JOBSHEET 8
QUEUE

1. KOMPETENSI

- Mahasiswa mampu memahami algoritma *Queue*.
- Mahasiswa mampu membuat dan mendeklarasikan struktur algoritma *Queue*.
- Mahasiswa mampu menerapkan dan mengimplementasikan algoritma *Queue* menggunakan array

2. ULASAN TEORI

***Queue* (antrian)** adalah sekumpulan elemen/data dimana proses **memasukkan/menambah** elemen/data dilakukan pada posisi **belakang** (*rear*) dan proses **mengeluarkan/mengambil** elemen/data dilakukan pada elemen/data di posisi **depan** (*front*). Istilah untu menggambarkan kondisi tersebut adalah FIFO (*First In First Out*). Jadi proses penambahan data hanya bisa dilakukan di posisi paling belakang, dan sebaliknya, proses penghapusan/pengambilan data dilakukan di posisi paling depan.

Sebagai contoh dapat disimbolkan suatu antrian Q dengan elemen/data sebanyak N di dalamnya (Q1, Q2,...,QN). Untuk data di posisi paling depan antrian Q disimbolkan FRONT(Q) dan bagian paling belakang sebagai REAR(Q). Jadi untuk antrean Q = [Q1, Q2, ..., QN] : FRONT(Q) = Q1 dan REAR(Q) = QN. Kita menggunakan notasi SIZE(Q) untuk menyatakan jumlah elemen di dalam antrean Q. SIZE (Q) mempunyai nilai berupa *integer*. Untuk antrean Q = [Q1,Q2,..., QN], maka SIZE (Q) = N.

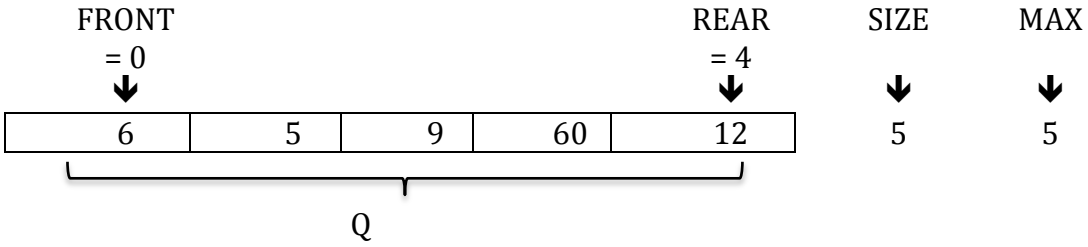
Terdapat beberapa operasi dasar dalam queue, antara lain:

- *Create* : proses awal pembuatan queue
- *Enqueue* : proses penambahan data dalam queue
- *Dequeue* : proses pengambilan data dari queue
- *Is Empty* : proses pengecekan apakah queue dalam kondisi kosong
- *Is Full* : proses pengecekan apakah queue dalam kondisi penuh

Di dalam pemrograman, untuk mengimplementasikan queue bisa menggunakan 2 pendekatan yang berbeda, yaitu:

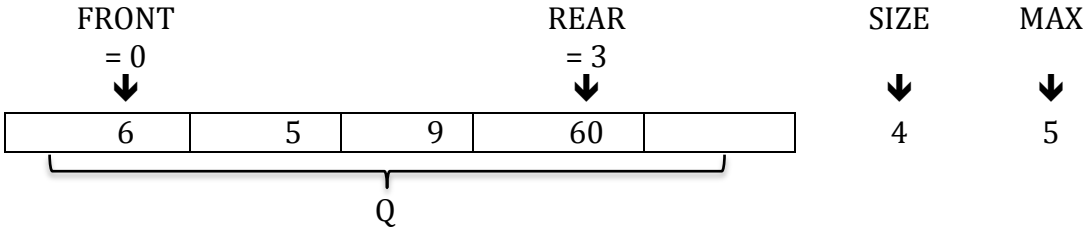
1. Menggunakan Array : panjang queue yang dibuat menggunakan array akan bersifat statis. Misalkan dibuat queue dengan panjang 5, maka maksimal queue tersebut bisa menampung 5 data. Pada materi ini, queue akan dibuat dengan menggunakan array.
2. Menggunakan Linked List : queue yang dibuat dengan menggunakan linked list bersifat lebih dinamis. Artinya jumlah data yang bisa dimasukan ke dalam queue bisa berkembang sesuai dengan yang diinginkan. Linked list akan dibahas pada bahasan selanjutnya di matakuliah ini.

Gambar di bawah ini menunjukan ilustrasi konsep queue yang diimplementasikan menggunakan Array.



- **FRONT** : atribut/variabel yang akan digunakan untuk menyimpan **nilai indeks array**, dimana data **terdepan** dari antrian berada
- **REAR** : atribut/variabel yang akan digunakan untuk menyimpan **nilai indeks array**, dimana data **paling belakang** dari antrian berada
- **SIZE** : atribut/variabel yang akan digunakan untuk menyimpan berapa banyak data yang ada dalam antrian
- **MAX** : atribut/variabel yang akan digunakan untuk menyimpan banyak data maksimal yang bisa disimpan di dalam queue
- **Q :** atribut/variabel yang akan digunakan untuk menyimpan data queue

Gambar di atas menunjukan antrian Q dengan jumlah maksimal data 5, dan sudah terisi penuh dan sudah tidak dapat menerima data antrian lagi. Sedangkan gambar di bawah ini menunjukan antrian Q belum penuh dan masih dapat menerima data antrian lagi di dalamnya.



OPERASI CREATE

Operasi Create merupakan operasi pembuatan queue di tahap awal. Pada awal pembuatan queue, maka yang perlu diinisialisasi adalah:

SIZE = 0
FRONT = REAR = -1

Jika dituliskan dalam pseudocode, prosedur/method/fungsi create adalah sebagai berikut:

```
PROCEDURE create
BEGIN
      q = dekclare array[max]
      size = 0
      front = rear = -1
END
```

Karena masih awal, jadi queue belum memiliki data di dalamnya, sehingga nilai SIZE = 0. Konsekuensinya, posisi FRONT dan REAR diinisialisasi ke index -1.

OPERASI ISEMPTY

Operasi IsEmpty digunakan untuk mengecek apakah suatu queue dalam kondisi kosong. Kondisi kosong akan terjadi ketika SIZE bernilai 0.

```
PROCEDURE isEmpty
BEGIN
      IF size == 0 THEN
            RETURN true
      ELSE
            RETURN false
      ENDIF
END
```

OPERASI ISFULL

Operasi IsFull digunakan untuk mengecek apakah suatu queue dalam kondisi penuh. Kondisi penuh akan terjadi ketika SIZE bernilai sama dengan nilai MAX (jumlah maksimal data yang bisa masuk ke dalam array/antrian).

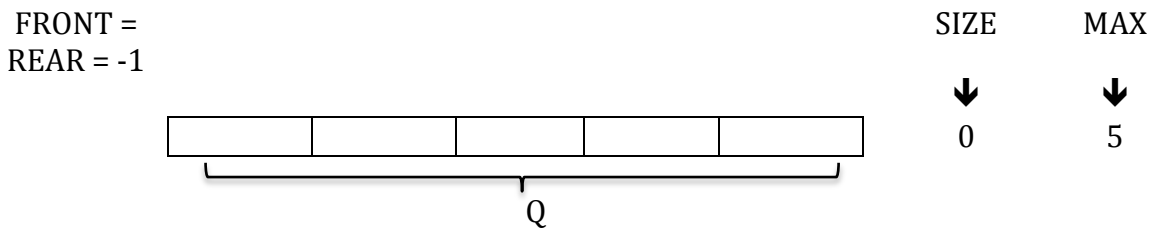
```
PROCEDURE isEmpty
BEGIN
      IF size == max THEN
            RETURN true
      ELSE
            RETURN false
      ENDIF
END
```

OPERASI ENQUEUE

Operasi Enqueue merupakan proses penambahan data baru ke dalam queue. Pada proses enqueue, data baru akan menempati pada posisi paling akhir dalam queue.

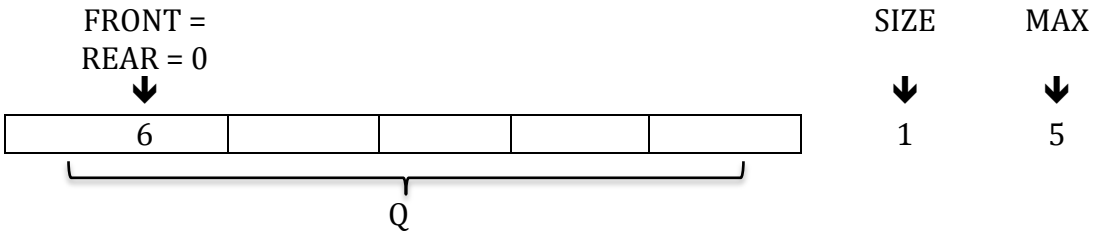
Operasi Enqueue ketika queue dalam kondisi kosong

Ketika ada suatu queue dalam kondisi kosong seperti diilustrasikan dalam gambar di bawah ini:



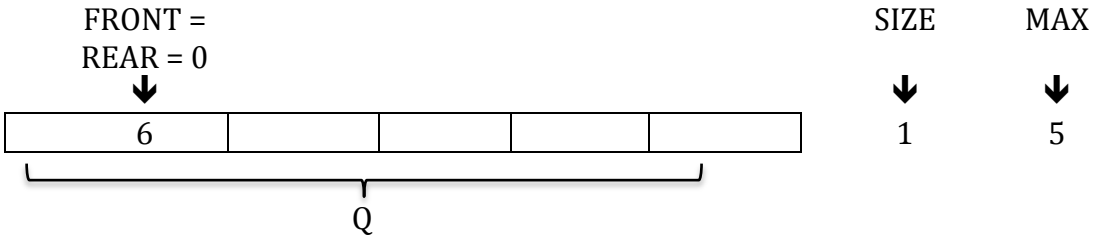
dan hendak dilakukan proses penambahan data, maka data baru dimasukan ke dalam queue pada indeks 0. Dan data tersebut akan menjadi data pada posisi FRONT dan sekaligus pada posisi REAR. Kemudian SIZE akan

bertambah 1. Misalkan data 6 dimasukan ke dalam queue tersebut, maka bisa diilustrasikan sebagai berikut:

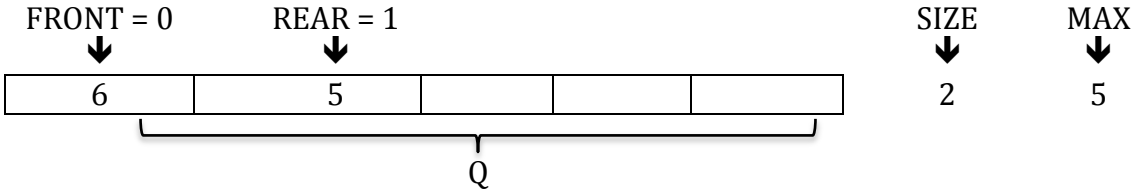


Operasi Enqueue ketika data paling belakang dari queue tidak berada di indeks terakhir array

Ketika ada suatu queue dimana data paling belakang dari queue tidak berada di indeks terakhir array, seperti ditunjukan pada gambar di bawah ini:

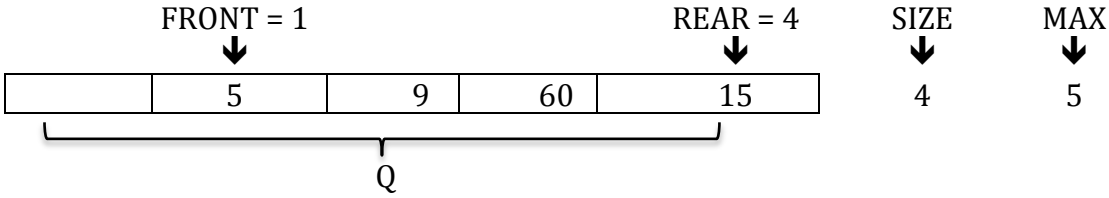


dan akan dimasukan data baru ke dalam queue tersebut, maka data baru tersebut akan menempati posisi setelah data paling belakang saat ini. Artinya data tersebut akan menempati indeks REAR+1, dan SIZE akan bertambah 1. Misal akan dimasukan data baru 5 ke dalam queue, maka bisa diilustrasikan sebagai berikut:



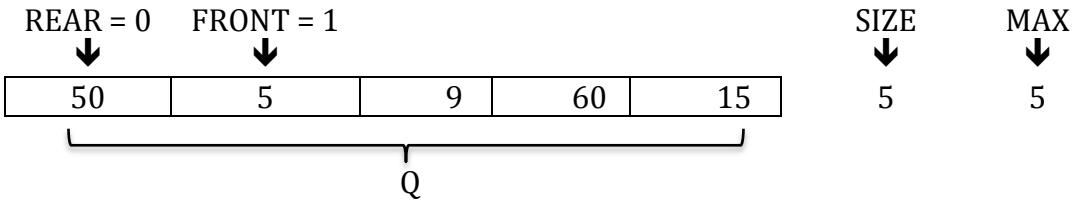
Operasi Enqueue ketika data paling belakang dari queue berada di indeks terakhir array

Ketika ada suatu queue dimana data paling belakang dari queue berada di indeks terakhir array, seperti ditunjukan pada gambar di bawah ini:



Dan akan dimasukan data baru ke dalam queue tersebut, maka data baru tersebut akan menempati posisi indeks 0. Artinya posisi REAR selanjutnya

adalah 0, dan SIZE akan bertambah 1. Misal akan dimasukan data baru 50 ke dalam queue, maka bisa diilustrasikan sebagai berikut:



Dari beberapa kemungkinan proses penambahan data ke dalam queue di atas, apabila digabungkan ke dalam langkah-langkah proses enqueue adalah sebagai berikut:

1. Memastikan bahwa queue tidak dalam kondisi penuh. Jika queue penuh, maka data tidak bisa dimasukan ke dalamnya.
2. Jika tidak penuh, maka proses penambahan data ke dalam queue bisa dilakukan.
    a. Terlebih dulu cek apakah queue dalam kondisi kosong (belum ada data sama sekali). Jika ternyata queue masih kosong, berarti data yang akan masuk menjadi data yang paling depan dan sekaligus menjadi data yang paling akhir dalam queue, yaitu pada posisi indeks 0. Artinya FRONT = REAR = 0
    b. Jika queue dalam kondisi tidak kosong (sudah ada data sebelumnya di dalam queue), yang selanjutnya dilakukan adalah:
        i. Cek apakah posisi REAR berada pada indeks terakhir array. Jika benar, maka posisi REAR selanjutnya adalah di indeks 0
        ii. Jika posisi REAR tidak berada pada indeks terakhir array, maka posisi REAR selanjutnya adalah REAR +1
    c. Masukan data ke dalam queue pada indeks REAR
    d. SIZE bertambah 1

Dan apabila dituliskan dalam pseudocode adalah sebagai berikut:

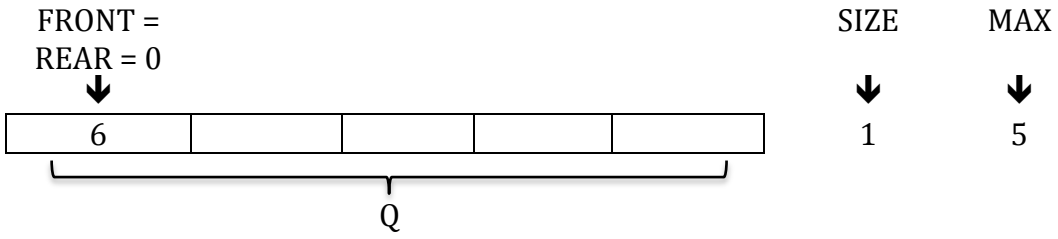
```
PROCEDURE enqueue (data)
BEGIN
      IF isFull() THEN
            PRINT "Antrian sudah penuh!!"
      ELSE
            IF size==0 THEN
                  front = rear = 0
            ELSE
                  IF rear==max-1 THEN
                        rear = 0
                  ELSE
                        rear++
                  ENDIF

            ENDIF
            q[rear] = data
            size++
      ENDIF
END
```

OPERASI DEQUEUE

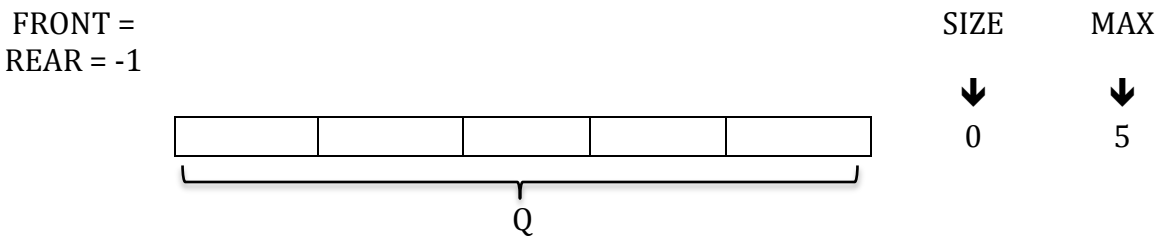
Operasi Dequeue merupakan proses pengambilan data dari dalam queue. Pada proses dequeue, data yang akan terambil adalah data yang menempati pada posisi paling depan (FRONT).

Operasi Dequeue ketika queue dalam kondisi kosong setelah data terambil

Ketika suatu queue, yang didalamnya hanya tertinggal 1 data saja, seperti diilustrasikan pada gambar di bawah ini:

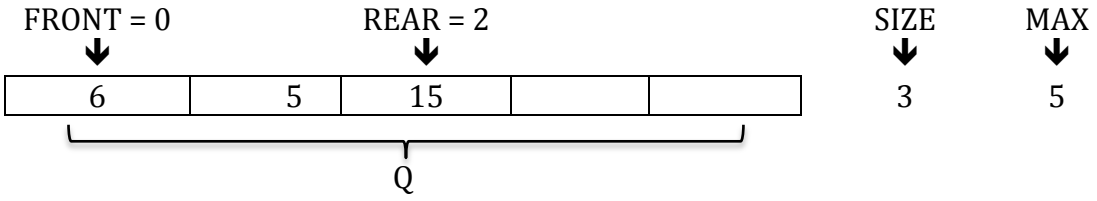


Dan hendak diambil data, maka data yang terambil adalah 6. Dan selanjutnya SIZE akan menjadi 0 dan posisi FRONT serta REAR akan diset menjadi -1. Dan saat tersebut queue berada pada kondisi kosong

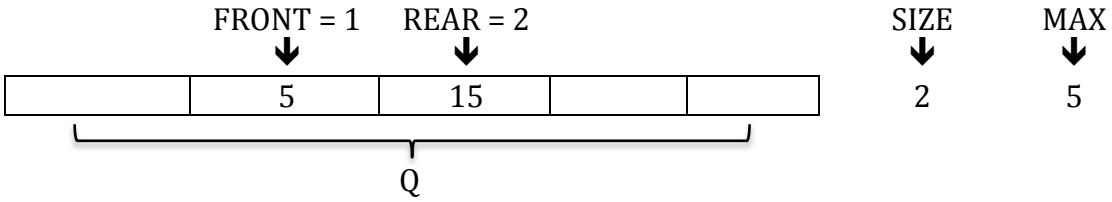


Operasi Dequeue ketika data yang paling depan dari queue tidak berada di indeks terakhir array

Ketika ada suatu queue dimana data paling depan dari queue tidak berada di indeks terakhir array, seperti ditunjukan pada gambar di bawah ini:

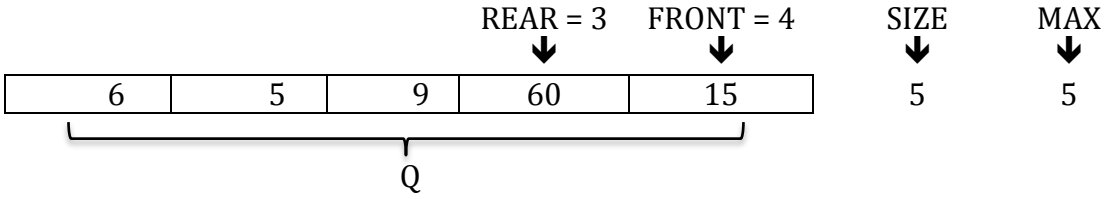


Ketika dilakukan pengambilan data dari queue tersebut, maka data yang terambil adalah 6. Selanjutnya SIZE berkurang 1, dan posisi FRONT akan bertambah 1 dari posisi FRONT sebelumnya.

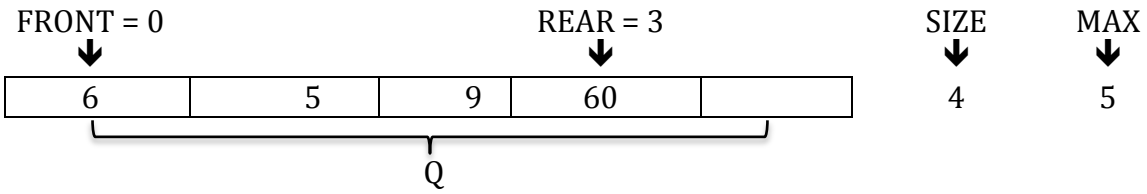


Operasi Dequeue ketika data paling depan dari queue berada di indeks terakhir array

Ketika ada suatu queue dimana data paling depan dari queue berada di indeks terakhir array, seperti ditunjukan pada gambar di bawah ini:



Ketika dilakukan pengambilan data dari queue tersebut, maka data yang terambil adalah 15. Selanjutnya SIZE berkurang 1, dan posisi FRONT akan bergeser ke indeks 0.



Dari beberapa kemungkinan proses pengambilan data dari queue di atas, apabila digabungkan ke dalam langkah-langkah proses dequeue adalah sebagai berikut:

1. Memastikan bahwa queue tidak dalam kondisi kosong. Jika queue kosong, maka tidak ada data untuk diambil darinya.
2. Jika tidak kosong, maka proses pengambilan data dari queue bisa dilakukan.

a. Ambil data yang ada di indeks FRONT, dimana data tersebut akan di return-kan dari proses ini
b. SIZE berkurang 1
c. Selanjutnya, ubah posisi FRONT:
    i. Cek apakah setelah diambil datanya, queue dalam kondisi kosong (SIZE = 0). Jika benar, maka posisi FRONT = REAR = -1
    ii. Jika setelah diambil datanya dan queue tidak kosong, yang selanjutnya dilakukan adalah:
        1. Cek apakah posisi FRONT saat ini berada di indeks terakhir array. Jika benar, maka FRONT selanjutnya ditaruh di indeks 0
        2. Jika posisi FRONT tidak berada di indeks terakhir array, maka posisi FRONT selanjutnya adalah FRONT sebelumnya ditambah 1

Dan apabila dituliskan dalam pseudocode adalah sebagai berikut:

```
PROCEDURE dequeue
BEGIN
      IF isEmpty() THEN
            PRINT "Antrian kosong !!"
            data = 0
      ELSE
            data = q[front]
            size--
            IF size==0 THEN
                  front = rear = -1
            ELSE
                  IF front==max-1 THEN
                        front = 0
                  ELSE
                        front++
                  ENDIF
            ENDIF
      ENDIF
      return data
END
```

3. PROSEDUR PERCOBAAN

PRAKTIKUM 1

Pada praktikum 1 ini akan dibuat class Queue yang di dalamnya terdapat operasi-operasi queue. Di dalamnyya juga terdapat data-data yang ada di dalam antrian.

1. Perhatikan diagram class Queue di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program class Queue.

Queue
max: int size: int

front: int rear: int q: int[]
Queue(max: int) create(): void isEmpty(): boolean isFull(): boolean enqueue(data: int): void dequeue(): int print(): void

2. Buat paket baru dengan nama **minggu8**
3. Buat class di dalam paket tersebut dengan nama **Queue**

```
1   package minggu8;
2
3   public class Queue {
4
5   }
```

4. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas

```
4       int max, size, front, rear;
5       int[] q;
```

5. Selanjutnya tambahkan konstruktor default dan method **create()** di dalam class tersebut. Di dalam konstruktor, terdapat statement yang memanggil/menjalankan method **create()**.

```
6
7       Queue(int m){
8           max = m;
9           create();
10      }
11
12      void create(){
13          q = new int[max];
14          size = 0;
15          front = rear = -1;
16      }
17
```

6. Buat method **isFull()** dan **isEmpty()**

```
17
18      boolean isEmpty(){
19          if (size==0)
20              return true;
21          else
22              return false;
23      }
24
25      boolean isFull(){
26          if (size==max)
27              return true;
28          else
29              return false;
30      }
31
```

7. Selanjutnya buat method **enqueue(int data)**

```
31
32      void enqueue(int data){
33          if(isFull()){
34              System.out.println("Antrian sudah penuh!!");
35          }else{
36              if(isEmpty()){
37                  front = rear = 0;
38              }else{
39                  if(rear==max-1){
40                      rear = 0;
41                  }else{
42                      rear++;
43                  }
44              }
45              q[rear] = data;
46              size++;
47          }
48      }
49
```

8. Selajutnya, buat method **dequeue()**

```
49
50      int dequeue(){
51          int data = 0;
52          if(isEmpty()){
53              System.out.println("Antrian kosong!!");
54          }else{
55              data = q[front];
56              size--;
57              if(isEmpty()){
58                  front = rear = -1;
59              }else{
60                  if(front==max-1){
61                      front = 0;
62                  }else{
63                      front++;
64                  }
65              }
66          }
67          return data;
68      }
69
```

9. Buat method **print()**. Method ini digunakan untuk menampilkan data antrian mulai dari posisi paling depan hingga posisi paling belakang.

```
69
70      void print(){
71          if (isEmpty()){
72              System.out.println("Antrian kosong!!");
73          }else{
74              int i = front;
75              while(i!=rear){
76                  System.out.print(q[i]+"-");
77                  i=(i+1)%max;
78              }
79              System.out.println(q[i]+" ");
                System.out.println("Jumlah antrian = "+size);
81          }
82      }
83
```

PRAKTIKUM 2

Pada praktikum 2 ini akan dibuat class QueueMain yang di dalamnya akan dibuat objek dari class Queue dan simulasi proses-proses di dalamnya.

1. Buat class QueueMain

```
1   package minggu8;
2
3   import java.util.Scanner;
4
5   public class QueueMain {
6
7   }
```

2. Buat method **menu()** di dalam class **QueueMain**

```
6
7       static void menu(){
8           System.out.println("Pilih operasi yang ingin dilakukan:");
9           System.out.println("1. Enqueue");
10          System.out.println("2. Dequeue");
11          System.out.println("3. Print");
12          System.out.println("4. Keluar");
13      }
14
```

3. Tambahkan method **main()** di dalam class **QueueMain.**

```
14
15      public static void main(String[] args) {
16          Scanner sc = new Scanner(System.in);
17          int pil = 0;
18          System.out.print("Masukan berapa maksimal data antrian = ");
19          int m = sc.nextInt();
20          Queue qobj = new Queue(m);
21          do{
22              menu();
23              pil = sc.nextInt();
24              switch(pil){
25              case 1: System.out.print("Masukan data baru = ");
26                      int dataIn = sc.nextInt();
27                      qobj.enqueue(dataIn);
28                      break;
29              case 2: int dataOut = qobj.dequeue();
30                      if(dataOut!=0)
31                          System.out.println("Datayang terambil = "+dataOut);
32                      break;
33              case 3: qobj.print();
34                      break;
35              }
36          }while(pil!=4);
37      }
38
```

4. Jalankan class **QueueMain** dan amati hasilnya.

5. PERTANYAAN

1. Jelaskan pengertian dari Queue!
2. Perhatikan class Queue, di dalamnya terdapat atribut q. Untuk apakah atribut tersebut?
3. Jelaskan apa kegunaan dari atribut max, size, front dan rear yang ada di dalam class Queue!

4. Perhatikan konstruktor class Queue, di dalam konstruktor tersebut terdapat statement untuk memanggil method create(). Apa tujuan pemanggilan tersebut method tersebut?
5. Perhatikan isi method create(), kenapa atribut front dan rear diinisialisasi ke -1 dan tidak ke 0?
6. Perhatikan isi dari method isEmpty() dan isFull(), kapan suatu queue dinyatakan kosong? Dan kapan pula queue dinyatakan penuh?
7. Perhatikan kembali isi dari method isFull(), jika kondisi di dalam IF diubah menjadi **size==max-1**, menurut Anda pengaruh apa yang akan terjadi?
8. Perhatikan method enqueue(), di dalamnya terdapat statement sbb:

```
if(rear==max-1){
        rear = 0;
}
```

   Untuk apakah proses tersebut?
9. Perhatikan kembali method enqueue(), stetment mana yang menunjukan bahwa data baru disimpan di dalam posisi terakhir queue?
10. Perhatikan method dequeue(), mengapa method tersebut tidak dibuat ber-tipe data void?
11. Pada method dequeue(), statement mana yang menunjukan bahwa pada proses pengambilan data , data paling depan yang terambil?
12. Pada method dequeue(),di dalamnya terdapat statement sbb:

```
if(front==max-1){
        front = 0;
}
```

   Untuk apakah proses tersebut?
13. Statement mana yang menunjukan bahwa ketika data terambil dari suatua queue, maka jumlah data yang ada di dalam berkurang satu?
14. Perhatikan method print(), mengapa pada awal proses perulangan, variabel i dimulai dari front (`int i=front`, bukan `int i=0`)?
15. Perhatikan method print(), untuk apakah proses `i=(i+1)%max` ?
16. Perhatikan method print(), kapan proses perulangan di dalam method print() akan berhenti?
17. Perhatikan method main(), statement mana saja yang menunjukan:
    a. Proses pembuatan objek dari class Queue
    b. Proses pemanggilan method enqueue()
    c. Proses pemanggilan method dequeue
    d. Proses pemanggilan method print()

6. TUGAS

1. Tambahkan method-method di bawah ini di dalam class Queue di atas.
    a. Method **printFront() : void** → untuk menampilkan data/nilai yang ada di posisi antrian paling depan
    b. Method **printRear() : void** → untuk menampilkan data/nilai yang ada di posisi antrian paling belakang
    c. Method **printPosition(data: int) : void** → untuk menampilkan di posisi ke berapa dalam antrian, suatu data/nilai berada
    d. Method **printDataByPos(position: int) : void** → untuk menampilkan data/nilai yang ada di suatu posisi antrian tertentu

Kemudian lakukan penyesuaian di dalam class QueueMain untuk mensimulasikan pemanggilan method-method yang baru ditambahkan tersebut!

2. Buatlah program antrian nasabah di suatu bank. Ketika seorang nasabah akan antri, maka ia harus menuliskan terlebih dulu no. rekening, dan nama. Jadi, antrian yang akan dibuat, berisi data-data nasabah berupa no. rekening dan nama. Sehingga pertama kali yang harus dibuat adalah class Nasabah sbb:

Nasabah
noRekening: String nama: String
Nasabah(noRek: String, nm: String) Nasabah() print(): void

Selanjutnya buatlah class Queue sbb

Queue
max: int front: int rear: int size: int q: Nasabah[]
Queue(max: int) create(): void isEmpty(): boolean isFull(): boolean enqueue(data: int): void dequeue(): int print(): void printFront(): void printRear(): void printPosition(nas: Nasabah): void printNasabah(posisi: int): void

Catatan:

- Method create(), isEmpty(), isFull(), enqueue(), dequeue() dan print(), kegunaannya sama seperti yang telah dibuat pada sesi praktikum
- Method printFront(): digunakan untuk menampilkan data Nasabah yang ada di posisi antrian paling depan
- Method printRear(): digunakan untuk menampilkan data Nasabah yang ada di posisi antrian paling belakang
- Method printPosition(): digunakan untuk menampilkan posisi antrian ke berapa, seorang Nasabah berada
- Method printNasabah(): digunakan untuk menampilkan data nasabah pada suatu posisi tertentu dalam antrian