**NAMA : INDAH KURNIA ILAHI**

**NIM : H071211001**

**TUGAS : PROGRAM CODE STACK AND QUEUE’**

**PROGRAM CODE STACK**

Stack atau tumpukan adalah sebuah koleksi objek yang menerapkan konsep **LIFO** (*Last In First Out*). Jadi data *terakhir* yang masuk adalah diproses terlebih dahulu. Berkebalikan denga Queue atau antrian yang mana data *yang masuk terlebih* *dahulu* akan diproses terlebih dahulu.

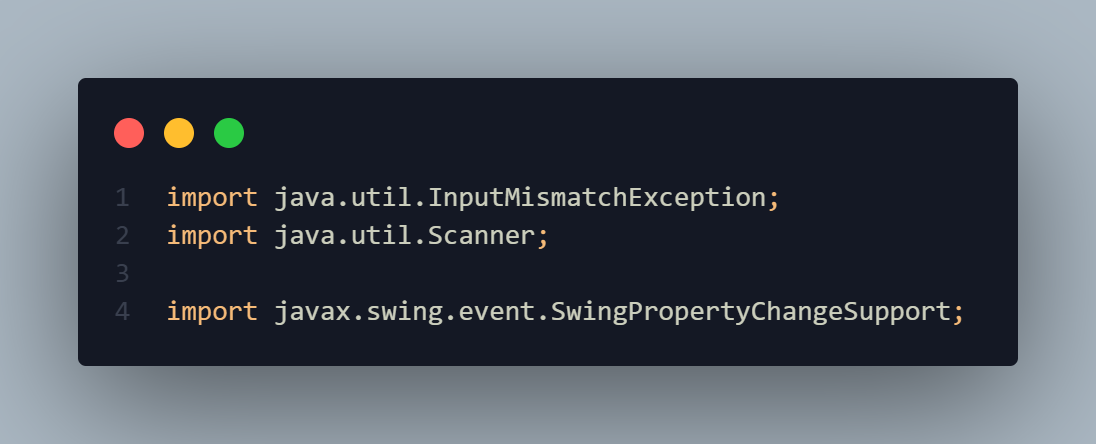
Ada dua operasi dasar dalam stack, yaitu **push** dan **pop**. Operasi **push**  digunakan untuk memasukkan data ke dalam stack sedangkan operasi **pop** digunakan untuk mengeluarkan data dari stack. Stack dapat penuh apabila metode penyimpanan datanya menggunakan array klasik atau array yang didefinisikan panjangnya. Sedangkan apabila menggunakan ArrayList, maka tumpukan bisa lebih besar kapasitas (maksimal hingga nilai terbesar integer).

Code Program di dalam Java :

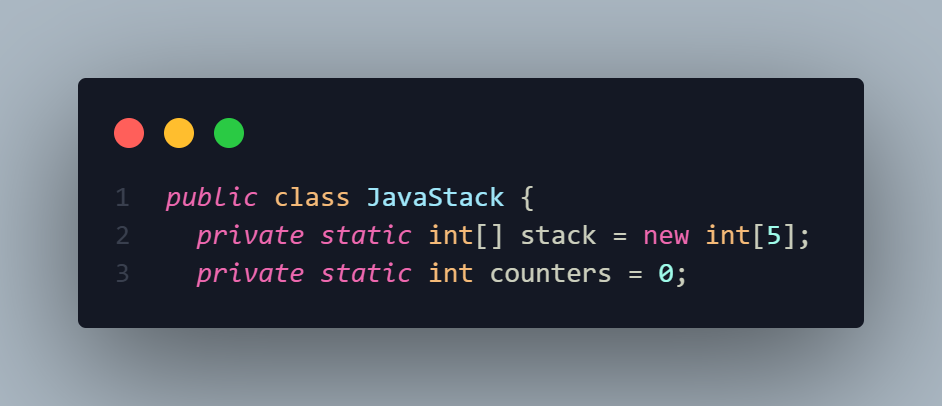
**Pertama**, buatlah class dengan nama JavaStack lengkap dengan main methodnya



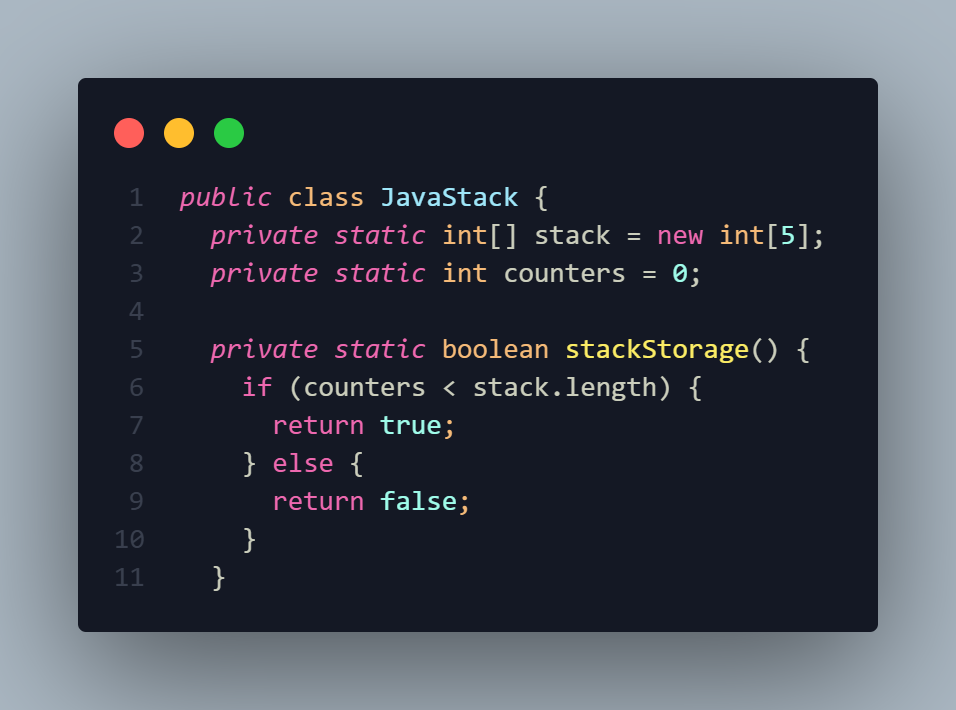
**Kedua**, import.java.util.Scanner dan java.util.InputMismatchException. Letakkan di atas nama class.



**Ketiga**, buatlah dua variable. Variable pertama adalah Integer Array dengan panjang 5 dengan nama *stack*, dan yang kedua adalah variable Integer dengan nama variable *counters* dan dengan nilai awal = 0. Variable Integer Array inilah storage yang digunakan untuk *stack* dan sebagai pointer dan counternya, saya menggunakan variable *counters*. Letakkan kedua variable setelah nama class.



**Keempat**, buat sebuah return value method untuk memeriksa apakah kapasitas stack penuh atau tidak. Method akan mengembalikan nilai TRUE jika masih terdapat tempat untuk menempatkan value dalam stack. Namun, apabila kapasitas stack sudah penuh, maka method akan mengembalikan nilai FALSE.



**Kelima,** buatlah sebuah method void untuk menghandle aksi *push*.



Kita lihat sejenak method diatas. Karena ingin menghandle inputan agar benar-benar terbebas dari error, maka terlebih dahulu pastikan bahwa proses inputan value adalah benar. Jika tidak, maka proses inputan value akan terus diulang hingga memperoleh masukan yang benar, yakni berupa angka. Oleh karena itu, buat variable integer loopX dengan nilai awal 0. Apabila variable loopX ini masih terus menerus bernilai 0, maka proses inputan akan terus diulang.

Kemudian, buat variable local integer dengan nama alpha dan dengan nilai awal 0 sebagai variable untuk menampung masukan.

Masuk ke dalam looping while, sudah dijelaskan tadi bahwa selama nilai loopX adalah 0, maka looping akan terus dilakukan, hingga mendapatkan input yang benar. Oleh karena itu, untuk memastikan bahwa inputannya benar, handle inputan Scanner dengan menggunakan *try catch* . ketika try, dan memperoleh inputan yang benar, maka loopX akan bernilai 1, dan looping selesai. Namun ketika try, ternyata memperoleh inputan yang salah (inputan selain angka), maka selain menampilkan pesan error, looping juga akan terus dilakukan, hingga memperoleh inputan yang benar.

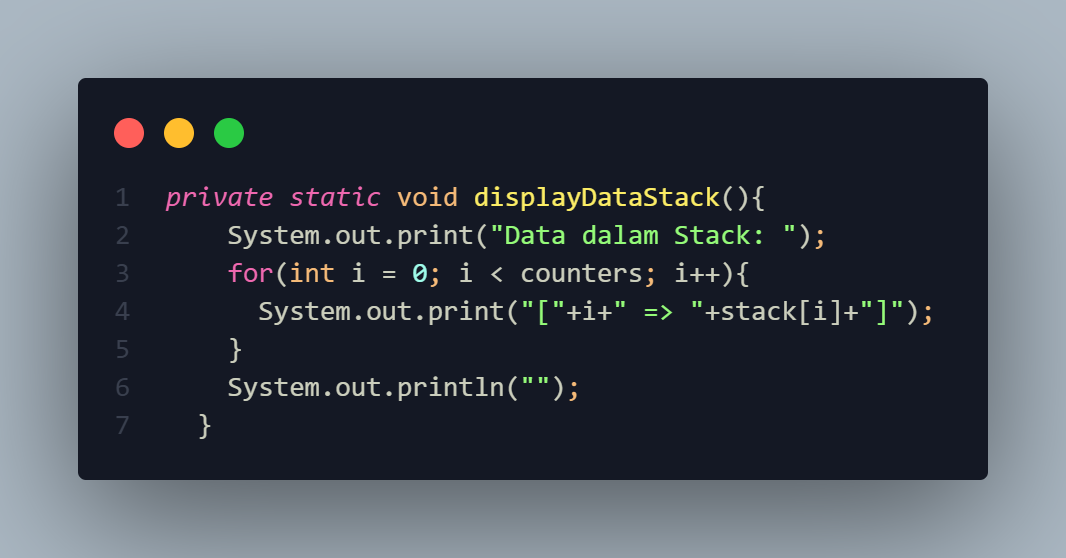
Setelah keluar dari looping, inputan dimasukkan kedalam array stack sebagai stack dan value counters bertambah 1. Yang artinya, sudah ada 1 value di dalam stack. Counters akan terus bertambah hingga batas titik maksimal array stack dapat menampung (5 value).

**Keenam,** buatlah sebuah method void untuk menghandle aksi *pop*.

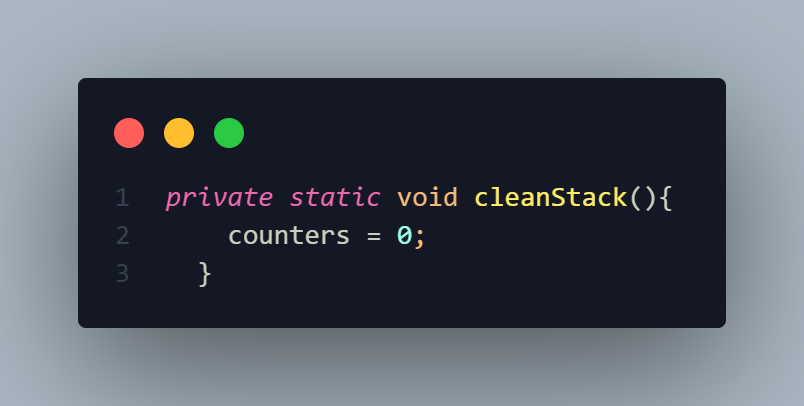


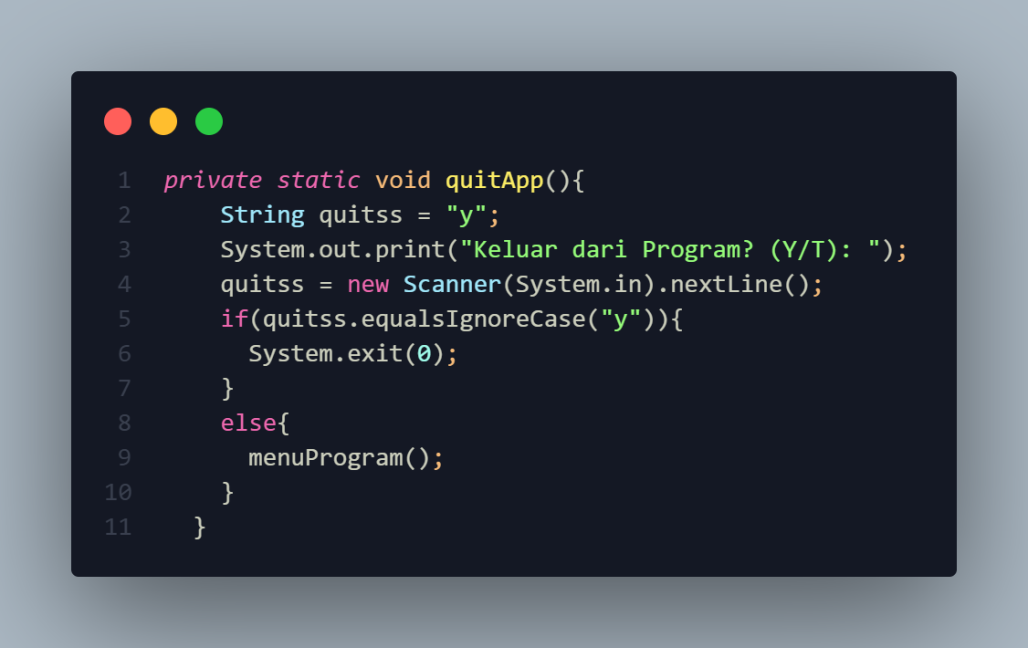
Dalam method tersebut, dapat kita lihat bahwa kita hanya mengurangi counters nya saja. Karena secara default, apabila nilai counters dikurangi, maka akan menunjuk ke elemen array sesuai dengan value counters-nya. Apabila ditambahkan value baru, maka elemen array terakhir yang tidak ditunjuk oleh counters akan diisi dengan value baru tersebut.

**ketujuh,** kita ingin menampilkan data yang tersimpan di dalam stack. Maka buatlah satu method void seperti dibawah ini



**Kedelapan,** kita ingin membuat aksi yang dapat membersihkan stack. Oleh karena itu, buatlah method void men-set counters ke = 0; sehingga dapat menunjuk langsung ke element array ke 0.



**Kesembilan,** buatlah method untuk menghandle fungsi exit

**Kesepuluh**, kita membuat menu untuk menambahkan interaksi antara user dan program



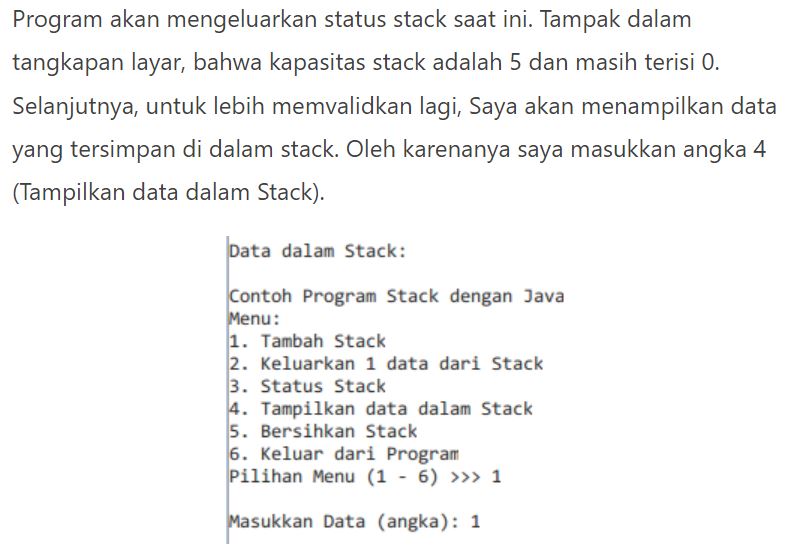
**Kesebelas,** untuk dapat menghandle pemilihan menu, kita buat method dengan fungsi switch didalamnya.

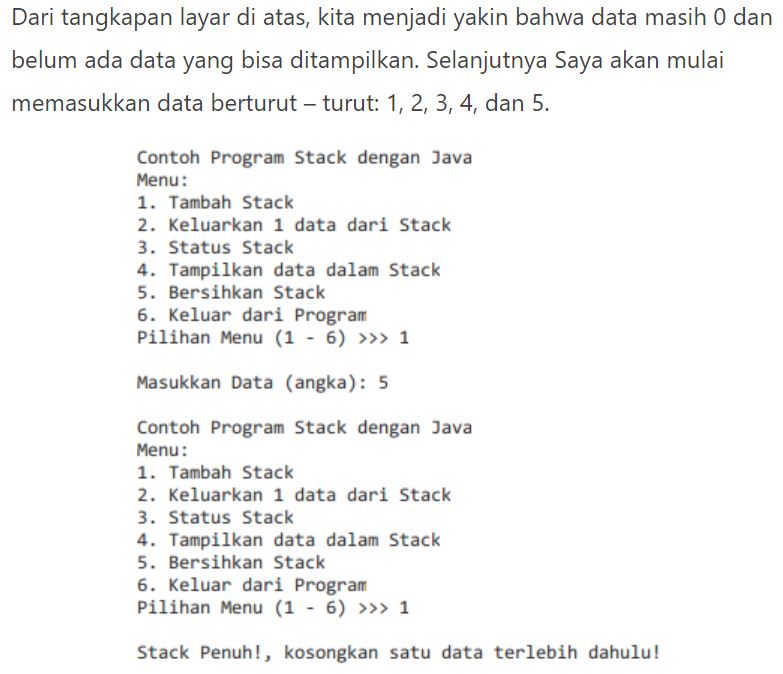


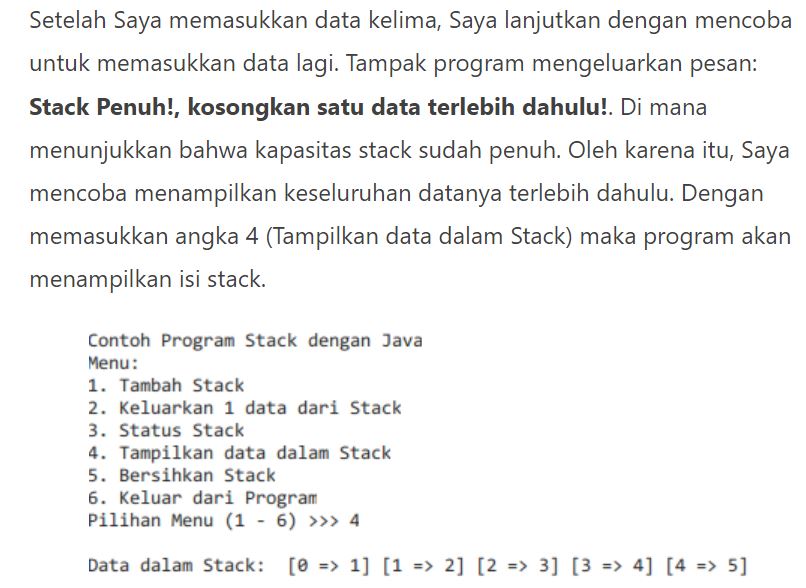
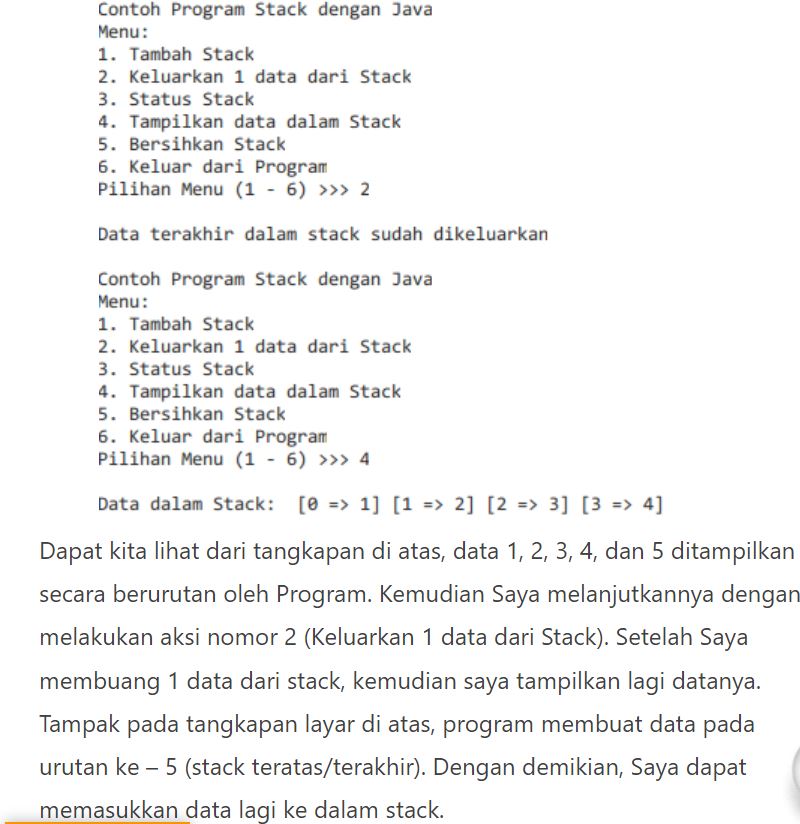
Terakhir, kita panggil method menuProgram(); di Main method

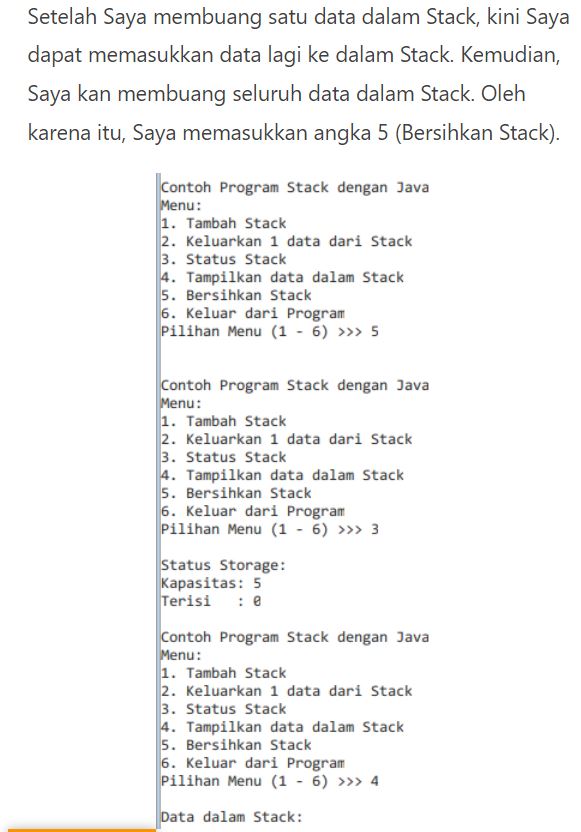


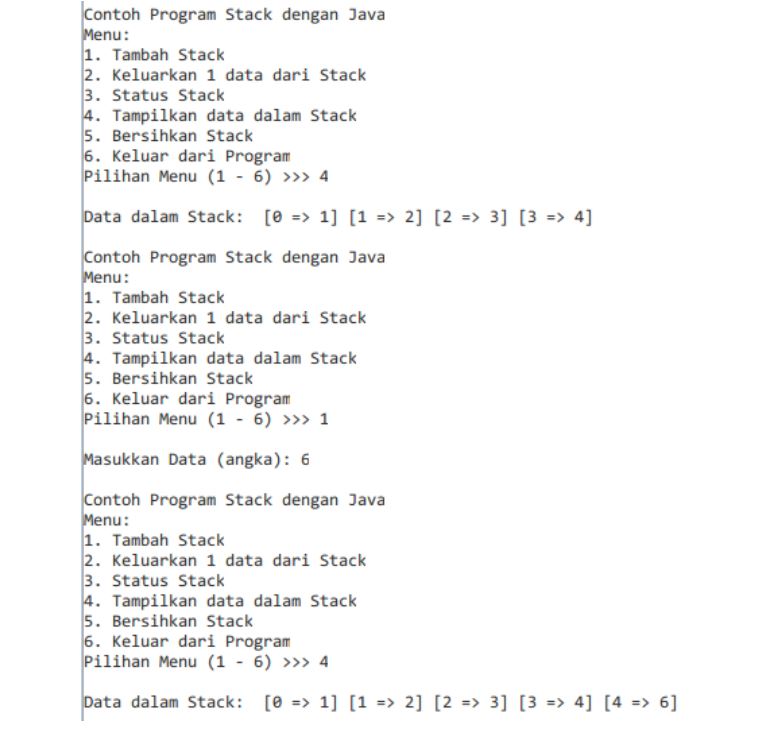
****

****

****

****

****

****

Stack menjadi benar-benar kosong setelah saya mengosongkannya. Sama sekali tidak ada data yang dapat ditampilkan

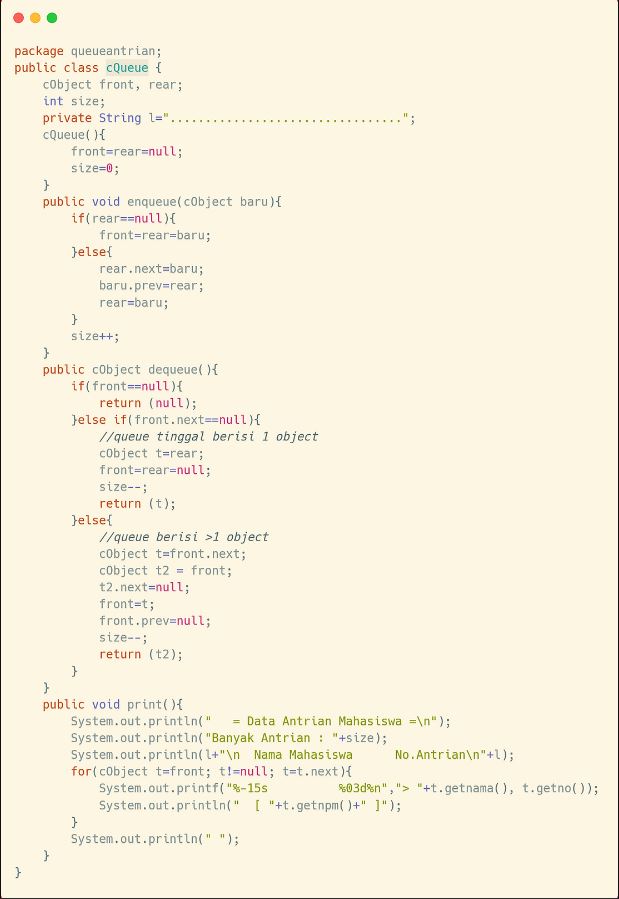
**PROGRAM CODE QUEUE**

Queue (antrian) adalah struktur data dimana data yang pertama kali dimasukkan adalah data yang pertama kali bisa dihapus. Atau bisa juga disebut dengan struktur data yang menggunakan mekanisme FIFO (First In First Out)

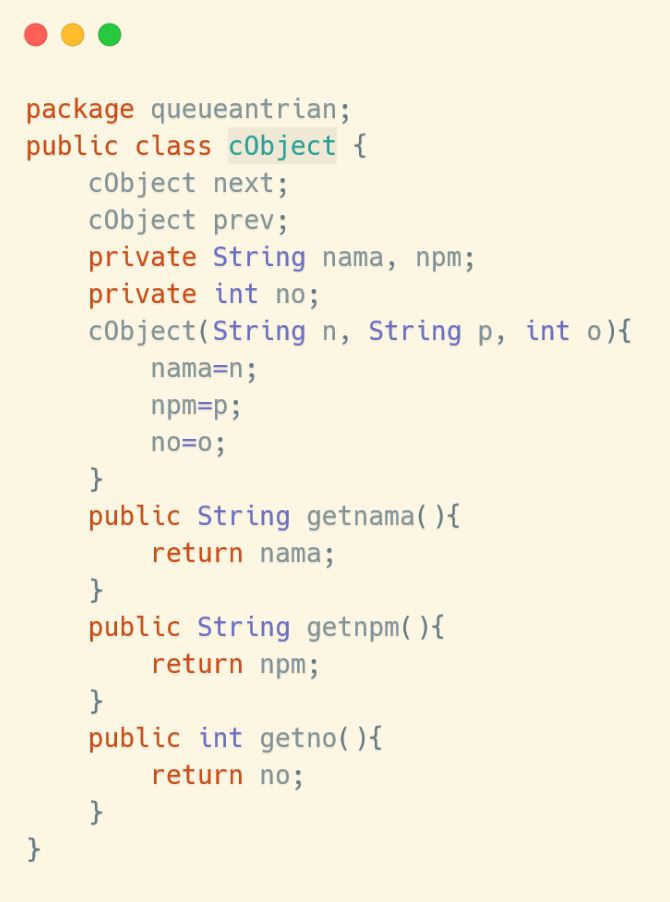
Source Code :

appQueue (Main Class)

cQueue



cObject



Hasil running :

