LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA
JOBSHEET 12

VANESA MARDIANA PUTRI

244107020129

KELAS TI-1B

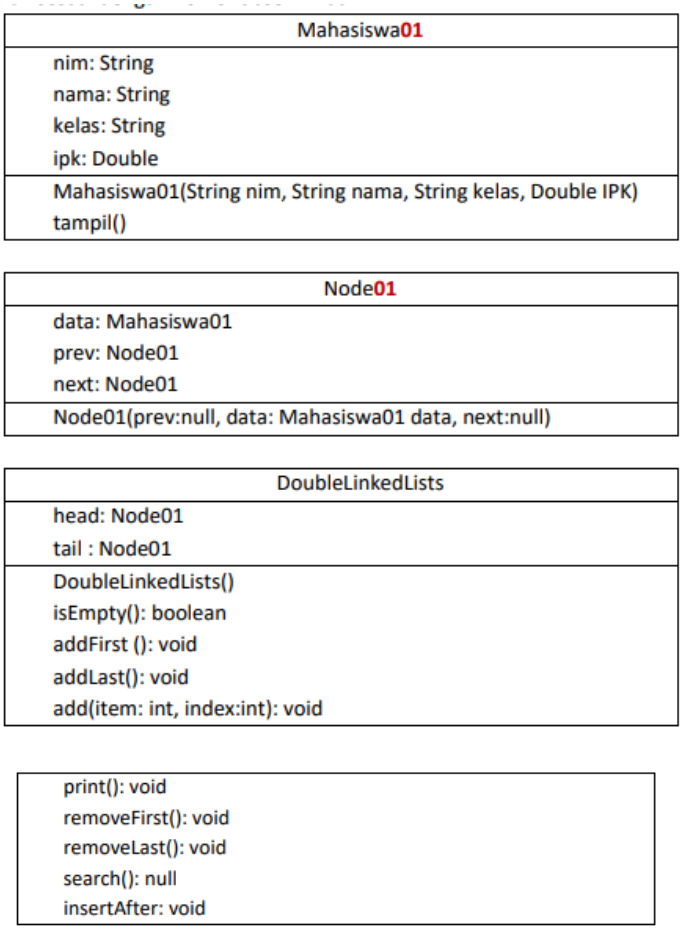
PRODI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

A. PERCOBAAN 1

- Langkah - langkah :
    1. Perhatikan diagram class di bawah ini (ganti 01 dengan nomor absen) :



    2. buat folder atau package baru bernama Jobsheet12 di dalam repository Praktikum ASD.

    3. Buat class di dalam paket tersebut dengan nama Mahasiswa01. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas. Tambahkan juga konstruktor dan method sesuai diagram di atas.

```
public class Mahasiswa23 {    Move this file to a named package.
    public String nama;    Make nama a static final constant or non-public and
    public String nim;    Make nim a static final constant or non-public and p
    public String kelas;    Make kelas a static final constant or non-public a
    public double ipk;    Make ipk a static final constant or non-public and p

    public Mahasiswa23(String nama, String nim, String kelas, double ipk) {
        this.nama = nama;
        this.nim = nim;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampil() {
        System.out.println(nama + "\t" + nim + "\t" + kelas + "\t" + ipk);    Repl
    }
}
```

4. Buat class di dalam paket tersebut dengan nama Node01. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas. Selanjutnya tambahkan konstruktor sesuai diagram di atas.

```
public class Node23 {    Move this file
    Mahasiswa23 data;
    Node23 next;
    Node23 prev;

    public Node23(Mahasiswa23 data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}
```

5. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan Node01. Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
public class DoubleLinkedList23 {
    Node23 head;
    Node23 tail;
```

6. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

```
public DoubleLinkedList23() {
    this.head = null;
    this.tail = null;
}
```

7. Buat method isEmpty(). Method ini digunakan untuk memastikan kondisi linked list kosong.

```
public boolean isEmpty() {
    return head == null;
}
```

8. Kemudian, buat method addFirst(). Method ini akan menjalankan penambahan data di bagian depan linked list.

```
public void addFirst(Mahasiswa23 data) {
  Node23 newNode = new Node23(data);
  if (isEmpty()) {
      head = tail = newNode;
  } else {
      newNode.next = head;
      head.prev = newNode;
      head = newNode;
  }
}
```

9. Selain itu pembuatan method addLast() akan menambahkan data pada bagian belakang linked list.

```
public void addLast(Mahasiswa23 data) {
  Node23 newNode = new Node23(data);
  if (isEmpty()) {
      head = tail = newNode;
  } else {
      tail.next = newNode;
      newNode.prev = tail;
      tail = newNode;
  }
}
```

10. Untuk menambahkan data pada posisi setelah node yang menyimpan data key, dapat dibuat dengan cara sebagai berikut.

```
public void insertAfter(String keyNim, Mahasiswa23 data) {
  Node23 current = head;

  while (current != null && !current.data.nim.equals(keyNim)) {
      current = current.next;
  }

  if (current == null) {
      System.out.println("Node dengan NIM " + keyNim + " tidak ditemukan.");    Rep
      return;
  }

  Node23 newNode = new Node23(data);

  if (current == tail) {
      current.next = newNode;
      newNode.prev = current;
      tail = newNode;
  } else {
      newNode.next = current.next;
      newNode.prev = current;
      current.next.prev = newNode;
      current.next = newNode;
  }

  System.out.println("Node Berhasil disisipkan setelah NIM " + keyNim);    Replace
}
```

11. Untuk mencetak isi dari linked lists dibuat method print(). Method ini akan mencetak isi linked lists berapapun size-nya.

```
public void print() {
  Node23 current = head;
  while (current != null) {
      current.data.tampil();
      current = current.next;
  }
}
```

12. Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.

```
public static void main(String[] args) {
  DoubleLinkedList23 list = new DoubleLinkedList23();
  Scanner scan = new Scanner(System.in);
  int pilihan;
```

13. Buatlah menu pilihan pada class main.

```
do {
  System.out.println("\nMenu Double Linked List Mahasiswa");
  System.out.println("1 Tambah berdasarkan index");
  System.out.println("2. Tambah di awal");
  System.out.println("3. Tambah di akhir");
  System.out.println("4. Hapus di awal");
  System.out.println("5. Hapus di akhir");
  System.out.println("6. Tampilkan data");
  System.out.println("7. Cari mahasiswa berdasarkan NIM");
  System.out.println("0. Keluar");
  System.out.print("Pilih Menu : ");
  pilihan = scan.nextInt();
  scan.nextLine();
```

14. Tambahkan switch case untuk menjalankan menu pilihan di atas.

```
switch (pilihan) {
  case 1 -> {
      Mahasiswa23 mhs = inputMahasiswa(scan);
      list.addFirst(mhs);
  }
  case 2 -> {
      Mahasiswa23 mhs = inputMahasiswa(scan);
      list.addLast(mhs);
  }
  case 3 -> list.removeFirst();    The method removeFirst() is undefine
  case 4 -> list.removeLast();    The method removeLast() is undefined
  case 5 -> list.print();
  case 6 -> {
      System.out.print("Masukkan NIM yang dicari: ");
      String nim = scan.nextLine();
      Node23 found = list.search(nim);    The method search(String) is
      if (found != null) {
          System.out.println("Data ditemukan:");
          found.data.tampil();
      } else {
          System.out.println("Data tidak ditemukan.");
      }
  }
}
```

15. Jangan lupa tambahkan while di bawah switch case dan close untuk menutup object scanner.

```
    } while(pilihan !=0);
    scan.close();
}
```

16. Ada satu karakter yang perlu ditambahkan agar code bisa berjalan. Silakan dianalisis kekurangannya dan ditambahkan sendiri.

```
static Mahasiswa23 inputMahasiswa(Scanner scan) {
    System.out.print(s:"Masukkan NIM: ");
    String nim = scan.nextLine();
    System.out.print(s:"Masukkan Nama: ");
    String nama = scan.nextLine();
    System.out.print(s:"Masukkan Kelas: ");
    String kelas = scan.nextLine();
    System.out.print(s:"Masukkan IPK: ");
    double ipk = scan.nextDouble();
    return new Mahasiswa23(nama, nim, kelas, ipk);
}
```

- Jawaban Pertanyaan :
    1. Perbedaannya berada pada, jika single linkedlist hanya memiliki next jika double linkedlist memiliki prev dan next.
    2. Next untuk ke node selanjutnya dan prev untuk ke node sebelumnya.
    3. Konstruktor ini memastikan linked list dimulai dalam keadaan kosong, siap untuk diisi dengan node-node baru nantinya.
    4. Memastikan penanganan kasus ketika linked list kosong sebelum menambahkan node baru di awal.
    5. Memastikan node yang baru ditambahkan di depan terhubung dengan benar ke node sebelumnya (head lama).
    6.
```
public void print() {
  if (isEmpty()) {
      System.out.println(x:"Linked List Kosong");    Rep
  } else {
```

    7. Memastikan semua pointer (prev dan next) tersambung dengan benar.
    8.
```
case 7 -> {
    System.out.print(s:"Masukkan NIM: ");
    String kyeNim = scan.nextLine();
    Mahasiswa23 mhs = inputMahasiswa(scan);
    list.insertAfter(kyeNim, mhs);
}
```

B. PERCOBAAN 2

- Langkah - langkah :
    1. Buatlah method removeFirst() di dalam class DoubleLinkedLists.

```
public void  removeFirst() {
    if (isEmpty()) {
        System.out.println(x:"List kosong, t
    }
    if (head == tail) {
        Mahasiswa23 data = head.data;    Rem
        head = tail = null;
    } else {
        Mahasiswa23 data = head.data;    Rem
        head = head.next;
        head.prev = null;
    }
}
```

2. Tambahkan method removeLast() di dalam class DoubleLinkedList.

```
public void removeLast() {
    if (isEmpty()) {
        System.out.println(x:"List kosong, tida
    }
    if (head == tail) {
        Mahasiswa23 data = tail.data;    Remove
        head = tail = null;
    } else {
        Mahasiswa23 data = tail.data;    Remove
        tail = tail.prev;
        tail.next = null;
    }
}
```

3. Tambahkan method search().

```
public Node23 search(String nim) {
    Node23 current = head;
    while (current != null) {
        if (current.data.nim.equals(nim)) {
            return current;
        }
        current = current.next;
    }
    return null;
}
```

4. Hasil run :

```
Menu Double Linked List Mahasiswa
1 Tambah berdasarkan index
2. Tambah di awal
3. Tambah di akhir
4. Hapus di awal
5. Hapus di akhir
6. Tampilkan data
7. Cari mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu : 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4.0

Menu Double Linked List Mahasiswa
1 Tambah berdasarkan index
2. Tambah di awal
3. Tambah di akhir
4. Hapus di awal
5. Hapus di akhir
6. Tampilkan data
7. Cari mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu : 5
Hermione        20304050        Gryffindor      4.0
```

```
Menu Double Linked List Mahasiswa
1 Tambah berdasarkan index
2. Tambah di awal
3. Tambah di akhir
4. Hapus di awal
5. Hapus di akhir
6. Tampilkan data
7. Cari mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu : 2
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4.0

Menu Double Linked List Mahasiswa
1 Tambah berdasarkan index
2. Tambah di awal
3. Tambah di akhir
4. Hapus di awal
5. Hapus di akhir
6. Tampilkan data
7. Cari mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu : 4
```

- Jawaban Pertanyaan :
    1. Digunakan untuk menghapus node pertama kemudian menghubungkan ke node selanjutnya.

2.

```
public void  removeFirst() {
    if (isEmpty()) {
        System.out.println(x:"List kosong, tidak bisa dihapus");    Replace this use of System.out by a l
    }
    Mahasiswa23 deletedData = head.data;
    if (head == tail) {
        Mahasiswa23 data = head.data;    Remove this unused "data" local variable.
        head = tail = null;
    } else {
        Mahasiswa23 data = head.data;    Remove this unused "data" local variable.
        head = head.next;
        head.prev = null;
    }
    System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah " + deletedData.nama);
}
```

C. TUGAS

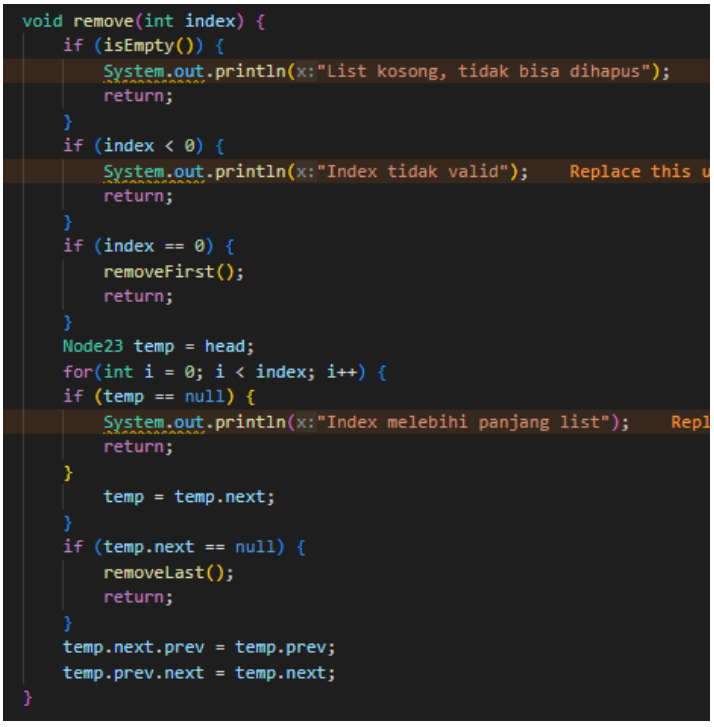
1. Tambahkan fungsi add() pada kelas DoubleLinkedList untuk menambahkan node pada indeks tertentu.

```
void add(Mahasiswa23 data, int index) {
    if (index == 0) {
        addFirst(data);
        return;
    }

    Node23 temp = head;
    for (int i = 0; i < index - 1; i++) {
    if (temp == null) {
    System.out.println(x:"Index melebihi panjang li
    return;
    }
    temp = temp.next;
    }

    if (temp.next == null) {
        addLast(data);
        return;
    }

    Node23 newNode = new Node23(data);
    temp.next.prev = newNode;
    newNode.next = temp.next;
    temp.next = newNode;
    newNode.prev = temp;
}
```

2. Tambahkan removeAfter() pada kelas DoubleLinkedList untuk menghapus node setelah data key.

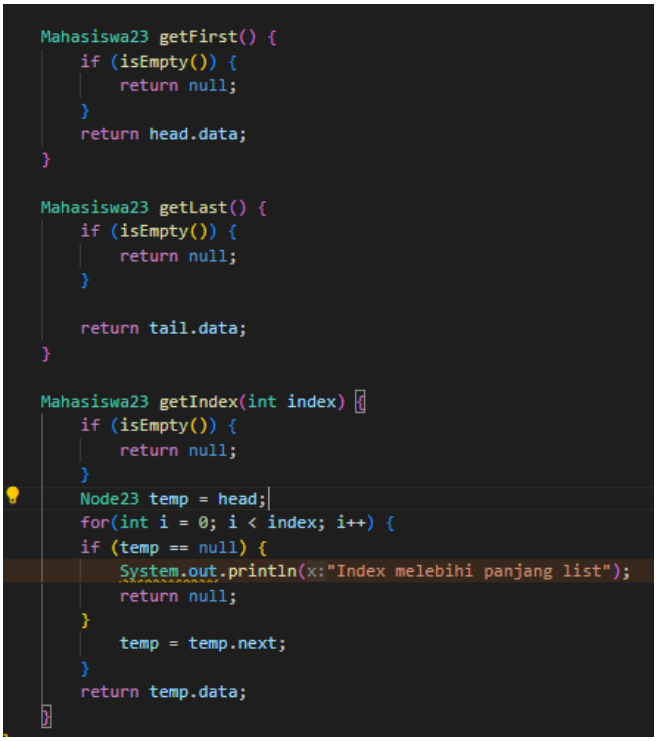
```
void removeAfter(String key) {
    if (isEmpty()) {
        System.out.println(x:"List kosong, tidak bisa dihapus");    Replace this use of Sy
        return;
    }
    Node23 temp = head;

    while (temp != null && !temp.data.nama.equalsIgnoreCase(key)) {
    temp = temp.next;
    }

    if (temp == null || temp.next == null) {
    System.out.println("Node setelah \"" + key + "\" tidak ditemukan atau tidak ada");
    return;
    }

    temp.next.prev = temp;
    temp.next = temp.next.next;

}
```

3. Tambahkan fungsi remove() pada kelas DoubleLinkedList untuk menghapus node pada indeks tertentu.



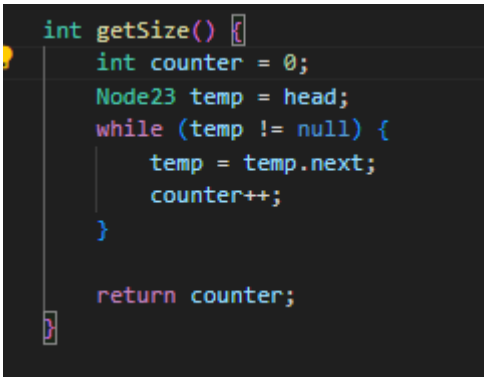
```
void remove(int index) {
    if (isEmpty()) {
        System.out.println(x:"List kosong, tidak bisa dihapus");
        return;
    }
    if (index < 0) {
        System.out.println(x:"Index tidak valid");    Replace this u
        return;
    }
    if (index == 0) {
        removeFirst();
        return;
    }
    Node23 temp = head;
    for(int i = 0; i < index; i++) {
    if (temp == null) {
        System.out.println(x:"Index melebihi panjang list");    Repl
        return;
    }
        temp = temp.next;
    }
    if (temp.next == null) {
        removeLast();
        return;
    }
    temp.next.prev = temp.prev;
    temp.prev.next = temp.next;
}
```

4. Tambahkan fungsi getFirst(), getLast() dan getIndex() untuk menampilkan data pada node head, node tail dan node pada indeks tertentu.



```
Mahasiswa23 getFirst() {
    if (isEmpty()) {
        return null;
    }
    return head.data;
}

Mahasiswa23 getLast() {
    if (isEmpty()) {
        return null;
    }

    return tail.data;
}

Mahasiswa23 getIndex(int index) {
    if (isEmpty()) {
        return null;
    }
    Node23 temp = head;
    for(int i = 0; i < index; i++) {
    if (temp == null) {
        System.out.println(x:"Index melebihi panjang list");
        return null;
    }
        temp = temp.next;
    }
    return temp.data;
}
```

5. Tambahkan kode program dan fungsi agar dapat membaca size/ jumlah data pada Double Linked List.


int getSize() {
int counter = 0;
Node23 temp = head;
while (temp != null) {
temp = temp.next;
counter++;
}
return counter;
}