

LAPORAN
Struktur Data Linear
Praktikum 13 : Stack & Queue Object



NAMA : Johanes Yogtan Wicaksono Raharja
NIM : 215314105

Program Studi INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA

1. Linked List

```
public class LinkedList {
    private ListNode head;
    private int size;

    public LinkedList() {
        head = new ListNode();
        head.next = head;
        head.prev = head;
        size = 0;
    }

    public void addBefore(Object x, ListNode Bantu) {
        ListNode baru = new ListNode(x);
        baru.next = Bantu;
        baru.prev = Bantu.prev;
        Bantu.prev.next = baru;
        Bantu.prev = baru;
        size++;
    }

    public Object remove(ListNode Bantu) {
        Bantu.prev.next = Bantu.next;
        Bantu.next.prev = Bantu.prev;
        size--;
        return Bantu.getElemen();
    }

    public void addFirst(Object x) {
        addBefore(x, head.next);
    }

    public void addLast(Object x) {
        addBefore(x, head);
    }

    public Object removeFirst() {
        return remove(head.next);
    }

    public Object removeLast() {
        return remove(head.prev);
    }

    public boolean isEmpty() {
        if (head == head.next) {
            return true;
        } else {
            return false;
        }
    }

    public ListNode search(Object Cari){
        ListNode Bantu = head.next;

        while(Bantu != head){
            if(Cari == Bantu.elemen){
                return Bantu;
            }
            Bantu = Bantu.next;
        }
        return null;
    }

    public int size(){
        return size;
    }

    @Override
    public String toString() {
        String temp = " ";

        ListNode Bantu = head.next;
        while (Bantu != head) {
            temp = temp + Bantu.elemen + " ";
            Bantu = Bantu.next;
        }
        return temp;
    }
}
```

2. List Node

```
public class ListNode {
    Object elemen;
    ListNode next;
    ListNode prev;

    ListNode() {}
    ListNode(Object Elemen) {
        this.elemen = Elemen;
    }
    ListNode(Object Elemen, ListNode Next, ListNode Prev) {
        this.elemen = Elemen;
        this.next = Next;
        this.prev = Prev;
    }
    public void setElemen(Object elemen) {
        this.elemen = elemen;
    }
    public Object getElemen() {
        return elemen;
    }
    public void setNext(ListNode next) {
        this.next = next;
    }
    public ListNode getNext() {
        return next;
    }
    public void setPrev(ListNode prev) {
        this.prev = prev;
    }
    public ListNode getPrev() {
        return prev;
    }
}
```

3. Stack Object

- Screenshot Listing Program
- Stack Statis

```
import java.util.NoSuchElementException;

public class StackStatis {
    Object[] elemen;
    int front = -1;
    int size = 0;

    StackStatis() {}
    StackStatis(int ukuran) {
        elemen = new Object[ukuran];
    }

    public boolean push(Object x) {
        if (size < elemen.length) {
            front++;
            elemen[front] = x;
            size++;
            return true;
        } else {
            return false;
        }
    }

    public Object pop() {
        if (!isEmpty()) {
            Object hapus = elemen[front];
            front--;
            size--;
            return hapus;
        } else {
            throw new NoSuchElementException();
        }
    }

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        if (front == -1) {
            return true;
        } else {
            return false;
        }
    }

    public void Cetak() {
        for (int i = 0; i < size; i++) {
            System.out.println(elemen[i]);
        }
    }

    public static void main(String[] args) {
        StackDinamis tump = new StackDinamis();
        tump.push(10);
        tump.push(5);
        tump.push(7);
        while (!tump.isEmpty()) {
            System.out.println(tump.pop());
        }
    }
}
```

- Stack Dinamis

```
public class StackDinamis {
    LinkedList tumpukan;

    StackDinamis() {
        tumpukan = new LinkedList();
    }

    public void push(Object elemen) {
        tumpukan.addFirst(elemen);
    }

    public Object pop() {
        return tumpukan.removeFirst();
    }

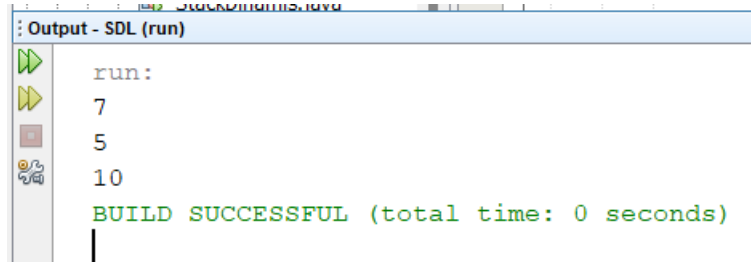
    public int size() {
        return tumpukan.size();
    }

    public boolean isEmpty() {
        return tumpukan.isEmpty();
    }

    public void Cetak() {
        System.out.println(tumpukan.toString());
    }

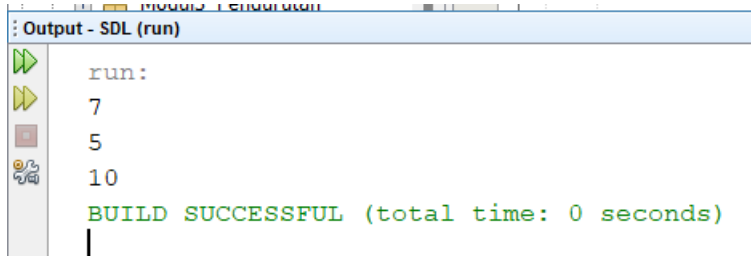
    public static void main(String[] args) {
        StackDinamis tump = new StackDinamis();
        tump.push(10);
        tump.push(5);
        tump.push(7);
        while (!tump.isEmpty()) {
            System.out.println(tump.pop());
        }
    }
}
```

- Main Statis



```
run:
7
5
10
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Main Dinamis



```
run:
7
5
10
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Penjelasan
- Pada Stack Static dideklarasikan array elemen bertipe objek yang akan diisi sesuai input yang dimasukan yaitu variabel `int front`, `size`. Kemudian terdapat 2 constructor untuk constructor pertama tidak memiliki parameter dan yang kedua memiliki parameter ukuran dengan tipe data `int`. Lalu, terdapat method `push ()` bertipe objek yang didalamnya terdapat aturan `if` untuk mengecek `size` apakah kurang dari panjang array elemen. Jika tidak maka akan mengembalikan nilai `false`, jika iya maka akan menambah nilai `size`. Kemudian ada method `pop ()` yang dimana program ini mengecek jika array kosong atau tidak. Jika kosong maka akan mengembalikan exception `NoSuchElementException()`, sedangkan jika tidak kosong maka akan mengurangi nilai `size` dan mengembalikan data temp. Kemudian ada method `isEmpty()` dengan tipe data `boolean` yang dimana berfungsi untuk mengecek apakah `front` sedang kosong atau tidak. Jika kosong dia akan mengembalikan nilai `true`, sedangkan jika tidak maka akan mengembalikan nilai `false`. Kemudian ada method `cetak()` yang digunakan untuk mencetak nilai yang dimasukan.
- Pada Stack Dinamis, method `push ()` memanggil method `addFirst()` milik `LinkedList` yang dimana, method `pop ()` juga memanggil method `remove ()` dan method `size()` kemudian memanggil juga method `size ()` milik `LinkedList` dimana pada method `sizeEmpty ()` memanggil method `isEmpty()` dan yang terakhir ada metode cetak.

4. Queue Object

- Screenshot Listing Program
- QueueStatic

```
import java.util.NoSuchElementException;

public class QueueStatic {

    Object[] elemen;
    int front;
    int rear;
    int size;

    public QueueStatic() {}

    public QueueStatic(int ukuran) {
        elemen = new Object[ukuran];
    }

    boolean enqueue(Object data) {
        if (size < elemen.length) {
            elemen[rear] = data;
            if (rear == elemen.length - 1) {
                rear = 0;
            } else {
                rear++;
            }
            size++;
            return true;
        }
        return false;
    }

    Object dequeue() {
        if (elemen.length != 0) {
            Object hapus = elemen[front];
            if (front == elemen.length - 1) {
                front = 0;
            } else {
                front++;
            }
            size--;
            return hapus;
        }
        throw new NoSuchElementException();
    }

    public int Size() {
        return size;
    }

    public boolean isEmpty() {
        if (size == 0) {
            return true;
        } else {
            return false;
        }
    }

    public void cetak() {
        for (int i = front; i < rear; i++) {
            System.out.println(" " + elemen[i]);
        }
    }

    public static void main(String[] args) {

        QueueStatic antrian = new QueueStatic(5);
        antrian.enqueue(14);
        antrian.enqueue(25);
        antrian.enqueue(58);
        while (!antrian.isEmpty()) {
            System.out.println(antrian.dequeue());
        }
    }
}
```

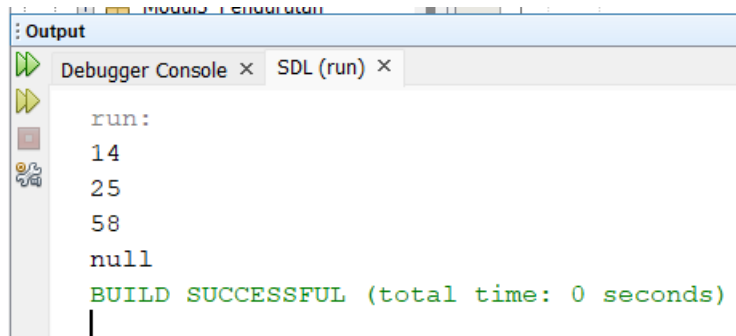
- Queue Dinamis

```
public class QueueDinamis {  
    LinkedList antrian;  
  
    QueueDinamis() {  
        antrian = new LinkedList();  
    }  
  
    boolean enqueue(Object data) {  
        antrian.addLast(data);  
        return false;  
    }  
  
    Object dequeue() {  
        return antrian.removeFirst();  
    }  
  
    public int Size() {  
        return antrian.size();  
    }  
  
    public boolean isEmpty() {  
        if (antrian.size() == -1) {  
            return true;  
        }  
        return false;  
    }  
  
    public void cetak() {  
        System.out.println(antrian.toString());  
    }  
  
    public static void main(String[] args) {  
        QueueDinamis antrian = new QueueDinamis();  
        antrian.enqueue(14);  
        antrian.enqueue(25);  
        antrian.enqueue(58);  
        while (!antrian.isEmpty()) {  
            System.out.println(antrian.dequeue());  
        }  
    }  
}
```

- Main Statis

```
Output - SDL (run)  
run:  
14  
25  
58  
BUILD SUCCESSFUL (total time: 0 seconds)  
|
```

- Main Dinamis



```
run:
14
25
58
null
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Penjelasan

Pada Queue Static dideklarasikan array elemen bertipe objek yang akan diisi sesuai input yang dimasukan yaitu variabel `int front`, `rear`, `size`. Kemudian terdapat 2 constructor untuk constructor pertama tidak memiliki parameter dan yang kedua memiliki parameter ukuran dengan tipe data `int`. Lalu, terdapat method `enqueue()` yang didalamnya terdapat aturan `if` untuk mengecek `size` apakah kurang dari panjang array elemen. Jika tidak maka akan mengembalikan nilai `false`, jika iya maka data inputan akan dimasukkan ke posisi `rear` pada elemen, jika posisi `rear` berada di paling akhir maka `rear` akan mengulang ke posisi 0, jika tidak maka `rear` akan bertambah, setelah penambahan data maka `size` bertambah lalu mengembalikan nilai `true`. Kemudian ada method `dequeue()` yang dimana program ini mengecek jika array kosong atau tidak. Jika kosong maka akan mengembalikan exception `NoSuchElementException()`, sedangkan jika tidak kosong maka program mendeklarasikan variabel hapus yang diletakkan di posisi di mana `front` berada, jika `front` sama dengan panjang array - 1 maka `front` akan mengulang ke posisi 0, jika tidak maka nilai `front` bertambah, setelah penghapusan, `size` berkurang lalu mengembalikan nilai dari hapus. Kemudian ada method `isEmpty()` dengan tipe data boolean yang dimana berfungsi untuk mengecek apakah antrian sedang kosong atau tidak. Jika kosong dia akan mengembalikan nilai `true`, sedangkan jika tidak maka akan mengembalikan nilai `false`. Kemudian ada method `cetak()` yang digunakan untuk mencetak nilai yang dimasukan.

Pada Queue Dinamis, method `enqueue()` memanggil method `addLast()` milik `LinkedList` yang dimana, method `addLast()` juga memanggil method `addBefore()` dan method `dequeue()` kemudian memanggil juga method `removeFirst()` milik `LinkedList` dimana pada method `removeFirst()` memanggil method `remove()`.