JOBSHEET X QUEUE

Ka Abi Muhammad R.F./12/244107020163

2.1 Percobaan 1 : Operasi Dasar Queue

2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti berikut ini :

```
int [] data;
int front;
int rear;
int size;
int max;
```

3. Buat method IsEmpty bertipe boolean yang digunakan untuk mengecek apakah queue kosong.

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

}
```

4. Buat method IsFull bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.

```
public boolean isFull() {
    if (size == max) {
        return true;
    }else{
        return false;
    }
}
```

5. Buat method peek bertipe void untuk menampilkan elemen queue pada posisi paling depan.

```
public void peek (){
    if (!isEmpty()) {
        System.out.println("Elemen terdepan: " + data[front]);

    }else {
        System.out.println(x:"Queue masih kosong");
    }

}
```

6. Buat method print bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.

```
public void print(){
    if (isEmpty()) {
        System.out.println(x:"Queue masih kosong");

    }else {
        int i = front;
        while (i != rear) {
            System.out.println(data[i]+ " ");
            i = (i+1) % max;
        }
        System.out.println(data[i] + " ");
        System.out.println("Jumlah elemen = " + size);
    }

}
```

7. Buat method clear bertipe void untuk menghapus semua elemen pada queue

```
public void clear(){
    if (!isEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println(x:"Queue berhasil dikosongkan");

    }else {
        System.out.println(x:"Queue masih kosong");
    }

}
```

8. Buat method Enqueue bertipe void untuk menambahkan isi queue dengan parameter dt yang bertipe integer

```
public void Enqueue(int dt){
    if (isFull()) {
        System.out.println(x:"Queue sudah penuh");

    }else {
        if (isEmpty()) {
            front=rear=0;

        }else{
            if (rear == max -1) {
                rear = 0;
            }else {
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }

}
```

9. Buat method Dequeue bertipe int untuk mengeluarkan data pada queue di posisi belakang

```
public int Dequeue(){
    int dt = 0;
    if (isEmpty()) {
        System.out.println(x:"Queue masih kosong");

    }else {
        dt= data[front];
        size--;

       if (isEmpty()) {
        front = rear = -1;
       }else{
        if (front == max-1) {
            front = 0;

        }else{
            front++;
        }
       }
    }
    return dt;

}
```

10. Selanjutnya, buat class baru dengan nama QueueMain tetap pada package Praktikum 1.

Buat method menu bertipe void untuk memilih menu program pada saat dijalankan.

```
public class QueueMain {
    public static void menu(){
        System.out.println(x:"Masukkan operasi yang diinginkan");
        System.out.println(x:"1. Enqueue");
        System.out.println(x:"2. Dequeue");
        System.out.println(x:"3. Print");
        System.out.println(x:"4. Peek");
        System.out.println(x:"5. Clear");
        System.out.println(x:"----------------------");
    }
    Run | Debug
```

11. Buat fungsi main, kemudian deklarasikan Scanner dengan nama sc.

12. Buat variabel n untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.

```
Run | Debug
public static void main(String[] args) {
    int pilih;
    Scanner sc = new Scanner (System.in);

    System.out.print(s:"Masukkan kapasitas queue: ");
    int n = sc.nextInt();

    Queue Q = new Queue(n);

    do {
```

13. Lakukan instansiasi objek Queue dengan nama Q dengan mengirimkan parameter n sebagai kapasitas elemen queue

```
Run | Debug
public static void main(String[] args) {
    int pilih;
    Scanner sc = new Scanner (System.in);

    System.out.print(s:"Masukkan kapasitas queue: ");
    int n = sc.nextInt();

    Queue Q = new Queue(n);

    do {
```

14. Deklarasikan variabel dengan nama pilih bertipe integer untuk menampung pilih menu dari pengguna.

15. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case untuk menjalankan operasi queue sesuai dengan masukan pengguna.

```
do {
    menu();
    pilih = sc.nextInt();
    switch (pilih) {
        case 1:
            System.out.print(s:"Masukkan data baru: ");
            int dataMasuk = sc.nextInt();
            Q.Enqueue(dataMasuk);
            break;
        case 2:
            int dataKeluar = Q.Dequeue();
            if (dataKeluar!= 0) {
                System.out.print("Data yang dikeluarkan: " + dataKeluar);
                break;
            }
        case 3:
            Q.print();
            break;

        case 4:
            Q.peek();
            break;

        case 5:
            Q.clear();
            break;

    }
} while (pilih == 1 || pilih == 2 ||pilih == 3 ||pilih == 4 ||pilih == 5 );
```

16. Compile dan jalankan class QueueMain, kemudian amati hasilnya.

2.1.3. Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?

Sebagai penanda kalau belum ada isinya, front dan rear -1 karena array dimulai dari 0, maka kalau pertama kali dijalankan akan bertambah ke 0, prinsip yang sama pada size

2. Pada method Enqueue, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

Ketika rear sudah sampai pada index paling belakang (kondisi max), set ulang rear = 0, supaya dapat “berputar" dan memanfaatkan ruang kosong di depan array

3. Pada method Dequeue, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

Sama, Digunakan supaya front bisa muter ke indeks awal array.

4. Pada method print, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (int i=0), melainkan int i=front?

Karena front tidak selalu di indeks ke 0

5. Perhatikan kembali method print, jelaskan maksud dari potongan kode berikut!

i = (i + 1) % max;

tujuan intinya adalah menncegah adanya indexarrayoutofbonds

6. Tunjukkan potongan kode program yang merupakan queue overflow!

```
if (isFull()) {
    System.out.println(x:"Queue sudah penuh");

}else {
```

7. Pada saat terjadi queue overflow dan queue underflow, program tersebut tetap dapat berjalan dan hanya menampilkan teks informasi. Lakukan modifikasi program sehingga pada saat terjadi queue overflow dan queue underflow, program dihentikan!

2.2. Percobaan 2 : Antrian Layanan Akademik

2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini. Dan tambahkan method tampilkanData berikut :

```
public Mahasiswa(String nim, String nama, String prodi, String kelas){
    this.nim = nim;
    this.nama = nama;
    this.prodi = prodi;
    this.kelas = kelas;

}
```

3. Salin kode program class Queue pada Praktikum 1 untuk digunakan kembali pada Praktikum 2 ini, ganti nama class-nya dengan AntrianLayanan. Karena pada Praktikum 1, data yang disimpan pada queue hanya berupa array bertipe integer, sedangkan pada Praktikum 2 data yang digunakan adalah object, maka perlu dilakukan modifikasi pada class AntrianLayanan tersebut.

```
Mahasiswa[] data;
int front;
int rear;
int size;
int max;
```

```
public AntrianLayanan (int max){
    this.max = max;
    this.data = new Mahasiswa[max];
    this.front = 0;
    this.rear = -1;
    this.size = 0;
}
```

4. Lakukan modifikasi pada class AntrianLayanan dengan mengubah tipe int[] data menjadi Mahasiswa[] data karena pada kasus ini data yang akan disimpan berupa object Mahasiswa. Modifikasi perlu dilakukan pada atribut, method Enqueue, dan method Dequeue.

```
public void tambahAntrian(Mahasiswa mhs){
    if (isFull()) {
        System.out.println(x:"Antrian penuh,tidak dapat menambah mahasiswa");
        return;
    }
    rear= (rear +1) % max;
    data [rear] = mhs;
    size++;
    System.out.println(mhs.nama + " berhasil masuk ke antrian");
}
```

5. Berikutnya method peek dan print yaitu untuk menampilkan data antrian layanan paling depan dan menampilkan semua data antrian layanan. Ditambahkan dengan method getJumlahAntrian yaitu menampilkan nilai size

```
public Mahasiswa layaniMahasiswa(){
    if (isEmpty()) {
        System.out.println(x:"Antrian kosong");
        return null;
    }
    Mahasiswa mhs = data[front];
    front = (front + 1) % max;
    size--;
    return mhs;
}
```

6. Selanjutnya, buat class baru dengan nama LayananAkademikSIAKAD tetap pada package yang sama. Buat fungsi main, deklarasikan Scanner dengan nama sc.

```
  Scanner sc = new Scanner (System.in);
  AntrianLayanan antrian = new AntrianLayanan(max:5);
  int pilihan;

  do {
```

7. Kemudian lakukan instansiasi objek AntrianLayanan dengan nama antrian dan nilai parameternya adalah nilai maksimal antrian yang ditentukan (misal sama dengan 5).

```
  Scanner sc = new Scanner (System.in);
  AntrianLayanan antrian = new AntrianLayanan(max:5);
  int pilihan;

  do {
```

8. Deklarasikan variabel dengan nama pilihan bertipe integer untuk menampung pilih menu dari pengguna.

```
  Scanner sc = new Scanner (System.in);
  AntrianLayanan antrian = new AntrianLayanan(max:5);
  int pilihan;

  do {
```

9. Tambahkan kode berikut untuk melakukan perulangan menu sesuai dengan masukan yang diberikan oleh pengguna.

```
do {
    System.out.println(x:"\n=== Menu Antrian Layanan Akademik ===");
    System.out.println(x:"1. Tambah mahasiswa ke Antrian");
    System.out.println(x:"2. Layani Mahasiswa");
    System.out.println(x:"3. Lihat Mahasiswa Terdepan");
    System.out.println(x:"4. Lihat semua antrian");
    System.out.println(x:"5. Jumlah Mahasiswa dalam Antrian");
    System.out.println(x:"6. Cek Antrian paling belakang");
    System.out.println(x:"0. keluar");
    System.out.print(s:"Pilih Menu: ");
    pilihan = sc.nextInt(); sc.nextLine();

    switch (pilihan) {
        case 1:
            System.out.print(s:"NIM: ");
            String nim =sc.nextLine();
            System.out.print(s:"Nama: ");
            String nama =sc.nextLine();
            System.out.print(s:"Prodi: ");
            String prodi =sc.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas =sc.nextLine();
            Mahasiswa mhs = new Mahasiswa(nim, nama, prodi, kelas);
            antrian.tambahAntrian(mhs);

            break;

        case 2:
        Mahasiswa dilayani = antrian.layaniMahasiswa();
        if (dilayani != null) {
            System.out.println(x:"Melayani mahasiswa: ");
            dilayani.tampilkanData();
        }
        break;

        case 3:
        antrian.lihatTerdepan();
        break;
```

10. Compile dan jalankan class LayananAkademikSIAKAD, kemudian amati hasilnya.

```
=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 1
NIM: 123
Nama: Aldi
Prodi: TI
Kelas: A
Aldi berhasil masuk ke antrian

=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 1
NIM: 124
Nama: Bobi
Prodi: TI
Kelas: G
Bobi  berhasil masuk ke antrian

=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 123 - Aldi - TI - A
2. 124 - Bobi  - TI - G
```

```
=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 2
Melayani mahasiswa:
123 - Aldi - TI - A

=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 124 - Bobi  - TI - G

=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 5
Jumlah dalam antrian: 1

=== Menu Antrian Layanan Akademik ===
1. Tambah mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat semua antrian
5. Jumlah Mahasiswa dalam Antrian
0. keluar
Pilih Menu: 0
Terima Kasih
PS C:\Users\kaabi\OneDrive\Desktop\Praktikum ASD\P2 Jobsheet 1
```

2.2.3 Pertanyaan

Lakukan modifikasi program dengan menambahkan method baru bernama LihatAkhir pada class AntrianLayanan yang digunakan untuk mengecek antrian yang berada di posisi belakang. Tambahkan pula daftar menu 6. Cek Antrian paling belakang pada class LayananAkademikSIAKAD sehingga method LihatAkhir dapat dipanggil!

```
case 6:
antrian.lihatAkhir();
break;
```

```
public void lihatAkhir() {
if (isEmpty()) {
    System.out.println(x:"Antrian kosong");
} else {
    System.out.print(s:"Mahasiswa di posisi terakhir: ");
    System.out.print(s:"NIM - NAMA - PRODI - KELAS");
    data[rear].tampilkanData();
}
```