  
Name : Azaria Cindy Sahasika

Number Id : 2341760169 / 06

Class : 1G – Business Information System

Lesson : Algorithm and Data Structure

Material : Jobsheet 10

Github Link : <https://github.com/azariacindy/algorithm-ds>

JOBSHEET X

QUEUE

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.

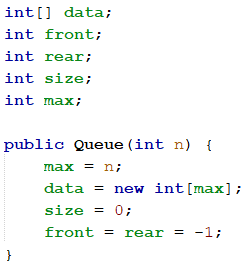
8.2.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class Queue berikut ini:

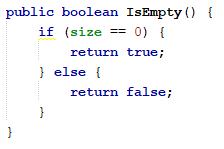
|  |
| --- |
| Queue |
| data: int[]  front: int  rear: int  size: int  max: int |
| Queue(n: int)  isFull(): boolean  isEmpty(): boolean  enqueue(dt: int): void  dequeue(): int  peek: void  print(): void  clear(): void |

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

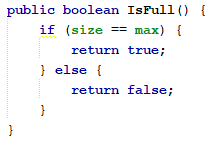
1. Buat package dengan nama **Praktikum1**, kemudian buat class baru dengan nama **Queue**.
2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.



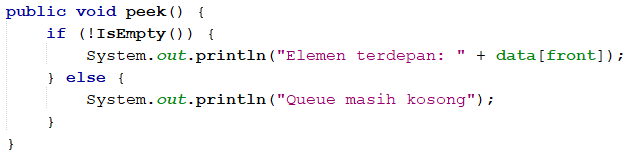
1. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah queue kosong.



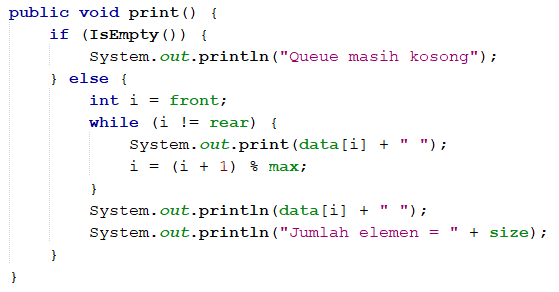
1. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.



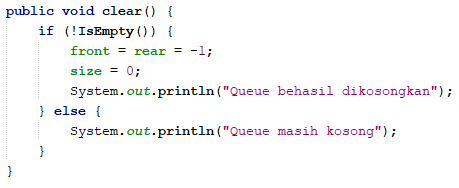
1. Buat method **peek** bertipe void untuk menampilkan elemen queue pada posisi paling depan.



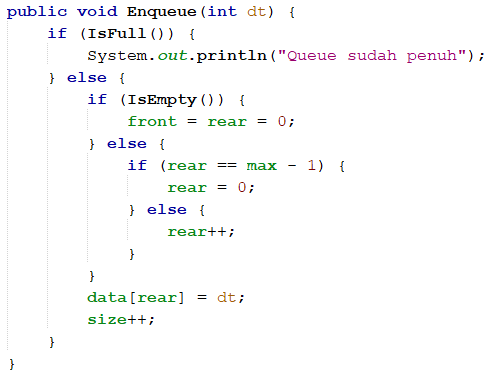
1. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.



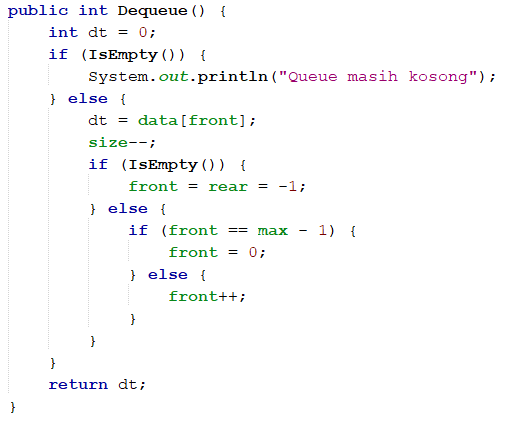
1. Buat method **clear** bertipe void untuk menghapus semua elemen pada queue



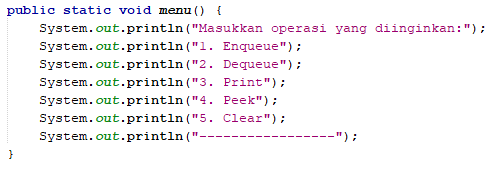
1. Buat method **Enqueue** bertipe void untuk menambahkan isi queue dengan parameter **dt** yang bertipe integer



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.



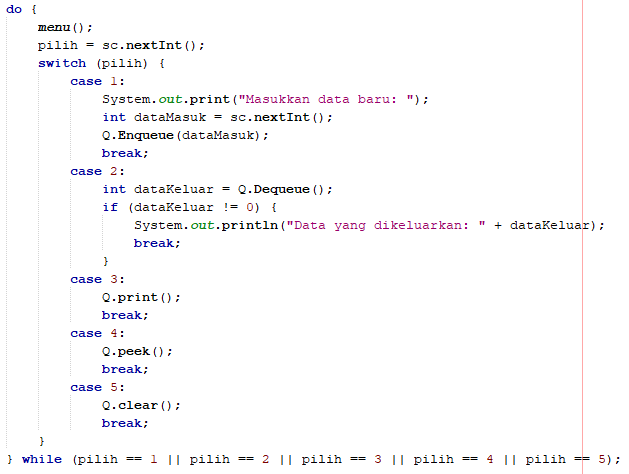
1. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.
2. Buat variabel **n** untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.



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



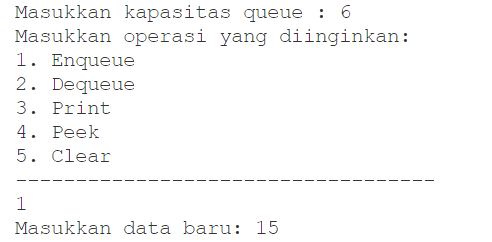
1. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
2. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan **switch-case**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

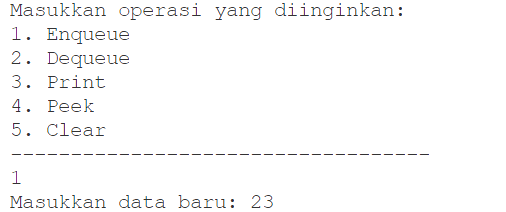


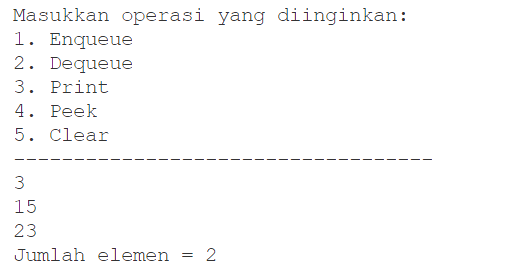
1. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

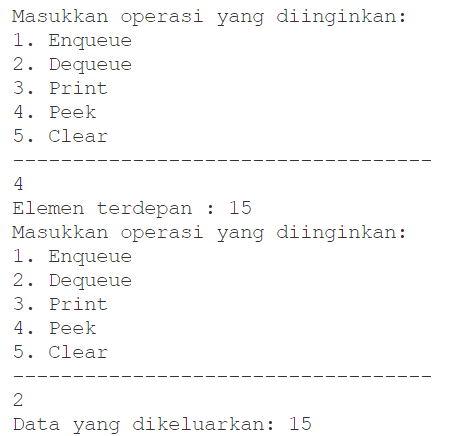
8.2.2 Verifikasi Hasil Percobaan

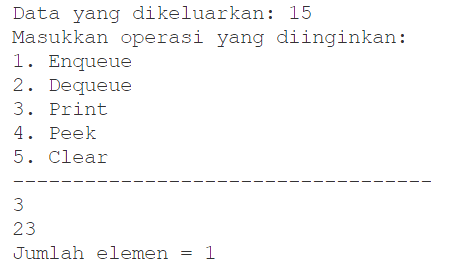
Samakan hasil compile kode program Anda dengan gambar berikut ini.

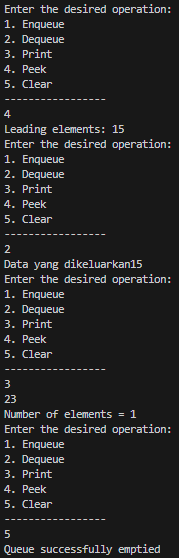
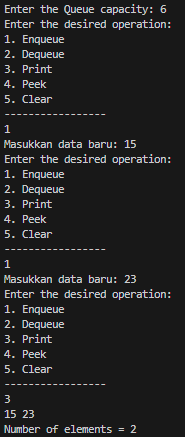












8.2.3 Pertanyaan

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

* Because initially the queue is empty, so there are no elements in front / behind. While the initial value of the size attribute is set to 0 at the beginning of an empty queue, there are no elements in the queue.

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



* To handle the situation when the 'rear' has reached the 'max-1' queue limit and needs to be repeated from the beginning of queue 0 if there is still free space in front.

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



* To handle situations when the 'front' has reached the 'max-1' queue limit and needs to be restarted from 0 if there are still elements behind.

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

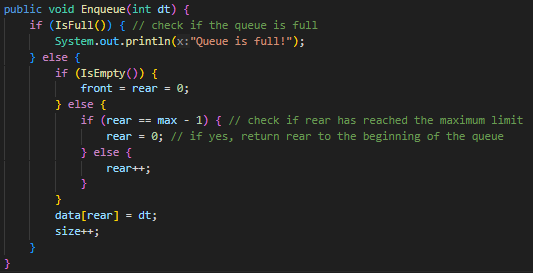
* Since the first index of the queue contains the first element printed, the loop starts from 'front' since the queue structure is cyclic (last element), the first element is the next element. So, it starts from 'front' and goes around until it reaches 'rear'.

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

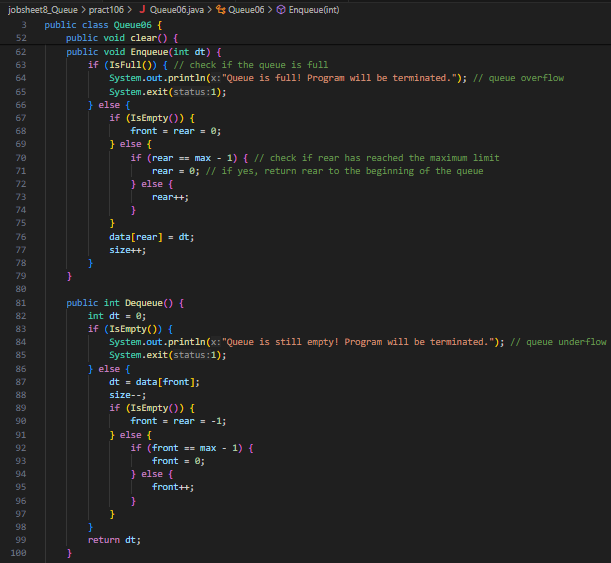


* To handle cyclic looping in the queue structure. If index 'i' returns to 0 after reaching 'max-1', thus creating a cyclic effect that allows to continue traversing the entire queue.

1. Tunjukkan potongan kode program yang merupakan queue overflow!



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!



8.3 Praktikum 2

**Waktu percobaan : 45 menit**

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

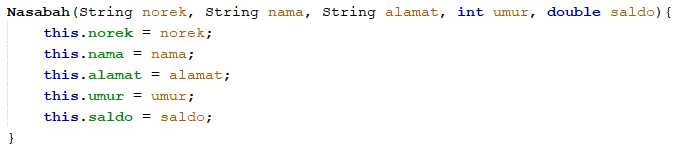
8.3.1 Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

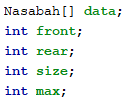
|  |
| --- |
| Nasabah |
| norek: String  nama: String  alamat: String  umur: int  saldo: double |
| Nasabah(norek: String, nama: String, alamat: String, umur: int, saldo: double) |

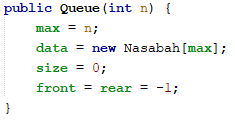
Berdasarkan diagram class tersebut, akan dibuat program class Nasabah dalam Java.

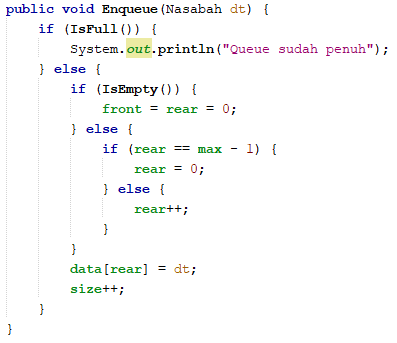
1. Buat package dengan nama **Praktikum2**, kemudian buat class baru dengan nama **Nasabah**.
2. Tambahkan atribut-atribut Nasabah seperti pada Class Diagram, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

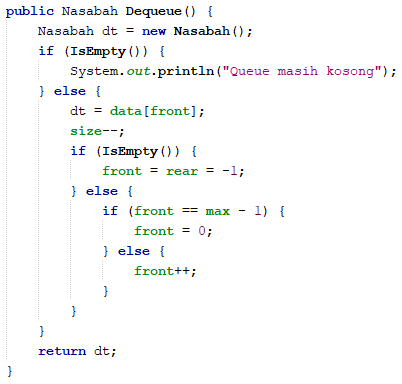


1. Salin kode program class **Queue** pada **Praktikum 1** untuk digunakan kembali pada **Praktikum 2** ini. Karena pada **Praktikum 1**, data yang disimpan pada queue hanya berupa array bertipe integer, sedangkan pada **Praktikum 2** data yang digunakan adalah object, maka perlu dilakukan modifikasi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

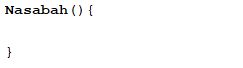




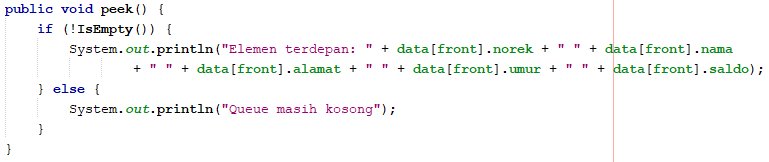


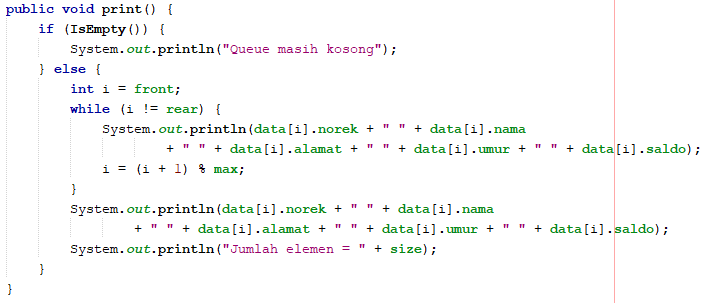


Baris program **Nasabah dt = new Nasabah();** akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class Nasabah.

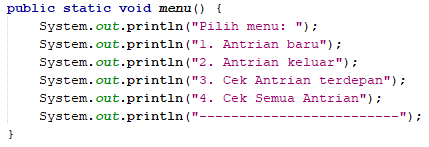


1. Karena satu elemen queue terdiri dari beberapa informasi (norek, nama, alamat, umur, dan saldo), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga meodifikasi perlu dilakukan pada method **peek** dan method **print**.





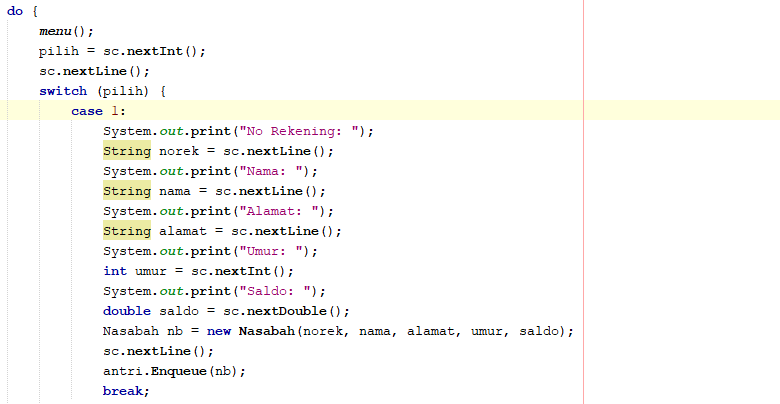
1. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum2**. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna

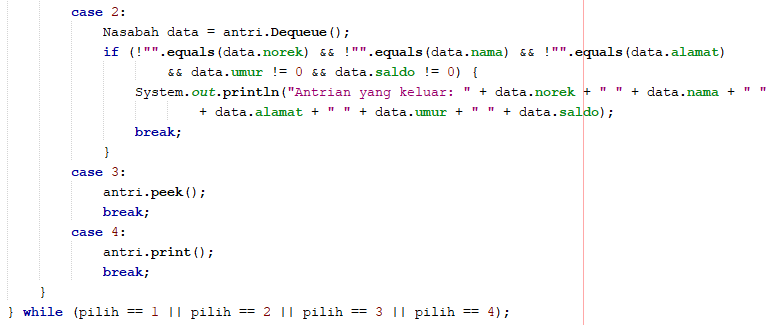


1. Buat fungsi **main**, deklarasikan Scanner dengan nama **sc**
2. Buat variabel **max** untuk menampung kapasitas elemen pada queue. Kemudian lakukan instansiasi objek queue dengan nama **antri** dan nilai parameternya adalah variabel **jumlah**.



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.

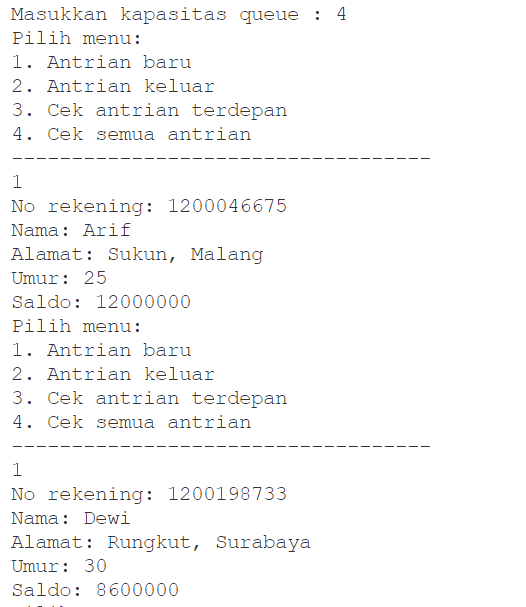


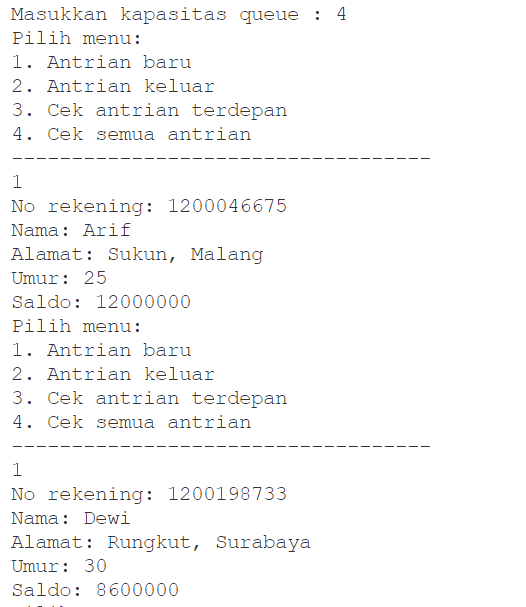


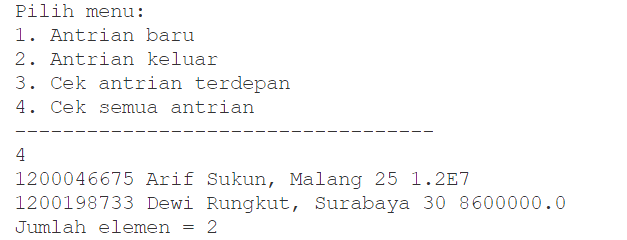
1. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

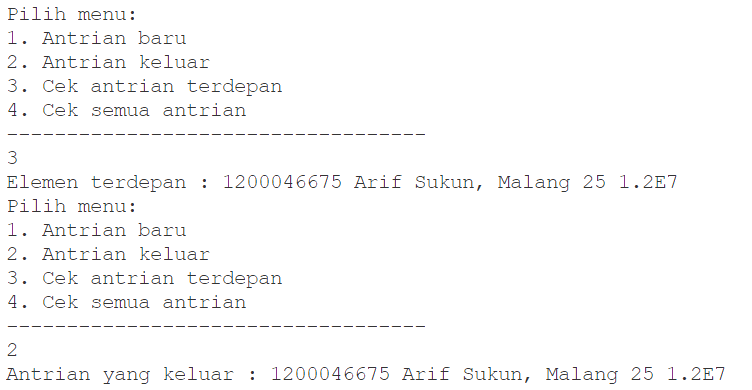
8.3.2 Verifikasi Hasil Percobaan

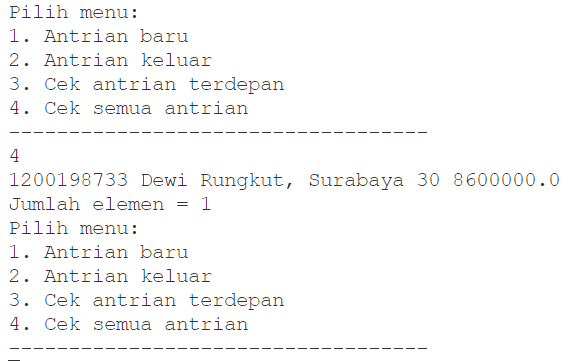
Samakan hasil compile kode program Anda dengan gambar berikut ini.

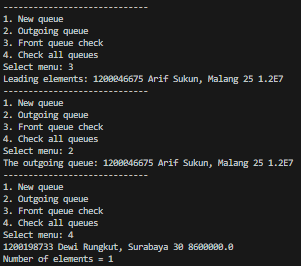
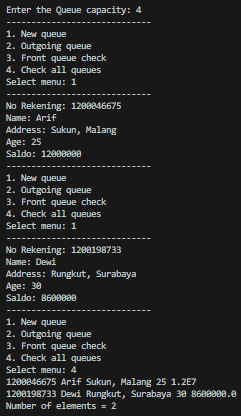






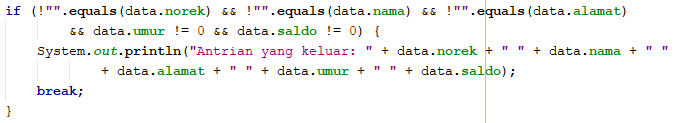






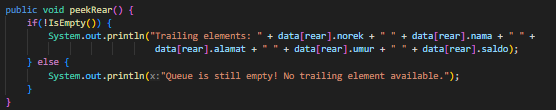
8.3.3 Pertanyaan

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



* To check if the 'customer' data to be output from the queue has valid attribute values. The conditions ensure that all customer attributes (norek, name, address) are not empty, the age and balance of the customer are not the default value of 0. If all conditions are met, then the customer data will be printed as output from the queue.

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

