

JOBSHEET VII STACK

PRAKTIKUM 1

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;  
}
```

```
public class Stack22 {  
  
    int data[];  
    int size;  
    int top;  
  
    public Stack22(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 isFull() {  
    if (top == size-1) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Valina 22

```
public boolean isEmpty() {
    if (top == -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");
    }
}
```

```
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 diekluarkan dari stack: "+x);
    } else {
        System.out.println(x: "Stock masih kosong");
    }
}
```

Valina 22

5. Tambahkan method peek() sebagai berikut:

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

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

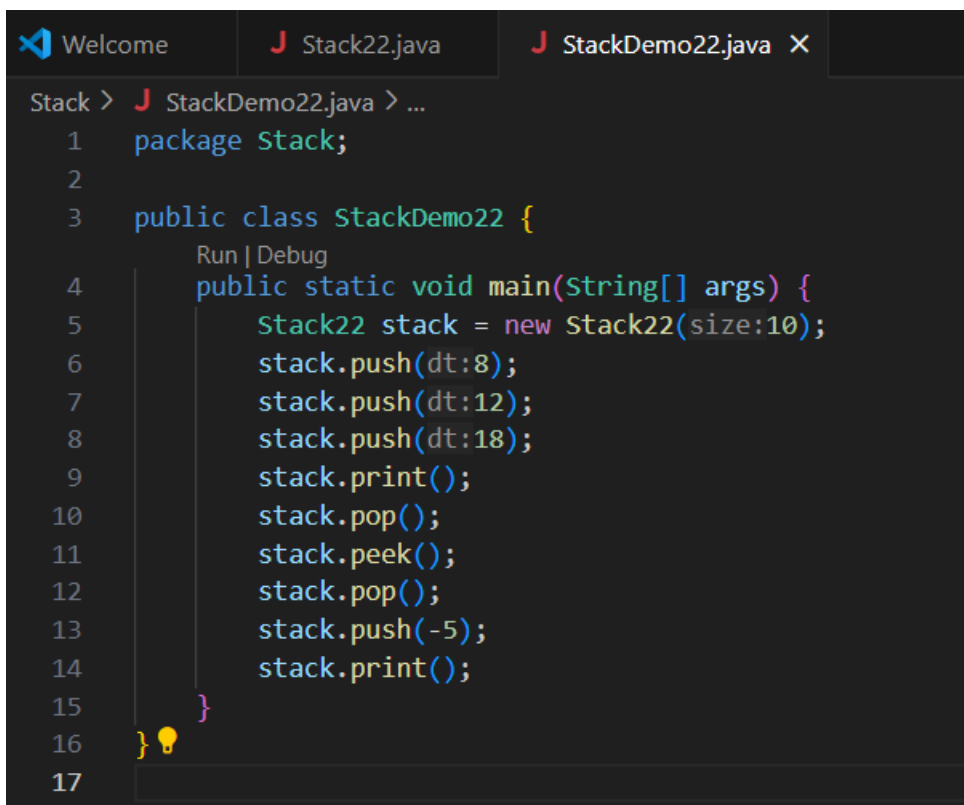
```
public void clear(){
    if (!isEmpty()){
        for (int i = top; i >= 0; i--){
            top--;
        }
        System.out.println(x:"Stack sudah dikosongkan");
    }else{
        System.out.println(x:"Stack masih kosong");
    }
}
```

```
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");
    }
}
```

Valina 22

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();
    }
}
```



```
Stack > StackDemo22.java > ...
1 package Stack;
2
3 public class StackDemo22 {
    Run | Debug
4     public static void main(String[] args) {
5         Stack22 stack = new Stack22(size:10);
6         stack.push(dt:8);
7         stack.push(dt:12);
8         stack.push(dt:18);
9         stack.print();
10        stack.pop();
11        stack.peek();
12        stack.pop();
13        stack.push(-5);
14        stack.print();
15    }
16 }
17
```

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

Valina 22

```
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

PS E:\1. KULIAH\SEMESTER 2\PRAKTIKUM ALGORITMA\JOBSHEET 7>
```

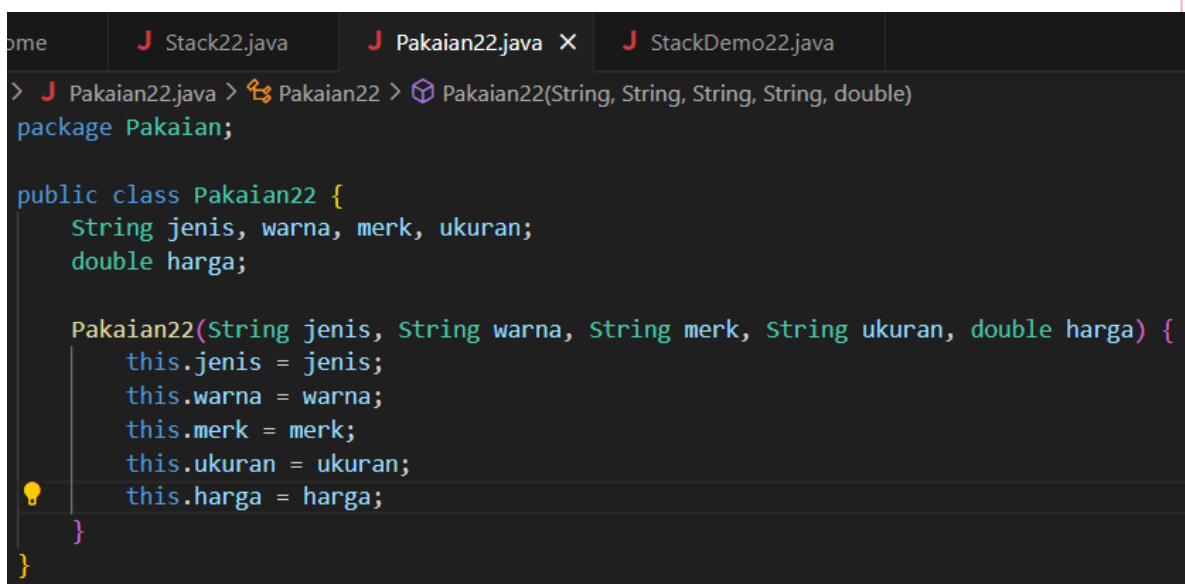
PRAKTIKUM 2

7.3.1. Langkah-langkah Percobaan

1. Buat class baru dengan nama **Pakaian**.
2. 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;
}
```



```
Stack22.java  Pakaian22.java X  StackDemo22.java
> Pakaian22.java > Pakaian22 > Pakaian22(String, String, String, String, double)
package Pakaian;

public class Pakaian22 {
    String jenis, warna, merk, ukuran;
    double harga;

    Pakaian22(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;
    }
}
```

Valina 22

3. 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;
}
```

```
int size;
int top;
Pakaian22 data[];

public Stack22(int size) {
    this.size = size;
    data = new Pakaian22[size];
    top = -1;
}
```

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

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

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

5. 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;
    }
}
```

Valina 22

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

6. 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!");
    }
}
```

```
public void push(Pakaian22 pkn) {
    if (!isFull()) {
        top++;
        data[top] = pkn;
    } else {
        System.out.println(x:"Isi stack penuh!");
    }
}
```

7. 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");
    }
}
```

Valina 22

```
public void pop() {
    if (!isEmpty()) {
        Pakaian22 x = data[top];
        top--;
        System.out.println("Data yang keluar: " + x.jenis + " " + x.warna +
            " " + x.merk + " " + x.ukuran + " " + x.harga);
    } else {
        System.out.println(x:"Stack masih kosong");
    }
}
```

8. 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);
}
```

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

9. 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("");
}
```

```
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 + " " + data[i].harga + " ");
    }
    System.out.println(x:"");
}
```

10. 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");
    }
}
```


Valina 22

```
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");
    }
}
```

11. 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);
```

12. Deklarasikan Scanner dengan nama **sc**

```
import java.util.Scanner;

Pakaian.StackMain22

public class StackMain22 {
    Run | Debug
    public static void main(String[] args) {
        Stack22 stk = new Stack22(size:5);
```

13. 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');
```

Valina 22

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

    Pakaian22 p = new Pakaian22(jenis, warna, merk, ukuran, harga);
    System.out.print(s:"Apakah Anda akan menambahkan data baru ke stack (y/n)? ");
    pilih = sc.next().charAt(index:0);
    sc.nextLine();
    stk.push(p);
} while (pilih == 'y');
```

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

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

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

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

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

7.3.2. Verifikasi Hasil Percobaan

Valina 22

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

Valina 22

```
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 Kemeja XL 127000.0
Isi stack:
Kemeja Putih Styves XL 127000.0
Kaos Hitam Nevada M 85000.0

PS E:\1. KULIAH\SEMESTER 2\PRAKTIKUM ALGORITMA\JOBSHEET 7> e;; cd 'e:\1. KULIAH\SEMES
```

Valina 22

7.3. Praktikum 3

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;
}
```

Valina 22

```
public Postfix22(int total) {
    n = total;
    top = -1;
    stack = new char[n];
    push(c: '(');
}

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;
    }
}
```

```
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.

Latihan 22

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

```
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;  
    }  
}
```

Latihan 22

```
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.

Valina 22

```

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;
}

```

```

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;
}

```

Valina 22

- 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 + ")";
```

```
Scanner sc = new Scanner(System.in);
String P, Q;

System.out.println(x:"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.

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

```
int total = Q.length();
```

```
int total = Q.length();
```

- 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);
```

```
Postfix22 post = new Postfix22(total);
P = post.konversi(Q);
System.out.println("Postfix: " + P);
```

- Compile dan jalankan class **PostfixMain** dan amati hasilnya.

Valina 22

7.4.2. Verifikasi Hasil Percobaan

Masukkan ekspresi matematika (infix):

$a+b*(c+d-e)/f$

Posftix: $abcd+e-*f/+$

Masukkan ekspresi matematika (infix):

$a+b*(c+d-e)/f$

Postfix: $abcd+e-*f/+$

PS E:\1. KULIAH\SEMESTER 2\PRAKTIKUM ALGORITMA\JOBSHEET 7>