JOBSHEET VII

STACK

7.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

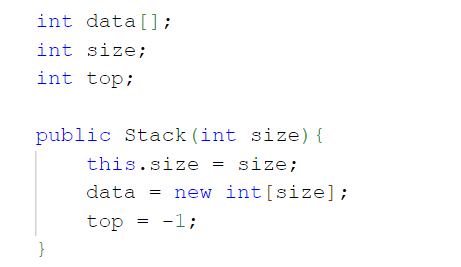
1. Mengenal struktur data Stack
2. Membuat dan mendeklarasikan struktur data Stack
3. Menerapkan algoritma Stack dengan menggunakan array
4. Praktikum 1

**Waktu percobaan: 30 menit**

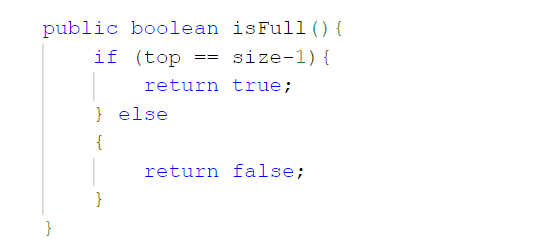
Pada percobaan ini, kita akan membuat program yang mengimplementasikan struktur data Stack dan operasi-operasi dasar pada struktur data Stack menggunakan array.

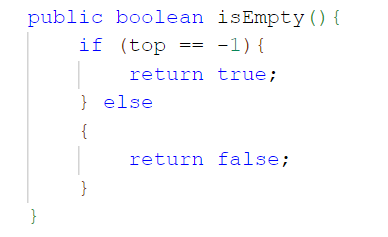
* + 1. Langkah-langkah Percobaan

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

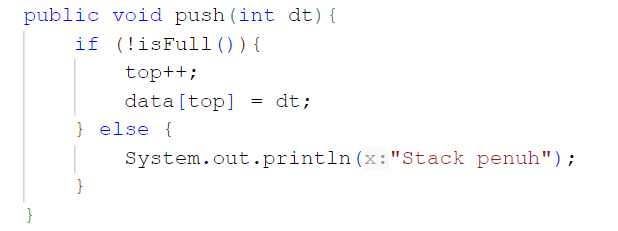


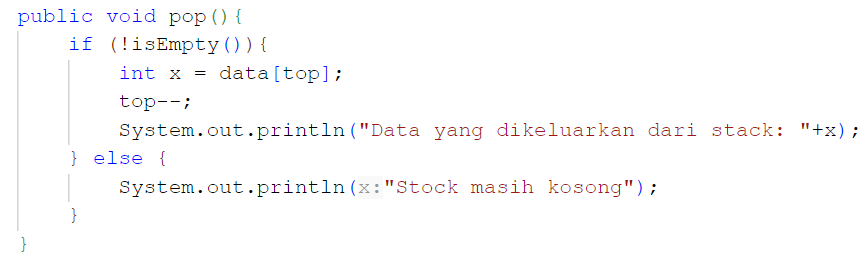
1. Lalu tambahkan method isFull() dan isEmpty() pada class Stack sebagai berikut:



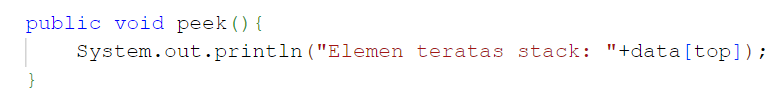


1. Tambahkan method push(int data) dan pop() sebagai berikut:

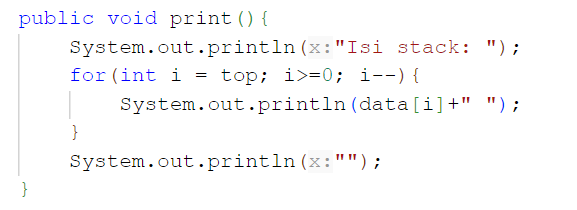


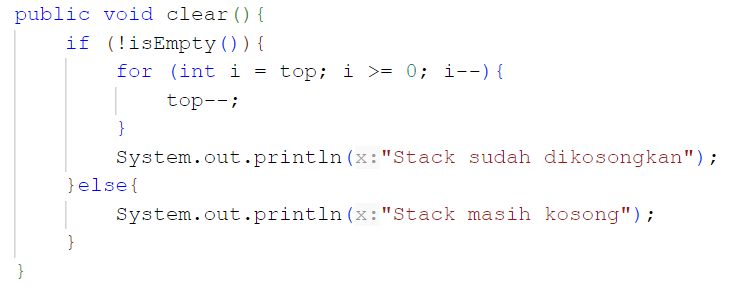


1. Tambahkan method peek() sebagai berikut:

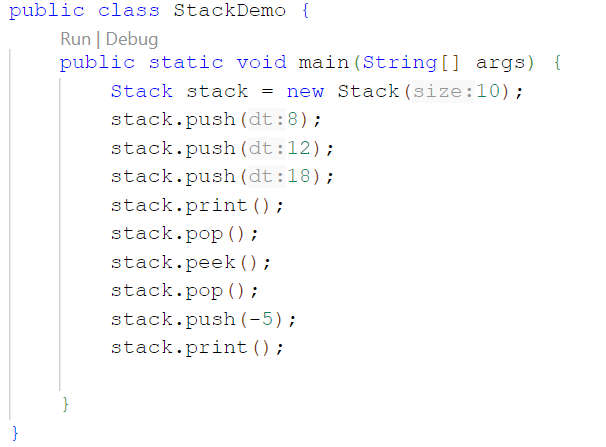


1. Tambahkan method print() dan clear() sebagai berikut:

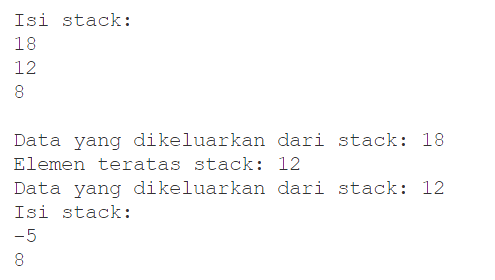


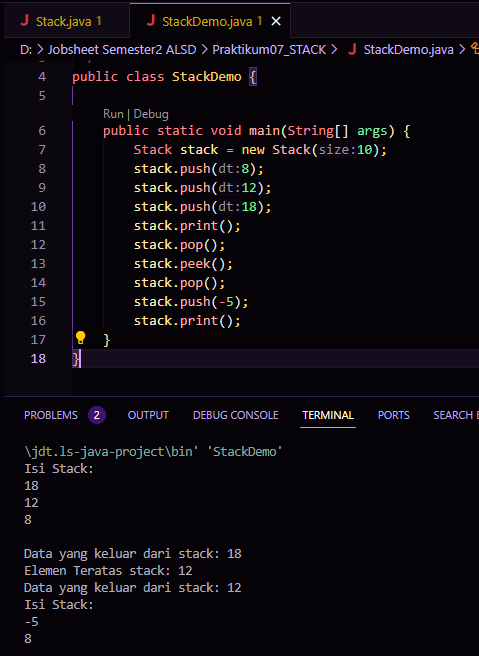


1. Buat file **StackDemo.java** untuk mengimplementasikan class StackDemo yang berisi fungsi main untuk membuat objek Stack dan mengoperasikan method-method pada class Stack.



1. Compile dan run class StackDemo.
   * 1. Verifikasi Hasil Percobaan





* + 1. Pertanyaan

1. Pada method pop(), mengapa diperlukan pemanggilan method isEmpty()? Apa yang terjadi jika tidak ada pemanggilan isEmpty()?

**Pada method pop(), pemanggilan isEmpty() digunakan untuk memeriksa apakah stack kosong sebelum mencoba mengeluarkan data. Jika tidak ada pemanggilan isEmpty(), ada risiko kesalahan indeks atau operasi yang tidak diinginkan jika mencoba mengakses data dari stack yang sebenarnya kosong.**

1. Jelaskan perbedaan antara method peek() dengan method pop() pada class Stack.

* peek(): Melihat elemen teratas dari stack tanpa menghapusnya dari stack.
* pop(): Menghapus dan mengembalikan elemen teratas dari stack.

1. Praktikum 2

**Waktu percobaan : 45 menit**

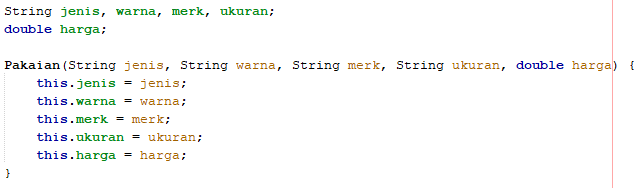
Pada percobaan ini, kita akan membuat program yang mengilustrasikan tumpukan pakaian yang disimpan ke dalam stack. Karena sebuah pakaian mempunyai beberapa informasi, maka implementasi Stack dilakukan dengan menggunakan array of object untuk mewakili setiap elemennya.

1. Langkah-langkah Percobaan
2. 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.

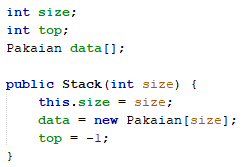


1. Setelah membuat class Pakaian, selanjutnya perlu dibuat class **Stack** yang berisi atribut dan method sesuai diagram Class Stack berikut ini:

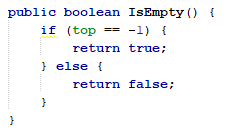
|  |
| --- |
| Stack |
| size: int  top: int  data[]: Pakaian |
| Stack(size: int)  IsEmpty(): boolean  IsFull(): boolean  push(): void  pop(): void  peek(): void  print(): void  clear(): void |

***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***

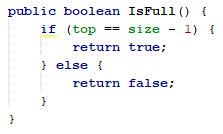
1. Buat class baru dengan nama Stack. Kemudian tambahkan atribut dan konstruktor seperti gambar berikut ini.



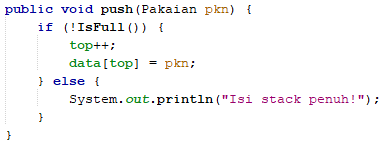
1. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah stack kosong.



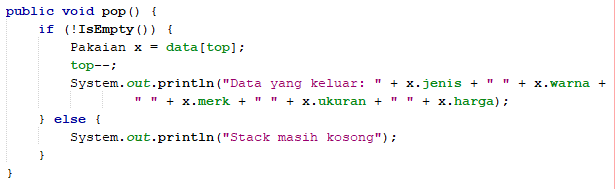
1. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah stack sudah terisi penuh.



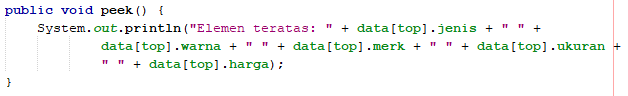
1. Buat method **push** bertipe void untuk menambahkan isi elemen stack dengan parameter **pkn** yang berupa object **Pakaian**



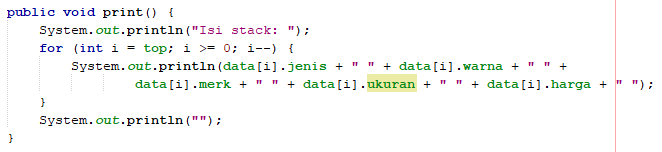
1. Buat method **Pop** bertipe void untuk mengeluarkan isi elemen stack. Karena satu elemen stack terdiri dari beberapa informasi (jenis, warna, merk, ukuran, dan harga), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut



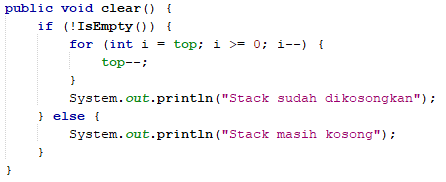
1. Buat method **peek** bertipe void untuk memeriksa elemen stack pada posisi paling atas.



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



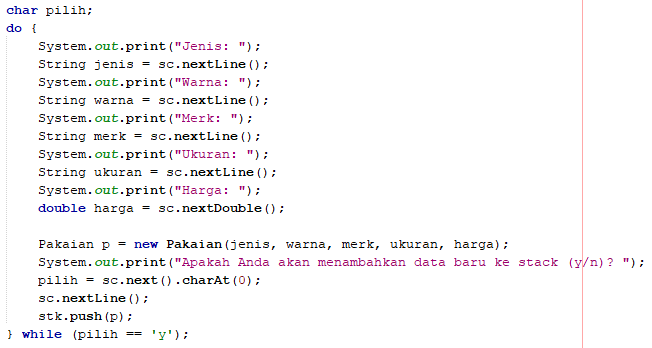
1. Buat method **clear** bertipe void untuk menghapus seluruh isi stack.



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.

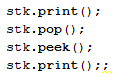


1. Deklarasikan Scanner dengan nama **sc**
2. Tambahkan kode berikut ini untuk menerima input data Pakaian, kemudian semua informasi tersebut dimasukkan ke dalam stack

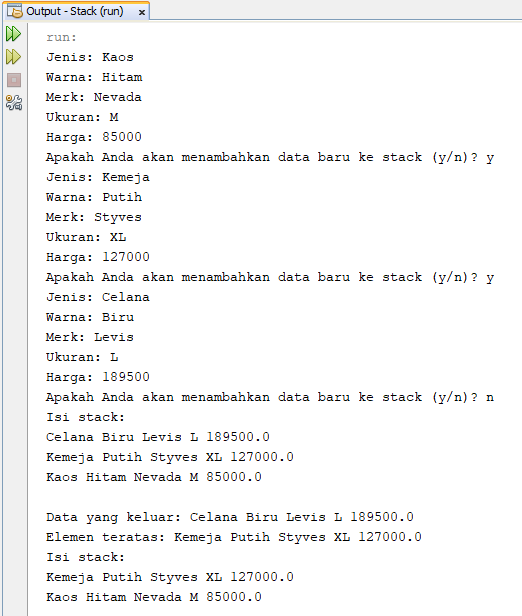


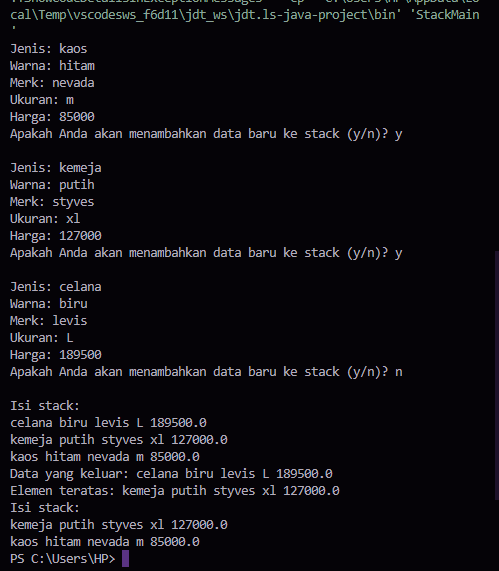
***Catatan****: sintaks sc.nextLine() sebelum sintaks st.push(p) digunakan untuk mengabaikan karakter new line*

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



1. Compile dan jalankan class **StackMain**, kemudian amati hasilnya.
2. Verifikasi Hasil Percobaan





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

**Stack11 stk = new Stack11(5);**

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



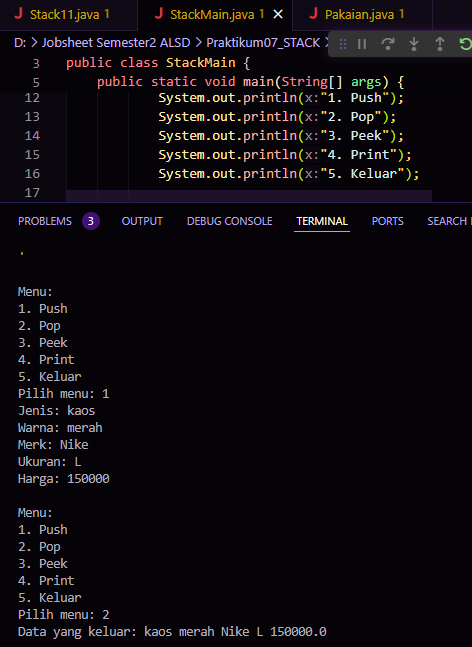
**Pada class StackMain, saat memanggil fungsi push, parameter p yang dikirimkan adalah objek Pakaian. Objek Pakaian tersebut berisi informasi tentang suatu pakaian, yaitu jenis, warna, merk, ukuran, dan harga.**

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

**Pada Kelas StackMain menggunakan do- while untuk memasukkan data pakaian dan mengulangi proses penambahan data tersebut ke tumpukan.**

**Perulangan ini berlanjut selama pengguna mengetik ``y'' untuk menambahkan data baru ke tumpukan. Ketika pengguna memasukkan input 'n', perulangan berhenti dan program melanjutkan ke bagian berikutnya setelah blok 'do- while'.**

1. 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!



1. Praktikum 3

**Waktu percobaan : 30 menit**

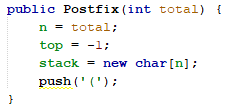
Pada percobaan ini, kita akan membuat program untuk melakukan konversi notasi infix menjadi notasi postfix.

1. Langkah-langkah Percobaan
2. Perhatikan Diagram Class berikut ini:

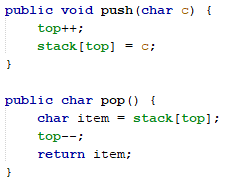
|  |
| --- |
| Postfix |
| n: int  top: int  stack: char[] |
| Postfix(total: int)  push(c: char): void  pop(): void  IsOperand(c: char): boolean  IsOperator(c: char): boolean  derajat(c: char): int  konversi(Q: String): string |

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

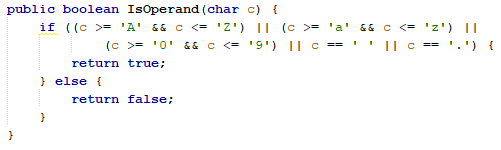
1. Buat class baru dengan nama **Postfix**. Tambahkan atribut **n**, **top**, dan **stack** sesuai diagram class Postfix tersebut.
2. Tambahkan pula konstruktor berparameter seperti gambar berikut ini.



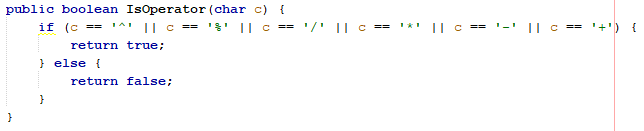
1. Buat method **push** dan **pop** bertipe void.



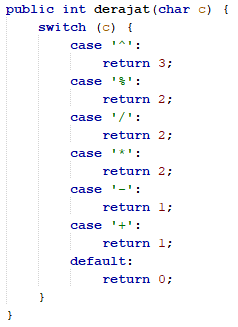
1. Buat method **IsOperand** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operand.



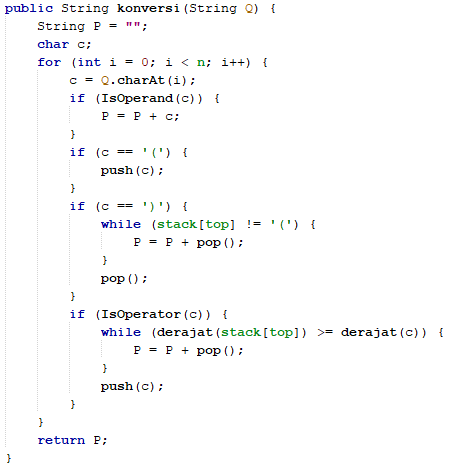
1. Buat method **IsOperator** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operator.



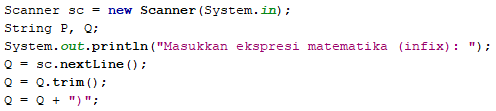
1. Buat method **derajat** yang mempunyai nilai kembalian integer untuk menentukan derajat operator.



1. 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.

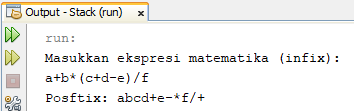
1. Buat variabel total untuk menghitung banyaknya karaketer pada variabel Q.

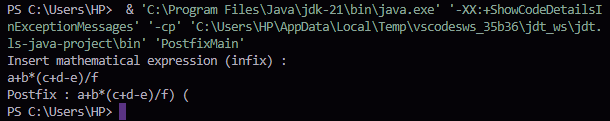


1. Lakukan instansiasi objek dengan nama **post** dan nilai parameternya adalah total. Kemudian panggil method **konversi** untuk melakukan konversi notasi infix Q menjadi notasi postfix P.



1. Compile dan jalankan class **PostfixMain** dan amati hasilnya.
2. Verifikasi Hasil Percobaan





1. Pertanyaan
2. Perhatikan class **Postfix**, jelaskan alur kerja method **derajat**!

**Langkah-langkah: Inisialisasi: Metode ini dimulai dengan menginisialisasi string kosong P untuk menyimpan ekspresi postfix.**

**Selain itu, karakter c dideklarasikan yang mengulangi setiap karakter dalam ekspresi infiks.**

**Memproses setiap karakter: Metode ini mengulang setiap karakter ekspresi infiks Q.**

**Untuk setiap karakter c: Jika c bukan operan (operator atau tanda kurung), maka segera ditambahkan ke ekspresi postfix P.**

**jika c adalah operan.**

**Metode ini memeriksa apakah itu adalah tanda kurung buka '('.**

**Jika ada, metode akan menambahkannya ke tumpukan untuk menandai awal subekspresi.**

**Jika c adalah tanda kurung tutup ")", maka metode akan mengeluarkan operator dari tumpukan dan menambahkannya ke ekspresi postfix P hingga tanda kurung buka yang cocok "(" ditemukan.**

**Kemudian tanda kurung buka dibuang.**

**Operator Preferensi Jika anak c kurang dari atau sama dengan, metode ini membandingkan prioritasnya dengan operator pilihan di bagian atas tumpukan, menghapus operator dari tumpukan, dan menambahkannya ke ekspresi postfix P.**

**Proses ini berlanjut hingga tumpukan kosong atau operator di puncak tumpukan mempunyai prioritas lebih rendah dari c.**

**Terakhir, operator c ditambahkan ke tumpukan.**

**Solusi: Setelah memproses semua karakter dalam ekspresi infiks Q, metode ini memeriksa apakah ada operator yang tersisa di tumpukan.**

**Jika ada, metode akan mencetaknya dan menambahkannya ke ekspresi postfix P.**

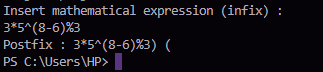
**Pengembalian: Metode ini mengembalikan ekspresi postfix P yang merupakan hasil konversi ekspresi infiks Q ke notasi postfix.**

1. Apa fungsi kode program berikut?



**Kode c = Q.charAt(i); berfungsi untuk mengambil karakter pada indeks ke-i dari string Q dan menyimpannya dalam variabel c.**

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



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

**dalam konversi ekspresi matematika dari infix ke postfix, tanda kurung tidak dimasukkan ke dalam ekspresi postfix karena urutan operasi ditentukan oleh posisi operand dan operator, tanpa memerlukan tanda kurung untuk menentukan prioritas.**

1. Tugas
2. Create a program with Stack implementation to insert a sentence and display the reversed version of the sentence as a result!

