JOBSHEET VIII

QUEUE

**NAMA : AHMAD HAYYIN B (01)**

**NIM : 2341760199**

8.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mengenal struktur data Queue
2. Membuat dan mendeklarasikan struktur data Queue
3. Menerapkan algoritma Queue dengan menggunakan array

8.2 Praktikum 1

**Waktu percobaan : 45 menit**

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Queue.

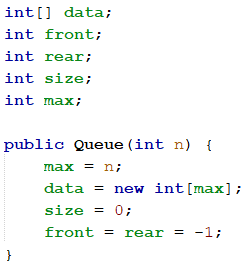
8.2.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class Queue berikut ini:

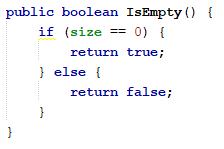
|  |
| --- |
| Queue |
| data: int[]  front: int  rear: int  size: int  max: int |
| Queue(n: int)  isFull(): boolean  isEmpty(): boolean  enqueue(dt: int): void  dequeue(): int  peek: void  print(): void  clear(): void |

Berdasarkan diagram class tersebut, akan dibuat program class Queue dalam Java.

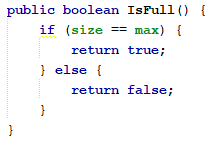
1. Buat package dengan nama **Praktikum1**, kemudian buat class baru dengan nama **Queue**.
2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.



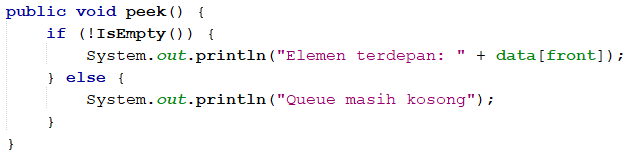
1. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah queue kosong.



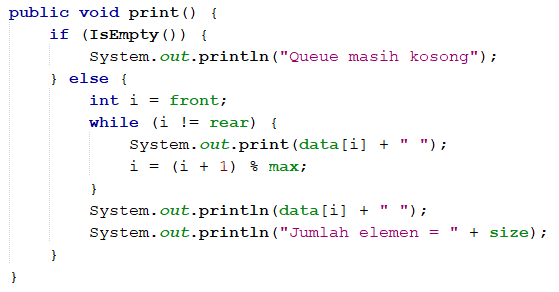
1. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.



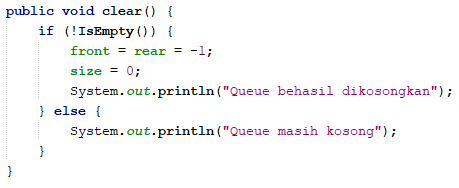
1. Buat method **peek** bertipe void untuk menampilkan elemen queue pada posisi paling depan.



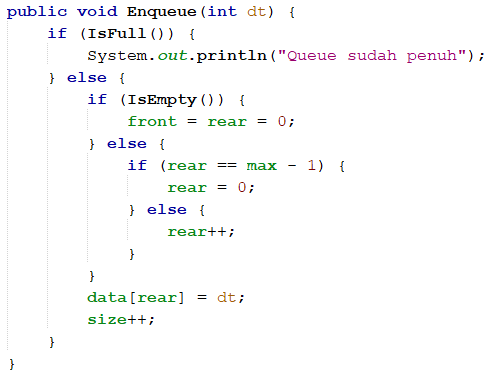
1. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.



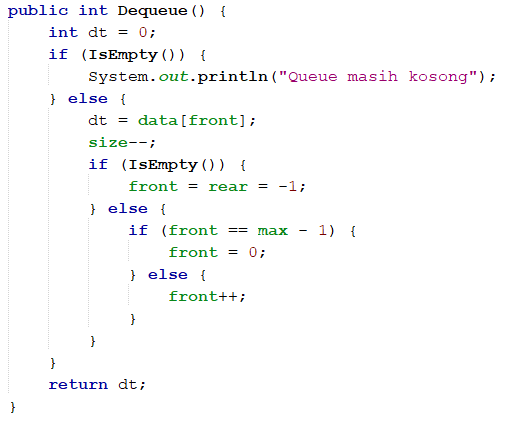
1. Buat method **clear** bertipe void untuk menghapus semua elemen pada queue



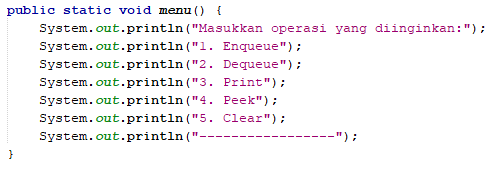
1. Buat method **Enqueue** bertipe void untuk menambahkan isi queue dengan parameter **dt** yang bertipe integer



1. Buat method **Dequeue** bertipe int untuk mengeluarkan data pada queue di posisi paling depan



1. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum1**. Buat method **menu** bertipe void untuk memilih menu program pada saat dijalankan.



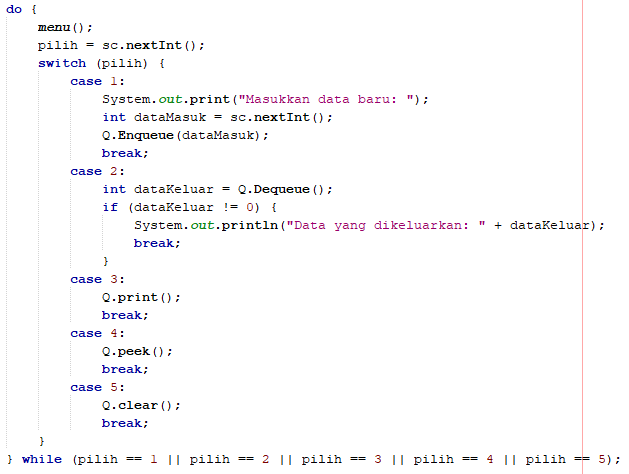
1. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.
2. Buat variabel **n** untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.



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



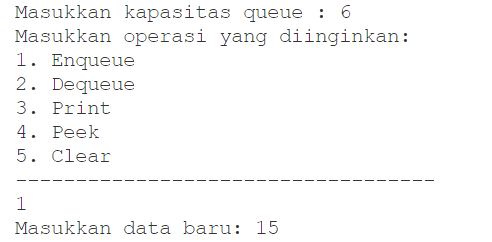
1. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
2. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan **switch-case**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

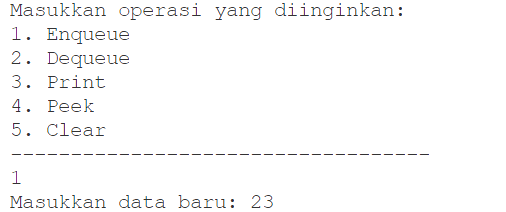


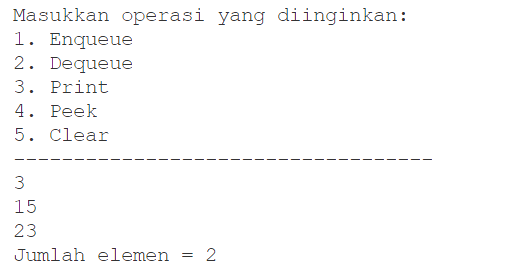
1. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

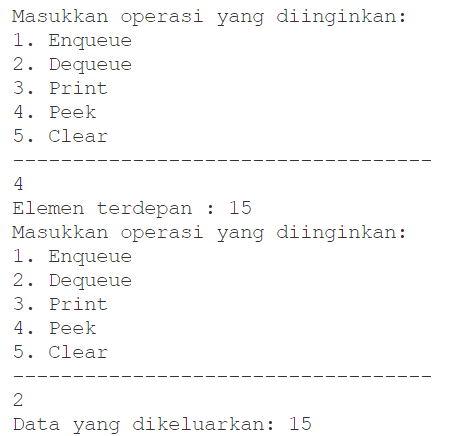
8.2.2 Verifikasi Hasil Percobaan

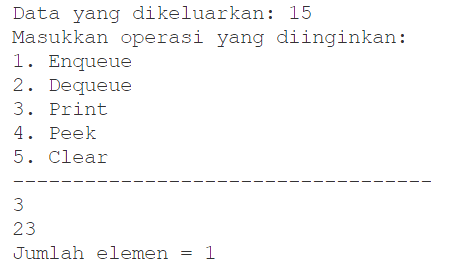
Samakan hasil compile kode program Anda dengan gambar berikut ini.

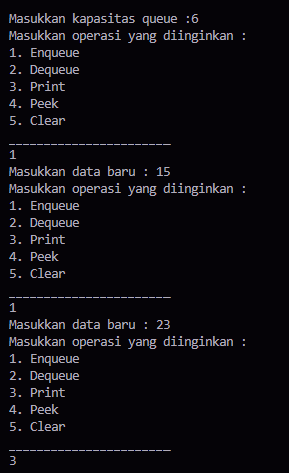


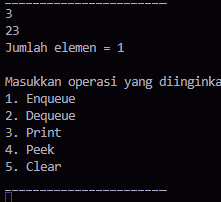
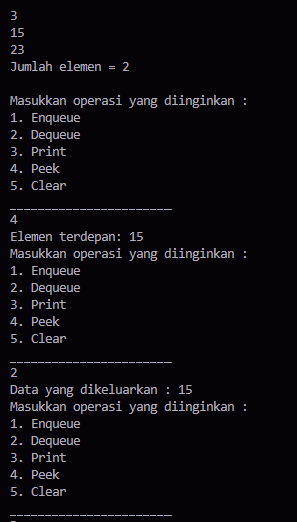












8.2.3 Pertanyaan

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

**Since the queue is empty in front and behind the constructor, the initial attribute is -1, meaning there are no elements in the constructor. We can distinguish between empty and element-based queues by setting the front and back to -1.**

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



**Potongan kode tersebut digunakan untuk mengimplementasikan antrian melingkar (circular queue) pada array. Ketika `rear` mencapai indeks maksimum (`max - 1`), yang menunjukkan bahwa array sudah penuh, kode tersebut mengatur `rear` kembali ke indeks awal (`0`) agar memungkinkan penambahan elemen baru di awal array jika terdapat ruang kosong setelah `rear`. Dengan cara ini, elemen-elemen dapat ditempatkan kembali ke awal array jika diperlukan, sehingga membuat implementasi antrian lebih efisien dalam penggunaan memori.**

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



**Jika bagian depan mencapai akhir array (max - 1), kode tersebut akan mengatur bagian depan kembali ke indeks awal (0) untuk memungkinkan penambahan elemen baru ke array jika ada ruang kosong setelahnya. Dengan cara ini, implementasi antrian dapat memanfaatkan ruang kosong setelah elemen-elemen yang ada diatur ulang.**

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

**`i` dimulai dari `front` agar elemen-elemen antrian dicetak sesuai urutan aslinya dari depan ke belakang, memastikan penambahan elemen sesuai dengan urutan pencetakan.**

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



**Pada setiap iterasi, nilai i akan bertambah satu (i + 1). Operator % digunakan untuk memastikan bahwa nilai i tetap berada dalam rentang indeks yang valid, yaitu antara 0 dan max - 1.**

Tunjukkan potongan kode program yang merupakan queue overflow!

**public void Enqueue(int 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++;**

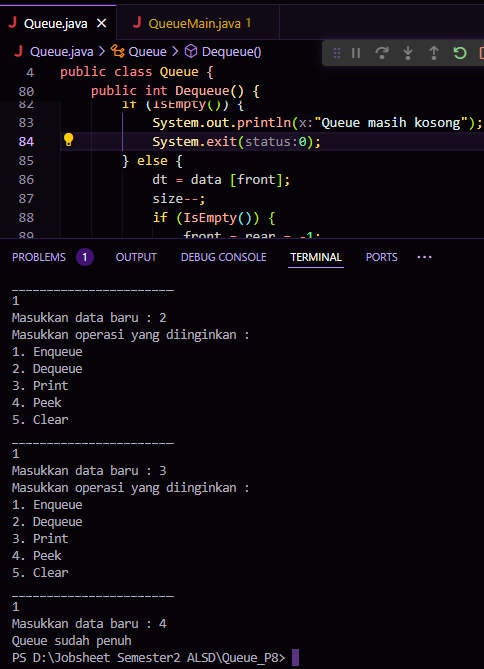
**}**

**}**

**Dalam potongan kode ini, IsFull() digunakan untuk memeriksa apakah antrian sudah penuh. Jika antrian sudah penuh, pesan "Queue sudah penuh" akan dicetak.**

1. 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!

**penambahan System.exit(0) pada kondisi queue overflow dan queue underflow, program akan dihentikan setelah menampilkan pesan informasi.**



8.3 Praktikum 2

**Waktu percobaan : 45 menit**

Pada percobaan ini, kita akan membuat program yang mengilustrasikan teller di bank dalam melayani nasabah.

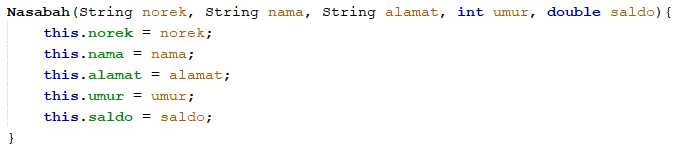
8.3.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

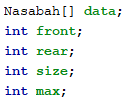
|  |
| --- |
| Nasabah |
| norek: String  nama: String  alamat: String  umur: int  saldo: double |
| Nasabah(norek: String, nama: String, alamat: String, umur: int, saldo: double) |

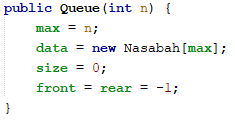
Berdasarkan diagram class tersebut, akan dibuat program class Nasabah dalam Java.

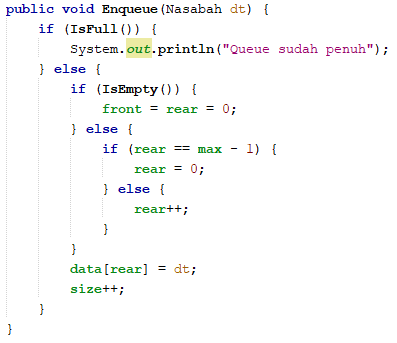
1. Buat package dengan nama **Praktikum2**, kemudian buat class baru dengan nama **Nasabah**.
2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

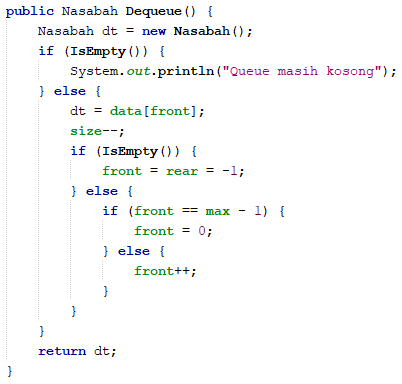


1. Salin kode program class **Queue** pada **Praktikum 1** untuk digunakan kembali pada **Praktikum 2** ini. Karena pada **Praktikum 1**, data yang disimpan pada queue hanya berupa array bertipe integer, sedangkan pada **Praktikum 2** data yang digunakan adalah object, maka perlu dilakukan modifikasi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

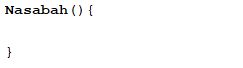




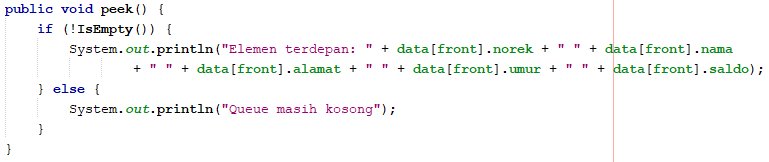


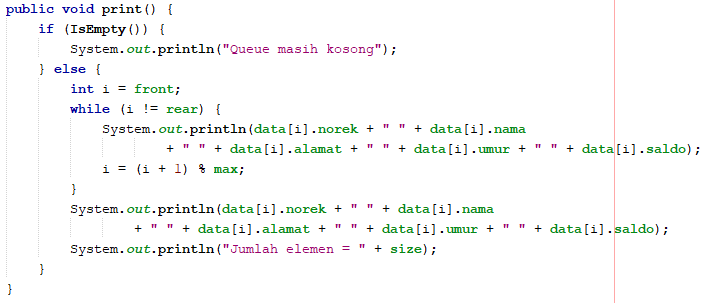


Baris program **Nasabah dt = new Nasabah();** akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class Nasabah.

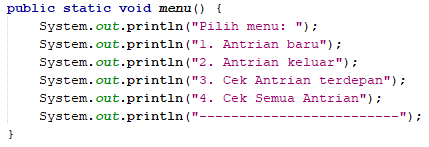


1. Karena satu elemen queue terdiri dari beberapa informasi (norek, nama, alamat, umur, dan saldo), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga meodifikasi perlu dilakukan pada method **peek** dan method **print**.





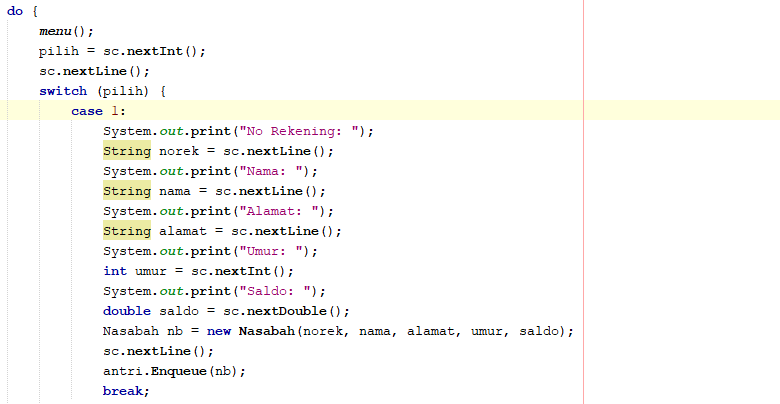
1. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum2**. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna

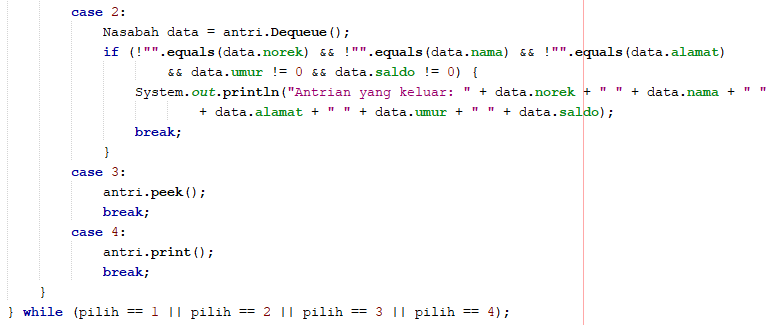


1. Buat fungsi **main**, deklarasikan Scanner dengan nama **sc**
2. Buat variabel **max** untuk menampung kapasitas elemen pada queue. Kemudian lakukan instansiasi objek queue dengan nama **antri** dan nilai parameternya adalah variabel **jumlah**.



1. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
2. Tambahkan kode berikut untuk melakukan perulangan menu sesuai dengan masukan yang diberikan oleh pengguna.

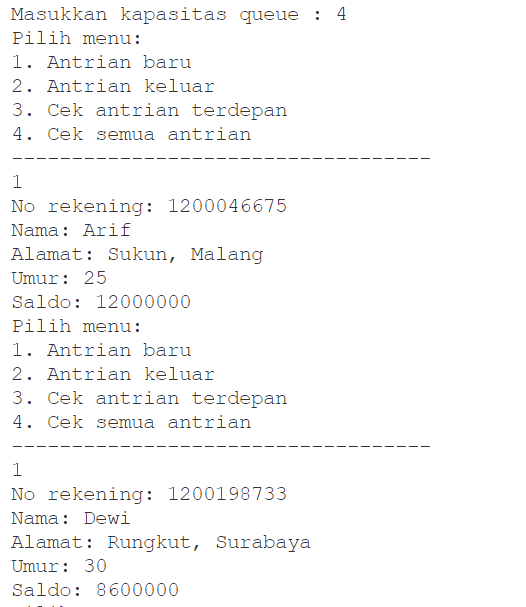


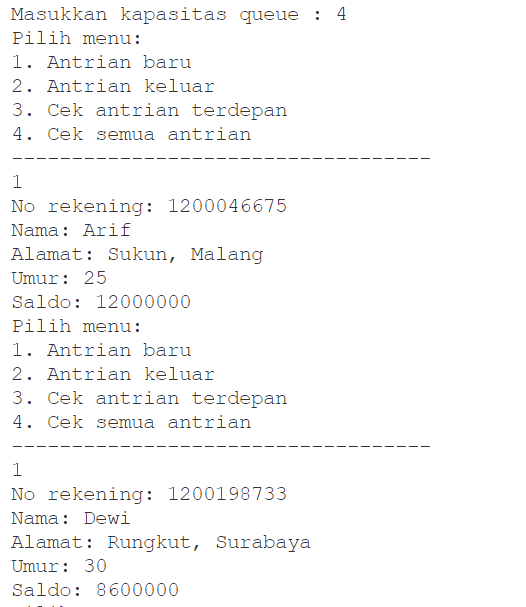


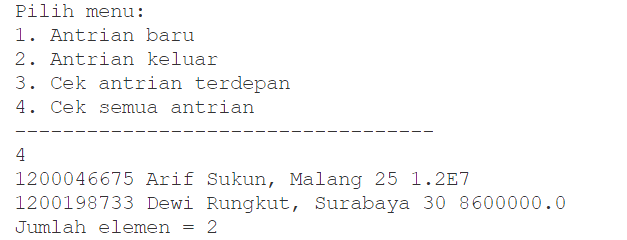
1. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

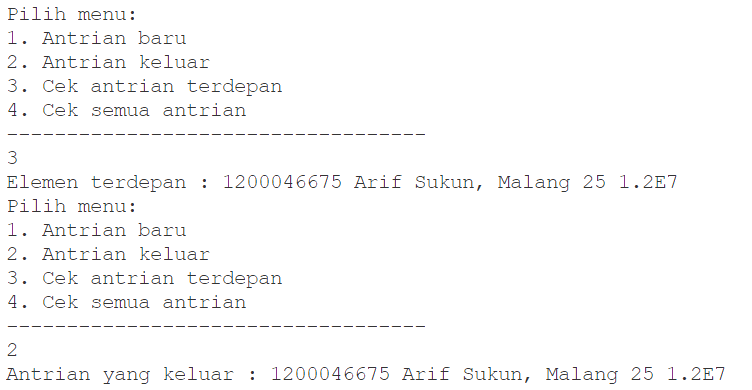
8.3.2 Verifikasi Hasil Percobaan

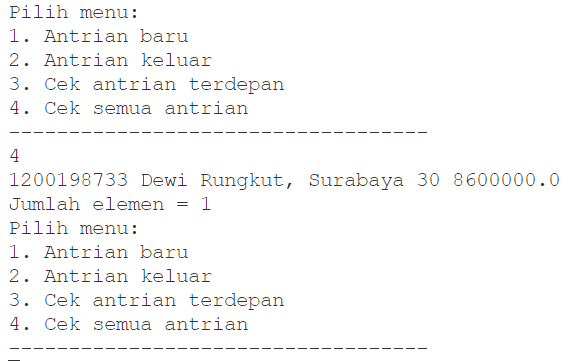
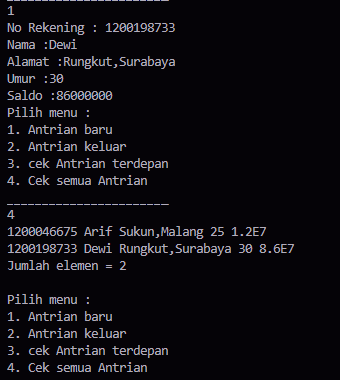
Samakan hasil compile kode program Anda dengan gambar berikut ini.

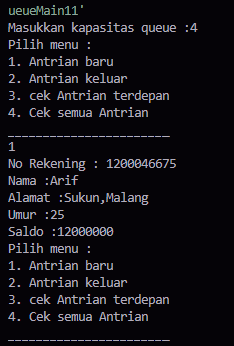


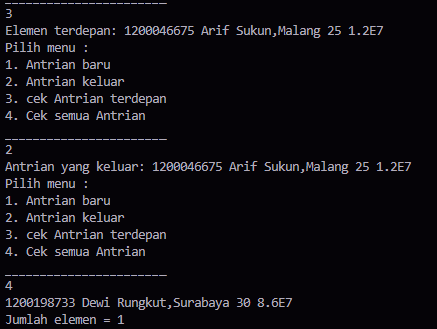






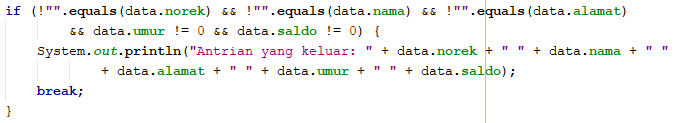




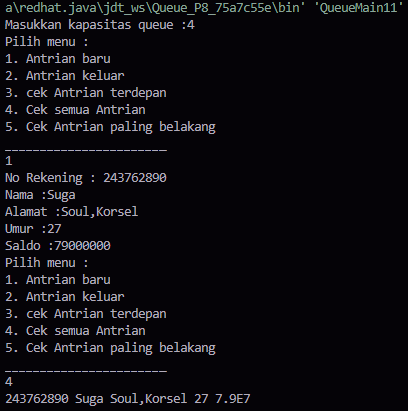


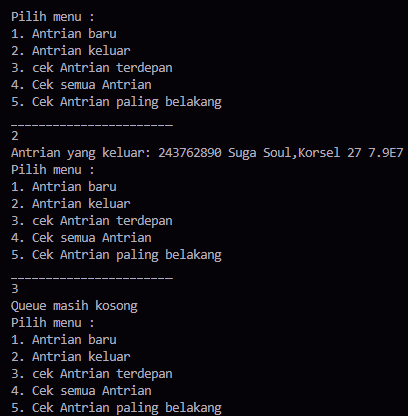
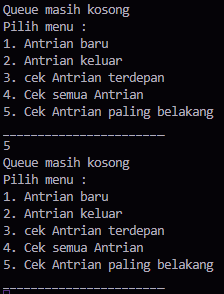
8.3.3 Pertanyaan

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



**Fungsi `if` tersebut memastikan bahwa data yang keluar dari antrian tidak kosong dan bahwa `umur` serta `saldo` memiliki nilai yang valid sebelum data tersebut ditampilkan sebagai "Antrian yang keluar".**

1. 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 **peekRear** dapat dipanggil!



8.4 Tugas

1. Buatlah program antrian untuk mengilustrasikan antrian pasien di sebuah klinik. Ketika seorang pasien akan mengantri, maka dia harus mendaftarkan nama, nomor identitas, jenis kelamin dan umur seperti yang digambarkan pada Class diagram berikut:

|  |
| --- |
| Pembeli |
| nama: String  noID: int  jenisKelamin: char  umur: int |
| Pasien (nama: String, noID: int, jenisKelamin: char, umur: int) |

Class diagram Queue digambarkan sebagai berikut:

|  |
| --- |
| Queue |
| antrian: Pasien[]  front: int  rear: int  size: int  max: int |
| Queue(n: int)  isEmpty(): boolean  isFull(): boolean  enqueue(antri: Pasien): void  dequeue(): int  print(): void  peek(): void  peekRear(): void  peekPosition(nama: String): void  daftarPasien(): void |

Keterangan method:

* Method create(), isEmpty(), isFull(), enqueue(), dequeue() dan print(), kegunaannya sama seperti yang telah dibuat pada Praktikum
* Method peek(): digunakan untuk menampilkan data Pasien yang berada di posisi antrian paling depan
* Method peekRear(): digunakan untuk menampilkan data Pasien yang berada di posisi antrian paling belakang
* Method peekPosition(): digunakan untuk menampilkan seorang pasien (berdasarkan nama) posisi antrian ke berapa
* Method daftarPasien(): digunakan untuk menampilkan data seluruh pasien

