Jobsheet 10 – Bisma Adhiaksa

Praktikum 1

1. Buat folder baru bernama P1Jobsheet10, lalu buat class dengan nama Queue

```
public class Queue {

    int data[];
    int front, rear;
    int size; //data yang sudah masuk
    int max; //ukuran Queue
```

2. Tambahkan atribut dari class tersebut
3. Lalu tambahkan konstruktor Queue dengan parameter

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

4. Buat method IsEmpty

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

5. Buat method IsFull

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

6. Buat method peek

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

7. Buat method print

```
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);
    }
}
```

8. Buat method clear

```
public void clear(){
    if(IsEmpty()){
        front = rear = -1;
        size = 0;
        System.out.println(x:"Queue berhasil diosongkan");
    }else{
        System.out.println(x:"Queue masih kosong");
    }
}
```

9. Buat method Enqueue

```
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++;
    }
}
```

10. Buat method Dequeue

```
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;
}
```

11. Lalu buat fungsi main dan buat method menu

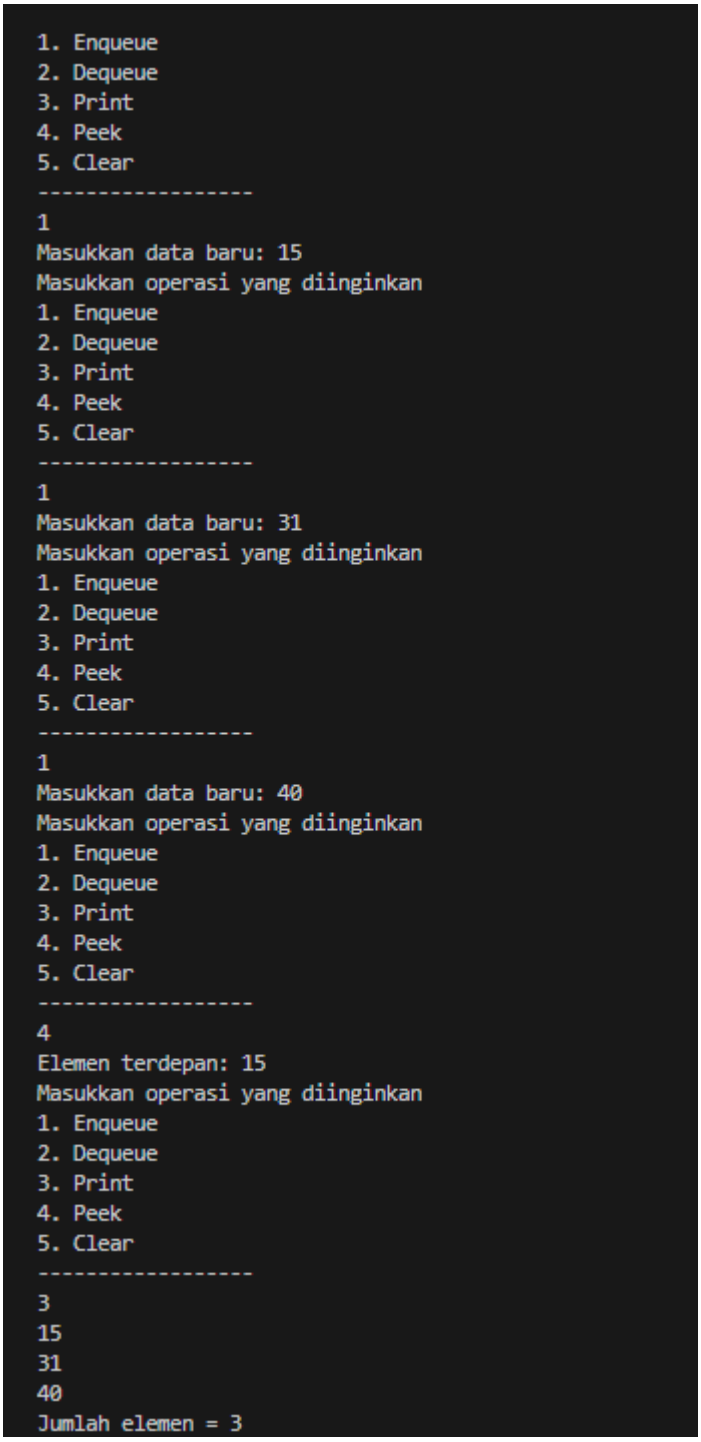
```
import java.util.Scanner;

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:"------------------");
    }
```

12. Buat switch case untuk memanggil method dari class Queue

```
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print(s:"Masukkan kapasitas queue: ");
    int n = sc.nextInt();
    Queue Q = new Queue(n);
    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 <= 5);
}
}
```

13.

1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
1
Masukkan data baru: 15
Masukkan operasi yang diinginkan
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
1
Masukkan data baru: 31
Masukkan operasi yang diinginkan
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
1
Masukkan data baru: 40
Masukkan operasi yang diinginkan
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
4
Elemen terdepan: 15
Masukkan operasi yang diinginkan
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
3
15
31
40
Jumlah elemen = 3

2.1.3. Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?
2. Pada method **Enqueue**, jelaskan maksud dan kegunaan dari potongan kode berikut!

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

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

```
if (front == max - 1) {
    front = 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!

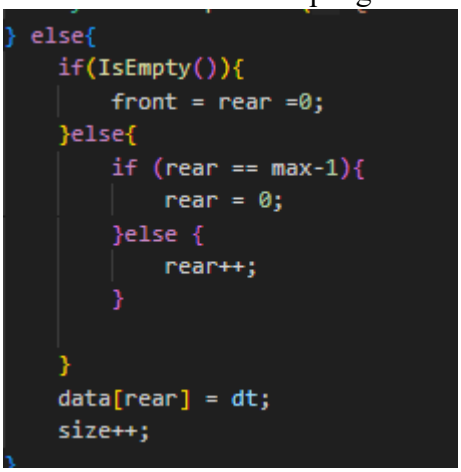
```
i = (i + 1) % max;
```

6. Tunjukkan potongan kode program yang merupakan queue overflow!
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!

Jawaban

1. Front adalah indeks dari nilai di posisi depan data, rear adalah indeks dari nilai di posisi belakng data, dan size adalah jumlah data ada di Queue. Di sini Queue menggunakan array sehingga front dan rear jika belum ada isinya harus berada di posisi -1, sedangkan untuk size karena belum ada data yang masuk, maka size bernilai 0
2. Rear adalah indeks dari nilai di posisi belakang dan max adalah ukuran Queue. Maksud dari kode tersebut adalah jika rear bernilai sama dengan max-1 maka rear akan menunjuk indeks ke 0. Jika rear bernilai sama dengan max-1 berarti posisi rear sekarang ada di indeks paling akhir array. Jika ingin menambahkan lagi harus ke posisi indeks depan lagi yang tidak memiliki nilai.
3. Pada kode tersebut jika front bernilai sama dengan max-1 maka nilai front akan diisi dengan nilai 0. Nilai 0 ini adalah indeks awal array. Pada posisi front bernilai sama dengan max-1 front menunjuk indeks terakhir dari array.
4. Karena jika i=0 pada indeks 0 bisa jadi tidak ada data. Maka i=front lebih masuk akal, karena front sudah pasti menunjuk indeks yang memiliki nilai.
5. Maksud dari kode tersebut adalah, i akan diisi nilai dari i+1 modulo max. ini berfungsi saat posisi front dan rear berada pada indeks yang mana nilai rear lebih kecil dari front. Sehingga saaat nilai i sudah mencapai indeks terakhir dan menjalankan program yang ada di soal nilai i akan menjadi 0.

6. Pada kode ini tidak ada pengecekan apakah indeks ke 0 punya nilai atau tidak



```
} else{
    if(IsEmpty()){
        front = rear =0;
    }else{
        if (rear == max-1){
            rear = 0;
        }else {
            rear++;
        }

    }
    data[rear] = dt;
    size++;
}
```

7.

```
public boolean Enqueue(int dt){
    if(IsFull()){
        System.out.println(x:"Queue sudah penuh");
        return false;
    } else{
        if(IsEmpty()){
            front = rear =0;
        }else{
            if (rear == max-1){
                rear = 0;
            }else {
                rear++;
            }

        }
        data[rear] = dt;
        size++;
        return true;
    }
}

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

```java
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print(s:"Masukkan kapasitas queue: ");
    int n = sc.nextInt();
    boolean lanjut = true;
    Queue Q = new Queue(n);
    int pilih;
    do {
        menu();
        pilih = sc.nextInt();
        switch (pilih) {
            case 1:
                System.out.print(s:"Masukkan data baru: ");
                int dataMasuk = sc.nextInt();
                lanjut = Q.Enqueue(dataMasuk);
                break;
            case 2:
                int dataKeluar = Q.dequeue();
                if (dataKeluar != 0) {
                    System.out.println("Data yang dikeluarkan: " + dataKeluar);
                    break;
                }else {
                    lanjut = false;
                }
            case 3:
                Q.print();
                break;
            case 4:
                Q.peek();
                break;
            case 5:
                Q.clear();
                break;
        }
    } while (pilih >= 1 && pilih <= 5 && lanjut);
}
```

Praktikum 2

1. Buat folder baru dengan nama P2Jobsheet10 lalu buat class Mahasiswa

```java
public class Mahasiswa{
    String nim, nama, prodi, kelas;

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

    public void tampilkanData(){
        System.out.println(nim + " - " + nama + " - " + prodi + " - " + " - " + kelas);
    }
}
```

2. Buat class baru dengan nama AntrianLayanan yang berisi method dari Queue

```java
public class AntrianLayanan {

    Mahasiswa data[];
    int front, rear;
    int size; //data yang sudah masuk
    int max; //ukuran Queue

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

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

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

```java
public void lihatTerdepan() {
    if (!IsEmpty()) {
        System.out.println(x:"Mahasiswa terdepan: ");
        System.out.println(x:"NIM - NAMA - PRODI - KELAS");
        data[front].tampilkanData();
    } else {
        System.out.println(x:"Antrian masih kosong");
    }
}

public void tampilkanSemua() {
    if (IsEmpty()) {
        System.out.println(x:"Antrian masih kosong");
    }
    System.out.println(x:"Daftar mahasiswa dalam antrian:");
    System.out.println(x:"NIM - NAMA - PRODI - KELAS");
    for (int i = 0; i < size; i++) {
        int index = (front +i) % max;
        System.out.print((i+1) +". ");
        data[index].tampilkanData();
    }
}
```

```
public int getJumlahAntrian(){
    return size;
}

public void clear() {
    if (IsEmpty()) {
        front = rear = -1;
        size = 0;
        System.out.println(x:"Queue berhasil diosongkan");
    } else {
        System.out.println(x:"Queue masih kosong");
    }
}
```

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

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

3. Lalu buat class main dengan nama LayananAkademikSIAKAD

```
java.util.Scanner;

class LayananAkademikSIAKAD {
Run | Debug | Run main | Debug main
blic static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  AntrianLayanan antrian = new AntrianLayanan(max:5);
  int pilihan;
```

```java
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:"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.layanMahasiswa();
              if (dilayani != null) {
                  System.out.print(s:"Melayani mahasiswa : ");
                  dilayani.tampilkanData();
              }
              break;
          case 3:
              antrian.lihatTerdepan();
              break;
          case 4:
              antrian.tampilkanSemua();
              break;
          case 5:
              System.out.println("Jumlah dalam antrian: " + antrian.getJumlahAntrian());
              break;
          case 0:
              System.out.println(x:"Terima Kasih.");
              break;
          default:
              System.out.println(x:"Pilihan tidak valid");
              ;
      }

  } while (pilihan != 0);
  sc.close();
```

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!

```
public void lihatTerakhir() {
    if (!IsEmpty()) {
        System.out.println(x:"Mahasiswa terdepan: ");
        System.out.println(x:"NIM - NAMA - PRODI - KELAS");
        data[rear].tampilkanData();
    } else {
        System.out.println(x:"Antrian masih kosong");
    }
}
```

```
case 6:
    antrian.lihatTerakhir();
    break;
case 0:
```

Tugas

Waktu : 120 Menit

Buatlah program antrian untuk mengilustasikan antrian persetujuan Kartu Rencana Studi (KRS) Mahasiswa oleh Dosen Pembina Akademik (DPA). Ketika seorang mahasiswa akan mengantri, maka dia harus mendaftarkan datanya (data mahasiswa seperti pada praktikum 2). Gunakan class untuk antrian seperti pada Praktikum 1 dan 2, dengan method-method yang berfungsi :

- Cek antrian kosong, Cek antrian penuh, Mengosongkan antrian.
- Menambahkan antrian, Memanggil antrian untuk proses KRS – setiap 1x panggilan terdiri dari 2 mahasiswa (pada antrian no 1 dan 2)
- Menampilkan semua antrian, Menampilkan 2 antrian terdepan, Menampilkan antrian paling akhir.
- Cetak jumlah antrian, Cetak jumlah yang sudah melakukan proses KRS
- Jumlah antrian maximal 10, jumlah yang ditangani masing-masing DPA 30 mahasiswa, cetak jumlah mahasiswa yang belum melakukan proses KRS.

Gambarkan **Diagram Class** untuk antriannya. Implementasikan semua method menggunakan menu pilihan pada fungsi **main**.

Diagram class

DPA
Ditampung[]: Mahasiswa Nip: String Nama: String Kode: String
DPA() Void menerima(Mahasiswa ditampung)

Mahasiswa
Nim: String Nama: String Prodi: String Kelas: String
Mahasiswa(nim: String, nama: String, prodi: String, kelas: String) Void tampilkanData()

AntrianLayanan
data []: Mahasiswa dosen: DPA front: int rear: int size: int max: int
Boolean IsEmpty() Boolean IsFull() Void lihat2Terdepan() Void lihatTerakhir() Void tampilkanSemua() Int getJumlahAntrian() Void clear() Void tambahAntrian(Mahasiswa mhs) Mahasiswa layaniMahasiswa() Mahasiswa layaniMahasiswa2() Int sudahDilayani()