JOBSHEET VII STACK

# 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

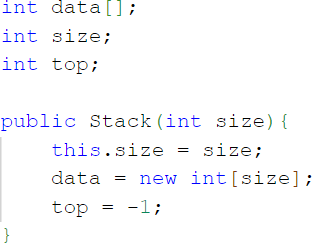
# Praktikum 1

## Waktu percobaan: 30 menit

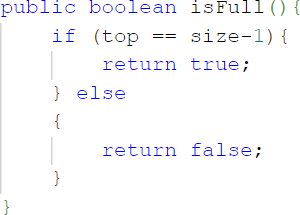
Pada percobaan ini, kita akan membuat program yang mengimplementasikan struktur data Stack dan operasi-operasi dasar pada struktur data Stack menggunakan array.

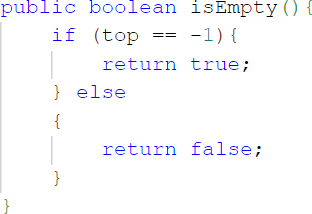
# Langkah-langkah Percobaan

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

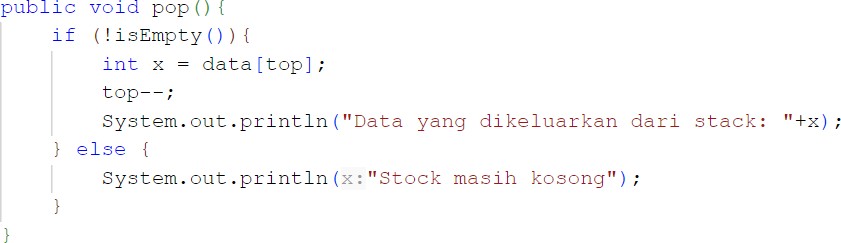
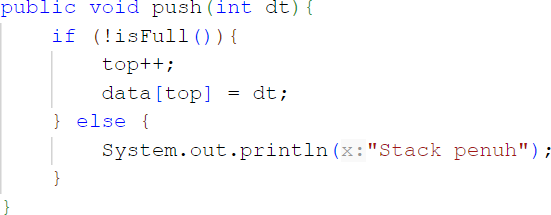


* + - 1. Lalu tambahkan method isFull() dan isEmpty() pada class Stack sebagai berikut:

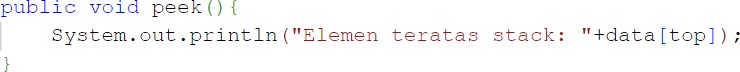




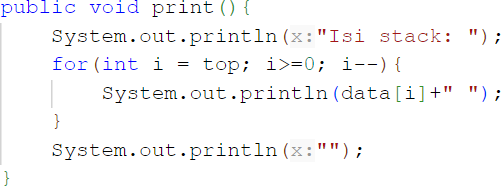
* + - 1. Tambahkan method push(int data) dan pop() sebagai berikut:

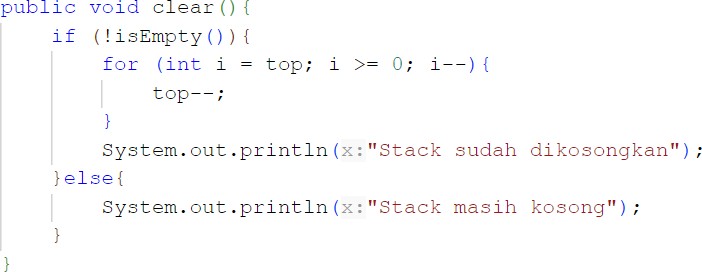


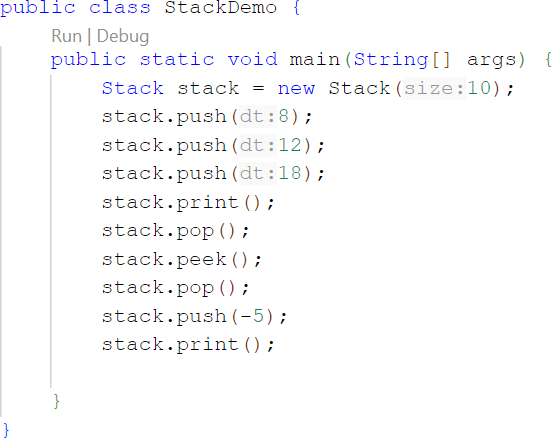
* + - 1. Tambahkan method peek() sebagai berikut:



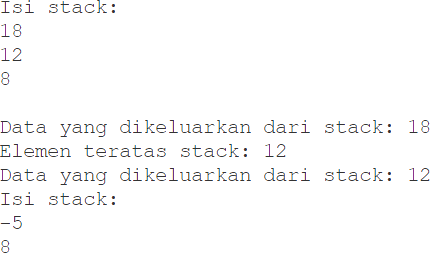
* + - 1. Tambahkan method print() dan clear() sebagai berikut:





* + - 1. Buat file **StackDemo.java** untuk mengimplementasikan class StackDemo yang berisi fungsi main untuk membuat objek Stack dan mengoperasikan method-method pada class Stack.
      2. Compile dan run class StackDemo.

# Verifikasi Hasil Percobaan



* + 1. **Pertanyaan**
       1. Pada method pop(), mengapa diperlukan pemanggilan method isEmpty()? Apa yang terjadi jika tidak ada pemanggilan isEmpty()?
       2. Jelaskan perbedaan antara method peek() dengan method pop() pada class Stack.

# 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.

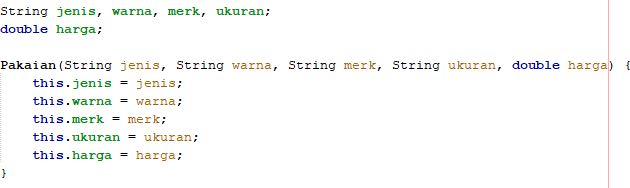
# Langkah-langkah Percobaan

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.

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.

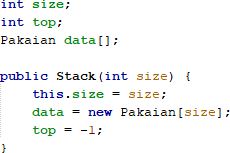


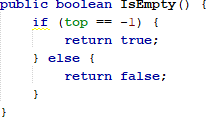
1. Setelah membuat class Pakaian, selanjutnya perlu dibuat class **Stack** yang berisi atribut dan method sesuai diagram Class Stack berikut ini:

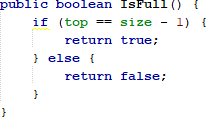
|  |
| --- |
| Stack |
| size: int top: int  data[]: Pakaian |
| Stack(size: int) IsEmpty(): boolean IsFull(): boolean  push(): void |

|  |
| --- |
| pop(): void peek(): void print(): void  clear(): void |

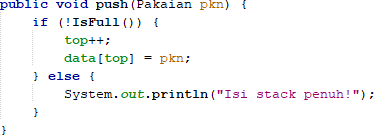
***Keterangan****: Tipe data pada variabell* ***data*** *menyesuaikan dengan data yang akan akan disimpan di dalam Stack. Pada praktikum ini, data yang akan disimpan merupakan array of object dari Pakaian, sehingga tipe data yang digunakan adalah* ***Pakaian***

1. Buat class baru dengan nama Stack. Kemudian tambahkan atribut dan konstruktor seperti gambar berikut ini.
2. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah stack kosong.

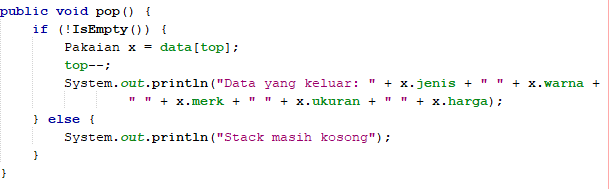


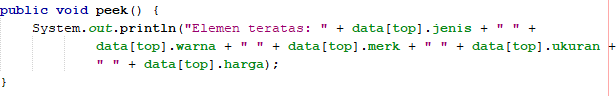
1. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah stack sudah terisi penuh.
2. Buat method **push** bertipe void untuk menambahkan isi elemen stack dengan parameter **pkn**

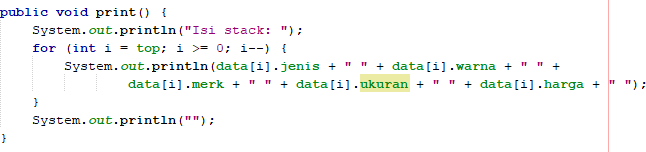
yang berupa object **Pakaian**



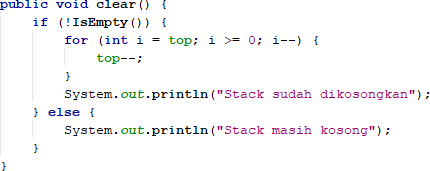
1. Buat method **Pop** bertipe void untuk mengeluarkan isi elemen stack. Karena satu elemen stack terdiri dari beberapa informasi (jenis, warna, merk, ukuran, dan harga), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut



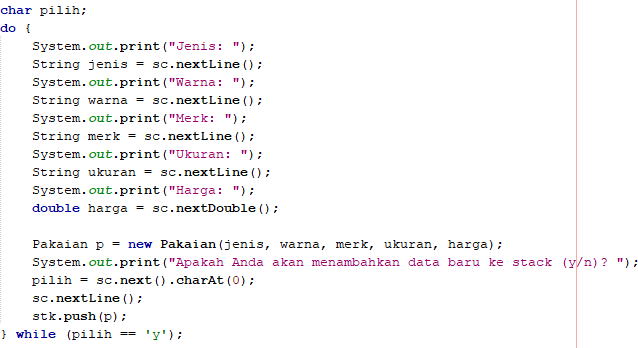
1. Buat method **peek** bertipe void untuk memeriksa elemen stack pada posisi paling atas.
2. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada stack.



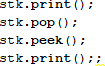
1. Buat method **clear** bertipe void untuk menghapus seluruh isi stack.



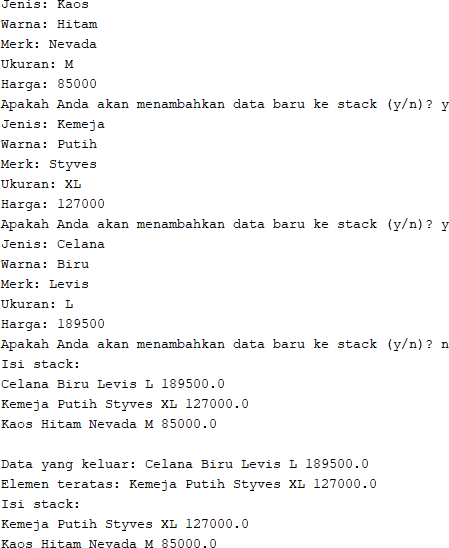
1. Selanjutnya, buat class baru dengan nama **StackMain**. Buat fungsi main, kemudian lakukan instansiasi objek dari class **Stack** dengan nama **stk** dan nilai parameternya adalah 5.
2. Deklarasikan Scanner dengan nama **sc**
3. Tambahkan kode berikut ini untuk menerima input data Pakaian, kemudian semua informasi tersebut dimasukkan ke dalam stack



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

1. Lakukan pemanggilan method print, method pop, dan method peek dengan urutan sebagai berikut.
2. Compile dan jalankan class **StackMain**, kemudian amati hasilnya.

# Verifikasi Hasil Percobaan



* + 1. **Pertanyaan**

1. Berapa banyak data pakaian yang dapat ditampung di dalam stack? Tunjukkan potongan kode program untuk mendukung jawaban Anda tersebut!
2. Perhatikan class **StackMain**, pada saat memanggil fungsi push, parameter yang dikirimkan adalah

**p**. Data apa yang tersimpan pada variabel **p** tersebut?



1. Apakah fungsi penggunaan **do-while** yang terdapat pada class **StackMain**?
2. 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!

# Praktikum 3

## Waktu percobaan : 30 menit

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

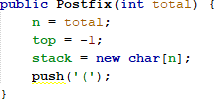
# Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

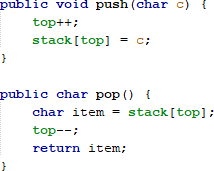
|  |
| --- |
| Postfix |
| n: int top: int  stack: char[] |
| Postfix(total: int) push(c: char): void pop(): void  IsOperand(c: char): boolean IsOperator(c: char): boolean derajat(c: char): int  konversi(Q: String): string |

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

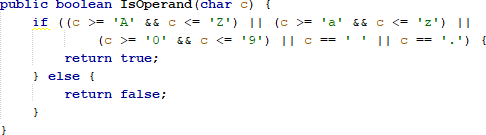
1. Buat class baru dengan nama **Postfix**. Tambahkan atribut **n**, **top**, dan **stack** sesuai diagram class Postfix tersebut.
2. Tambahkan pula konstruktor berparameter seperti gambar berikut ini.



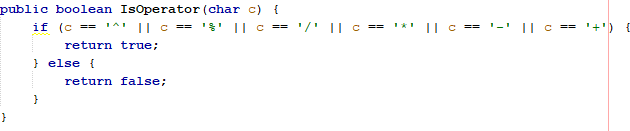
1. Buat method **push** dan **pop** bertipe void.

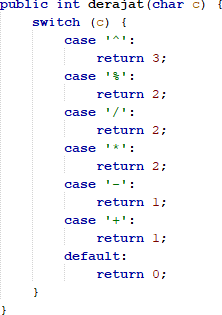


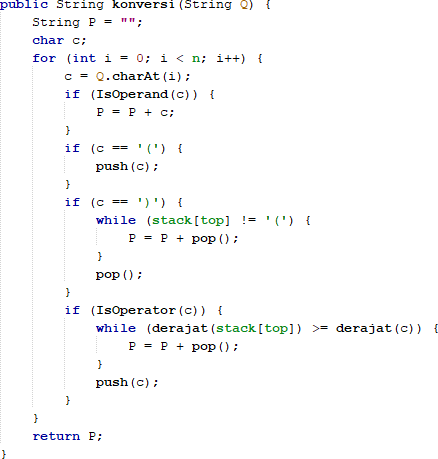
1. Buat method **IsOperand** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operand.

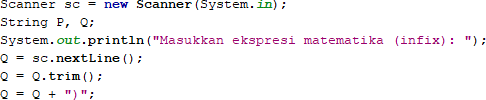


1. Buat method **IsOperator** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operator.



1. Buat method **derajat** yang mempunyai nilai kembalian integer untuk menentukan derajat operator.
2. Buat method konversi untuk melakukan konversi notasi infix menjadi notasi postfix dengan cara mengecek satu persatu elemen data pada **String Q** sebagai parameter masukan.



1. 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.

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

1. Buat variabel total untuk menghitung banyaknya karaketer pada variabel Q.



1. 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.



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

# Verifikasi Hasil Percobaan



* + 1. **Pertanyaan**

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



1. Jalankan kembali program tersebut, masukkan ekspresi **5\*4^(1+2)%3**. Tampilkan hasilnya!
2. Pada soal nomor 3, mengapa tanda kurung tidak ditampilkan pada hasil konversi? Jelaskan!

# Tugas

1. Perhatikan dan gunakan kembali kode program pada **Praktikum 2**. Tambahkan method **getMax** pada class **Stack** yang digunakan untuk mencari dan menampilkan data pakaian dengan harga tertinggi dari semua data pakaian yang tersimpan di dalam stack!
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 sesuai dengan jumlah struk yang akan ditukarkan dengan voucher. Informasi yang tersimpan pada struk belanja terdiri dari:

* + Nomor transaksi
  + Tanggal pembelian
  + Jumlah barang yang dibeli
  + Total harga bayar

Tampilkan informasi struk belanja yang masih tersimpan di dalam stack!