PRAKTIKUM 23
QUEUE

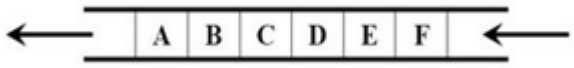
A. TUJUAN

Mahasiswadiharapkanmampu :

1. Memahamikonsepdanimplementasidari Queue
2. Memahamioperasi-operasipada queue

B. DASAR TEORI

Antrian (Queue) dapatdiartikansebagaisuatukumpulan data yang seolah-olah terlihatsepertiada data yang diletakkan di sebelah data yang lain sepertipadagambar 01.Pada gambar, data masukmelaluilorong di sebelahkanandanmasukdariterowongansebelahkiri. Hal inimembuatantrianbersifat FIFO (First In First Out), bedadengan stack yang berciri LIFO.



Gambar 23.1 Konsep Queue sepertiAntrian

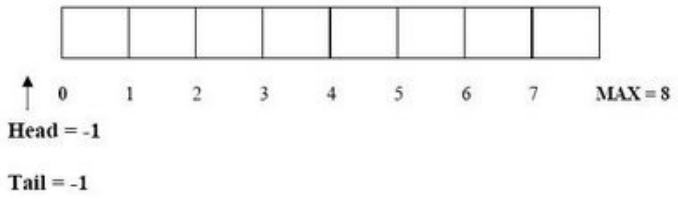
Antriandapatdibuatbaikdengan array maupundenganstruct.Padapembuatan antriandengan array, antrian yang disajikanbersifat statis.Ini disebabkanolehjumlahmaksimal array sudahditentukansejakdeklarasiawal.

Ada 4 operasidasar yang dapatdilakukanpadastruktur data antrean, yakni:

1. CREATE(antrean)
2. ISEMPTY(antrean)
3. INSERT(elemen,antrean)
4. REMOVE(antrean)

Algoritma

Terdapatsatubuahpintumasuk di suatuujungdansatubuahpintukeluar di ujungSatunyaSehinggamembutuhkanvariabel Head dan Tail



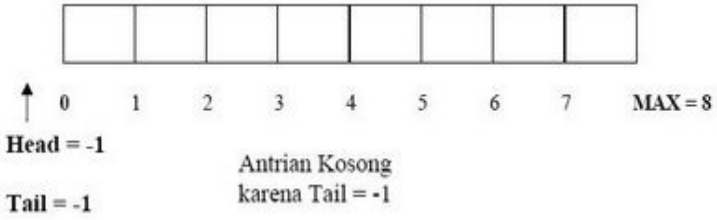
Gambar 23.2PenerapanQueue di Array

Create()

- Untukmenciptakandanmenginisialisasi Queue
- Dengancaramembuat Head dan Tail = -1

IsEmpty()

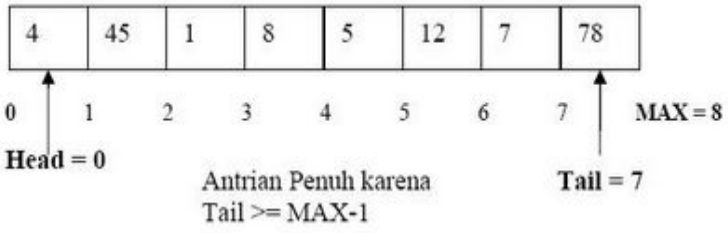
- UntukmemeriksaapakahAntriansudahpenuhataubelum
- Dengancaramemeriksanilai Tail, jika Tail = -1 maka empty
- Kita tidakmemeriksa Head, karena Head adalahtandauntukkepala
- Antrian (elemenpertamadalamantrian) yang tidakakanberubah-ubah
- PergerakanpadaAntrianterjadidenganpenambahanelemenAntrian
- kebelakang, yaitumenggunakannilai Tail



Gambar 23.3 Queue Kosong

IsFull()

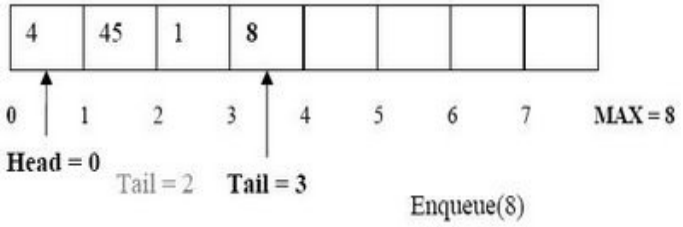
- UntukmengecekapakahAntriansudahpenuhataubelum
- Dengancaramengeceknilai Tail, jika Tail >= MAX-1 (karena MAX-1 adalahbataselemen array pada C) berartisudahpenuh



Gambar 23.4 Queue padasaatPenuh

Enqueue(data)

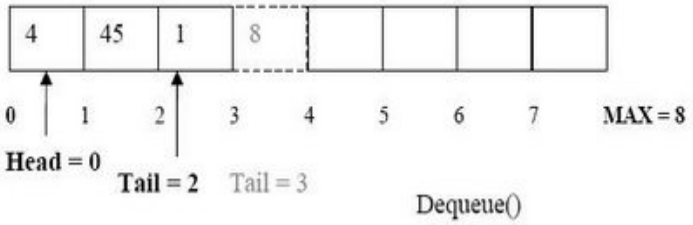
- UntukmenambahkanelemenkedalamAntrian, penambahanelemenselaluditambahkan di elemen paling belakang
- Penambahanelemenselalumenggerakanvariabel Tail dengancara increment counter Tail



Gambar 23.5Memasukkan data keQueue

Dequeue()

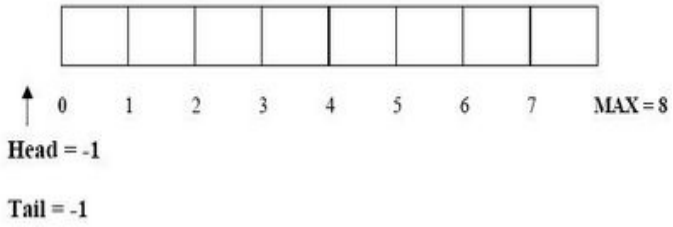
- Digunakanuntukmenghapuselementerdepan/pertamadariAntrian
- Dengancaramengurangi counter Tail danmenggesersemuaelemenantriankedepan.
- Penggeserandilakukandenganmenggunakan looping



Gambar 23.6Mengambil data dariQueue

Clear()

- Untukmenghapuselemen-elemenAntriandengancaramembuat Tail dan Head = -1
- Penghapusanelemen-elemenAntriansebenarnyatidakmenghapusarraynya, namunhanyamengesetindekspengaksesan-nyakenilai -1 sehinggaelemen-elemenantriantidaklagiterbaca



Gambar 23.7fungsiClear()

- Untukmenampilkannilai-nilaielemenAntrian

Menggunakan looping dari head s/d tail Perbedaanantara stack dan queue terdapatpadaaturanpenambahandanpenghapusanelemen. Pada stack, operasipenambahandanpenghapusanelemendilakukan di satuujung. Elemen yang terakhir kali dimasukkanakanberada paling dekatdenganujungataudianggap paling atassehinggapadaoperasipenghapusan, elementeratastersebutakandihapus paling awal, sifatdemikiandikenaldengan LIFO. Pada queue, operasitersebutdilakukan di tempat yang berbeda.Penambahanelemenselaludilakukanmelaluisalahsatuujung, menempatiposisi di belakangelemenelemen yang sudahmasuksebelumnyaataumenjadielemen paling belakang.Sedangkanpenghapusanelemendilakukan di ujung yang berbeda, yaitupadaposisielemen yang masuk paling awalatauelementerdepan.Sifat yang demikiandikenaldengan FIFO.

C. TUGAS PENDAHULUAN

Buatlah resume 1 halamanmengenai**Queue**danberikanpenjelasannya.!

D. PERCOBAAN

```
1.  import java.io.BufferedReader;
2.  import java.io.IOException;
3.  import java.io.InputStreamReader;
4.  import java.util.LinkedList;
5.  import java.util.Queue;
6.
7.  public class Antrian {
8.
9.      private static intukuran;
10.     private static Queue<Integer> queue;
11.
12.     public static void main(String[] args) {
13. System.out.print("Berapaukuran QUEUE diinginkan? ");
14. ukuran = inputData();
15. buatQueue();
16. bacaData();
17. tulisData();
18.
19.     }
20.
21.     private static void buatQueue() {
22.         queue = new LinkedList<Integer>();
23.     }
24.
25.     private static intinputData() {
26. BufferedReaderbfr = new BufferedReader(
27.                 new InputStreamReader(System.in));
```

```
28.         String angkaInput = null;
29.         try {
30. angkaInput = bfr.readLine();
31.         } catch (IOException e) {
32. e.printStackTrace();
33.         }
34. int Data = Integer.valueOf(angkaInput).intValue();
35.         return Data;
36.     }
37.
38.     private static void tulisData() {
39.         Integer data;
40. System.out.println("\nUrutankeluarelemendari QUEUE : ");
41.         for (int i = 0; i <ukuran; i++) {
42.             data = queue.remove();
43. System.out.println("Data ke-" + (i + 1) + " : " + data);
44.         }
45.         data = queue.size();
46. System.out.println("Ukuran QUEUE sekarangadalah " + data);
47.     }
48.
49.     private static void bacaData() {
50.         Integer data;
51.         for (int i = 0; i <ukuran; i++) {
52. System.out.print("Data ke-" + (i + 1) + " : ");
53.             data = inputData();
54. queue.add(data);
55.         }
56.         data = queue.size();
57. System.out.println("Ukuran QUEUE sekarangadalah " + data);
58.     }
59. }
```

E. LATIHAN

1. Implementasikan Interface Queue menggunakan List

Tabel 23.1 Interface Queue

Interface Queue	
booleanisEmpty()	Mengembalikannilai true jika queue kosongdan false jika queue terdapat minimal satuelemen
T peek()	Mengembalikanelemen di depan queue. Jika queue kosongmelempar exceptionyaitu: throws NoSuchElementException.
T pop()	Menghapuselemendi depan queue danmengembalikannilainya. Jika queuekosongmakamelempar exception yaitu: NoSuchElementException.
void push(T item)	Menyisipkan item di akhir queue
int size()	Mengembalikanjumlahelemendari queue

```
public class LinkedQueue<T> implements Queue<T> {

    private LinkedList<T>qlist = null;

    public LinkedQueue() {
qlist = new LinkedList<T>();
    }

    @Override
    public booleanisEmpty() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public T peek() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public T pop() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void push(T item) {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public int size() {
throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

2. Implementasikan Interface Bounded Queue menggunakan Array dan Circular Bounded Queue

Bounded Queue adalah queue yang berisielemendengan size tertentu.

Tabel 23.2Interface BQueueadalah interface BoundedQueue

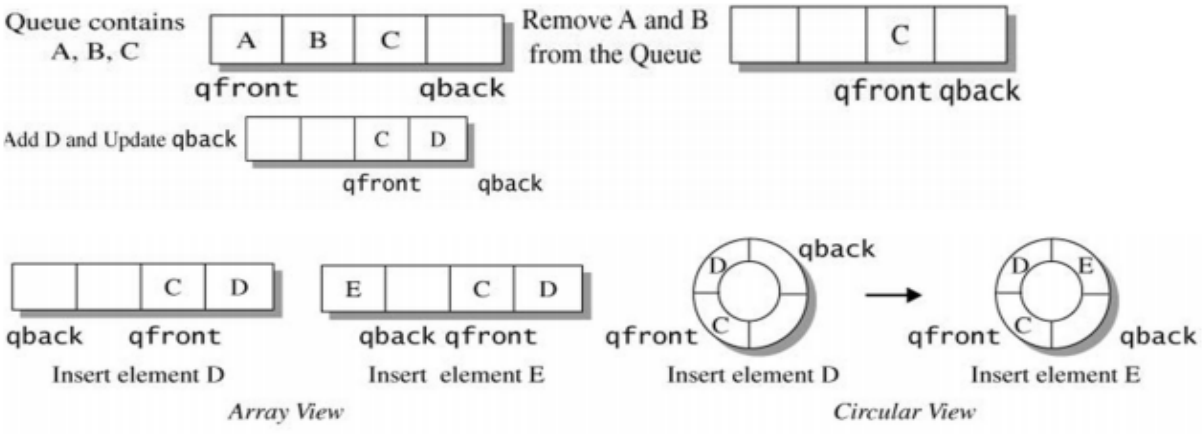
Interface BQueue	
boolean full()	Mengembalikannilai true jika queue penuhsesuaidengan size yang ditentukandan false jika queue belumpenuh

Circular Bounded Queue

Cara mensimulasikanantriansecara circular dalam array linear menggunakan arithmetic modular. Arithmetic modular menggunakanekspresirumus (X % N)

untukmenjagabesarnyanilai X pada range 0:N-1. Jikaindekstelahsampaipada N denganpenambahanataupengurangantersebut, makaindeksakandisetpadaangka 0.

Hal yang samajugadilakukanpada Front jikadilakukanpengambilan item dariantrian. Setelahmengambil item dariantrian, kitamelakukan increment terhadap Front untukpenunjukanpadaposisisesudahnya. Apabilaindekstelahberadapada N, makaindeksdisetjugapadaangka 0.



Gambar 23.8Proses memasukkandanmenghapus data pada circular queue

```
Move qback forward: qback = (qback + 1) % qcapacity;
Move qfront forward: qfront = (qfront + 1) % qcapacity;
```

```
public class ArrQueue<T> implements BQueue<T> {

    private T Arr[];
    private intqfront = 0;
    private intqback = 0;
    private intqcapacity = 0;

    public ArrQueue() {
Arr = (T[]) new Object[50];
qcapacity = 50;
    }

    public ArrQueue(int size) {
Arr = (T[]) new Object[size];
qcapacity = size;
    }

    @Override
    public booleanisEmpty() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
```

```
    public T peek() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public T pop() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void push(T item) {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public int size() {
throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public boolean full() {
throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.