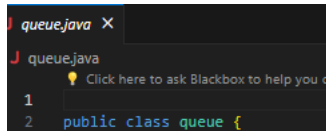


Laporan Jobsheet 8

Muhammad ircham

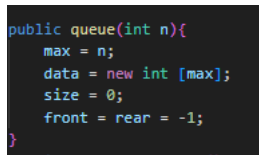
2314760115

1. buat class baru dengan nama Queue.



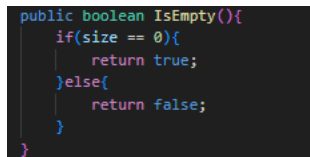
```
queue.java X
queue.java
Click here to ask Blackbox to help you c
1
2 public class queue {
```

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



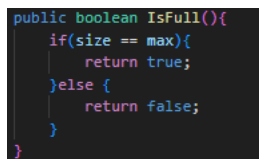
```
public queue(int n){
    max = n;
    data = new int [max];
    size = 0;
    front = rear = -1;
}
```

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



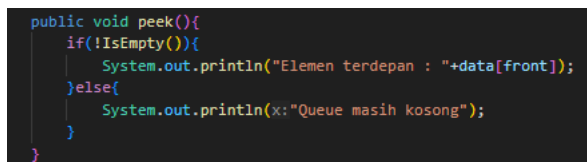
```
public boolean isEmpty(){
    if(size == 0){
        return true;
    }else{
        return false;
    }
}
```

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");
        System.exit(status:0);
    }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 paling depan

```
public int dequeue(){
    int dt = 0;
    if(IsEmpty()){
        System.out.println(x:"Queue masih kosong");
        System.exit(status:0);
    }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. Buat method menu bertipe void untuk memilih menu program pada saat dijalankan.

```
public class queueMain {  
    public static void menu(){  
        System.out.println(x:"\nMasukkan Operasi yang diinginkan ");  
        System.out.println(x:" 1. Enqueue\n 2. Dequeue\n 3. Print\n 4. Peek\n 5. Clear\n 0. Exit");  
        System.out.println(x:"-----");  
    }  
}
```

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

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
}
```

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

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

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

```
queue Q = new queue(n);
```

14. Deklarasikan variabel dengan nama pilih bertipe integer untuk menampung pilih menu dari pengguna. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisimenggunakan switch-case untuk menjalankan operasi queue sesuai dengan masukan pengguna.

```
int pilih;  
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.println("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.

```
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
0. Exit
-----
1
Masukkan data baru : 4
Queue sudah penuh
PS C:\Users\RYZEN\Desktop\Praktikum1>
```

Pertanyaan

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

- atribut front dan rear bernilai -1, dimaksudkan untuk menunjukkan bahwa sizenya masih dalam kondisi kosong/0,
- atribut size bernilai 0, Karena setiap array di mulai dari index ke 0,

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

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

- potongan kode tersebut berguna untuk jika rear/data berada pada posisi max-1/index terakhir dari array, maka disaat ada penambahan data baru, maka akan di tempatkan pada index ke -0.

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

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

- potongan kode tersebut berguna untuk jika front berada pada posisi max-1 atau index terakhir dari array, maka disaat ada penambahan data baru, maka akan di tempatkan pada index ke -0.

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

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

- Karena posisi front atau data terdepan tidak selalu pada indeks ke-0, sedangkan perulangan dimulai dengan posisi frontnya

6. Tunjukkan potongan kode program yang merupakan queue overflow!

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

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!

```
0;
    }else{
        front++;
    }
}
return dt;
}
```

Praktikum2

1. buat class baru dengan nama Nasabah.

```
1
2 public class nasabah {
```

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

```
nasabah(String norek, String nama, String alamat, int umur, double saldo){
    this.norek = norek;
    this.nama = nama;
    this.alamat = alamat;
    this.umur = umur;
    this.saldo = saldo;
}
```

3. Salin kode program class Queue pada Praktikum 1 untuk digunakan kembali pada Praktikum 2 ini. 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 Queue tersebut.

4. Lakukan modifikasi pada class Queue dengan mengubah tipe int[] data menjadi Nasabah[] data karena pada kasus ini data yang akan disimpan pada queue berupa object Nasabah. Modifikasi perlu dilakukan pada atribut, method Enqueue, dan method Dequeue. ZjBaris program Nasabah dt = new Nasabah(); akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class Nasabah.

```

public void enqueue(nasabah 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++;
    }
}

public nasabah dequeue(){
    nasabah dt = new nasabah();
    if(IsEmpty()){
        System.out.println(x:"Queue masih kosong");
    }else{
        dt = data[front];
    }
}

```

6. Karena satu elemen queue terdiri dari beberapa informasi (norek, nama, alamat, umur, dan saldo), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga modifikasi perlu dilakukan pada method peek dan method print.

```

public void peek(){
    if(!IsEmpty()){
        System.out.println("Elemen terdepan : "+data[front].norek+ " "+data[front].nama+" "+data[front].alamat+" "+data[front].umur+" "+data[front].saldo);
    }else{
        System.out.println(x:"Queue masih kosong");
    }
}

public void peekRear(){
    if(!IsEmpty()){
        System.out.println("Elemen terdepan : "+data[rear].norek+ " "+data[rear].nama+" "+data[rear].alamat+" "+data[rear].umur+" "+data[rear].saldo);
    }else{
        System.out.println(x:"Queue masih kosong");
    }
}

public void print(){
    if(IsEmpty()){
        System.out.println(x:"Queue masih kosong");
    }else {
        int i = front;
        while(i != rear){
            System.out.println(data[i].norek + " "+data[i].nama + " "+data[i].alamat + " "+data[i].umur + " "+data[i].saldo);
            i = (i + 1) % max;
        }
        System.out.println(data[i].norek + " "+data[i].nama + " "+data[i].alamat + " "+data[i].umur + " "+data[i].saldo);
        System.out.println("Jumlah elemen = "+ size);
    }
}

```

7. Selanjutnya, buat class baru dengan nama QueueMain tetap pada package Praktikum2. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna

```

public class QueueMain {

    public static void menu(){
        System.out.println(x:"\nPilih Menu ");
        System.out.println(x:" 1. Antrian baru\n 2. Antrian Keluar\n 3. Cek Antrian Terdepan\n 4. Cek Semua Antrian\n 5. Cek Antrian Paling Blekang\n 6. Cek Posisi Antrian\n 7. Cek Data Mahasiswa Berdasarkan Antrian");
    }
}

```

8. Buat fungsi main, deklarasikan Scanner dengan nama sc

```

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
}

```

9. Buat variabel max untuk menampung kapasitas elemen pada queue. Kemudian lakukan instansiasi objek queue dengan nama antri dan nilai parameternya adalah variabel jumlah.

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

    System.out.print(s:"Masukkan kapasitas queue : ");
    int jumlah = sc.nextInt();
    queue antri = new queue(jumlah);
}
```

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

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

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

    int pilih;
```

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

```
do{
    menu();
    pilih = sc.nextInt();
    sc.nextLine();

    switch(pilih){
        case 1:
            System.out.print(s:"No Rekening\t: ");
            String norek = sc.nextLine();
            System.out.print(s:"Nama\t\t: ");
            String nama = sc.nextLine();
            System.out.print(s:"Alamat\t\t: ");
            String alamat = sc.nextLine();
            System.out.print(s:"Umur\t\t: ");
            int umur = sc.nextInt();
            System.out.print(s:"Saldo\t\t: ");
            int saldo = sc.nextInt();
            nasabah nb = new nasabah(norek, nama, alamat, umur, saldo);
            sc.nextLine();
            antri.enqueue(nb);
            break;
        case 2:
            nasabah data = antri.dequeue();
            if(!"".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat) && data.umur !=0 && data.saldo !=0){
                System.out.println("Antrian yang keluar : " + data.norek+ " "+ data.nama+" "+ data.alamat+" "+data.umur+" "+data.saldo);
            }
            break;
        case 3:
            antri.peek();
            break;
        case 4:
            antri.print();
            break;
        case 5:
            antri.peekRear();
            break;
    }
}while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4);
```

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

```
Pilih Menu
1. Antrian baru
2. Antrian Keluar
3. Cek Antrian Terdepan
4. Cek Semua Antrian
5. Cek Antrian Paling Blekang
6. Cek Posisi Antrian
7. Cek Data Mahasiswa Berdasarkan Antrian
0. Exit
-----
5
Queue masih kosong
```

Pertanyaan

1. Pada class QueueMain, jelaskan fungsi IF pada potongan kode program berikut!

```
if (!"".equals(data.norek) && !"".equals(data.nama) && !"".equals(data.alamat)
    && data.umur != 0 && data.saldo != 0) {
    System.out.println("Antrian yang keluar: " + data.norek + " " + data.nama + " "
        + data.alamat + " " + data.umur + " " + data.saldo);
    break;
}
```

- equals adalah membandingkan dua string jika kedua string sama maka akan mereturn true dan sebaliknya
- !"".equals(data.norek) ... digunakan untuk mengecek apakah norek pada data tidak sama dengan string kosong ("") dst
- jika semua kondisi bernilai true maka akan menampilkan pada konsol isi dari data tersebut lalu berhenti (break)
- break berfungsi untuk mengakhiri kode program agar tidak terus melakukan proses

2. Lakukan modifikasi program dengan menambahkan method baru bernama peekRear pada class

Queue yang digunakan untuk mengecek antrian yang berada di posisi belakang! Tambahkan

pula daftar menu 5. Cek Antrian paling belakang pada class QueueMain sehingga method

```
import java.util.Scanner;

public class queueMain {

    public static void menu(){
        System.out.println("\nPilih Menu ");
        System.out.println(" 1. Antrian baru\n 2. Antrian Keluar\n 3. Cek Antrian
Terdepan\n 4. Cek Semua Antrian\n 5. Cek Antrian Paling Blekang\n 6. Cek Posisi
Antrian\n 7. Cek Data Mahasiswa Berdasarkan Antrian\n 0. Exit");
        System.out.println("-----");
    }

    public static void main(String[] args) {
```



```

Scanner sc = new Scanner(System.in);

System.out.print("Masukkan kapasitas queue : ");
int jumlah = sc.nextInt();
queue antri = new queue(jumlah);

int pilih;
do{
    menu();
    pilih = sc.nextInt();
    sc.nextLine();

    switch(pilih){
        case 1:
            System.out.print("No Rekening\t: ");
            String norek = sc.nextLine();
            System.out.print("Nama\t\t: ");
            String nama = sc.nextLine();
            System.out.print("Alamat\t\t: ");
            String alamat = sc.nextLine();
            System.out.print("Umur\t\t: ");
            int umur = sc.nextInt();
            System.out.print("Saldo\t\t: ");
            int saldo = sc.nextInt();
            nasabah nb = new nasabah(norek, nama, alamat, umur, saldo);
            sc.nextLine();
            antri.enqueue(nb);
            break;
        case 2:
            nasabah data = antri.dequeue();
            if(!"".equals(data.norek) && !"".equals(data.nama)
&& !"".equals(data.alamat) && data.umur !=0 && data.saldo !=0){
                System.out.println("Antrian yang keluar : " + data.norek+
" "+ data.nama+" "+ data.alamat+" "+data.umur+" "+data.saldo);
            }
            break;
        case 3:
            antri.peek();
            break;
        case 4:
            antri.print();
            break;
        case 5:
            antri.peekRear();
            break;
    }
}

```

```

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

}

public class queue {
    int max, size, front, rear;
    nasabah[] data;

    public queue(int n){
        max = n;
        data = new nasabah [max];
        size = 0;
        front = rear = -1;
    }
    public boolean IsEmpty(){
        if(size == 0){
            return true;
        }else{
            return false;
        }
    }
    public boolean IsFull(){
        if(size == max){
            return true;
        }else {
            return false;
        }
    }
    public void peek(){
        if(!IsEmpty()){
            System.out.println("Elemen terdepan : "+data[front].norek+ "
"+data[front].nama+" "+data[front].alamat+" "+data[front].umur+"
"+data[front].saldo);
        }else{
            System.out.println("Queue masih kosong");
        }
    }
    public void peekRear(){
        if(!IsEmpty()){
            System.out.println("Elemen terdepan : "+data[rear].norek+ "
"+data[rear].nama+" "+data[rear].alamat+" "+data[rear].umur+"
"+data[rear].saldo);
        }else{
            System.out.println("Queue masih kosong");
        }
    }
}

```

```

    }
}
public void print(){
    if(IsEmpty()){
        System.out.println("Queue masih kosong");
    }else {
        int i = front;
        while(i != rear){
            System.out.println(data[i].norek + " "+data[i].nama + "
"+data[i].alamat + " "+data[i].umur + " "+data[i].saldo);
            i = (i + 1) % max;
        }
        System.out.println(data[i].norek + " "+data[i].nama + "
"+data[i].alamat + " "+data[i].umur + " "+data[i].saldo);
        System.out.println("Jumlah elemen = "+ size);
    }
}
}
public void clear(){
    if(!IsEmpty()){
        front = rear = -1;
        size = 0;
        System.out.println("queue berhasil dikosongkan");
    }else{
        System.out.println("Queue masih kosong");
    }
}
}
public void enqueue(nasabah dt){
    if(IsFull()){
        System.out.println("Queue sudah penuh");
    }else{
        if(IsEmpty()){
            front = rear = 0;
        }else{
            if (rear == max -1){
                rear = 0;
            }else{
                rear++;
            }
        }
        data[rear] = dt;
        size++;
    }
}
}
public nasabah dequeue(){
    nasabah dt = new nasabah();
}

```

```
if(IsEmpty()){
    System.out.println("Queue masih kosong");
}else{
    dt = data[front];
    size--;
    if(IsEmpty()){
        front = rear = -1;
    }else{
        if(front == max -1){
            front = 0;
        }else{
            front++;
        }
    }
}
return dt;
}
```