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

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Stack.

7.2.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class Stack berikut ini:

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

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

2. Buat project baru dengan nama **Jobsheet7**, buat package dengan nama **Praktikum1**, 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. Selanjutnya, buat class baru dengan nama **StackMain** tetap pada package **Praktikum1**. Buat fungsi main, kemudian lakukan instansiasi objek dengan nama **stk** dan nilai parameternya adalah 5.

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

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

```
stk.push(15);
stk.push(27);
stk.push(13);
```

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

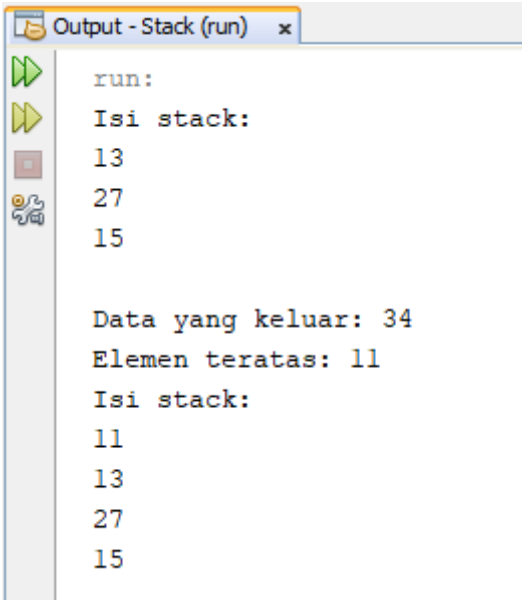
```
stk.print();
```

15. Lakukan pengisian data kembali dengan memanggil method push untuk menambahkan dua data baru, kemudian panggil method **pop** untuk mengeluarkan sebuah data, kemudian cek data teratas dengan method **peek**, dan tampilkan kembali seluruh data menggunakan method **print**.

```
stk.push(11);
stk.push(34);
stk.pop();
stk.peek();
stk.print();
```

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

7.2.2 Verifikasi Hasil Percobaan



7.2.3 Pertanyaan

1. Perhatikan class **StackMain**, apakah fungsi angka 5 pada potongan kode program berikut?

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

2. Lakukan penambahan data ke stack sebanyak dua kali, menggunakan angka 18 dan 40. Tampilkan hasilnya!
3. Pada soal nomor 2, mengapa data yang dimasukkan ke dalam Stack hanya angka 18, sedangkan angka 40 tidak dimasukkan? Jelaskan!

7.3 Praktikum 2

Pada percobaan ini, kita akan membuat program yang mengilustrasikan tumpukan buku yang disimpan ke dalam stack. Karena sebuah buku 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 Buku berikut ini:

Buku
judul: String namaPengarang: String tahunTerbit: int jmlHalaman: int harga: int
Buku(jd: String, nm: String, thn: int, hal: int, hg: int)

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

2. Buat package dengan nama **Praktikum2**, kemudian buat class baru dengan nama **Buku**.
3. Tambahkan atribut-atribut Buku seperti pada Class Diagram Buku, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

```
String judul, namaPengarang;
int tahunTerbit, jmlHalaman, harga;

Buku(String jd, String nm, int thn, int hal, int hg) {
    judul = jd;
    namaPengarang = nm;
    tahunTerbit = thn;
    jmlHalaman = hal;
    harga = hg;
}
```

4. Salin kode program class **Stack** pada **Praktikum1** untuk digunakan kembali pada **Praktikum2** ini. Karena pada **Praktikum1**, data yang disimpan pada stack hanya berupa array bertipe integer, sedangkan pada Praktikum2 data yang digunakan adalah object, maka perlu dilakukan modifikasi pada class **Stack** tersebut.
5. Lakukan modifikasi pada class **Stack** dengan mengubah tipe **int data[]** menjadi **Buku data[]** karena pada kasus ini data yang akan disimpan pada stack berupa object Buku. Modifikasi perlu dilakukan pada **atribut**, **konstruktor**, method **push**, dan method **pop**.

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

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

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

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

6. Karena satu elemen stack terdiri dari beberapa informasi (judul, nama pengarang, tahun terbit, jumlah halaman, dan harga), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga modifikasi perlu dilakukan pada method **pop**, method **peek**, dan method **print**. Ketika mencetak, semua atribut dari objek buku harus dipanggil.

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

public void peek() {
    System.out.println("Elemen teratas: " + data[top].judul + " " +
            data[top].namaPengarang + " " + data[top].tahunTerbit + " "
            + data[top].jmlHalaman + " " + data[top].harga);
}

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

7. Selanjutnya, buat class baru dengan nama **StackMain** tetap pada package **Praktikum2**. Buat fungsi main, kemudian lakukan instansiasi objek dengan nama **st** dan nilai parameternya adalah 8.
8. Deklarasikan Scanner dengan nama **sc**
9. Tambahkan kode berikut ini untuk menerima input data Buku, kemudian semua informasi tersebut dimasukkan ke dalam stack

```
char pilih;
do {
    System.out.print("Judul: ");
    String judul = sc.nextLine();
    System.out.print("Nama Pengarang: ");
    String nama = sc.nextLine();
    System.out.print("Tahun Terbit: ");
    int tahun = sc.nextInt();
    System.out.print("Jumlah Halaman: ");
    int jml = sc.nextInt();
    System.out.print("Harga: ");
    int hrg = sc.nextInt();

    Buku bk = new Buku(judul, nama, tahun, jml, hrg);
    System.out.print("Apakah Anda akan menambahkan data baru ke stack (y/n)? ");
    pilih = sc.next().charAt(0);
    sc.nextLine();
    st.push(bk);
} while (pilih == 'y');
```

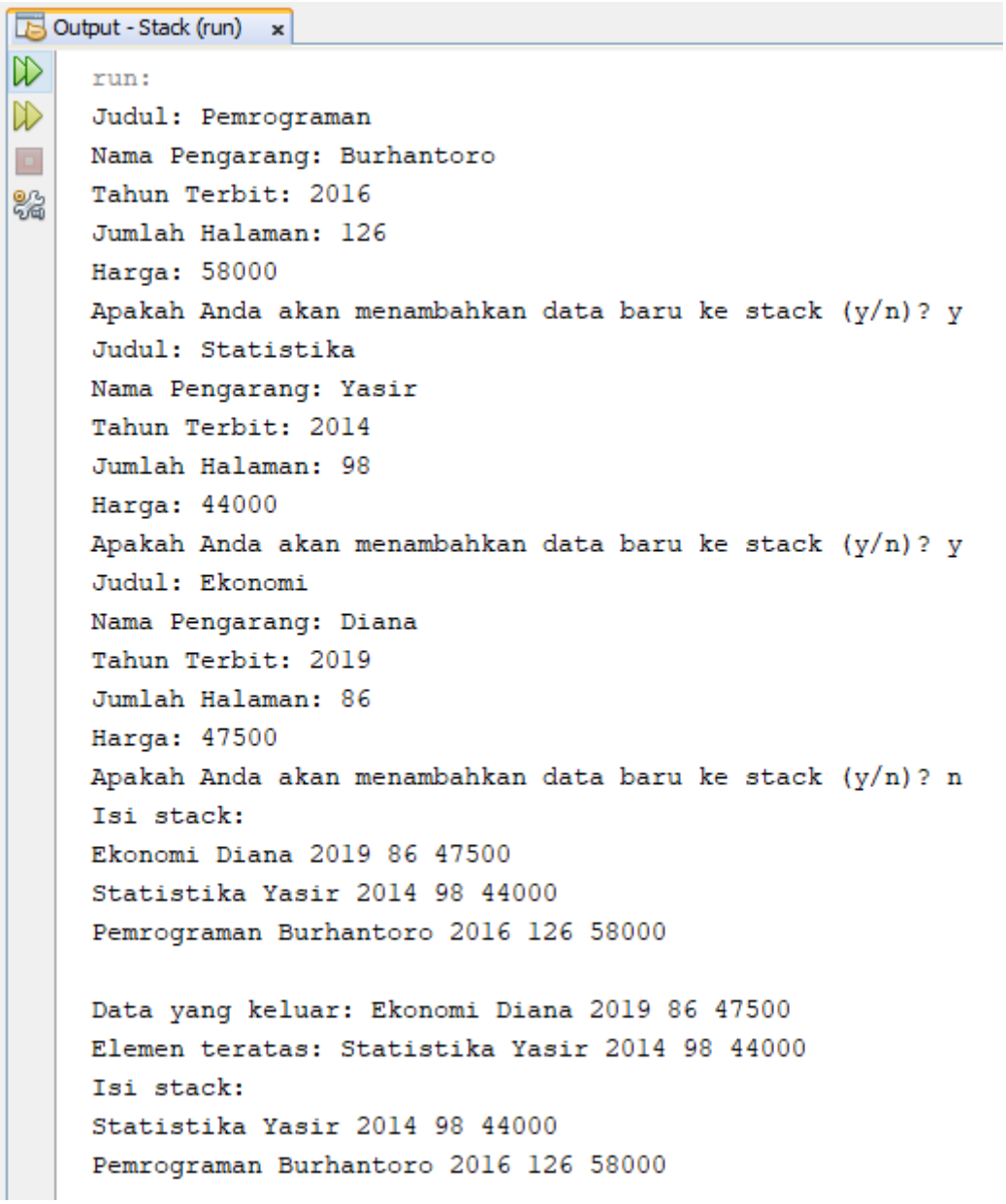
***Catatan**: sintaks sc.nextLine() sebelum sintaks st.push(bk) digunakan untuk mengabaikan karakter new line*

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

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

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

7.3.2 Verifikasi Hasil Percobaan



7.3.3 Pertanyaan

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

   ```
   st.push(bk);
   ```

2. Tunjukkan potongan kode program untuk menentukan kapasitas penampungan stack!
3. Apakah fungsi penggunaan **do-while** yang terdapat pada class **StackMain**?
4. Modifikasi kode program pada class **StackMain** sehingga pengguna dapat memilih operasi-operasi pada stack (push, pop, peek, atau print) melalui pilihan menu program!

7.4 Praktikum 3

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 package dengan nama **Praktikum3**, kemudian 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** tetap pada package **Praktikum3**. 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.

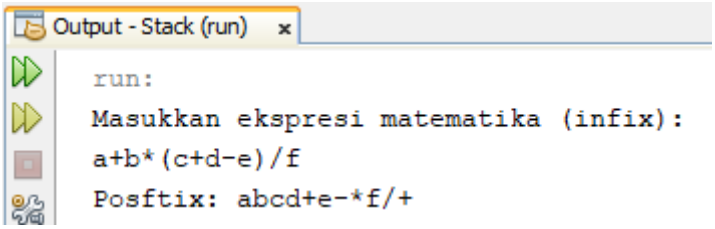
```
int total = Q.length();
```

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



7.4.3 Pertanyaan

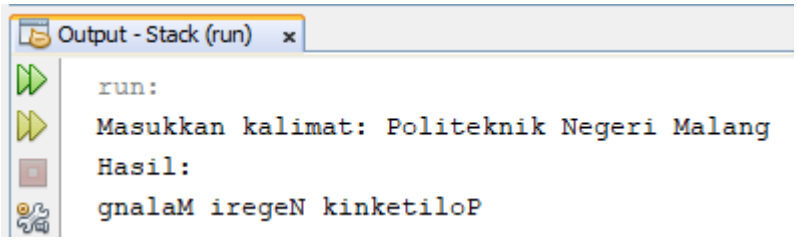
1. Perhatikan class **Postfix**, jelaskan alur kerja method **derajat**!
2. Apa fungsi kode program berikut?

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

3. Jalankan kembali program tersebut, masukkan ekspresi **3*5^(8-6)%3**. Tampilkan hasilnya!
4. Pada soal nomor 2, mengapa tanda kurung tidak ditampilkan pada hasil konversi? Jelaskan!

7.5 Tugas

1. Buat program dengan menggunakan konsep Stack untuk memasukkan sebuah kalimat, kemudian keluaran yang ditampilkan berupa kalimat dengan urutan karakter terbalik!



2. Setiap hari Minggu, Dewi pergi berbelanja ke salah satu supermarket yang berada di area rumahnya. Setiap kali selesai berbelanja, Dewi menyimpan struk belanjaannya di dalam laci. Setelah dua bulan, ternyata Dewi sudah mempunyai delapan struk belanja. Dewi berencana mengambil lima struk belanja untuk ditukarkan dengan voucher belanja.

   Buat sebuah program stack untuk menyimpan data struk belanja Dewi, kemudian lakukan juga proses pengambilan data struk belanja. Informasi yang tersimpan pada struk belanja terdiri dari:

   - Nomor transaksi
   - Tanggal pembelian
   - Jumlah barang yang dibeli
   - Total harga bayar