String jenis, warna, merk, ukuran;
    double harga;

    Pakaian06 (String jenis, String warna, String merk, String ukuran, double harga){
        this.jenis = jenis;
        this.warna = warna;
        this.merk = merk;
        this.ukuran = ukuran;
        this.harga = harga;
    }
}

```

CLASS STACK PAKAIAN

StackPakaian06.java > StackPakaian06 > push(Pakaian06)

```java
public class StackPakaian06 {
    int size;
    int top;
    Pakaian06 data[];

    public StackPakaian06(int size){
        this.size = size;
        data = new Pakaian06[size];
        top = -1;
    }
    public boolean IsEmpty(){
        if (top== -1){
            return true;
        } else {
            return false;
        }
    }
    public boolean IsFull(){
        if (top == size-1){
            return true;
        } else {
            return false;
        }
    }
```

```java
    public void push(Pakaian06 p){
        if(!IsFull()){
            top++;
            data[top] = p;
        } else {
            System.out.println(x:"Isi stack penuh!");
        }
    }
    public void pop(){
        if(!IsEmpty()){
            Pakaian06 x = data[top];
            top--;
            System.out.println("Data yang keluar : " + x.jenis + " " + x.warna + " " + x.merk + " " + x.ukuran + " " +
        } else {
            System.out.println(x:"Stack masih kosong");
        }
    }
    public void peek (){
        System.out.println("Elemen teratas : " + data[top].jenis + " " + data[top].warna + " " +data[top].merk + " " +
    }
```

```java
    public void print(){
        System.out.println(x:"Isi Stack : ");
    for (int i = top; i >= 0; i--) {
        System.out.println(data[i].jenis + " " + data[i].warna + " " + data[i].merk + " " + data[i].ukuran + " " + dat
    }
    System.out.println(x:" ");
    }
    public void clear(){
        if (!IsEmpty()) {
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println(x:"Stack sudah dikosongkan");
        } else {
            System.out.println(x:"Stack masih kosong");
        }
    }
}
```

CLASS MAIN

```
5   public class StackMain06 {
7       public static void main(String[] args) {
9           StackPakaian06 stk = new StackPakaian06(size:5);
10          char pilih;
11          do {
12              System.out.print(s:"Jenis : ");
13              String jenis = sc.nextLine();
14              System.out.print(s:"Warna : ");
15              String warna = sc.nextLine();
16              System.out.print(s:"Merk : ");
17              String merk = sc.nextLine();
18              System.out.print(s:"Ukuran : ");
19              String ukuran = sc.nextLine();
20              System.out.print(s:"Harga : ");
21              double harga = sc.nextDouble();
22
23              Pakaian06 p = new Pakaian06(jenis, warna, merk, ukuran, harga);
24              System.out.print(s:"Apakah Anda akan menambahkan data baru ke stack (y/n)? ");
25              pilih = sc.next().charAt(index:0);
26              sc.nextLine();
27              stk.push(p);
28          } while (pilih == 'y');
29              stk.print();
30              stk.pop();
31              stk.peek();
32              stk.print();;
```

HASIL OUTPUT :

```
Jenis : Kaos
Warna : Hitam
Merk : Nevada
Ukuran : M
Harga : 85000
Apakah Anda akan menambahkan data baru ke stack (y/n)? y
Jenis : Kemeja
Warna : Putih
Merk : Styvee
Ukuran : XL
Harga : 127000
Apakah Anda akan menambahkan data baru ke stack (y/n)? y
Jenis : Celana
Warna : Biru
Merk : Levis
Ukuran : L
Harga : 189500
Apakah Anda akan menambahkan data baru ke stack (y/n)? n
Isi Stack :
Celana Biru Levis L 189500.0
Kemeja Putih Styvee XL 127000.0
Kaos Hitam Nevada M 85000.0

Data yang keluar : Celana Biru Levis L 189500.0
Elemen teratas : Kemeja Putih Styvee XL 127000.0
Isi Stack :
Kemeja Putih Styvee XL 127000.0
Kaos Hitam Nevada M 85000.0
```

7.3.3. Pertanyaan

1. Berapa banyak data pakaian yang dapat ditampung di dalam stack? Tunjukkan potongan kode program untuk mendukung jawaban Anda tersebut!

   - Banyak data pakaian yang ditampung yaitu berjumlah 5 elemen

```
8           Scanner sc = new Scanner (System.in);
9           StackPakaian06 stk = new StackPakaian06(size:5);
10          char pilih;
```

2. Perhatikan class **StackMain**, pada saat memanggil fungsi push, parameter yang dikirimkan adalah **p**. Data apa yang tersimpan pada variabel **p** tersebut?

```
stk.push(p);
```

   - Berisi data pakaian yang dimasukkan melalui input. Data pakaian ini meliputi jenis pakaian, warna, merk, ukuran, dan harga.

3. Apakah fungsi penggunaan **do-while** yang terdapat pada class **StackMain**?
   - Yaitu untuk mengulang proses meminta input data pakaian dan menambahkannya ke dalam stack hingga memilih untuk tidak menambahkan data lagi.
4. Modifikasi kode program pada class **StackMain** sehingga pengguna dapat memilih operasi-operasi pada stack (push, pop, peek, atau print) melalui pilihan menu program dengan memanfaatkan kondisi IF-ELSE atau SWITCH-CASE!
   - **HASIL INPUT :**

     CLASS MAIN YANG DI TAMBAHKAN

```
StackMain06.java > StackMain06 > main(String[])
 5  public class StackMain06 {
 7      public static void main(String[] args) {
 8          Scanner sc = new Scanner (System.in);
 9          StackPakaian06 stk = new StackPakaian06(size:5);
10          char pilih;
11          do {
12              System.out.println(x:"Menu:");
13              System.out.println(x:"1. Push");
14              System.out.println(x:"2. Pop");
15              System.out.println(x:"3. Peek");
16              System.out.println(x:"4. Print");
17              System.out.println(x:"5. Keluar");
18              System.out.print(s:"Pilih operasi yang ingin dilakukan: ");
19              int choice = sc.nextInt();
20              sc.nextLine();
21
22              switch (choice) {
23              case 1:
24              System.out.print(s:"Jenis : ");
25              String jenis = sc.nextLine();
26              System.out.print(s:"Warna : ");
27              String warna = sc.nextLine();
28              System.out.print(s:"Merk : ");
29              String merk = sc.nextLine();
30              System.out.print(s:"Ukuran : ");
31              String ukuran = sc.nextLine();
```

```
32              System.out.print(s:"Harga : ");
33              double harga = sc.nextDouble();
34              sc.nextLine();
35
36              Pakaian06 p = new Pakaian06(jenis, warna, merk, ukuran, harga);
37              stk.push(p);
38                      break;
39                  case 2:
40                      stk.pop();
41                      break;
42                  case 3:
43                      stk.peek();
44                      break;
45                  case 4:
46                      stk.print();
47                      break;
48                  case 5:
49                      System.out.println(x:"Terima kasih!");
50                      System.exit(status:0); // Keluar dari program
51                  default:
52                      System.out.println(x:"Pilihan tidak valid.");
53              }
54
55              System.out.print(s:"Apakah Anda ingin melanjutkan (y/n)? ");
```

```
51                  default:
52                      System.out.println(x:"Pilihan tidak valid.");
53              }
54
55              System.out.print(s:"Apakah Anda ingin melanjutkan (y/n)? ");
56              pilih = sc.next().charAt(index:0);
57              sc.nextLine(); // Membersihkan newline
58          } while (pilih == 'y');
59      }
60  }
```

➢ **HASIL OUTPUT :**

```
Menu:
1. Push
2. Pop
3. Peek
4. Print
5. Keluar
Pilih operasi yang ingin dilakukan: 1
Jenis : Kaos
Warna : Hitam
Merk : Nevada
Ukuran : L
Harga : 200000
Apakah Anda ingin melanjutkan (y/n)? y
Menu:
1. Push
2. Pop
3. Peek
4. Print
5. Keluar
Pilih operasi yang ingin dilakukan: 3
Elemen teratas : Kaos Hitam Nevada L 200000.0
Apakah Anda ingin melanjutkan (y/n)? y
Menu:
1. Push
2. Pop
3. Peek
4. Print
5. Keluar
```

7.4. Praktikum 3

Waktu percobaan : 30 menit

Pada percobaan ini, kita akan membuat program untuk melakukan konversi 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(c: char): void pop(): void IsOperand(c: char): boolean IsOperator(c: char): boolean derajat(c: char): int konversi(Q: String): string

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

2. Buat class baru dengan nama **Postfix**. Tambahkan atribut **n**, **top**, dan **stack** sesuai diagram class Postfix tersebut.
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 **derajat** yang mempunyai nilai kembalian integer untuk menentukan derajat operator.

```
public int derajat(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 pada **String Q** sebagai parameter masukan.

```
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();
            }
            pop();
        }
        if (IsOperator(c)) {
            while (derajat(stack[top]) >= derajat(c)) {
                P = P + pop();
            }
            push(c);
        }
    }
    return P;
}
```

9. Selanjutnya, buat class baru dengan nama **PostfixMain**. 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. Deklarasikan variabel Scanner dengan nama sc, kemudian panggil fungsi *built-in* **trim** yang digunakan untuk menghapus adanya spasi di depan atau di belakang teks dari teks persamaan yang dimasukkan oleh pengguna.

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

Penambahan string **“)”** digunakan untuk memastikan semua simbol/karakter yang masih berada di stack setelah semua persamaan terbaca, akan dikeluarkan dan dipindahkan ke postfix.

10. Buat variabel total untuk menghitung banyaknya karaketer 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. Compile dan jalankan class **PostfixMain** dan amati hasilnya.

7.4.2. Verifikasi Hasil Percobaan

```
Masukkan ekspresi matematika (infix):
a+b*(c+d-e)/f
Posftix: abcd+e-*f/+
```

HASIL INPUT :

CLASS POSTFIX

```
Posthix06.java > Posthix06 > n
1   public class Postfix06 {
2       int n, top;
3       char [] stack;
4       public Postfix06 (int total){
5           n = total;
6           top = -1;
7           stack = new char[n];
8           push(c:'(');
9       }
10      public void push (char c){
11          top++;
12          stack[top] =c;
13      }
14      public char pop(){
15          char item = stack[top];
16          top--;
17          return item;
18      }
19      public boolean IsOperand (char c){
20          if((c >= 'A' && c <= 'Z')||(c >= 'a' && c <= 'z') || (c >= '0' && c<= '9'|| (c == ' ' && c <= '.'))){
21              return true;
22          } else {
23              return false;
24          }
25      }
```

```
26      public boolean IsOperator (char c) {
27          if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+') {
28              return true;
29          } else {
30              return false;
31          }
32      }
33      public int derajat (char c) {
34          switch (c) {
35              case '^':
36              return 3;
37              case '%':
38              return 2;
39              case '/':
40              return 2;
41              case '*':
42              return 2;
43              case '-':
44              return 1;
45              case '+':
46              return 1;
47              default :
48              return 0;
49          }
50      }
```

```
1   public class Postfix06 {
51      public String konversi (String Q) {
52          String P = " ";
53          char c;
54          for (int i= 0; i < n; i++) {
55              c = Q.charAt(i);
56              if (IsOperand(c)){
57                  P = P + c;
58              }
59              if (c == '(') {
60                  push (c);
61              }
62              if (c == ')') {
63                  while (stack [top] != '('){
64                      P = P + pop ();
65                  }
66                  pop();
67              }
68              if (IsOperator(c)){
69                  while (derajat(stack[top]) >= derajat (c)) {
70                      P = P + pop();
71                  }
72                  push(c);
73              }
74          }
75          return P;
```

CLASS POSTFIX MAIN

```
PostFixMain.java > PostFixMain
1   import java.util.Scanner;
2   public class PostFixMain {
        Run | Debug
3       public static void main(String[] args) {
4           Scanner sc = new Scanner (System.in);
5           String P, Q;
6           System.out.println(x:"Masukkan ekspresi matematika (infix) : ");
7           Q = sc.nextLine();
8           Q = Q.trim();
9           Q = Q + ")";
10
11          int total = Q.length();
12          Postfix06 post = new Postfix06 (total);
13          P = post.konversi(Q);
14          System.out.println("Postfix : " + P);
15      }
16  }
```

HASIL OUTPUT :

```
.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
0249eb69d4f7cc\redhat.java\jdt_ws\Pratikum07_7e2865\bin' 'PostFix
Masukkan ekspresi matematika (infix) :
a+b*(c+d-e)/f
Postfix :  abcd+e-*f/+
PS C:\Users\TOSHIBA\Semester 2\Pratikum07>
```

7.4.3. Pertanyaan

1. Perhatikan class **Postfix**, jelaskan alur kerja method **derajat**!
   - Untuk menentukan derajat prioritas dari operator matematika yang diberikan. Derajat prioritas ini digunakan dalam konversi infix ke postfix untuk memastikan urutan operasi yang benar.
2. Apa fungsi kode program berikut?

```
c = Q.charAt(i);
```

   - Untuk mengambil karakter pada posisi tertentu dalam string Q dan menyimpannya ke dalam variabel c

3. Jalankan kembali program tersebut, masukkan ekspresi **5*4^(1+2)%3**. Tampilkan hasilnya!

   - Hasil Output :

```
va\jdt_ws\Pratikum07_7e2865\bin  PostFixMain
Masukkan ekspresi matematika (infix) :
5*4^(1+2)%3
Postfix :  5412+^*3%
PS C:\Users\TOSHIBA\Semester 2\Pratikum07>
```

4. Pada soal nomor 3, mengapa tanda kurung tidak ditampilkan pada hasil konversi? Jelaskan!

   - Karena dalam algoritma konversi dari infix ke postfix, tanda kurung berperan dalam mengatur urutan evaluasi operasi. Namun, dalam ekspresi postfix, urutan operasi sudah terdefinisi berdasarkan posisi operand dan operator, sehingga tanda kurung tidak diperlukan.