**LAPORAN PRAKTIKUM PEMROGRAMAN LANJUT**

**Abstract Data Type**



Oleh:

Nama : Muhammad Rizky Fahlevi

NIM : 1957301063

Kelas : TI 2.A  
Dosen Pembimbing : Musta’inul Abdi, SST., M.Kom.

NIP : 19911030 20190310 1 5



**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER**

**TAHUN 2021**

LEMBAR PENGESAHAN

No. Tugas : 04/TIK/TI 2.A/P.Pemrograman Lanjut/2021

Judul : Abstract Data Type

Nama : Muhammad Rizky Fahlevi

NIM : 1957301063

Kelas : TI 2.A

Jurusan : Teknologi Informasi Dan Komputer

Prodi : Teknik Informatika

Tanggal Tugas : 26 Maret 2021

Tanggal penyerahan : 28 Maret 2021

Nilai :

|  |  |
| --- | --- |
|  | Buketrata, 28 Maret 2021 |
|  | Dosen Pembimbing, |
|  |  |
|  | Musta’inul Abdi, SST., M.Kom.  19911030 20190310 1 5 |
|  |  |

DAFTAR ISI

LEMBAR PENGESAHAN i

DAFTAR ISI ii

DAFTAR GAMBAR iii

**BAB I : PENDAHULUAN 1**

**1.1 Tujuan 1**

**1.2 Dasar Teori 1**

**1.2.1 Rekursif Vs. Iterasi 1**

**1.2.2 Factorials: Contoh 3**

**1.2.3 Print n in any Base: Contoh yang lain 3**

**BAB II : PEMBAHASAN 6**

**2.1 Factorial 6**

**2.1.1 Program Faktorial dengan Rekursif 6**

**2.2 Palindrom 8**

**2.2.1 Program Palindrom dengan Rekursif 9**

**BAB III : PENUTUP 12**

**3.1 Kesimpulan 12**

**DAFTAR PUSTAKA 13**

DAFTAR GAMBAR

Gambar 1.1 Contoh Factorial 2

**BAB I**

**PENDAHULUAN**

1. **Tujuan**

* Mengetahui perbedaan antara stacks dan queue
* Mengimplementasikan suatu implementasi sequensial dari stacks dan queues
* Mengimplementasikan suatu implementasi linked dari stacks and queues

1. **Dasar Teori**

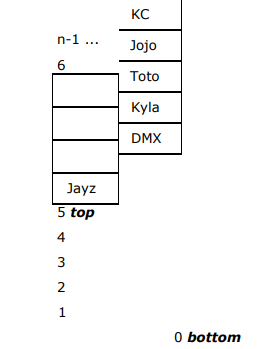
Abstract Data Type (ADT) adalah kumpulan dari elemen-elemen data yang disajikan dengan satu set operasi yang digambarkan pada elemen-elemen data tersebut. Stacks, queues dan binary trees adalah tiga contoh dari ADT. Dalam bab ini, Anda akan mempelajari tentang stacks dan queues.

**1.2.1 Stacks**

Stack adalah satu set atau urutan elemen data dimana manipulasi data dari elemen elemen hanya diperbolehkan pada tumpukan teratas dari stack. Hal ini merupakan perintah pengumpulan data secara linier yang disebut “last in, first out” (LIFO). Stacks berguna untuk bermacam-macam aplikasi seperti pattern recognition dan pengkonversian antar notasi infix, postfix dan prefix .

Dua operasi yang dihubungkan dengan stacks adalah operasi push dan pop. Push berarti memasukkan data kedalam stacks yang paling atas dimana pop sebagai penunjuk/pointer untuk memindahkan elemen ke atas stacks. Untuk memahami bagaimana cara kerja stacks, pikirkan bagaimana Anda dapat menambah atau memindakan sebuah data dari tumpukan data. Pikiran Anda akan memberitahu Anda untuk menambah atau memindahkan data hanya pada stack yang paling atas karena jika menggunakan cara lain, dapat menyebabkan tumpukan stack akan terjatuh.

Dibawah ini merupakan ilustrasi bagaimana tampilan dari stacks



Gambar 1.1 Illustrasi Stacks

Stack akan berarti penuh jika jangkauan cell teratas disimbolkan dengan n-1. Jika nilai teratas / top sama dengan -1, stack berarti kosong.

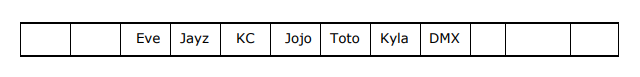
**1.2.2 Queues**

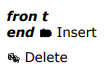
Queues adalah contoh lain dari ADT. Hal ini merupakan perintah pengumpulan data yang disebut “first-in, first-out”. Aplikasi ini meliputi jadwal pekerjaan dalam operating system, topological sorting dan graph traversal.

Enqueue dan dequeue merupakan operasi yang dihubungkan dengan queues. Enqueu menunjuk pada memasukkan data pada akhir queue dimana dequeue berarti memindahkan elemen dari queue tersebut. Untuk mengingat bagaimana queue bekerja, ingatlah arti khusus dari queue yaitu baris. Berikut ini bagaimana cara kerja queue. Siapa yang akan mendapatkan kesempatan pertama untuk bertemu bintang idolanya dari mereka yang sedang menunggu dalam sebuah barisan? Seharusnya orang pertama yang berada pada barisan tersebut. Orang ini mendapat kesempatan pertama untuk meninggalkan barisan. Hubungkan hal tersebut dengan bagaimana queue bekerja.

Berikut ini merupakan ilustrasi dari bagaimana tampilan dari queue

0 1 2 3 4 5 6 7 8 9 ... n-1





Gambar 1.2 Illustrasi Queue

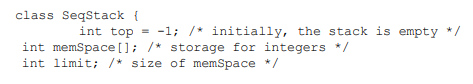
Queue akan kosong jika nilai end kurang dari front. Sementara itu, akan penuh jika end sama dengan n-1.

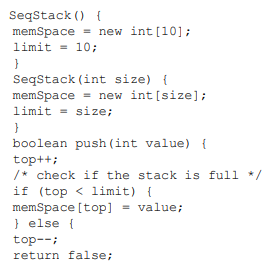
**1.2.3 Sequential and Linked Representation**

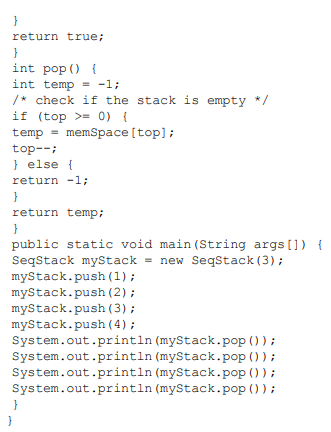
ADTs biasanya dapat diwakilkan menggunakan sequential dan linked representation. Hal ini memudahkan untuk membuat sequential representation dengan menggunakan array. Bagaimanapun juga, masalah dengan menggunakan array adalah pembatasan size, yang membuatnya tidak fleksibel. Dengan menggunakan array, sering terjadi kekurangan atau kelebihan space memory. Mempertimbangkan hal tersebut, Anda harus membuat sebuah array dan mendeklarasikannya agar mampu menyimpan 50 elemen. Jika user hanya memasukkan 5 elemen, maka 45 space pada memory akan sia-sia. Disisi lain, jika user ingin memasukkan 51 elemen, space yang telah disediakan didalam array tidak akan cukup.

Dibandingkan dengan sequential representation, linked representation lebih sedikit rumit tetapi lebih fleksibel. Linked representation menyesuaikan memory yang dibutuhkan oleh user. Penjelasan lebih lanjut pada linked representation akan didiskusikan pada bab berikutnya.

**1.2.4 Sequential Representation dari Integer Stack**





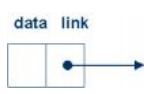


**1.2.5 Linked Lists**

Sebelum mengimplementasikan linked representation dari stacks, pertama mari kita pelejari bagaimana membuat linked representation. Dalam hal ini, kita akan menggunakan linked lists.

Linked list merupakan struktur dinamis yang berlawanan dengan array, yang merupakan struktur statis. Hal ini berarti linked list dapat tumbuh dan berkurang dalam size yang bergantung pada kebutuhan user. Linked list digambarkan sebagai kumpulan dari nodes, Yang masing-masing berisi data dan link atau pointer ke node berikutnya didalam list.

Gambar dibawah ini menunjukkan tampilan dari node.



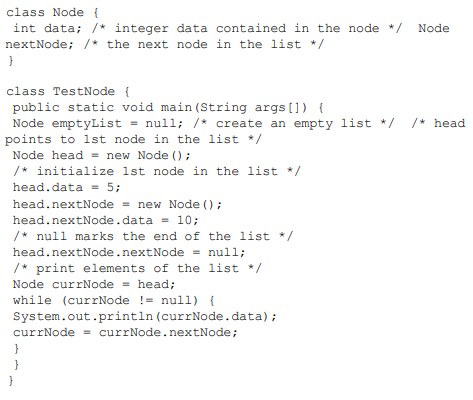
Gambar 1.3 Sebuah Node

Berikut ini merupakan contoh dari non-empty linked list dengan 3 node.



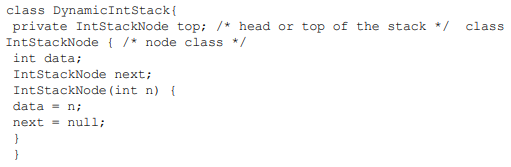
Gambar 1.4 Non-empty linked list dengan tiga node

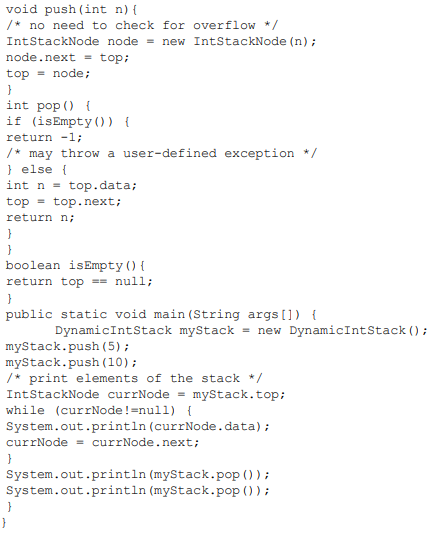
Berikut ini bagaimana class node diimplementasikan. Class ini dapat digunakan untuk membuat linked list.



**1.2.6 Linked Representation dari Integer Stack**

Sekarang Anda telah mempelajari tentang linked list. Maka Anda telah siap untuk menerapkan apa yang telah Anda pelajari untuk implementasi linked representation dari stack.







Gambar 1.5 Implementasi linked dari stack

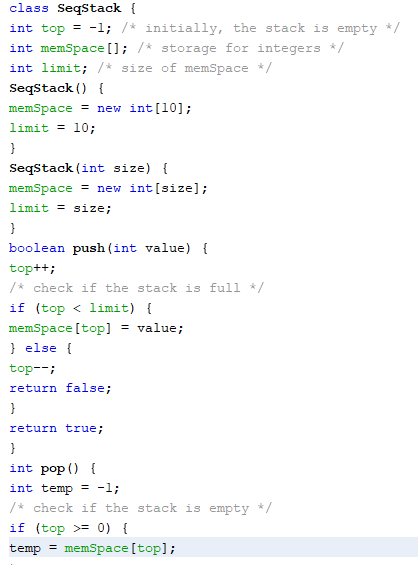
**BAB II**

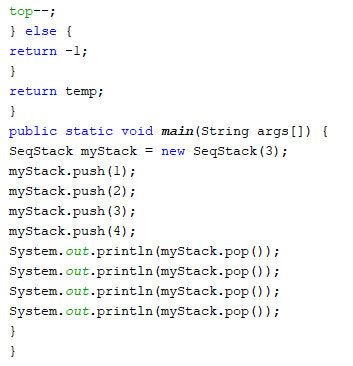
**PEMBAHASAN**

1. **Sequential Representation dari Integer Stack**

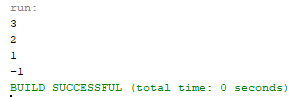
**2.1.1 Program**

**Code:**





**Hasil:**

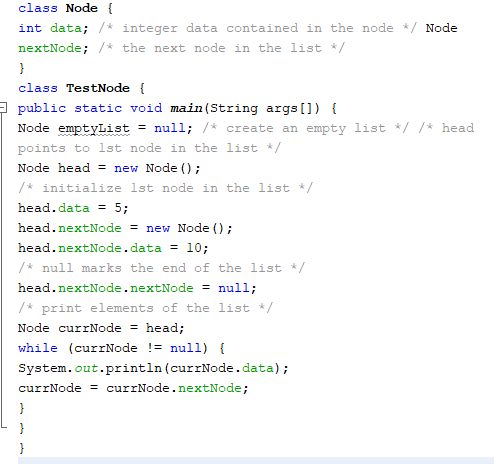


**Analisa:**

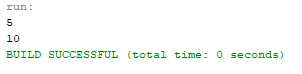
1. **Linked List**

**2.2.1 Program**

**Code:**



**Hasil:**

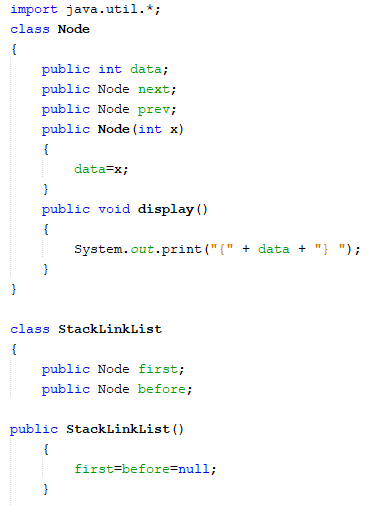


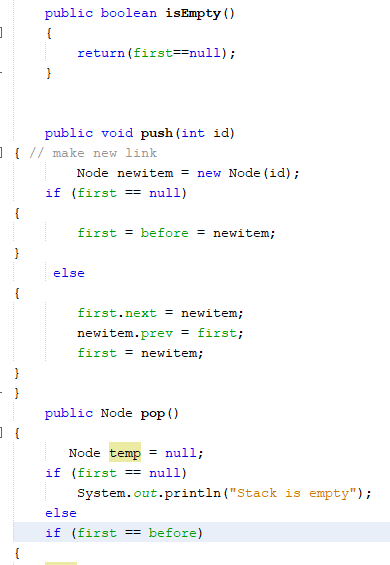
**Analisa:**

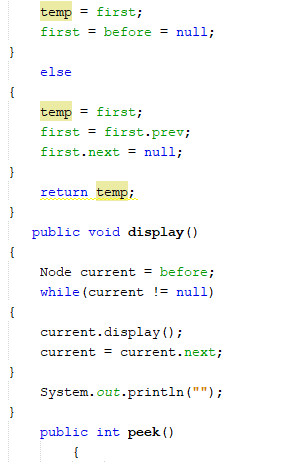
1. **Linked List Stack**

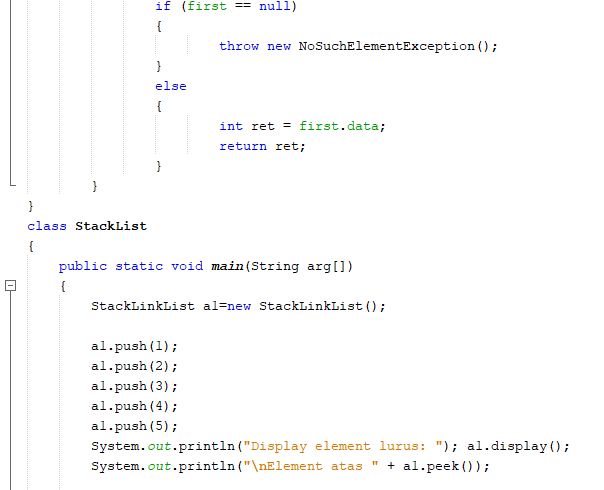
**2.3.1 Program Linked List Stack**

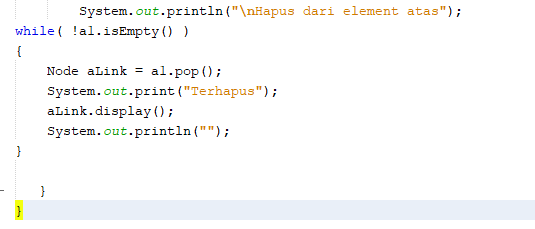
**Code:**



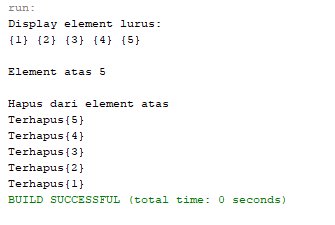








**Hasil:**

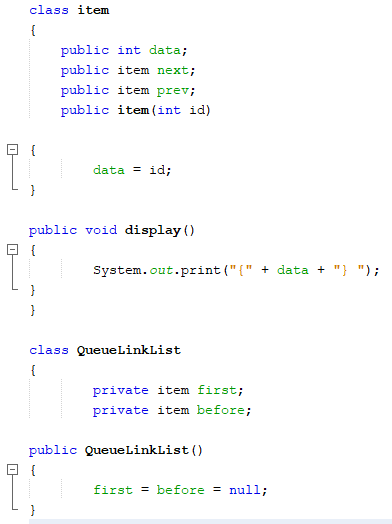


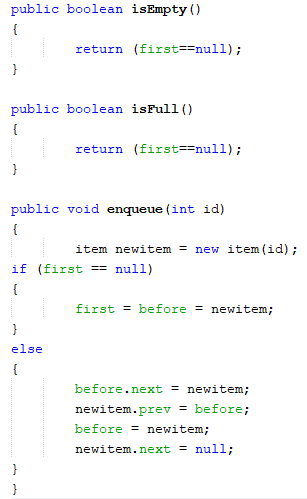
**Analisa:**

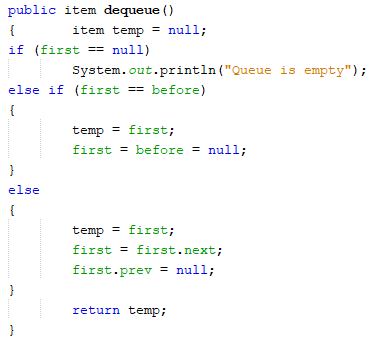
1. **Linked List Queue**

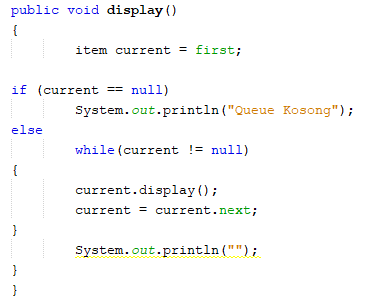
**2.4.1 Program Linked List Queue**

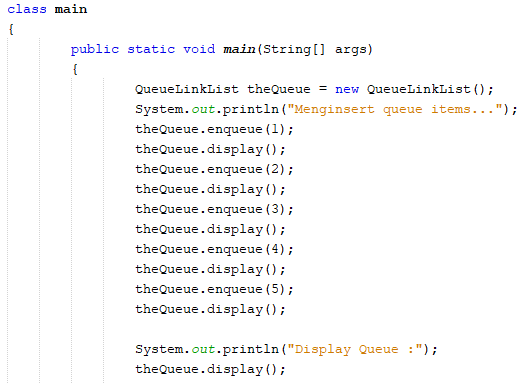
**Code:**

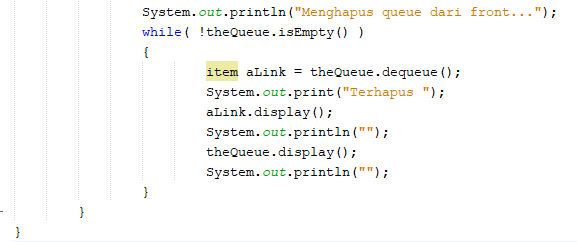




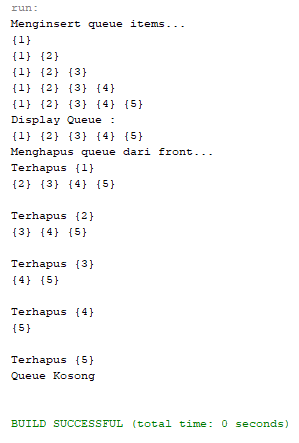








**Hasil:**



**Analisa:**

**BAB III**

**PENUTUP**

1. **Kesimpulan**

**DAFTAR PUSTAKA**

1. Thamura, Frans. JENI Intro2 Bab03 Teknik Pemrograman Lanjut. Diakses pada 7 Maret 2021 melalui <https://docplayer.info/30937572-Bab-3-modul-ini-mengenalkan-suatu-teknik-pemrograman-yang-lebih-tinggi-dalam-bagian-ini-anda-akan-mempelajari-rekursif-dan-tipe-data-abstrak.html>