TUGAS BESAR

ALGORITMA DAN STRUKTUR DATA

Oleh:

Ayudya Nur Rahmadina	(DS-01-01 - 1206210003)
Halim Arif Cahyono	(DS-01-01 - 1206210009)
Angel br Tarigan	(DS-01-01 - 1206210013)
Miranthy Pramita Ningtyas	(DS-01-01 - 1206210015)
Annas Thasya Haafizhah S.A.	(DS-01-01 - 1206210022)

PROGRAM STUDI SAINS DATA

FAKULTAS TEKNOLOGI INFORMASI DAN BISNIS

INSTITUT TEKNOLOGI TELKOM SURABAYA

2022

KATA PENGANTAR

Segala puji dan syukur senantiasa kami hadirkan kepada Tuhan Yang Maha Esa, yang mana telah melimpahkan rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan tugas besar mengenai sistem antrian.

Adapun maksud dan tujuan dari penyusunan laporan ini adalah untuk memenuhi tugas besar mata kuliah Algoritma dan Struktur Data, yang mana laporan ini membahas tentang sistem antrian dengan menggunakan bahasa pemrograman Python. Penulisan laporan ini bertujuan memberikan informasi berdasarkan pengetahuan dan hasil kerja kelompok kami.

Dalam kesempatan ini kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dan bekerjasama dalam membuat laporan ini. Kami menyadari bahwa laporan ini jauh dari kesempurnaan, baik dari cara penulisan maupun isi yang terkandung didalamnya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang bersifat membangun dari pembaca demi kesempurnaan laporan ini. Akhir kata kami berharap semoga laporan ini dapat bermanfaat bagi para pembacanya.

Surabaya, 1 Juli 2022
Penyusun,

Kelompok 4

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pesatnya teknologi, terutama teknologi komputer sudah tak bisa dipungkiri lagi, bagi yang mengikuti perkembangannya, ia tidak akan dipandang sebelah mata. Sebaliknya, bagi yang tidak mengikuti perkembangannya, bersiaplah untuk mundur secara sukarela dari panggung kompetisi. Ibarat wabah, teknologi komputer sudah menyusupi hampir semua bidang kehidupan manusia. Komputer pada dasarnya adalah mesin yang tidak bisa digunakan untuk apa-apa. Maka, komputer seharusnya digabungkan dengan matematika yaitu menggunakan algoritma dan struktur data. Definisi dari algoritma sendiri adalah sederetan langkah-langkah logis yang disusun secara sistematis untuk memecahkan suatu masalah. Sedangkan, definisi struktur data adalah suatu cara penyimpanan, pengorganisasian, dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut digunakan secara efisien. Sehingga, harus digabungkan agar menjadikan komputerisasi yang bergerak dengan baik. Di dalam algoritma dan struktur data banyak materi yang disampaikan, diantaranya ada *Object Oriented Programming* (OOP), *Stack*, dan *Queue* yang akan kami bahas di tugas besar ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang tugas besar di atas maka bisa dirumuskan beberapa masalah berikut ini:

1. Apa yang dimaksud dengan *Object Oriented Programming* (OOP)?
2. Apa yang dimaksud dengan *Stack* dan *Queue*?
3. Bagaimana implementasi dari OOP, *Stack*, dan *Queue* terhadap studi kasus yang telah diberikan yaitu program antrian parkir mobil di area IT Telkom Surabaya, sehingga membantu pengelola untuk mengoptimalkan area parkir dengan membuat sistem antrian?

1.3 Tujuan

Berdasarkan laporan tugas besar ini akan diperoleh tujuan sebagai berikut.

- Memenuhi penilaian tugas besar pada mata kuliah Algoritma dan Struktur Data.
- Memahami pengertian dan kegunaan *Object Oriented Programming* (OOP), *Stack*, dan *Queue*
- Dapat menerapkan *Object Oriented Programming* (OOP), *Stack*, dan *Queue* ke dalam pemrograman yang akan dibahas.

1.4 Manfaat Penelitian

Laporan ini diharapkan dapat memberikan manfaat bagi pihak-pihak yang membutuhkan, baik secara teoritis maupun praktis, diantaranya:

1. Manfaat Teoritis

   laporan ini diharapkan dapat menambah wawasan dan pengetahuan mengenai algoritma dan struktur data tentang *Object Oriented Programming* (OOP), *Stack*, dan *Queue*, serta juga diharapkan sebagai sarana pengembangan ilmu pengetahuan yang secara teoritis dipelajari di bangku perkuliahan.
2. Manfaat Praktis
   A. Bagi penulis laporan ini diharapkan dapat menjadi sarana yang bermanfaat dalam mengimplementasi dan menerapkan *Object Oriented Programming* (OOP), *Stack*, dan *Queue* dalam pemrograman.
   B. Bagi tugas besar selanjutnya laporan ini diharapkan dapat memberikan kontribusi dalam pengembangan teori mengenai pengertian dan penerapan *Object Oriented Programming* (OOP), *Stack*, dan *Queue*.
   C. Bagi kampus laporan ini diharapkan dapat memenuhi nilai mata kuliah algoritma dan struktur data dalam menerapkan *Object Oriented Programming* (OOP), *Stack*, dan *Queue* yang digunakan dalam proses belajar mengajar khususnya Algoritma dan Struktur Data.

1.5 Batasan Masalah

Adapun batasan masalah dari laporan tugas besar ini yaitu mengenai definisi dari fungsi OOP, *Stack*, dan *Queue* serta penggunaan dari fungsi tersebut pada algoritma dari studi kasus antrian parkir mobil, sehingga membantu pengelola untuk mengoptimalkan area parkir dengan membuat sistem antrian.

BAB II

TINJAUAN PUSTAKA

2.1 *Object Oriented Programming* (OOP)

2.1.1 Sejarah OOP

Sejarah perkembangan OOP dimulai pada tahun 1966 saat Ole Johan Dhaldan Kristen Nygaard dari universitas Oslo, norwegia menerbitkan sebuah jurnal kertas kerja dengan judul "SIMULA An Algol Based Language".

Pemrograman Berorientasi Objek (OOP) merupakan metode pemrograman dimana pengembang harus mendefinisikan tipe dari struktur data dan juga tipe dari operasi yang dapat di aplikasikan ke struktur data. dengan demikian struktur data menjadi objek yang dapat memiliki data dan fungsi. beberapa kemampuan utama dari pemrograman OOP antara lain :

1. Pemrograman OOP menekankan pada data dari pada prosedur karena data diperlakukan sebagai elemen yang penting dan tidak boleh mengalir secara bebas dalam program.
2. Data disembunyikan dari akses program oleh fungsi-fungsi (function)eksternal.
3. Program dapat dibagi-bagi ke dalam objek-objek yang lebih kecil.
4. Objek dapat berkomunikasi satu dengan yang lain melalui function.
5. Data baru dan function dapat dengan mudah ditambahkan pada saat dibutuhkan.
6. Konsep pemrogramannya mengikuti pendekatan bottom up.

2.1.2 Pengertian OOP (*Object Oriented Programming*)

Banyak orang pertama kali belajar program menggunakan bahasa yang tidak berorientasi objek. Program non-OOP mungkin salah satu daftar panjang dari perintah. Lebih program yang kompleks akan kelompok daftar perintah ke dalam fungsi atau subrutin masing-masing yang mungkin melakukan tugas tertentu.Dengan desain semacam ini, biasanya untuk data program untuk dapat diakses dari setiap bagian dari program tersebut. Sebagai program tumbuh dalam ukuran, memungkinkan fungsi apapun untuk memodifikasi setiap bagian dari data berarti bahwa bug dapat memiliki dampak yang luas jangkauannya.

Sebaliknya, pendekatan berorientasi objek mendorong para programmer untuk tempat data di mana tidak langsung dapat diakses oleh seluruh program. Sebaliknya data diakses dengan memanggil tertulis fungsi khusus, yang biasa disebut metode, baik yang dibundel dengan data atau warisan dari “objek kelas” dan bertindak sebagai perantara

untuk mengambil atau memodifikasi data tersebut. Pemrograman yang membangun yang menggabungkan data dengan satuset metode untuk mengakses dan mengelola data tersebut disebut objek.

Sebuah program berorientasi objek biasanya akan mengandung berbagai jenis objek, masing-masing jenis yang sesuai untuk jenis tertentu dari data yang kompleks untuk dikelola atau mungkin ke objek dunia nyata atau konsep seperti rekening bank, pemain hoki, atau bulldozer. Sebuah program mungkin berisi beberapa salinan dari setiap jenis objek, satu untuk setiap objek dunia nyata program ini berurusan dengan OOP. Sebagai contoh, ada bisa menjadi salah satu rekening bank untuk setiap account objek dunia nyata di sebuah bank tertentu.Setiap salinan dari objek rekening bank akan sama dalam metode ini menawarkan untuk memanipulasi atau membaca data, tetapi data dalam setiap objek akan berbeda mencerminkan sejarah yang berbeda dari setiap account.

Objek dapat dianggap sebagai pembungkus data mereka dalam satu set fungsi yang dirancang untuk memastikan bahwa data yang digunakan tepat, dan untuk membantu dalam menggunakan. Metode ini objek biasanya akan mencakup pemeriksaan dan perlindungan yang khusus untuk jenis data objek berisi. Sebuah objek juga dapat menawarkan sederhana digunakan, metode standar untuk melakukan operasi tertentu pada data, sementara menyembunyikan secara spesifik tentang bagaimana tugas-tugas yang dicapai. Dengan cara ini perubahan dapat dibuat dengan struktur internal atau metode objek tanpa memerlukan bahwa sisa program dimodifikasi.

Pendekatan ini juga dapat digunakan untuk menawarkan metode standar di berbagai jenis objek. Sebagai contoh, beberapa jenis benda mungkin menawarkan metode cetak. Setiap jenis objek yang mungkin menerapkan metode cetak dalam cara yang berbeda, yang mencerminkan jenis data yang berbeda masing-masing berisi, tetapi semua metode cetak yang berbeda mungkin disebut dengan cara standar yang sama dari tempat lain di program ini.Fitur-fitur ini menjadi berguna terutama ketika lebih dari satu programmer berkontribusi kode untuk proyek atau ketika tujuannya adalah untuk menggunakan kembali kode di antara proyek.Pemrograman berorientasi objek memiliki akar yang dapat ditelusuri ke tahun 1960-an. Sebagai perangkat keras dan software menjadi semakin kompleks, pengelolaan sering menjadi perhatian. Para peneliti mempelajari cara untuk menjaga kualitas software dan pemrograman berorientasi objek yangdikembangkan sebagian untuk mengatasi masalah-masalah umum dengan sangat menekankan diskrit, unit dapat digunakan kembali logika. Teknologi ini berfokus pada data daripada proses, dengan program yang terdiri dari modul mandiri(kelas), setiap contoh (objek) yang berisi semua informasi yang dibutuhkan untuk memanipulasi data struktur sendiri (anggota).

Hal ini berbeda dengan yang ada pemrograman modular yang telah dominan selama bertahun-tahun yang difokuskan pada fungsi dari sebuah modul, bukan data spesifik, tetapi juga disediakan untuk penggunaan kembali kode, dan cukup dapat

digunakan kembali unit-diri dari logika pemrograman, memungkinkan kolaborasi melalui penggunaan modul terkait(subrutin). Pendekatan yang lebih konvensional, yang masih tetap, cenderung untuk mempertimbangkan data dan perilaku secara terpisah. Program berorientasi objek dengan demikian dapat dilihat sebagai kumpulan objek berinteraksi, yang bertentangan dengan model konvensional, di mana program dipandang sebagai daftar tugas(subrutin) untuk melakukan. Dalam OOP,setiap objek dapat menerima pesan, pengolahan data, dan mengirim pesan ke objek lainnya. Setiap objek dapat dilihat sebagai "mesin" independen dengan peran yang berbeda atau tanggung jawab. Tindakan(metode) pada objek-objek yang terkait erat dengan objek. Sebagai contoh, OOP struktur data cenderung membawa operator sendiri main dengan mereka (atau setidaknya mewarisi mereka dari objek yang sama atau kelas).

2.2 *Stack* dan *Queue*

2.2.1 Pengertian *Stack*

stack adalah salah satu struktur data yang digunakan untuk menyimpan sekumpulan objek ataupun variabel. Sesuai namanya yaitu *stack*, tidak heran apabila objek yang terkumpul terlihat seperti tumpukan. Karakteristik *stack* sendiri bersifat LIFO (*last in first out)*. Artinya, data yang terakhir masuk merupakan data yang akan keluar terlebih dahulu.

Stack dibuat dari sebuah list yang dibatasi sifat-sifat dan aksi /operasi yang dilakukan pada list tersebut. Aksi atau operasi yang bisa dilakukan dengan Stack:

- isEmpty(), untuk mengecek apakah *Stack* sedang kosong atau tidak.
- push(x), untuk melakukan insert data x ke *Stack.*
- pop(), untuk melakukan delete 1 data di *Stack* yang berada di paling atas.
- peek(), untuk mengintip atau melihat data di *Stack* yang berada di paling atas.
- size(), untuk melihat jumlah data atau ukuran dari sebuah *Stack.*

Stack biasa digunakan dalam mengontrol operasi dalam sebuah sistem operasi. Selain itu *stack* juga merupakan algoritma yang baik yang dapat digunakan untuk membuat phaser (membaca urutan operasi dari sebuah persamaan matematika).

2.2.2 Pengertian *Queue*

Queue adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung yang disebut sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain. Contoh queue paling sederhana dapat dilihat pada antrian. Prinsip kerja dari queue adalah prinsip "*First In First Out*" (FIFO) atau "masuk pertama keluar pertama". Sedangkan prinsip "masuk terakhir

keluar pertama" atau "*Last In First Out*" (LIFO), digunakan pada tumpukan atau stack. Pada queue terdapat satu pintu masuk di salah satu ujung dan satu pintu keluar di ujung lainnya. Maka ada petunjuk yang menunjukkan awal dan akhir.

Operasi-Operasi Pada *Queue*:

- Create, untuk menciptakan dan menginisialisasi *Queue* dengan cara membuat Head dan Tail = -1
- IsEmpty, dipakai untuk memeriksa penuh tidaknya sebuah antrian dengan cara memeriksa nilai tail, jika tail = -1 maka empty. Kita tidak memeriksa head, karena head adalah tanda kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah. Pergerakan pada antrian terjadi dengan penamb ahan elemen antrian di bagian belakang, yaitu menggunakan nilai tail.
- IsFull, dipakai untuk mengecek penuh tidaknya antrian dengan cara mengecek nilai tail, jika tail >= MAX-1 (karena MAX-1 adalah batas elemen array pada C) maka sudah penuh.
- EnQueue, digunakan untuk penambahan elemen ke dalam antrian, penambahan elemen selalu ditambahkan di elemen paling belakang. Penambahan elemen selalu menggerakan variabel tail dengan cara increment counter tail terlebih dahulu
- DeQueue, dipakai untuk menghapus elemen terdepan (head) dari queue dengan cara menggeser semua elemen antrian ke bagian depan dan mengurangi tail dengan 1 penggusuran yang dilakukan dengan menggunakan pengulangan.
- Clear, digunakan untuk menghapus elemen-elemen antrian dengan cara membuat tail dan head bernilai -1. Penghapusan elemen-elemen antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesannya menjadi -1 sehingga elemen-elemen antrian tidak lagi terbaca.
- Tampil, dipakai untuk menampilkan nilai-nilai elemen antrian menggunakan pengulangan dari head hingga tail

Operasi-Operasi Queue dengan Linear Array

- IsEmpty, fungsi IsEmpty berguna untuk mengecek apakah queue sudah terisi atau masih kosong. hal ini dilakukan dengan mengecek apakah tail bernilai -1 atau tidak. Nilai -1 menandakan bahwa queue masih kosong.
- IsFull, fungsi IsFull berguna untuk mengecek apakah queue sudah penuh atau masih bias menampung data dengan cara mengecek apakah nilai tail sudah sama dengan jumlah maksimal queue. Jika nilai keduanya sama, berarti antrian sudah penuh.

- EnQueue, fungsi EnQueue berguna untuk memasukkan sebuah elemen dalam sebuah antrian.
- DeQueue, fungsi DeQueue berfungsi untuk mengambil sebuah elemen dari antrian. Operasi ini sering disebut juga serve. Hal ini dilakukan dengan cara memindahkan sejauh satu langkah ke posisi di depannya sehingga otomatis elemen yang paling depan akan tertimpa dengan elemen yang terletak di belakangnya.
- Clear, fungsi Clear berguna untuk menghapus semua elemen dalam queue dengan jalan mengeluarkan semua elemen tersebut satu per satu hingga queue kosong dengan memanfaatkan fungsi DeQueue.

BAB III

METODOLOGI PENELITIAN

3.1 Sumber Data

Sumber data merupakan tempat mendapatkan data yang diinginkan. Sumber data sangat penting untuk diketahui, supaya dapat menghindari kesalahan dalam pemilihan data yang sesuai dengan tujuan penelitian ini. Seperti halnya dalam penelitian ini untuk mengetahui fungsi OOP, *Stack*, dan *Queue* pada suatu algoritma dari studi kasus 1 yaitu antrian parkir mobil, sumber data ini menggunakan pengumpulan data dari jurnal dan artikel terpercaya.

3.2 Langkah Analisis

Pada lembar kerja mahasiswa ini akan dilakukan beberapa langkah analisis meliputi latar belakang penelitian, rumusan masalah, implementasi dari fungsi OOP, *Stack*, dan *Queue* dalam pengerjaan studi kasus antrian parkir mobil, kemudian menganalisis hasil pengerjaan mengenai studi kasus tersebut.

3.3 Alur Penelitian

BAB IV

ANALISIS DAN PEMBAHASAN

Soal

Permasalahan 1

Buatlah sebuah program untuk antrian parkir mobil di area IT Telkom Surabaya. Karena parkiran di Kampus sangat terbatas, Anda membantu pengelola untuk mengoptimalkan area parkir mobil dengan membuat system antrian mobil. Menu pada parkiran tersebut adalah sebagai berikut:

1. Antrian Parkir Mobil, merupakan menu yang digunakan untuk memasukkan nomor-nomor plat mobil yang akan diparkir. Hanya bisa menerima inputan berupa angka saja dan akan melakukan pengecekan apabila ada nomor plat yang sama, maka akan memberikan peringatan.
2. Keluarkan Mobil, merupakan menu yang digunakan untuk memilih plat nomor mobil yang akan dikeluarkan. Apabila mobil yang dipilih berada pada antrian ke 3, maka lakukan kondisi bahwa mobil ke 1 dan ke 2 dikeluarkan terlebih dahulu. Namun apabila plat nomor yang akan dikeluarkan berada pada posisi antrian paling depan, maka mobil langsung dikeluarkan.
3. Tampilkan Antrian Mobil, merupakan menu yang digunakan untuk menampikan isi antrian parkiran.
4. Exit, menu yang digunakan untuk keluar dari program.

Pembahasan:

Input

```
class queue:
    def __init__(self):
        #self.antrian = queue.antrian
        self.list_mobil = []
        self.antrian = 0
```

Langkah pertama membuat class yang mendeklarasikan variabel dengan fungsi constructor yaitu (__init__). Selanjutnya, menggunakan keyword self dimasing-masing string yang akan digunakan yaitu list_mobil dan antrian. Selanjutnya, mendefinisikan 4 method yaitu :

```
def tambahAntri(self, plat_mobil):
    if self.isDuplicate(plat_mobil):
        print('plat ini sudah parkir sebelumnya') # peringatan
    else:
        self.antrian += 1
        self.list_mobil.append(plat_mobil)


    print("\nTambah Antrian Berhasil")
    print("-----------------------")
    print('| antrian | ', '    plat_mobil      |')
    print("-----------------------")

    self.daftarAntri()
    for mobil in self.list_mobil:
         print('|', self.antrian, '|', mobil.plat_mobil, '|')
```

Membuat method pertama yaitu tambahAntri, setiap mendefinisikan sebuah method, parameter keyword self harus selalu ada dan ditambahkan parameter plat_mobil. Selanjutnya, membuat kondisi if jika plat mobil yang diinputkan sama maka akan mengeluarkan ‘plat ini sudah parkir sebelumnya’ dan kondisi else jika plat mobil tidak sama dan maka ditambahkan dalam list_mobil tersebut serta akan mengeluarkan ‘tambah antrian berhasil’ dan list antrian.

```
def isDuplicate(self, plat_mobil):
    for plat in self.list_mobil:
        if plat == plat_mobil:
            return True
        else:
            return False
```

Membuat method kedua yaitu isDuplicate, setiap mendefinisikan sebuah method, parameter keyword self harus selalu ada dan ditambahkan parameter plat_mobil. Selanjutnya membuat perulangan for, di dalam perulangan for ada kondisi if jika plat yang diinputkan sama dengan plat mobil akan berhasil dan kondisi else jika plat yang diinputkan tidak sama dengan plat mobil maka akan salah.

```
def daftarAntri(self):
    print("\nDaftar Antrian mobil")
    print("----------------------")
    print('| NO | ', 'plat_mobil  |')
    print("----------------------")

    for index, mobil in enumerate(self.list_mobil):
        print(f'| {index + 1} | {mobil} |')
        print('|', self.antrian, '|', mobil , '|')
```

Membuat method ketiga yaitu daftarAntri, setiap mendefinisikan sebuah method, parameter keyword self harus selalu ada. Selanjutnya, membuat print list/daftar antrian mobil. Lalu, membuat perulangan for untuk mengisi list tersebut.

```
    def hapusantrian(self):
        list_mobil.pop(0)
        print("\nDaftar Antrian mobil")
        print("----------------------")
        print('| NO | ', 'plat_mobil  |')
        print("----------------------")

        for mobil in list_mobil:
             print('|', mobil.antrian, '|', mobil.plat_mobil, '|')
```

Membuat method keempat yaitu hapusAntrian, setiap mendefinisikan sebuah method, parameter keyword self harus selalu ada. Di Dalam method ini berisi pop yaitu berfungsi mengembalikan nilai elemen. Selanjutnya, membuat print list/daftar antrian mobil. Lalu, membuat perulangan for untuk mengisi list tersebut bahwa fungsi constructornya sudah berhasil.

```
def start():
    return True

parkir = queue()

while (True):
    print('\n========================\n')
    print('Menu:')
    print('1. Tambah antrian')
    print('2. Lihat antrian sekarang')
    print('3. hapus antrian')
    print('4. keluar')

    pilih = int(input("Masukan pilihan: "))
```

Langkah kedua, membuat fungsi start yang dan syntax while yang berisi print menu yang akan dijalankan serta membuat variabel pilih yang berisi inputan untuk memilih menu tersebut.

```
if pilih == 1:
    plat_mobil = int(input("\nMasukkan plat_mobil lengkap: "))
    parkir.tambahAntri(plat_mobil)

elif pilih == 2:
     daftarAntri()

elif pilih == 3:
     hapusantrian()

elif pilih == 4:
    print("\nArigatou Gozaimasu ^-^")
    break
```

Langkah ketiga,

- Jika pemilihan menu tadi memilih angka 1 maka akan mengeluarkan kondisi if yang berisi untuk menambahkan antrian mobil ke dalam list.
- Jika memilih angka 2 maka akan menggunakan kondisi elif yang menghasilkan list/daftar antrian.
- Jika memilih angka 3 maka akan menggunakan kondisi elif yang berisi penghapusan plat antrian.
- Jika memilih angka 4 maka akan menggunakan kondisi elif yang menghasilkan kata terimakasih ‘ Arigatou Gozaimasu ‘

Output:

```
=========================

Menu:
1. Tambah antrian
2. Lihat antrian sekarang
3. hapus antrian
4. keluar
Masukan pilihan: 1

Masukkan plat_mobil lengkap: 123

Tambah Antrian Berhasil
----------------------
| antrian |      plat_mobil      |
----------------------

Daftar Antrian mobil
---------------------
| NO |  plat_mobil  |
---------------------
| 1 | 123 |

=========================

Menu:
1. Tambah antrian
2. Lihat antrian sekarang
3. hapus antrian
4. keluar
Masukan pilihan: 1

Masukkan plat_mobil lengkap: 123
duplikat

Tambah Antrian Berhasil
----------------------
| antrian |      plat_mobil      |
----------------------


Tambah Antrian Berhasil
----------------------
| antrian |      plat_mobil      |
----------------------

Daftar Antrian mobil
---------------------
| NO |  plat_mobil  |
---------------------
| 1 | 123 |
| 2 | 234 |
```

```
=========================

Menu:
1. Tambah antrian
2. Lihat antrian sekarang
3. hapus antrian
4. keluar
Masukan pilihan: 2

Daftar Antrian mobil
----------------------
| NO |  plat_mobil  |
----------------------
| 1 | 123 |
| 2 | 234 |

=========================

Menu:
1. Tambah antrian
2. Lihat antrian sekarang
3. hapus antrian
4. keluar
Masukan pilihan: 3

masukkan nomor antrian mobil yang akan dikeluarkan = 1

mobil telah dikeluarkan
=========================

Menu:
1. Tambah antrian
2. Lihat antrian sekarang
3. hapus antrian
4. keluar
Masukan pilihan: 4

Arigatou Gozaimasu ^-^
```

BAB V

PENUTUP

5.1. Kesimpulan

Program Berorientasi Objek atau dalam bahasa Inggris dikenal dengan OOP *(Object Oriented Programming*) merupakan bahasa pemrograman komputer untuk menyelesaikan masalah program dengan menyediakan objek-objek (terdiri dari beberapa attribute dan method) yang saling berkaitan dan disusun kedalam satu kelompok yang disebut dengan class. OOP adalah paradigma pemrograman yang cukup dominan saat ini, karena mampu memberi solusi kaidah pemrograman modern.

Stack adalah list yang operasi penyisipan dan penghapusan elemennya dilakukan di satu ujung datau bisa juga disebut tumpukan. Stack dapat dideklarasikan dengan sebuah record uang mempunyai elemen sebuah array data untuk menyimpan elemen stacak dan sebuah variabel top untuk menunjuk elemen stack teratas (top element). Operasi pada stack adalah *isEmpty, push, pop, peek dan size*. *Queue* adalah sekumpulan data yang beroperasi dengan cara *First In First Out* (FIFO) yang dimana barisan pertama yang masuk merupakan elemen yang pertama keluar. Pada Queue ada operasi-operasi dasar, yaitu; *Create, IsEmpty, IsFull, EnQueue, DeQueue, Clear*, Tampil. Hasil Analisis Antrian Mobil menggunakan methode dari OOP.

5.2. Saran

Ketika menggunakan OOP tidak dapat menerapkan di semua bahasa pemrograman karena ini bukan bahasa universal. Tetapi, Bahasa OOP (*Object Oriented Programming*) memungkinkan kamu memecah program menjadi program berukuran bit yang dapat diselesaikan dengan mudah. Jika, *stack* dan *queue* memori terbatas tetapi mengontrol memorinya secara sendiri.

DAFTAR PUSTAKA

https://glints.com/id/lowongan/stack-adalah/#.Ys2Jd3Iza3A

Danuri, M.

https://amikjtc.com/jurnal/index.php/jurnal/article/view/35

Marzuqi, A.

https://www.academia.edu/29852283/MAKALAH_STRUKTUR_DATA