PENERAPAN STACK DAN QUEUE PADA ARRAY DAN LINKED LIST DALAM JAVA

Research · June 2020

CITATIONS
0

READS
19,424

5 authors, including:

Johnson Sihombing
STMIK GANESHA BANDUNG
3 PUBLICATIONS **12** CITATIONS

SEE PROFILE

PENERAPAN STACK DAN QUEUE PADA ARRAY DAN LINKED LIST DALAM JAVA

Johnson Sihombing
Program Studi Manajemen Informatika DIV
Politeknik Piksi Ganesha, Jl. Jend. Gatot Soebroto No. 301 Bandung
Email : john97.sihombing@gmail.com

ABSTRAK

Pada pemrograman struktur data, *stack* dan *queue* adalah dua jenis struktur data non primitif bertipe data abstrak yang digunakan untuk menyimpan elemen data baik di *array* maupun di *linked list* yang sebenarnya didasarkan pada beberapa kejadian di kehidupan nyata. *Stack* menggunakan metode LIFO (*last in first out*) untuk mengakses dan menambahkan elemen data sedangkan *queue* menggunakan metode FIFO (*First in first out*) untuk mengakses dan menamhkan elemen data. Penelitian menggunakan tipe data integer untuk dimasukkan ke dalam array dan linked list dan dilakukan pengujian terhadapnya dengan membuat program aplikasi menggunakan bahasa pemrograman Java, yaitu suatu bahasa pemrograman dengan konsep object oriented. Dengan menggunakan versi JDK (*Java Develompent Kit*), maka *class* Java untuk mengoperasikan *linked list*, *queue* dan *stack* sudah tersedia. Untuk mengakses data dari *class- class* tersebut dapat dikonversi menjadi *array* atau *iterator* (list satu arah). Java sudah menyedikan *LinkedList* yang digunakan sebagai *queue*, *Stack* yang digunakan sebagai *stack*, dan *ArrayDeque* sebagai *deque* (gabungan antara queue dan *stack*).

Kata Kunci : stack, queue, array, linked list, Java

ABSTRACT

In data structure programming, stack and queue are two types of non-primitive data structures of abstract data type that are used to store data elements both in arrays and in linked lists that are actually based on several events in real life. Stack uses the LIFO (last in first out) method to access and add data elements while the queue uses the FIFO (First in first out) method to access and add data elements. The study uses integer data types to be included in arrays and linked lists and tested against it by creating application programs using the Java programming language, which is a programming language with an object oriented concept. By using the JDK (Java Develompent Kit) version, the Java class to operate the linked list, queue and stack is available. To access data from these classes can be converted into an array or iterator (one-way list). Java already provides a LinkedList that is used as a queue, a Stack that is used as a stack, and ArrayDeque as a deque (a combination of queue and stack).

***Keywords** : stack, queue, array, linked list, Java*

PENDAHULUAN

1. Latar Belakang Masalah

Java merupakan salah satu bahasa pemrograman yang terkenal dengan basis *object oriented* (berorientasi objek). Dengan metode tersebut, maka dikenal pula struktur data bentukan yang sering disebut ADT (*Abstract Data Type*) sebagai pengganti struktur data primitif yang telah di identifikasi dan di deklarasikan dalam bahasa pemrograman tertentu (misalnya char, integer, string, dan lain-lain). Penelitian yang dilakukan penulis adalah ADT Statis (ADT yang menggunakan lokasi memori secara tetap) pada *stack* dan *queue* pada *array* dan *linked list*.

Stack bersifat LIFO (*Last In First Out*) dan objek yang terakhir masuk ke dalam *stack* akan menjadi benda pertama yang dikeluarkan dari *stack* itu. Sedangkan *queue* bersifat FIFO (First In First Out) dengaa cara kerja yang

berbanding terbalik dengan *stack*. Kedua struktur data tersebut dapat disajikan dalam *array* dan *linked list*. Khusus untuk *stack*, penyajian dalam bemtuk *array* masih kurang tepat. *Array* bisa digunakan kalau elemen *stack* tidak melebihi batas maksimum. Tipe data yang bisa digunakan adalah *record*.

TINJAUAN PUSTAKA

1. Array

A. Pengertian *Array*

Array merupakan suatu kumpulan data terstruktur yang berupa sejumlah data sejenis (memiliki jenis data yang sama) yang jumlahnya tetap dan diberi suatu nama tertentu.

Sebuah *array* dapat dibayangkan sebagai sekumpulan kotak yang menyimpan sekumpulan elemen bertipe sama secara berurutan (*sequential*) di dalam memori komputer. *Array* juga dapat digambarkan sebagai elemen-elemen yang disusun secara vertikal sehingga dinamai tabel. Setiap elemen *array* data diacu melalui indeksnya.

Karena elemen disimpan secara berturutan, indeks *array* haruslah suatu tipe yang juga mempunyai keterurutan (memiliki suksesor dan ada predesesor), misalnya tipe *integer* atau karakter. Jika indeksnya adalah *integer* maka keterurutan indeks sesuai dengan urutan *integer*. Jika indeks array adalah karakter maka keterurutan indeks sesuai dengan urutan karakter. Tiap elemen *array* langsung diakses dengan menspesifikasikan nama *array*. *Array* dapat berupa array 1 dimensi, 2 dimensi, dan bahkan n-dimensi.

B. Deklarasi Array

Untuk dapat mengakses array, maka terlebih dahulu ditentukan deklarasi terhadap suatu array yang berbentuk seperti berikut ini :

tipe_data
nama_var_array[ukuran];

Keterangan :

tipe_data : menyatakan jenis tipe data dari elemen yang tersimpan di *array* (int, char, float, dan lain-lain).

nama_var_array : menyatakan nama variabel yang dipakai.

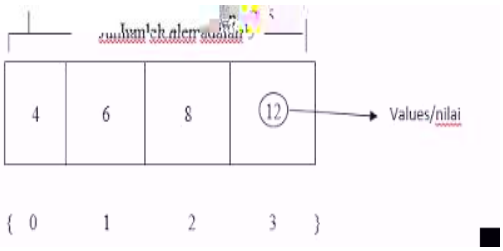
ukuran : menunjukkan jumlah maksimal elemen *array*.

Contoh : Int nilai[6].

C. *Array* Berdimensi Satu

Merupakan sekumpulan item data yang disusun secara baik menjadi suatu rangkaian dan diacu atau ditunjuk oleh satu *identifier*. Contoh : Nilai = (56 42 89 65 48). .

Item data individual dalam *array* bisa ditunjuk secara terpisah dengan menyatakan posisinya dalam array itu. Misalnya Nilai(1) menunjuk ke 56, Nilai(2) menunjuk ke 42, dan seterusnya. Bilangan yang ditulis dalam tanda kurung menandakan posisi item individual dalam *array* (disebut juga *subscript* / indeks). Variabel bisa digunakan sebagai *subscript*, misalnya Nilai(i). Jika i = 2 maka menunjuk ke Nilai(2) yaitu 42. Jika i = 4 maka menunjuk ke Nilai(4) yaitu 65. Sedangkan item data individual dalam suatu *array* sering disebut elemen**.**



Gambar 1. Visualisasi array berdimensi satu

D. *Array* Berdimensi Dua

Array tersebut sering di analogikan dan digambarkan sebagai bentuk matriks, dengan indeks pertama berfungsi sebagai baris dan indeks kedua digunakan untuk kolom. Beberapa kolom dan baris pada array dua dimensi memiliki elemen yang bertipe sama. Pada a*rray* dua dimensi terdapat dua jumlah elemen yang terdapat didalam kurung siku dan keduanya boleh tidak sama. Bentuk umum deklarasi *array* dua dimensi :

Tipe_Data Nama_Variabel [index-1][index-2]

	0	1	2	3
0	2	4	10	12
1	5	-7	9	20

Gambar 2. Visualisasi array berdimensi dua

Pada gambar diatas, kolom adalah {2 dan 5} {4 dan -7} {10 dan 9} {12 dan 20 } dan baris adalah { 2,4 ,10, 12 } dan { 5, -7, 9, 20}.

E. Linked List

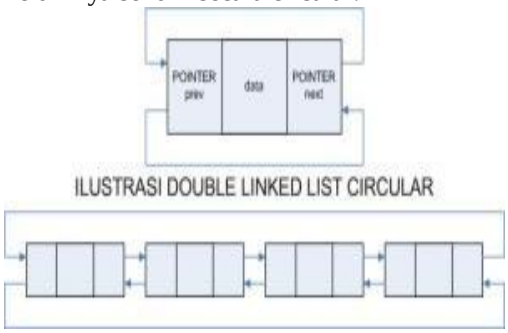
1. Pengertian Linked List

Linked list adalah suatu bentuk struktur data yang berupa sekumpulan elemen data yang bertipe sama dimana tiap elemen saling berkaita atau dihubungkan dengan elemen lain melalui suatu *pointer*. *Pointer* itu sendiri adalah alamat elemen data yang tersimpan di memori. Penggunaan *pointer* untuk mengacu elemen berakibat elemen-elemen bersebelahan secara logik walau tidak bersebelahan secara fisik di memori**.**

Linked list terdiri dari *node-node*

5. Doubly Circular Linked List

Merupakan sutau *double linked list* yang *pointer next* dan *pointer prev*-nya menunjuk ke dirinya sendiri secara circular.

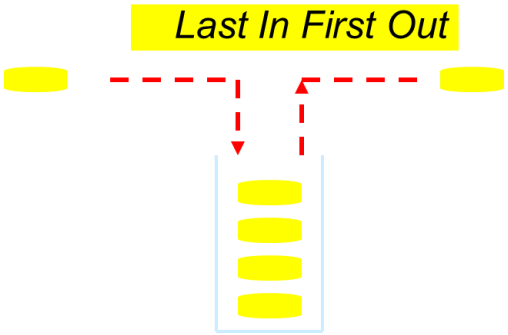


Gambar 7. Ilustrasi doubly circular linked list

2. Stack

A. Pengertian *Stack*

Stack atau tumpukan adalah suatu stuktur data yang penting dalam pemrograman dengan metode pemrosesan yang bersifat LIFO (*Last In First Out*) dimana objek/benda yang terakhir masuk ke dalam *stack* akan menjadi benda pertama yang dikeluarkan dari *stack*. Dengan model demikian, maka hanya bagian paling atas saja dari *stack* (*TOP*) yang bisa di akses. Salah satu kelebihan *stack* adalah bahwa struktur data tersebut dapat di implementasikan baik pada *array* maupun pada *linked list*.



Gambar 7. Visualisasi operasi stack

Adapuu operasi-operasi/fungsi yang dapat dilakukan pada *stack* adalah sebagai berikut :

1) **Push :** digunakan untuk menambah item pada stack pada tumpukan paling atas
2) **Pop :** digunakan untuk mengambil item pada stack pada tumpukan paling atas
3) **Clear :** digunakan untuk mengosongkan stack
4) **IsEmpty :** fungsi yang digunakan untuk mengecek apakah stack sudah kosong
5) **IsFull :** fungsi yang digunakan untuk mengecek apakah stack sudah penuh

B. Deklarasi Struktur Data *Stack*

Agar pemrograman berjalan baik, *stack* harus dideklarasikan terlebih dahulu dengan bentuk seperti berikut :

```
#define maxsize 100
// mendefinisikan maks ukuran data
// dlm stack
typedef struct {
int     top;                // indeks TOP
char    items [ maxsize ] // array
} stack;
// nama tipe data baru yg dibuat
// adalah stack
```

Untuk notasi deklarasi fungsi-fungsi stack dapat dilihat di bawah ini :

1) Fungsi Initialize :

```
void initialize ( stack *s)
 // operasi initialize dg parameter
 // s bertipe pointer stack
 {   s -> top = -1;
                // top = -1      stack dlm
 kondisi empty
 }
```

2) Fungsi Pop :

```
void pop ( stack *s, char *x )
 {
 if (s->top > maxsize) // stack is full
                printf("\nERROR: the stack
 is full!");
     else {
                s->top = s->top + 1;
                s->items [ s->top ] = x;
                printf("\nPUSH
 SUCCEED");
     }
 }
```

3) Fungsi Push :

```
 void push ( stack *s, char x )
 {
     if (s->top > maxsize) // stack is full
                printf("\nERROR: the stack
 is full!");
     else {
                s->top = s->top + 1;
                s->items [ s->top ] = x;
 printf("\nPUSH SUCCEED"); } }
```

4) Fungsi Show :

```
 void show( stack *s )
```

```
{
    printf("\nISI STACK :\n");
    for(int i=s->top; i>=0; i--)
            printf("\t%c\n", s->items[i]);
    printf("\n");
}
```

Seperti yang telah dijelaskan sebelumnya bahswa struktur data *stack* dapat di impelementasikan pada *array* dan linked list. Berikut adalah table perbedaan antara *stack* yang ada di *array* dan *linked list* :

Tabel 1. Perbedaan Stack Linked List versus Stack Array

Stack Dengan Linked List	Stack Dengan Array
operasi : create()	
void create { top := nil ; }	void create { top := 0; }
operasi : empty()	

function empty :
boolean; {
empty := false ;
if (top = nil)
empty := true ; }
	function empty :
	boolean; {
	empty := false ;
	if (top = 0) empty :=
	true ; “

operasi : full()	

tidak ada istilah full pada stack. program tidak menentukan jumlah elemen stack yang mungkin ada. kecuali dibatasi oleh pembuat program dan jumlah memory yang tersedia. tempat akan sesuai dengan banyaknya elemen yang mengisi stack.	function full :
	boolean; {
	full := false ;
	if (top = max) full :=
	true ;
	}

operasi : push()	

void push
(elemen :
typedata) ;
now:point ;
now(now) ;
now^.isi :=
elemen ;
if (empty)) {
now^.next := nil ;
else
now^.next := top
;
top := now ;
}
	void push (elemen :
	typedata) ;
	if (not full) {
	top := top + 1 ;
	stack [top] := elemen ;
	}
	;

operasi : pop()	

void pop (elemen
: typedata) ;{
now : point ;
if (not empty) {
elemen :=
now^.isi ;
now := top ;
top := top^.next ;
dispose(now) ;
}
	void pop (elemen :
	typedata) ;
	if (not empty) {
	elemen := stack [top] ;
	top := top – 1 ;
	}

operasi : clear	

void clear ;
trash : typedata ;
while (not empty)
do pop(trash) ;
}
	void clear ;
	top := 0 ;
	}

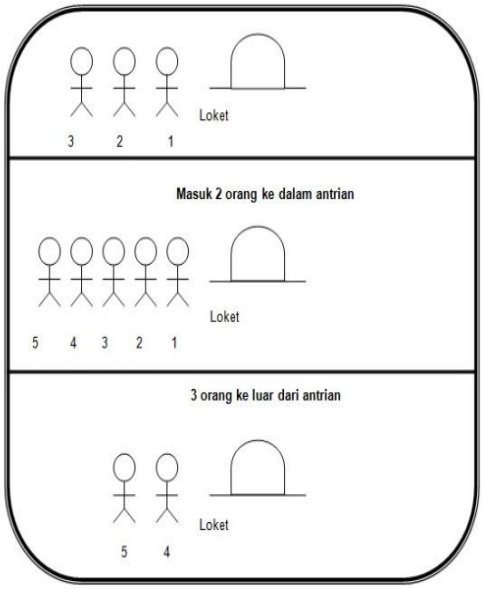
Pada tabel diatas, dapat dilihat bahwa pada *linked list* tidak dikenal istilah *full*. Hal ini berkaitan dengan penggunaan alokasi memori pada *linked list* yang lebih dinamis jika dibandingkan dengan *array*, sehingga pemborosan memory dapat dihindari. Program tidak menentukan jumlah elemen *stack* yang mungkin ada. Kecuali dibatasi oleh pembuat program dan jumlah memory yang tersedia. Tempat akan sesuai dengan banyaknya elemen yang mengisi stack

3. Queue

A. Pengertian *Queue*

Kebalikan dari *stack*, *queue* (antrian) adalah suatu jenis struktur data yang dapat diproses dengan sifat FIFO (*First In First Out*), dimana elemen yang pertama kali masuk

ke antrian akan keluar pertama kalinya. Ada dua jenis operasi yang bias dilakukan di antrian : enqueue (memasukkan elemen baru ke dalam elemen) dan dequeue (adalah mengeluarkan satu elemen dari suatu antrian). Antrian dapat dibuat dengan menggunakan: *Liniear Array* dan *Circular Array*.



Gambar 8. Ilustrasi proses queue

B. Deklarasi Struktur Data *Queue*

Queue dapat dideklarasikan dengan bentuk seperti berikut :

```
define maxsize 100
typdef struct {
int       jumlah;  //jumlah data
int       depan;   //ujung depan
int       belakang;          //ujung belakang
char      data [ maxsize ]; //array isi queue
}  queue;
```

Sedangkan bentuk fungsi-fungsi queue dapat dilihat berikut ini :

1) Fungsi Initialize :

```
void initialize ( queue *q )
{
q -> jumlah = 0;
q -> depan = 0;
q -> belakang = 0;
}
```

2) Fungsi Is_Empty, yang antara lain digunakan untuk :
   a. Untuk memeriksa apakah Antrian sudah penuh atau belum
   b. Dengan cara memeriksa nilai *Tail*, jika *Tail* = -1 maka *empty*
   c. Tidak perlu memeriksa *Head*, karena *Head* adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yan:g tidak akan berubah

Berikut adalah deklarasi fungsi.Is_Empty :

```
int Is_Empty (queue  *q)  {
if (q      jumlah == 0)
     return (1);
else
     return (0);
}
```

3) Fungsi Is_Full, berfungsi untuk :
   a. Untuk mengecek apakah Antrian sudah penuh atau belum
   b. Dengan cara mengecek nilai Tail, jika Tail >= MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh

Deklarasi fungsi.Is_Empty :

```
int Is_Full (queue  *q)  {
if (q      jumlah == MAX)
     return (1);
else
     return (0);
}
```

4) Fungsi Enqueue , dengan tujuan :
   a. Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang
   b. Penambahan elemen selalu menggerakan variabel *Tail* dengan cara *increment counter Tail.*

Deklarasi fungsi.Enqueue :

```
void enqueue ( char X, queue *q )
{
if ( Is_Full(q) )
     printf("\nERROR:  queue  sudah
     penuh\");
else
     {
     q->data[q->belakang] = X;
     q->belakang          =          (q-
>belakang+1)%maxsize;
++(q->count);
     }
}
```

5) Fungsi Dequeue :
   a. Digunakan untuk menghapus elemen terdepan/pertama dari Antrian

b. Dengan cara mengurangi *counter Tail* dan menggeser semua elemen antrian kedepan.
c. Penggeseran dilakukan dengan menggunakan *looping*

Deklarasi fungsi.Dequeue :

```
void dequeue ( queue *q, char X )
{
if ( Is_Empty(q) )
    printf("\nERROR: queue sudah kosong\");
else
    {
    X = q->data[q->depan];
    q->depan = (q->depan+1)%maxsize;
                --(q->count);
    }       }
```

6) Fungsi Tampil , yang berfungsi untuk menampilkan nilai-nilai elemen dalam Antrian dam menggunakan looping dari *head* samapai *tail.*

Deklarasi fungsi Tampil :

```
void show_queue(queue *q)
{
    printf("\nIsi Queue:\n");
    for(int i=q->depan; i<q->belakang; i++)
    printf("%c ", q->data[i]);
    printf("\n"); }
```

Note: *script* ini khusus untuk normal *queue*

Tabel di bawah ini akan menampilkan Perbedaan *Queue* yang menggunakan *Linked List* versus *Queue – Array* :

Tabel 2. Perbedaan Implementasi Queue Linked List versus Queue Array

Queue Dengan Linked List	Queue Dengan Array
Memiliki kompleksitas pada pengimplemantasian	Implementasi sederhana
Pengalokasian memori dinamis	Ukuran memori harus ditentukan ketika sebuah objek queue dideklarasikan
Menggunaka 2 buah pionter, *qFront* dan *qRear*, untuk menandai posisi depan dan belakang dari queue	Pemborosan tempat (memori) ketika menggunakan jumlah data yang lebih sedikit dari alokasi memory
	Tidak dapat menambahkan data melebihi maksimal ukuran array yang telah dideklarasikan

PENELITIAN TERDAHULU

Tabel 2. Penelitian Terkait

No	Penulis	Tahun	Publikasi	Judul	Hasil
1	Arif Aliyanto	2011	Seminar Nasional Aplikasi Teknologi Informasi 2011 (SNATI 2011) ISSN: 1907-5022 Yogyakarta, 17-18 Juni 2011 ISSN: 1907-5022	Sistem Pembelajaran Algoritma Stack Dan Queue Dengan Pendekatan Problem Based Learning Untuk Mendukung Pembelajaran Struktur Data	Konsep pembelajaran dapat memudahkan dan perceptan mahasiswa dalam memahami konsep algoritma *stack* dan *queue*
Kesimpulan:: Sistem pembelajaran algoritma ***Stack*** dan ***Queue*** yang dibuat telah memenuhi ketiga aspek penilaian multimedia pembelajaran yaitu aspek rekayasa perangkat lunak (RPL), aspek desain pembelajaran dan aspek komunikasi visual					
2	M Zaky Hadi, Taufik Djatna, dan Sugiarto2	2017	Jurnal Teknologi Industri Pertanian 27 (3):298-309 (2017) ISSN: 0216-3160 EISSN: 2252-3901	Pemodelan Antrian Sistem Pengambilan Pesanan Produk Pada Gudang Minuman Ringan Dengan Sistem Rak *Drive-In*	Hasil studi kasus telah sesuai dengan hasil observasi kondisi gudang industri dan model ini digunakan untuk menganalisis jumlah operator yang harus ditugaskan untuk meningk

					atkan kinerja gudang
Kesimpulan : pembuatan model sitem mampu menentukan kinerja gudang : rendahnya kinerja pada proses *racking*, transportasi produk menuju area pembongkaran, dan proses pembongkaran produk untuk sejumlah operator yang saat ini ditugaskan serta adanya operator menganggur pada proses di gudang/ Model juga dapat menentukan status produk berdasarkan *hold*, siap rilis dan kadaluarsa di gudang					
3	Rachmat Selamet	2016	Media Informatika Vol. 15 No.3 (2016)	Implementasi Struktur Data List, Queue, Dan Stack Dalam Java	LinkedList pada Java digunakan sebagai queue, Stack yang sebagai stack, dan ArrayDeque sebagai deque. PriorityQueue digunakan untuk priorotas kasus dengani kondisi khusus
Kesimpulan : 1. Sstruktur data menjadi lebih mudah di implementasikan karena telah tersedia kelas-kelas yang dibutuhkan di dalam nya 2. Sruktur list, queue dan stack dapat diakses dengan mengkonversi menjadi bentuk array 3. Struktur Queue paling banyak digunakan dalam pengolahan data karena pengaksesan data dimulai dari awal 4. Pemrosesan Stack dimulai dari data terakhir. Kasus yang umum menggunakan stack adalah pembuatan proses undo/redo pada program.					

PERENCANAAN PENGJUIAN

A. Java Netbeans IDE

NetBeans adalah suatu *Integrated Development Environment* (IDE) berbasiskan Java dari Sun Microsystems yang berjalan di atas *Swing*. *Swing* merupakan sebuah teknologi Java untuk pengembangan aplikasi *desktop* yang dapat bejalan di berbagai macam *platforms* seperti Windows, Linux, Mac OS X and Solaris.

Netbeans merupakan *software development* yang *Open Source*, dengan kata lain software ini di bawah pengembangan bersama, bebas biaya. NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra. Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.

Suatu IDE adalah lingkup pemrograman yang diintegrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan pembangun *Graphic User Interface* (GUI), suatu *text* atau kode *editor*, suatu *compiler* atau *interpreter* dan suatu *debugger*. The NetBeans IDE adalah sebuah lingkungan pengembangan, yang merupakan sebuah *tools* untuk pemrogram menulis, mengkompilasi, mencari kesalahan dan menyebarkan program. Netbeans IDE ditulis dalam Java namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas *Netbeans IDE*. *Netbeans IDE* adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan. *NetBeans IDE* mendukung pengembangan semua tipe aplikasi Java (*J2SE*, *web*, *EJB*, dan aplikasi *mobile*). Fitur lainnya adalah sistem proyek berbasis *Ant*, *kontrol versi*, dan *refactoring*

B. Perangkat Keras Pengujian

Spesifikasi perangkat keras (*hardware*) yang digunakan penulis untuk membuat program aplikasi adalah sebagai berikut :

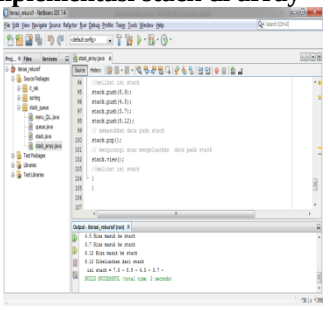
OS : Windows 7/
Processor : Intel® ATOM™) CPU N435 @ 1.60 GHz
RAM : 2.00 GB
System type : 32-bit Operating System

HASIL DAN PEMBAHASAN

A. Pengujian Struktur Data Stack

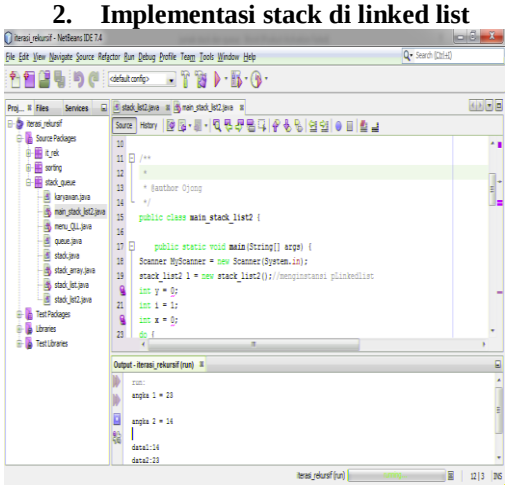
Sesuai dengan judul jurnal yang penulis teliti, maka proses pengujian terhadap *stack* dapat dilakukan pada *array* dan *linked list* dengan membuat program aplikasi menggunakan bahasa pemrograman Java.

1. Implementasi stack di array



Gambar 9. Koding dan output stack di array

2. Implementasi stack di linked list

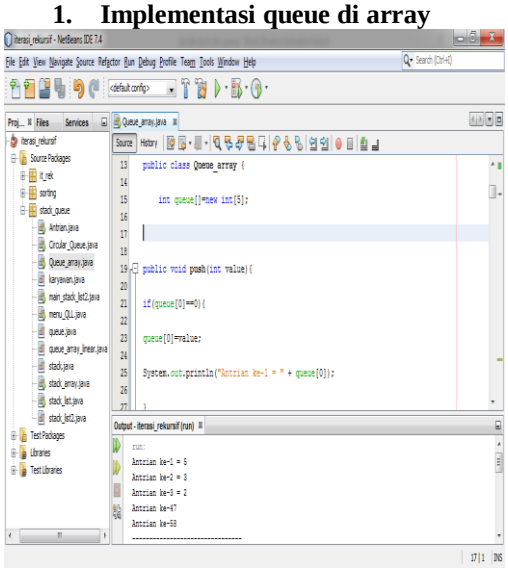


Gambar 10. Koding dan output stack di linked list

B. Pengujian Struktur Data Queue

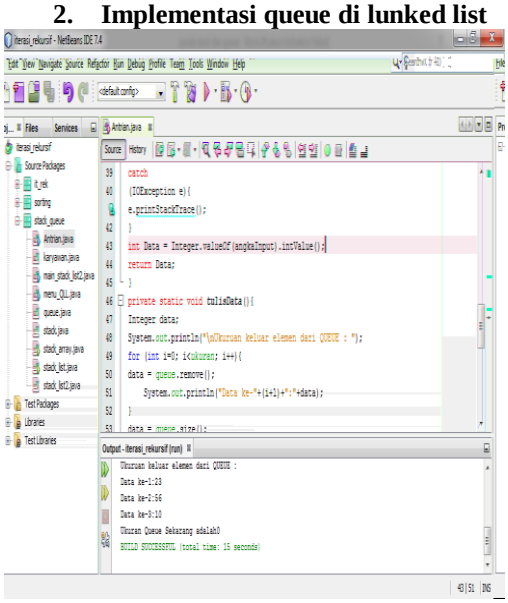
Sama seperti pada stack, pengujian terhadap *queue* juga dapat dilakukan pada *array* dan *linked list* .

1. Implementasi queue di array



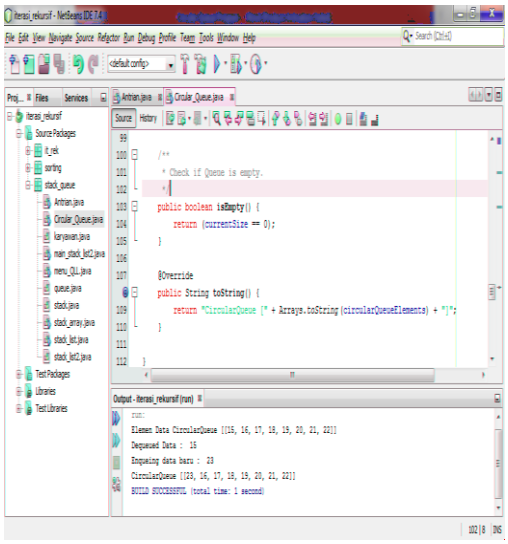
Gambar 11. Koding dan output queue di array

2. Implementasi queue di lunked list



Gambar 12. Koding dan output queue di linked list

3. Implementasi queue di circular linked list



Gambar 13. Koding dan output quue di circular linked list

PENUTUP

Kesimpulan

Tahap pengujian yang telah dilaksanakan dengan implementasi terhadap strukur data *queue* dan *stack* baik yang digunakan di *array* maupun *linked list*, penulis mendapat beberapa baik kesimpulan, antarap lain :

1. Implementasi *stack* dan *queu*e di *array* memiilki waktu akses rata-rata lebih cepat daripada waktu akses yang di implementsikan di *linked list*.
2. Kasus khusus jika jumlah maksimal elemen data nya tidak diketahui pada *stack* dan *queue* yang berukuran besar, maka disarankan untuk menggunakan *linked list.*
3. Selain itu, khusus untuk perangkat keras yang memiliki memori terbatas, *linked list* memiliki performa yang lebih bagus.

.

Saran

Adapun saran-saran yang dapat ditulis adalah :

1. Pada tiap kasus *stack* dan *queue*, buatlah algoritma dan deklarasi struktur data nya yang lebih efektif, lebih cepat waktu proses dan tidak membutuhkan memori banyak.
2. Menggunakan *software* yang lebih canggih dalam pengimplementasian di masa mendatang.

DAFTAR PUSTAKA

Prihartono. (2016), *Pedoman Praktis Penulisan Paper Jurnal Ilmiah*, Bandung : Piksi Ganesha Press.

Aliyantoi, Arif. (2011). *Sistem Pembelajaran Algoritma Stack Dan Queue Dengan Pendekatan Based Learning Untuk Mendukung Pembelajaran Struktur Data*. Seminar Aplikasi Teknologi Informasi 2011 (SNATI 2011). ISSN : 1907-5022.

Hadi, M. Zaky., Taufik Djatna dan Sugiarto. (2017). *Pemodelan Antrian Sistem engambilan Pesanan Produk Pada Gudang Minuman Ringan Dengan Sistem Rak Drive-In*. Jurnal Teknologi Industri Pertanian 27 (3):298-309 (2017).

ISSN: 0216-3160 EISSN: 2252-3901.

Nitesh, Manbir Singh dan Rahul Yadav. (2014). Research Paper On Stack And Queue.

© 2014 IJIRT | Volume 1 Issue 7 | ISSN: 2349-6002. Deparment Of Electronics And Communication Dronacharya College Of Engineering Farruknagar, Khetawas, Gurgaon.

Selamet, Rachmat. (2016). *Implementasi Sttruktur Data List, Queue Dan Stack Dalam Java*. Media Informatika Vol. 15 No.3 (2016)

Mandasari, Rizza Indah Mega dan Cahyana. (2016). *Modul Praktikum Implementasi Struktur Data (Hal. 42 – 56)*. Universitas Telkom Bandung

Yatini Indra B, Nasution Erliansyah. 2006. *Algoritma Dan Struktur Data*. Yogyakarta : Penerbit Graha Ilmu.

Munir, Rinaldi. (2003). *Matematika Diskrit*. Departemen Teknik Informatika, Institut Teknologi Bandung.

Ficher, M.M., dan Getis,A. (2010)). *Handbook of Applied Spatial Analysis Software Tools, Methods and Applications*. Berlin Heidelberg : Springer-Verlag

Barakbah, Ali Ridho., Tita Karlita dan Ahmad Syauqi Ahsan. (2013). *Buku Ajar Logika Dan*

*Algoritm*a. Program Studi Teknik InformatikaDepartemen Teknik Informatika dan KomputerPoliteknik Elektronika Negeri Surabaya

[OC] Oracle Technology. (2013). *Netbeans IDE 7.3*. Tersedia pada https://netbeans.org/ community/releases/73/. [11 Juli 2013]

TH, Cormen., Leiserson CE, Rivest R dan Stein C. (2009). *Introduction to Algorithms, 3ed*. Massachusetts: MIT Pers.