# 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**

1. D4 MIE akan membuat cabang miexue, silahkan membantu membuat aplikasi pemesanan menu nya. Pesanan jumlah nya tidak tentu tergantung kebutuhan setiap pelanggan. Maka dari itu, dibutuhkan linked list karena jumlahnya dinamis.
    a. Isi dari linked list adalah
        i. Nama menu
        ii. Harga
    b. Bantu mi gacoan untuk menerima pesanan dan menghitung total harga.
    c. Menu Miexue adalah
        i. Miexue Ice Cream -> 5000
        ii. Boba Shake → 16000
        iii. Mi Sundae → 14000
        iv. Mi Ganas →11000
        v. Creamy Mango Boba-> 22000
    d. Menu dari aplikasi nya
        i. Tambah pesanan ke keranjang
        ii. Tampilkan pesanan yang sudah ditambahkan
        iii. Jumlah Harga yang dibayarkan

Contoh input :

1. Input : pesan miexue
   Output : "miexue ice cream sudah ditambahkan ke keranjang"
2. Input : pesan mi ganas
   Output : "mi ganas sudah ditambahkan ke keranjang"
3. Input : tampilkan pesanan
   Output : 1 . miexue ice cream 5000 rupiah
   2. mi ganas 11000 rupiah
4. Input : bayar pesanan
   Output : "Total biaya yang harus dibayarkan adalah 16000 rupiah, terimakasih sudah memesan"

**Link GitHub :**

**https://github.com/gerrymoeis/tugas-struktur-data/tree/tugas-1**

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
```

## 2. 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.

## 2. 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.

```python
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.

```python
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.