**LAPORAN TUGAS STRUKTUR DATA**

**LINKED LIST – MIEXUE**



Disusun oleh:

1. Gerry Moeis M.D.P (23091397164)
2. Ahmad Aryobimo (23091397151)
3. Dea Ayu Novita Putri (23091397173)

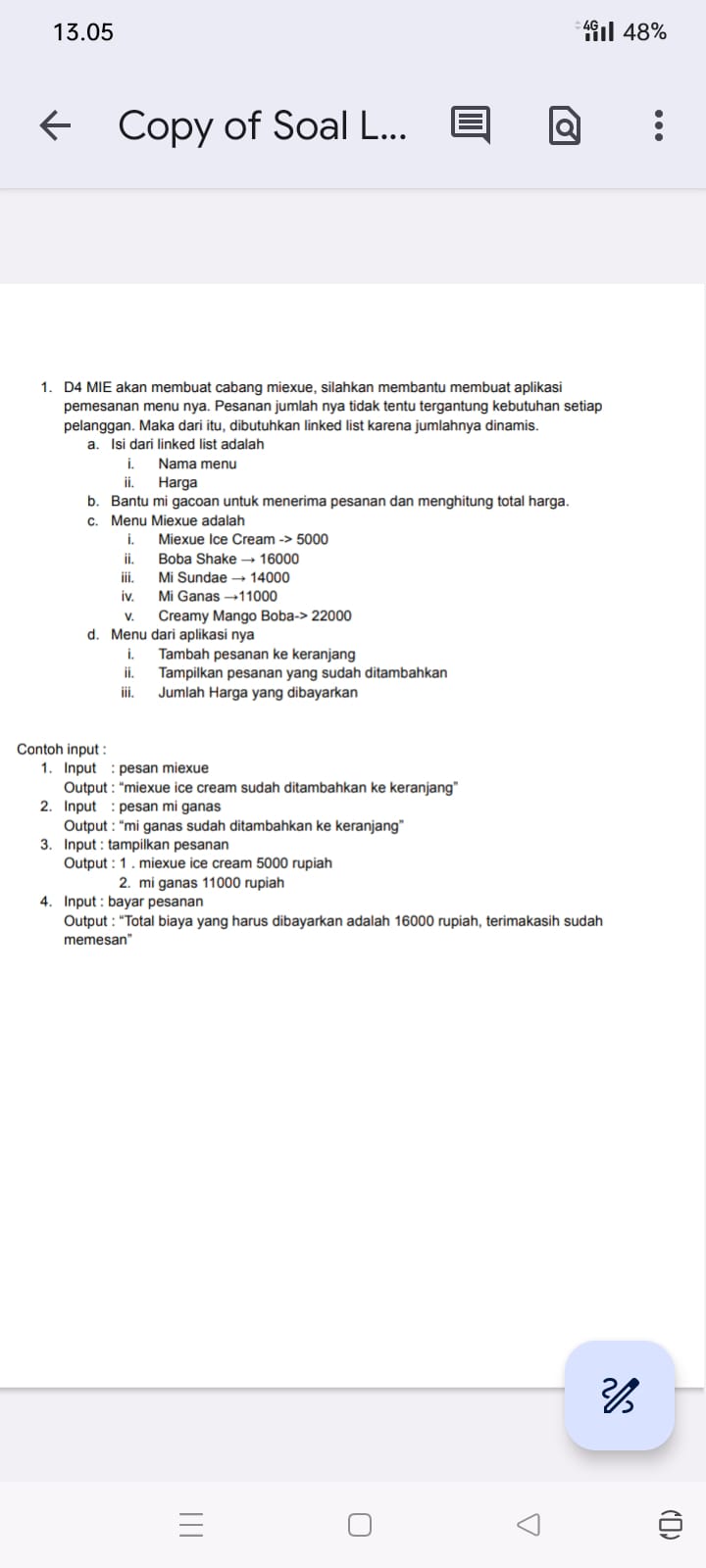
**Progam Studi D4 Manajemen Informatika**

**Fakultas Vokasi**

**Universitas Negeri Surabaya**

**2024**

* **SOAL**



Dalam mengerjakan tugas struktur data pada studi kasus pemesanan mixue ini kami memakai praktek import. Disini kami memiliki dua file:

1. **kelompok\_10\_linked\_list.py**

Berisikan class Node dan LinkedList sebagai blueprint untuk studi kasus Miexue kami

class Node:

    def \_\_init\_\_(self, data=None, next=None, prev=None):

        self.data = data

        self.next = next

        self.prev = prev

class LinkedList:

    def \_\_init\_\_(self, data=None):

        self.head = Node(data) if data else None

        self.tail = self.head if data else None

        self.message = None

    # DISPLAYING VALUES

    def display(self, reversed=False):

        if self.head is None and self.tail is None:

            print(self.message) if self.message else print("Linked List is Empty")

            return

        linked\_list = []

        if reversed:

            node = self.tail

            while node:

                linked\_list.append(node.data)

                node = node.prev

        else:

            node = self.head

            while node:

                linked\_list.append(node.data)

                node = node.next

        return linked\_list

    def length(self):

        count = 0

        node = self.head

        while node:

            node = node.next

            count += 1

        return count

    # CREATING AND UPDATING

    def create(self, data):

        node = Node(data, self.head)

        if self.head is not None:

            self.head.prev = node

        self.head = node

        if self.tail is None: self.tail = self.head

    def create\_values(self, data\_list):

        for data in data\_list:

            self.create(data)

    def append(self, data):

        if self.head is None: return self.create(data)

        node = Node(data, prev=self.tail)

        self.tail.next = node

        self.tail = node

    def append\_values(self, data\_list):

        for data in data\_list:

            self.append(data)

    def insert\_at(self, data, index):

        if index < 0 or index >= self.length():

            raise Exception("Index out of Range")

        if index == 0: return self.create()

        if index == -1: return self.append()

        count = 0

        temp = self.head

        while temp:

            if count == index - 1:

                node = Node(data, temp.next, temp)

                temp.next.prev = node

                temp.next = node

                break

            temp = temp.next

            count += 1

    # DELETE OR REMOVING

    def delete(self):

        self.head = self.head.next

        self.head.prev = None

    def pop(self):

        self.tail = self.tail.prev

        self.tail.next = None

    def remove\_at(self, index):

        if index < 0 or index >= self.length():

            raise Exception("Index out of Range")

        if index == 0: self.delete()

        count = 0

        node = self.head

        while node:

            if count == index - 1:

                node.next = node.next.next

                node.next.prev = node

            node = node.next

            count += 1

1. **kelompok\_10\_pemesanan\_miexue**

File ini sebagai program utama yang akan dijalankan untuk memproses pemesanan Miexue

from linked\_list import LinkedList

class DaftarPesanan(LinkedList):

    def \_\_init\_\_(self, data=None):

        super().\_\_init\_\_(data)

        self.message = "Daftar Pesanan Masih Kosong, Yok Belanja"

    def tampilkan\_menu(self, menus, message="Silahkan dipilih menunya (input 'nomer'nya): "):

        print(menus)

        pesanan = input(message)

        return int(pesanan) - 1 if pesanan.isnumeric() else pesanan

    def rincian\_pesanan(self, list\_menu):

        rincian = []

        node = self.head

        while node:

            rincian.append(node.data.id)

            node = node.next

        rincian\_set = set(rincian)

        for pesanan in rincian\_set:

            print(f"{list\_menu[pesanan][0]} - Rp{list\_menu[pesanan][1]}: {rincian.count(pesanan)}x = Rp{rincian.count(pesanan) \* list\_menu[pesanan][1]}")

    def total\_pesanan(self):

        total = 0

        node = self.head

        while node:

            total += node.data.harga

            node = node.next

        print(f"Total Harga Pesanan: Rp{total}")

class Pesanan:

    def \_\_init\_\_(self, id, nama\_menu, harga):

        self.id = id

        self.nama\_menu = nama\_menu

        self.harga = harga

    def \_\_str\_\_(self):

        return f"{self.id + 1}. {self.nama\_menu.title()} - Rp{self.harga}"

MIXUE\_CABANG\_MI = {

    "Mixue Ice Cream": 5\_000,

    "Boba Shake": 16\_000,

    "Mi Sundae": 14\_000,

    "Mi Ganas": 11\_000,

    "Creamy Mango Boba": 22\_000,

}

list\_menu = list(MIXUE\_CABANG\_MI.items())

menus = ""

for i, menu in enumerate(list\_menu):

    menus += f"\n{i+1}. {menu[0]} - Rp{menu[-1]}"

daftar\_pesanan = DaftarPesanan()

pesanan = daftar\_pesanan.tampilkan\_menu(menus)

daftar\_pesanan.append(Pesanan(pesanan, list\_menu[pesanan][0], list\_menu[pesanan][-1]))

while True:

    pesanan = daftar\_pesanan.tampilkan\_menu(menus, message="Ada lagi? (ketik 'exit' untuk mengakhiri pesanan): ")

    if pesanan == "exit":

        break

    daftar\_pesanan.append(Pesanan(pesanan, list\_menu[pesanan][0], list\_menu[pesanan][-1]))

daftar\_pesanan.rincian\_pesanan(list\_menu)

daftar\_pesanan.total\_pesanan()

* **Penjelasan Tiap Command**

1. **kelompok\_10\_linked\_list.py**

class Node:

    def \_\_init\_\_(self, data=None, next=None, prev=None):

        self.data = data

        self.next = next

        self.prev = prev

Kode diatas bertujuan untuk membuat blueprint atau cetak biru pada objek yang kita perlukan, atribut yang dibuat meliputi data, next, dan prev.

class LinkedList:

    def \_\_init\_\_(self, data=None):

        self.head = Node(data) if data else None

        self.tail = self.head if data else None

        self.message = None

Pada studi kasus pemesanan Miexue ini kami memakai struktur data linked list untuk mengolah data-datanya.

 def create(self, data):

        node = Node(data, self.head)

        if self.head is not None:

            self.head.prev = node

        self.head = node

        if self.tail is None: self.tail = self.head

Setelah kita menginisiasi atribut – atributnya selanjutnya kita membuat metode create untuk menambahkan data pada bagian awal.

  def append(self, data):

        if self.head is None: return self.create(data)

        node = Node(data, prev=self.tail)

        self.tail.next = node

        self.tail = node

Untuk metode kali ini kita menggunakan metode append untuk menambahkan data pada bagian akhir.

1. **kelompok\_10\_pemesanan\_miexue.py**

from kelompok\_10\_linked\_list import LinkedList

class DaftarPesanan(LinkedList):

    def \_\_init\_\_(self, data=None):

        super().\_\_init\_\_(data)

        self.message = "Daftar Pesanan Masih Kosong, Yok Belanja"

Pertama kita import terlebih dahulu class LinkedList dari file kelompok\_10\_linked\_list. Lalu kita membuat objek daftar pesanan yang menjadi turunan dari class LinkedList.

    def tampilkan\_menu(self, menus, message="Silahkan dipilih menunya (input 'nomer'nya): "):

        print(menus)

        pesanan = input(message)

        return int(pesanan) - 1 if pesanan.isnumeric() else pesanan

Setelah itu kita menambahkan metode tampilkan menu yang menampilkan keseluruhan pilihan menu dan mengembalikan menu pilihan user.

 def rincian\_pesanan(self, list\_menu):

        rincian = []

        node = self.head

        while node:

            rincian.append(node.data.id)

            node = node.next

        rincian\_set = set(rincian)

        for pesanan in rincian\_set:

            print(f"{list\_menu[pesanan][0]} - Rp{list\_menu[pesanan][1]}: {rincian.count(pesanan)}x = Rp{rincian.count(pesanan) \* list\_menu[pesanan][1]}")

Kita membuat metode rincian pesanan untuk menampilkan detail pesanan yang dipesan user dengan melakukan looping dari pesanan pertama (self.head) sampai pesanan terakhir.

 def total\_pesanan(self):

        total = 0

        node = self.head

        while node:

            total += node.data.harga

            node = node.next

        print(f"Total Harga Pesanan: Rp{total}")

Untuk metode terakhir kita membuat metode total pesanan untuk menghitung dan menampilkan total keseluruhan harga yang harus dibayar user.

class Pesanan:

    def \_\_init\_\_(self, id, nama\_menu, harga):

        self.id = id

        self.nama\_menu = nama\_menu

        self.harga = harga

    def \_\_str\_\_(self):

        return f"{self.id + 1}. {self.nama\_menu.title()} - Rp{self.harga}"

Sekarang kami membuat class Pesanan yang memiliki atribut id, nama\_menu, dan harga. Setelah itu, kami membuat metode special yaitu “\_\_str\_\_” untuk memberikan label ke masing-masing objek pesanan.

MIXUE\_CABANG\_MI = {

    "Mixue Ice Cream": 5\_000,

    "Boba Shake": 16\_000,

    "Mi Sundae": 14\_000,

    "Mi Ganas": 11\_000,

    "Creamy Mango Boba": 22\_000,

}

list\_menu = list(MIXUE\_CABANG\_MI.items())

menus = ""

for i, menu in enumerate(list\_menu):

    menus += f"\n{i+1}. {menu[0]} - Rp{menu[-1]}"

Disini kami menyimpan data menu dalam bentuk dictionary, lalu mengubah nya menjadi 3D list yang berisikan masing-masing menu dan di simpan dalam variable list\_menu.

Kemudian kami membuat variable menus untuk membuat dan memformat pesan yang akan ditampilkan kepada user.

daftar\_pesanan = DaftarPesanan()

first = True

while True:

    pesanan = daftar\_pesanan.tampilkan\_menu(menus) if first else daftar\_pesanan.tampilkan\_menu(menus, message="Ada lagi? (ketik 'exit' untuk mengakhiri pesanan): ")

    if pesanan == "exit":

        break

    elif pesanan not in range(0, len(list\_menu)):

        print("Pesanan yang dipilih tidak ada di Menu (input salah)")

        continue

    first = False

    daftar\_pesanan.tambahkan\_pesanan(Pesanan(pesanan, list\_menu[pesanan][0], list\_menu[pesanan][-1]))

Setelah menyiapkan beberapa hal yang diperlukan di atas seperti class, list, dll, sekarang kami membuat variable daftar\_pesanan yang menjadi objek hasil class DaftarPesanan

Untuk mengambil input pesanan dari user kami menggunakan while loop yang akan mengecek input pesanan user valid atau tidak dan bila user meng input “exit” maka while loop akan berhenti.

Jika input user valid, maka pesanan akan ditambahkan ke objek daftar\_pesanan yang telah dibuat sebelumnya dan tampilkan menu apa yang dipesan user tersebut.

if daftar\_pesanan.length() <= 0:

    print(f"Terima kasih telah berkunjung")

else:

    daftar\_pesanan.rincian\_pesanan(list\_menu)

    daftar\_pesanan.total\_pesanan()

Disini sebelum kami menampilkan pesanan user, akan dicek terlebih dahulu apabila user tidak memesan apa-apa maka hanya ditampilkan pesan “Terima kasih telah berkunjung”. Tetapi jika user telah memesan sesuatu maka akan ditampilkan rincian pesanan dan total harga yang perlu dibayar.