JOBSHEET X QUEUE

Satria Wiguna / TI\_1D/26

# 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

# Praktikum 1

## Waktu percobaan : 45 menit

Pada percobaan ini, kita akan mengimplementasikan penggunaan class Queue.

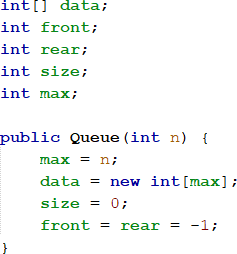
# Langkah-langkah Percobaan

1. Perhatikan Diagram Class Queue berikut ini:

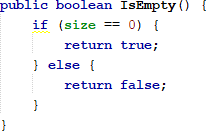
|  |
| --- |
| Queue |
| data: int[] front: int rear: int size: int  max: int |
| Queue(n: int) isFull(): boolean isEmpty(): boolean  enqueue(dt: int): void dequeue(): int  peek: void print(): void  clear(): void |

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

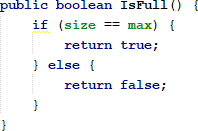
1. Buat package dengan nama **Praktikum1**, kemudian buat class baru dengan nama **Queue**.
2. Tambahkan atribut-atribut Queue sesuai diagram class, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.



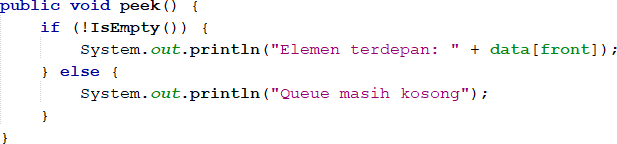
1. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah queue kosong.



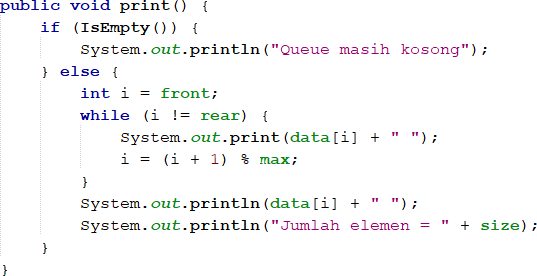
1. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah queue sudah penuh.



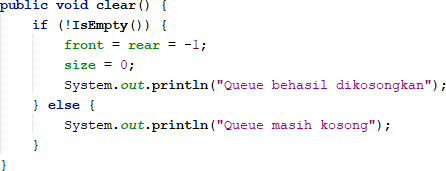
1. Buat method **peek** bertipe void untuk menampilkan elemen queue pada posisi paling depan.

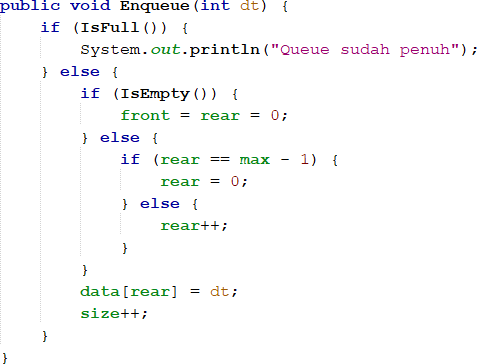


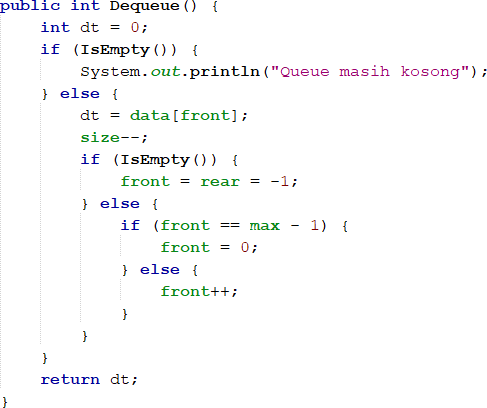
1. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada queue mulai dari posisi front sampai dengan posisi rear.

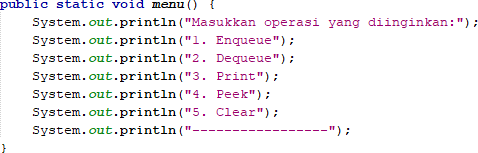


1. Buat method **clear** bertipe void untuk menghapus semua elemen pada queue



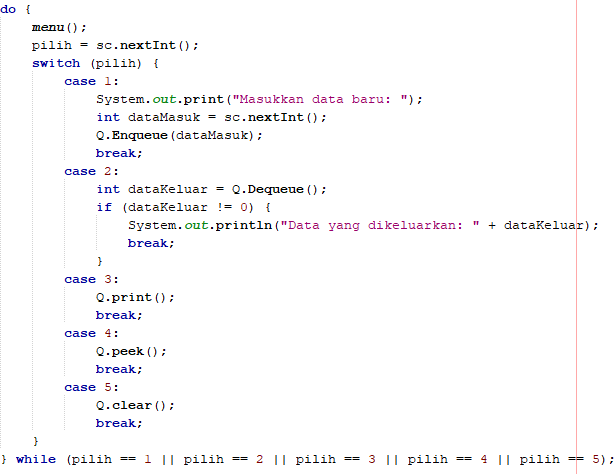
1. Buat method **Enqueue** bertipe void untuk menambahkan isi queue dengan parameter **dt** yang bertipe integer
2. Buat method **Dequeue** bertipe int untuk mengeluarkan data pada queue di posisi belakang



1. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum1**. Buat method **menu** bertipe void untuk memilih menu program pada saat dijalankan.
2. Buat fungsi **main**, kemudian deklarasikan Scanner dengan nama **sc**.
3. Buat variabel **n** untuk menampung masukan berupa jumlah maksimal elemen yang dapat disimpan pada queue.
4. Lakukan instansiasi objek Queue dengan nama **Q** dengan mengirimkan parameter **n** sebagai kapasitas elemen queue

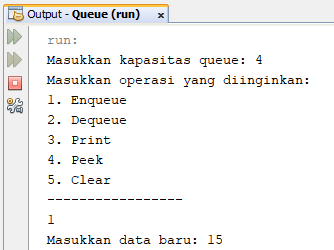


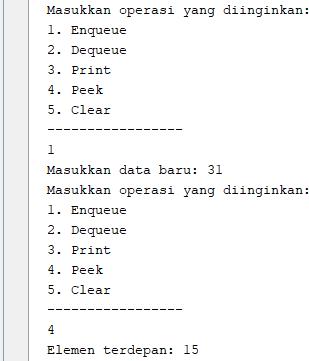
1. Deklarasikan variabel dengan nama **pilih** bertipe integer untuk menampung pilih menu dari pengguna.
2. Lakukan perulangan menggunakan do-while untuk menjalankan program secara terus menerus sesuai masukan yang diberikan. Di dalam perulangan tersebut, terdapat pemilihan kondisi menggunakan switch-case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



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

# Verifikasi Hasil Percobaan

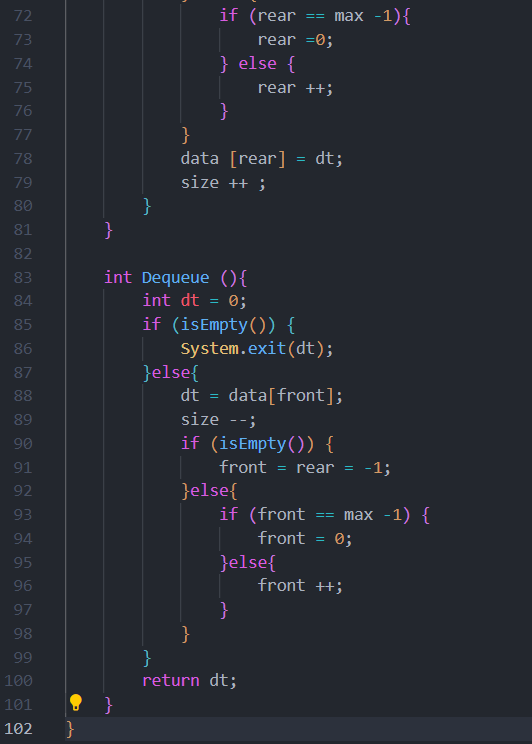
Samakan hasil compile kode program Anda dengan gambar berikut ini.



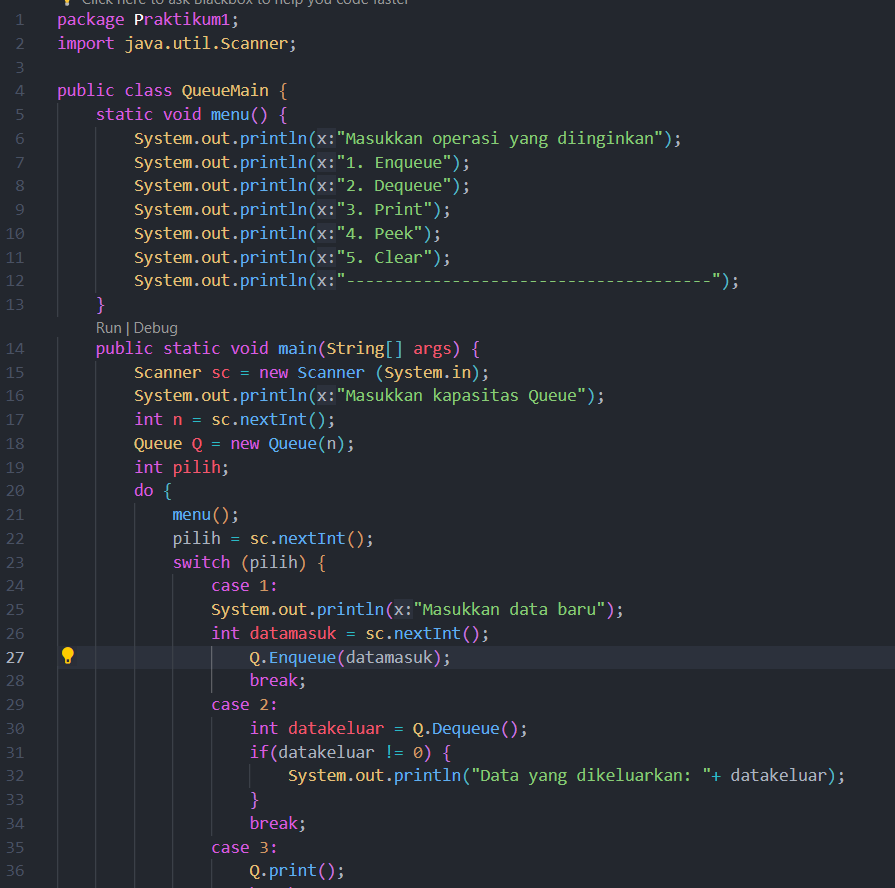
**Queue :**

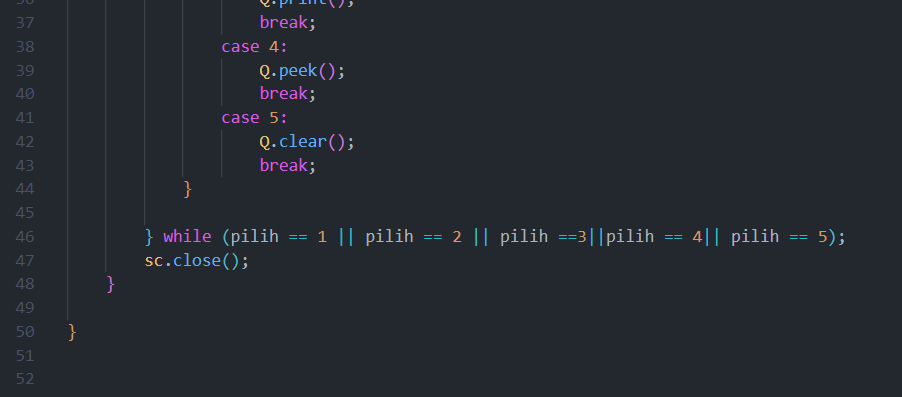
****

****

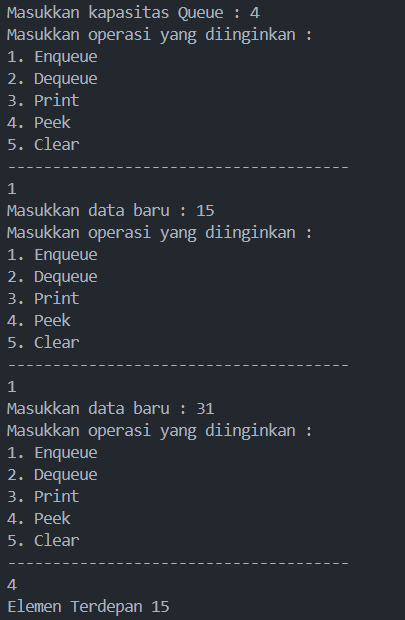
****

**QueueMain :**

****

****

**Verivikasi Hasil :**

****

# Pertanyaan

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

Pada konstruktor, nilai awal atribut front dan rear diatur ke -1 sedangkan size diatur ke 0 untuk menandakan bahwa pada awalnya antrian (queue) dalam keadaan kosong.

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



Potongan kode if (rear == max - 1) { rear = 0; dalam method Enqueue memiliki maksud dan kegunaan untuk menangani kondisi di mana rear mencapai batas maksimal dari array antrian (max - 1) kode tersebut membantu dalam mengimplementasikan antrian sirkular

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



Potongan kode if (front == max - 1) { front = 0; } ini membantu dalam mengimplementasikan antrian sirkular, di mana penghapusan elemen dari depan antrian akan terus dilakukan dengan melanjutkan dari awal array setelah mencapai batas akhir dari array yang digunakan untuk menyimpan elemen-antrian.

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

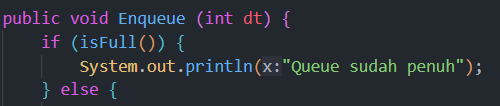
pada method print, perulangan dimulai dari front karena antrian tidak selalu dimulai dari indeks 0. karena elemen mungkin dimulai dari posisi lain selain 0 tergantung pada bagaimana elemen-elemen dimasukkan dan dihapus dari antrian.

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

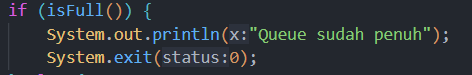


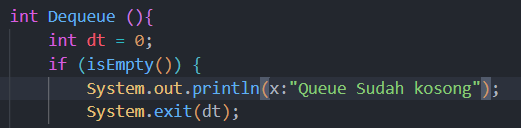
Potongan kode i = (i + 1) % max; dalam method print digunakan untuk menggerakkan variabel i ke elemen berikutnya dalam antrian, dengan memperhitungkan sifat sirkular dari antrian.

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!





# Praktikum 2

## Waktu percobaan : 45 menit

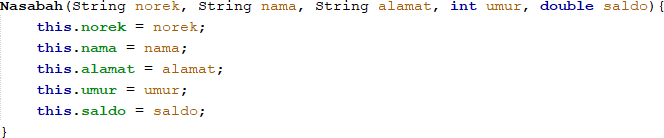
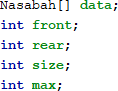
Pada percobaan ini, kita akan membuat program yang mengilustrasikan teller di bank dalam melayani nasabah.

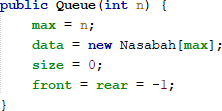
# Langkah-langkah Percobaan

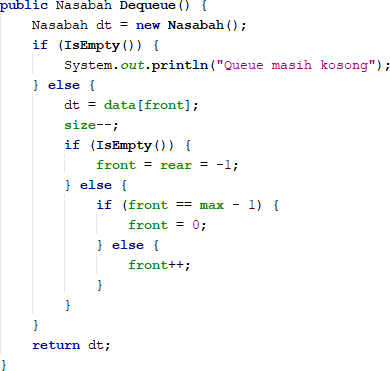
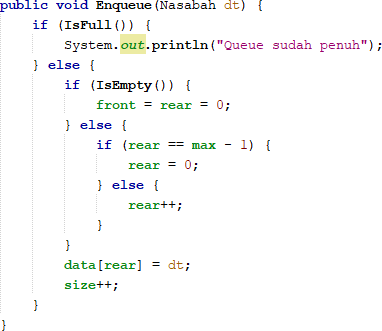
1. Perhatikan Diagram Class berikut ini:

|  |
| --- |
| 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.
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**.

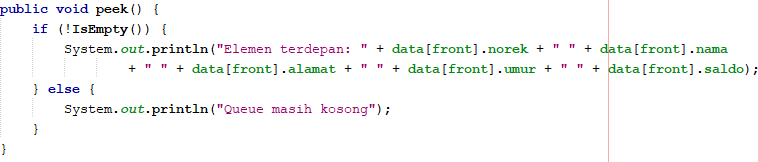


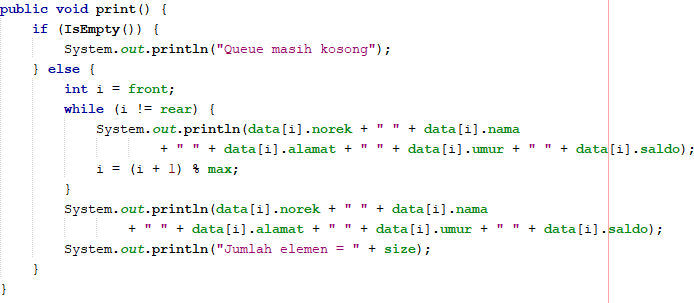


Baris program **Nasabah dt = new Nasabah();** akan ditandai sebagai error, untuk mengatasinya, tambahkan konstruktor default di dalam class Nasabah.

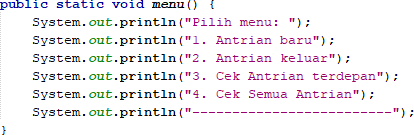


1. Karena satu elemen queue terdiri dari beberapa informasi (norek, nama, alamat, umur, dan saldo), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut, sehingga meodifikasi perlu dilakukan pada method **peek** dan method **print**.

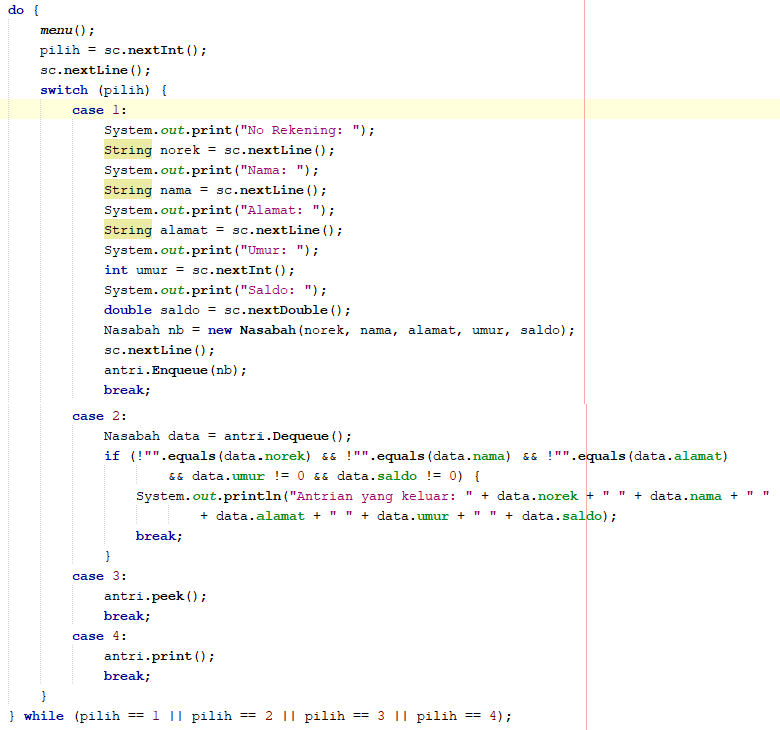




1. Selanjutnya, buat class baru dengan nama **QueueMain** tetap pada package **Praktikum2**. Buat method menu untuk mengakomodasi pilihan menu dari masukan pengguna



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



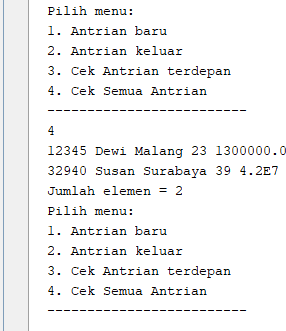
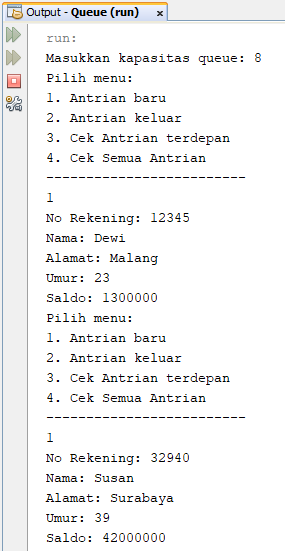
1. Compile dan jalankan class **QueueMain**, kemudian amati hasilnya.

# Verifikasi Hasil Percobaan

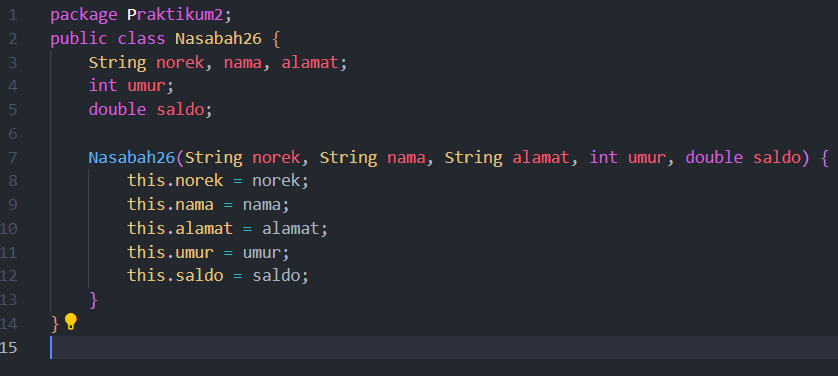
Samakan hasil compile kode program Anda dengan gambar berikut ini.



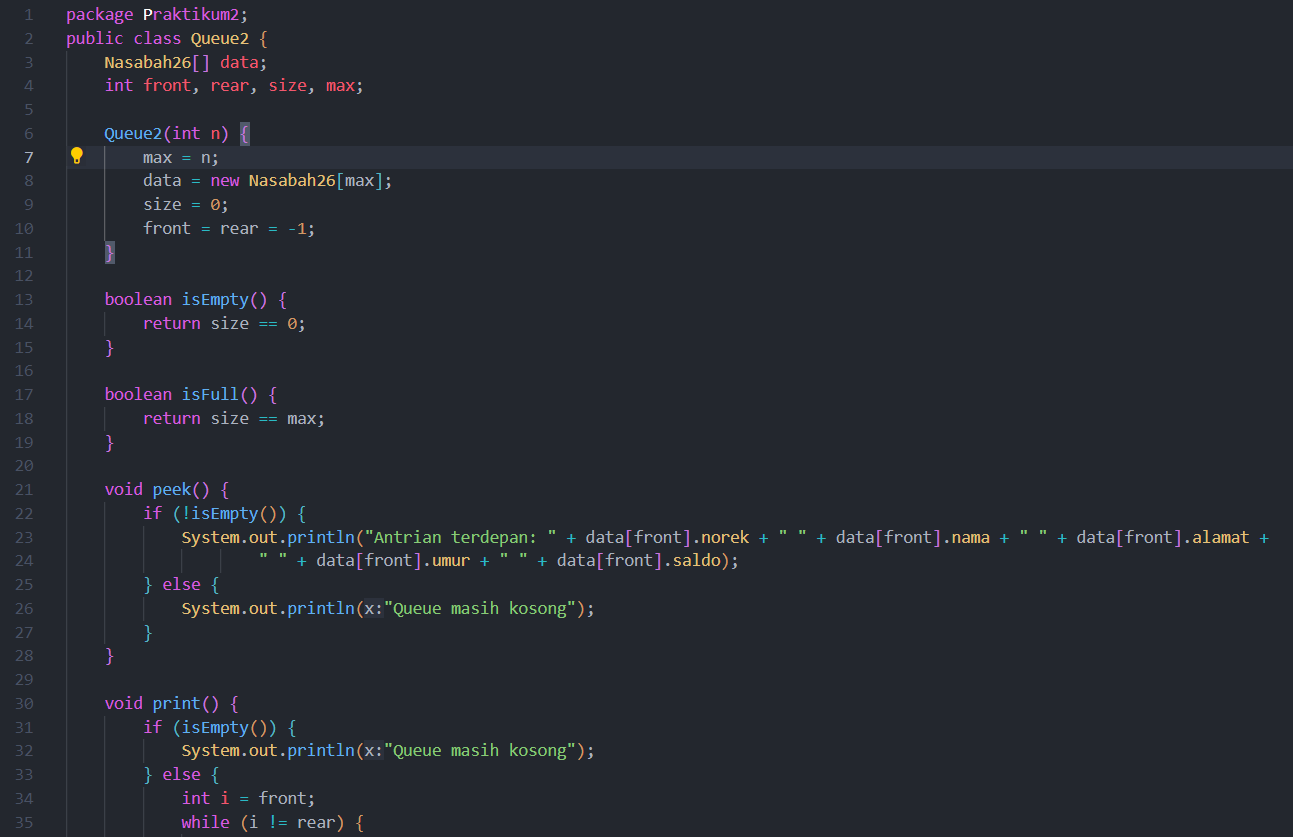
**Algoritma dan Struktur Data 2023-2024**

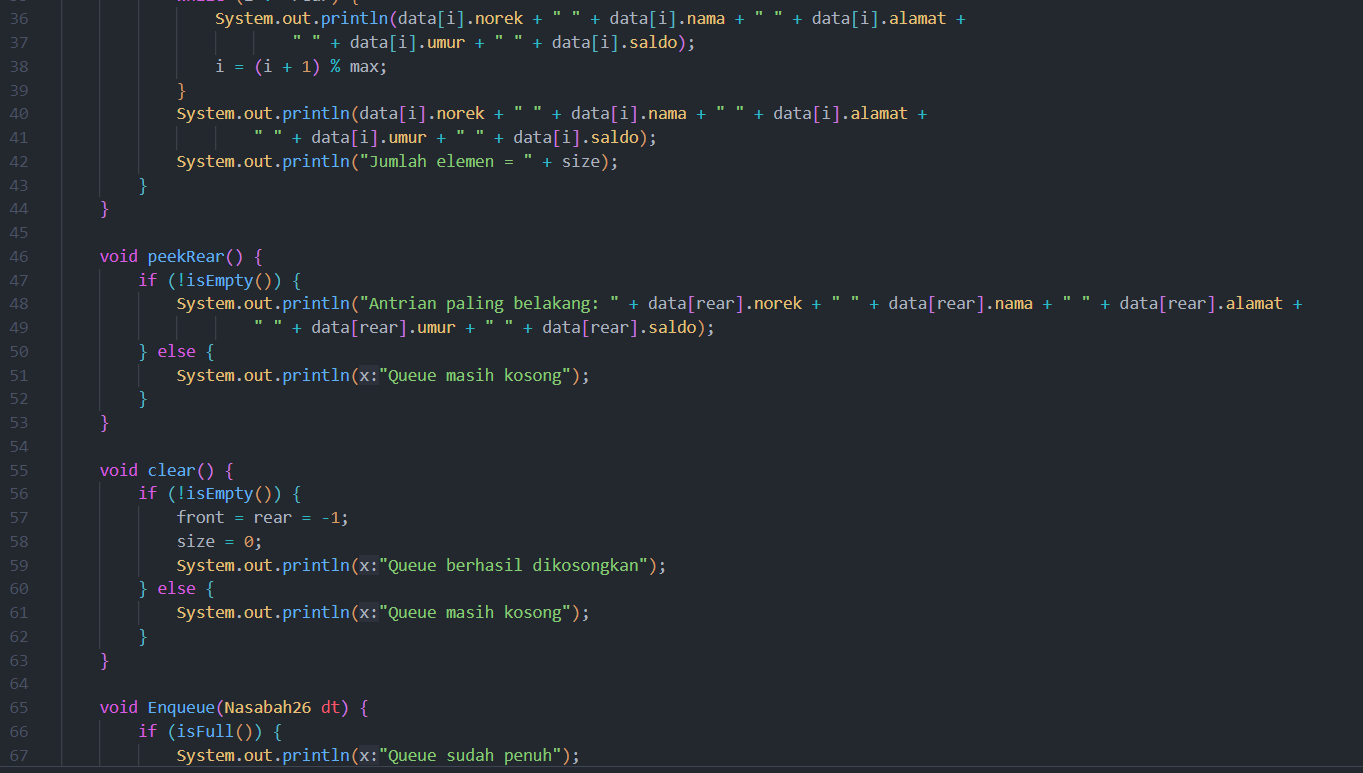


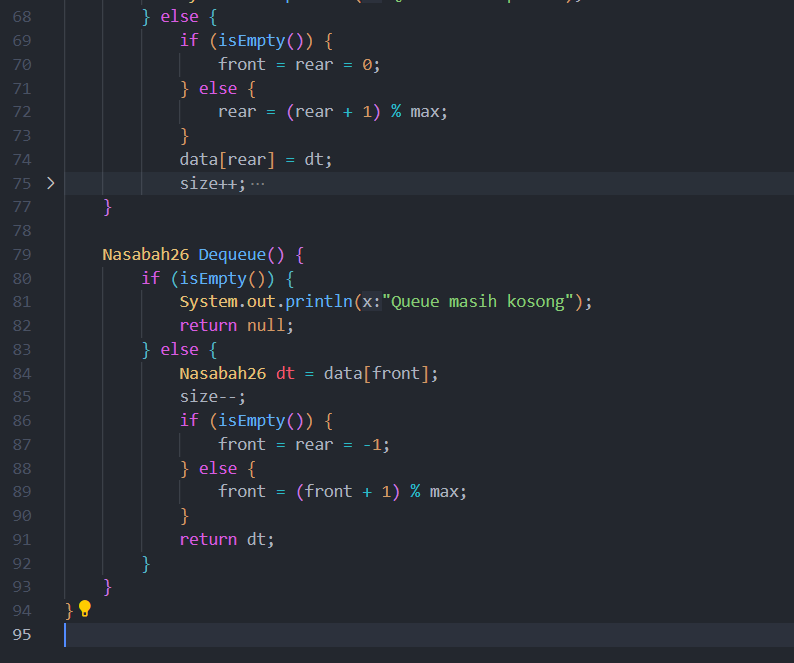
Nasabah Class :

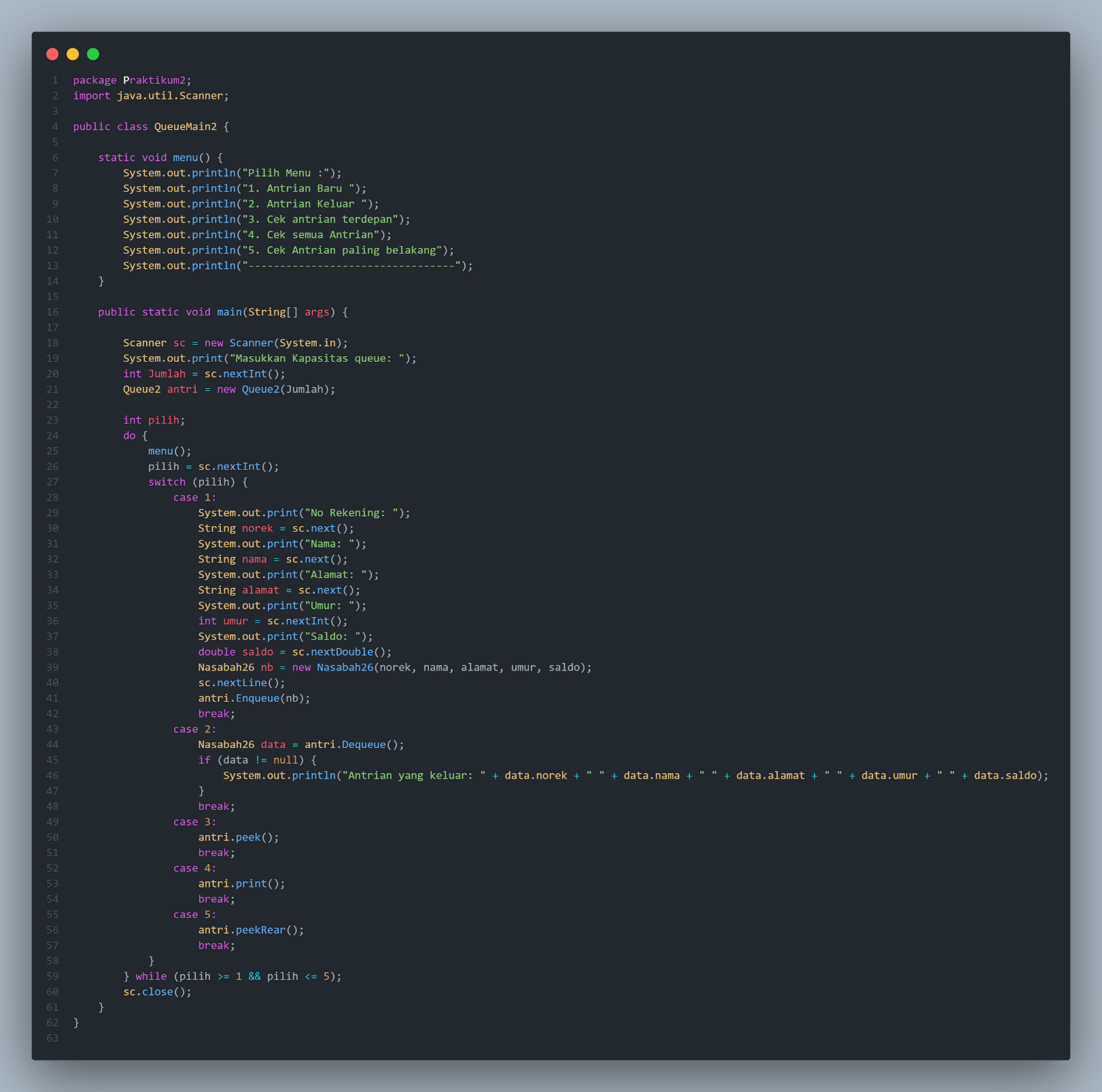


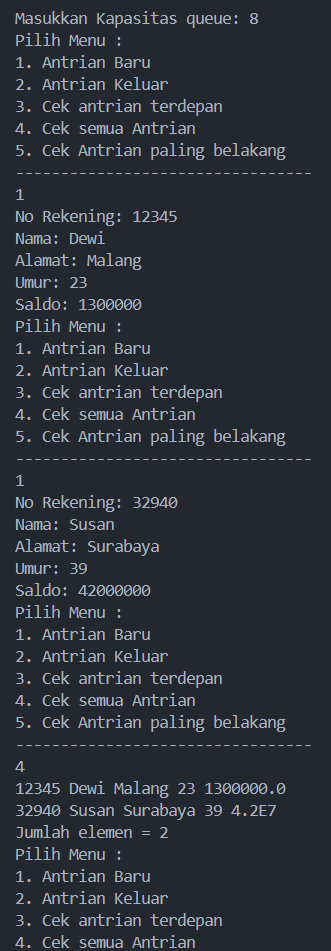
Queue26 Class :



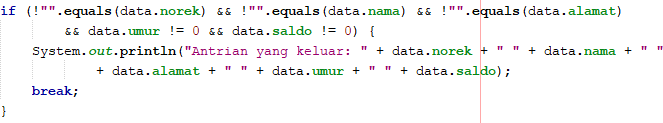




QueueMain26 : 



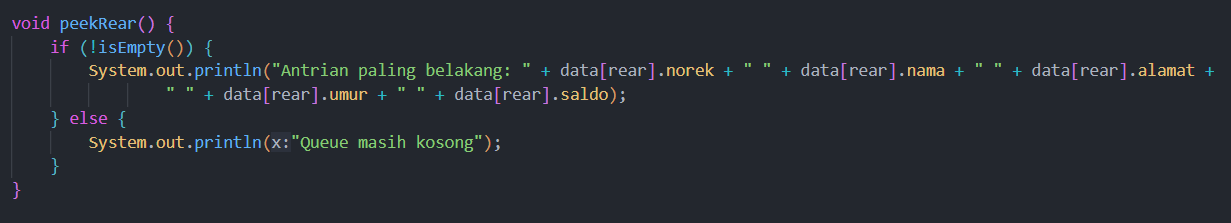
# Pertanyaan

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

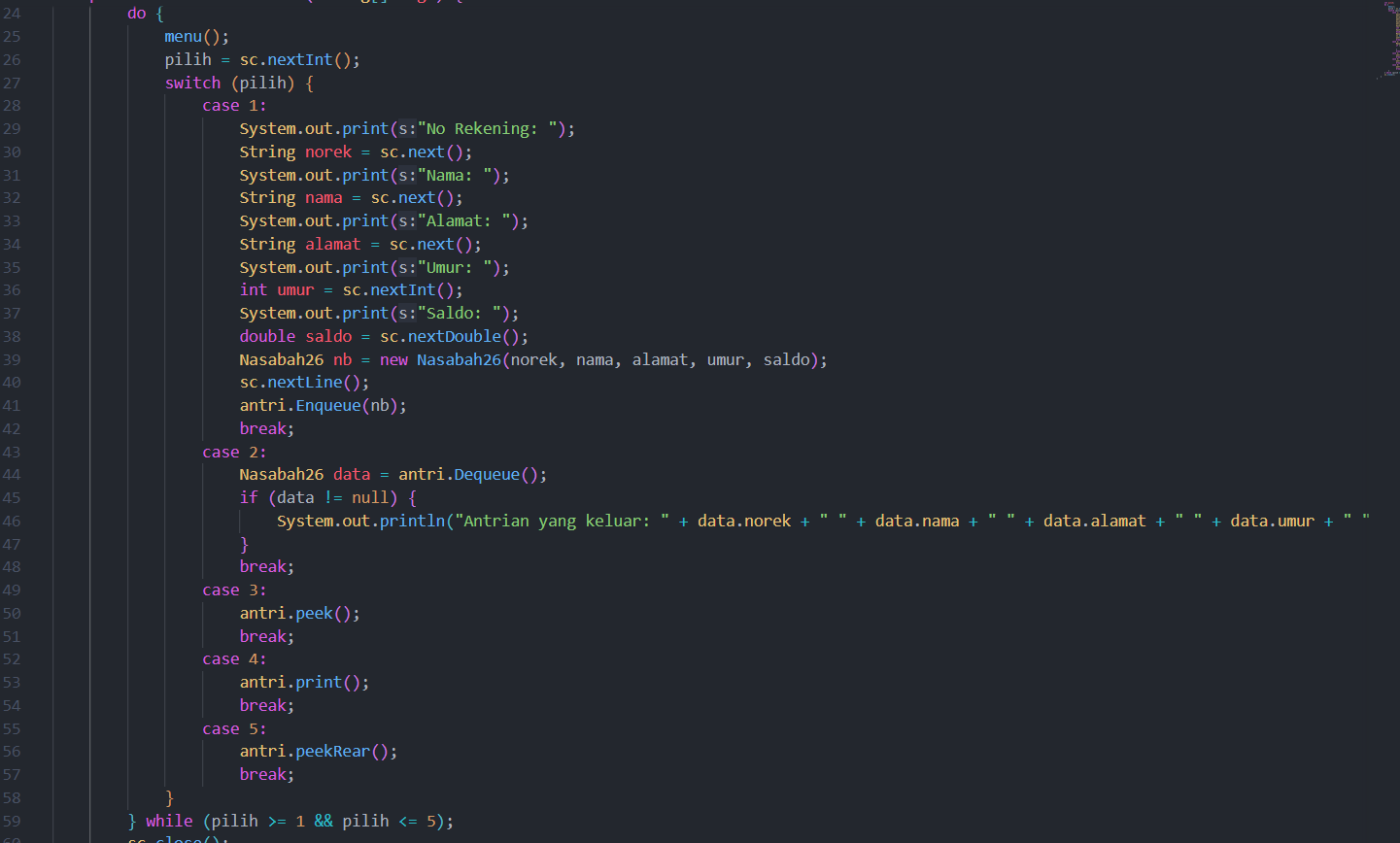
kondisi if tersebut memeriksa apakah semua atribut dari objek data memiliki nilai yang tidak null (untuk string) dan tidak sama dengan 0 (untuk integer dan double). Jika semua kondisi terpenuhi, maka pesan yang berisi informasi tentang nasabah yang keluar akan dicetak.

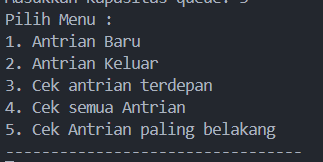
1. Lakukan modifikasi program dengan menambahkan method baru bernama **peekRear** pada class Queue yang digunakan untuk mengecek antrian yang berada di posisi belakang! Tambahkan pula daftar menu **5. Cek Antrian paling belakang** pada class **QueueMain** sehingga method **peekRear** dapat dipanggil!

Queue26 class



QueueMain26 class





# Tugas

1. Buatlah program antrian untuk mengilustasikan pesanan disebuah warung. Ketika seorang pembeli akan mengantri, maka dia harus mendaftarkan nama, dan nomor HP seperti yang digambarkan pada Class diagram berikut:

|  |
| --- |
| Pembeli |
| nama: String  noHP: int |
| Pembeli(nama: String, noHP: int) |

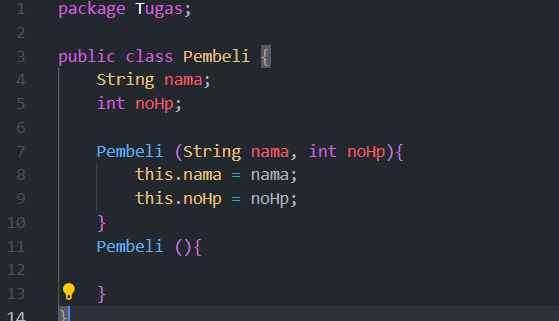
Class diagram Queue digambarkan sebagai berikut:

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

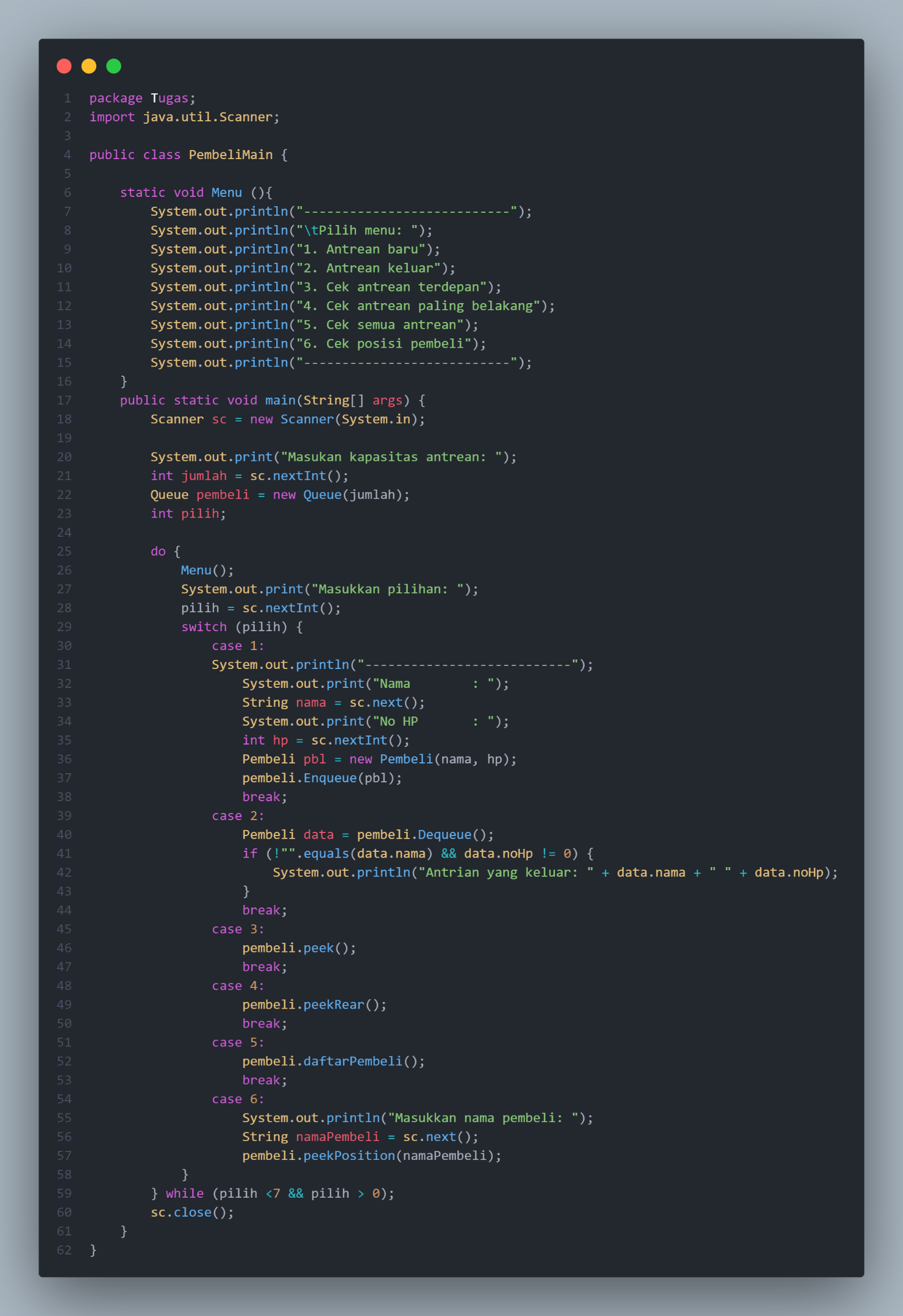
Keterangan:

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

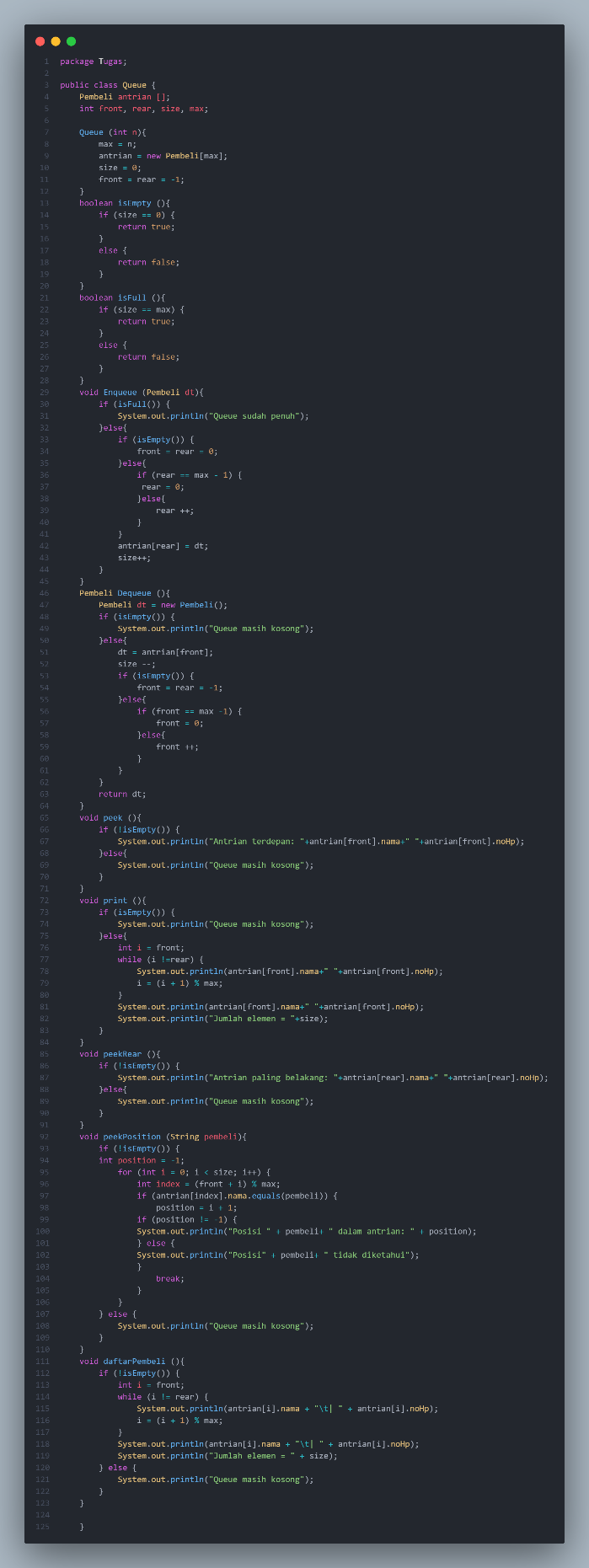
Pembeli Class



PembeliMain Class



Queue Class

****

Hasil :

