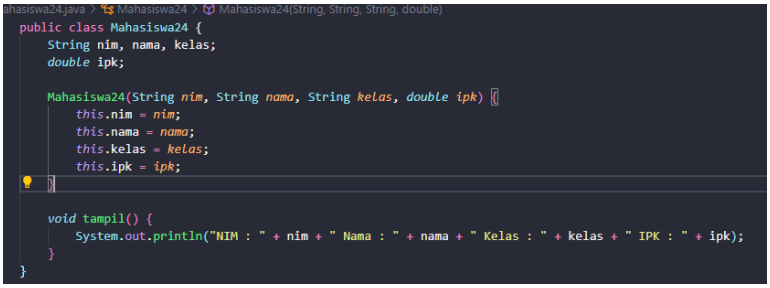
Nama: Wahda Adella Putri Febriana

Kelas: 1B / 24

NIM: 244107020156

Praktikum 1

1. Buat File Mahasiswa24.java dan isinya



```java
public class Mahasiswa24 {
    String nim, nama, kelas;
    double ipk;

    Mahasiswa24(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    void tampil() {
        System.out.println("NIM : " + nim + " Nama : " + nama + " Kelas : " + kelas + " IPK : " + ipk);
    }
}
```

2. Buat File Node24.java beserta isinya

```java
public class Node24 {
    Mahasiswa24 data;
    Node24 prev;
    Node24 next;

    Node24(Mahasiswa24 data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}

class DoubleLinkedList24 {
    Node24 head;
    Node24 tail;

    DoubleLinkedList24() {
        head = null;
        tail = null;
    }

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



```java
void addFirst(Mahasiswa24 data) {
    Node24 newNode = new Node24(data);
    if(isEmpty()) {
        head = tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
}

void addLast(Mahasiswa24 data) {
    Node24 newNode = new Node24(data);
    if(isEmpty()) {
        head = tail = newNode;
    } else {
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
    }
}
```

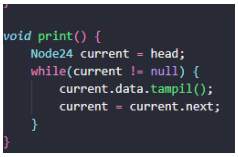
```java
void insertAfter(String keyNim, Mahasiswa24 data) {
    Node24 current = head;

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

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

    Node24 newNode = new Node24(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);
}
```



```java
void print() {
    Node24 current = head;
    while(current != null) {
        current.data.tampil();
        current = current.next;
    }
}
```

3. Buat file DDLMain24.java beserta isinya

```java
import java.util.Scanner;

public class DLLMain24 {
    static Mahasiswa24 inputMahasiswa(Scanner scan) {
        System.out.print("Masukkan NIM: ");
        String nim = scan.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = scan.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = scan.nextLine();
        System.out.print("Masukkan IPK: ");
        double ipk = scan.nextDouble();
        return new Mahasiswa24(nama, nim, kelas, ipk);
    }

    Run | Debug
    public static void main(String[] args) {
        DoubleLinkedList24 list = new DoubleLinkedList24();
        Scanner input = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("\nMenu Double Linked List Mahasiswa");
            System.out.println("1. Tambah di awal");
            System.out.println("2. Tambah di akhir");
            System.out.println("3. Tambah di awal");
            System.out.println("4. Tambah di akhir");
            System.out.println("5. Tampilkan Data");
            System.out.println("6. Cari Mahasiswa Berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilihan Menu: ");
            pilihan = input.nextInt();
            input.nextLine();
```

```java
switch (pilihan) {
    case 1 -> {
        Mahasiswa24 mhs = inputMahasiswa(input);
        list.addFirst(mhs);
    }
    case 2 -> {
        Mahasiswa24 mhs = inputMahasiswa(input);
        list.addLast(mhs);
    }
    case 3 -> list.removeFirst();    The method removeFirst()
    case 4 -> list.removeLast();    The method removeLast() i
    case 5 -> list.print();
    case 6 -> {
        System.out.print("Masukkan NIM yang dicari: ");
        String nim = input.nextLine();
        Node24 found = list.search(nim);    The method search
        if (found != null) {
            System.out.println("Data ditemukan:");
            found.data.tampil();
        } else {
            System.out.println("Data tidak ditemukan.");
        }
    }
    case 0 -> System.out.println("Keluar dari program.");
    default -> System.out.println("Pilihan tidak valid!");
}
while (pilihan != 0);
ut.close();
```

Pertanyaan

1. Single Linked List Hanya punya 1 penghubung (next), Double linked list punya dua arah penghubung (next & prev)
2. Next untuk menunjuk node berikutnya yang ada dalam list, dan prev untuk menunjuk node sebelumnya yang ada di list
3. Untuk inisialisasi linked list kosong
4. Jika linked list kosong maka node baru akan menjadi head dan tail nya
5. Node yang sebelumnya menjadi head, akan dihubungkan balik ke node baru melalui pointer prev
6.
```java
void print() {
    if (isEmpty()) {
        System.out.println("Linked list masih kosong.");
    } else {
        Node24 current = head;
        System.out.println("Isi linked list:");
        while (current != null) {
            System.out.println("NIM: " + current.data.nim + ", Nama: " + current.data.nama);
            current = current.next;
        }
    }
}
```
7. Memulai pencarian pertama dengan var current, lalu melakukan while sampai ketemu dengan key, jika tidak ditemukan mengeluarkan pesan tidak ditemukan, jika ditemukan buat node baru dengan data mhs yang ingin disisipkan, jika current merupakan node terakhir maka

sambungkan newnode ke akhir dan perbarui tail, jika current bukan di akhir sisipkan newnode di Tengah dan update semua pointer prev dan next agar linked tetep 2 arah

8.
```
}
case 7 -> {
    System.out.prix:nt("Masukkan NIM yang ingin disisipi setelahnya: ");
x:  String keyNim = input.nextLine();
    Mahasiswa24 mhs = inputMahasiswa(input);
    list.insertAfter(keyNim, mhs);
}
case 0 -> System.out.println("Keluar dari program.");
```

Praktikum 2

1. Tambahkan kode berikut di file Node24.java

```
void removeFirst() {
    if(isEmpty()) {
        System.out.println(x:"List kosong tidak bisa di hapus");
        return;
    }
    if(head == tail) {
        head = tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }
}

void removeLast() {
    if(isEmpty()) {
        System.out.println(x:"List kosong tidak bisa dihapus");
        return;
    }
    if(head == tail) {
        head = tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }
}
```

Hasil percobaan

```
Masukkan NIM: 2030
Masukkan Nama: Hermio
Masukkan Kelas: Gryy
Masukkan IPK: 4,0

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa Berdasarkan NIM
7. Tambah setelah NIM tertentu
0. Keluar
Pilihan Menu: 3

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa Berdasarkan NIM
7. Tambah setelah NIM tertentu
0. Keluar
Pilihan Menu: 5
Linked list masih kosong.
```

Pertanyaan

1. Digunakan untuk menghapus node pertama , memindahkan head ke head.next lalu membuat head.prev nya null
2. Modifikasi kode

```
void removeFirst() {
    if(isEmpty()) {
        System.out.println(x:"List kosong tidak bisa di hapus");
        return;
    }

    Mahasiswa24 dataTerhapus = head.data;

    if(head == tail) {
        head = tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }

    System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah " + dataTerhapus.nama);
}
```

Tugas Praktikum

1.
```
void add(int index, Mahasiswa24 data) {
    if (index < 0) {
        System.out.println(x:"Indeks tidak valid.");
        return;
    }
    if (index == 0) {
        addFirst(data);
        return;
    }

    Node24 current = head;
    int count = 0;
    while (current != null && count < index) {
        current = current.next;
        count++;
    }

    if (current == null) {
        addLast(data);
    } else {
        Node24 newNode = new Node24(data);
        newNode.prev = current.prev;
        newNode.next = current;
        if (current.prev != null) current.prev.next = newNode;
        current.prev = newNode;
    }
}
```

2.
```
void removeAfter(String keyNim) {
    Node24 current = head;
    while (current != null && !current.data.nim.equals(keyNim)) {
        current = current.next;
    }

    if (current == null || current.next == null) {
        System.out.println(x:"Tidak ada node setelah NIM tersebut.");
        return;
    }

    Node24 toDelete = current.next;
    if (toDelete == tail) {
        tail = current;
        current.next = null;
    } else {
        current.next = toDelete.next;
        toDelete.next.prev = current;
    }

    System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus
}
```

3.
```
void remove(int index) {
    if (index < 0 || isEmpty()) {
        System.out.println(x:"Indeks tidak valid atau list kosong.");
        return;
    }

    if (index == 0) {
        removeFirst();
        return;
    }

    Node24 current = head;
    int count = 0;
    while (current != null && count < index) {
        current = current.next;
        count++;
    }

    if (current == null) {
        System.out.println(x:"Indeks melebihi jumlah node.");
        return;
    }

    if (current == tail) {
        removeLast();
    } else {
        current.prev.next = current.next;
        current.next.prev = current.prev;
        System.out.println("Node pada indeks " + index + " berhasil dihapus.");
```

4.
```
void getIndex(int index) {
    if (index < 0 || isEmpty()) {
        System.out.println(x:"Indeks tidak valid atau list kosong.");
        return;
    }

    Node24 current = head;
    int count = 0;
    while (current != null && count < index) {
        current = current.next;
        count++;
    }

    if (current == null) {
        System.out.println(x:"Indeks melebihi jumlah node.");
    } else {
        System.out.println("Data pada indeks " + index + ":");
        current.data.tampil();
    }
```

```
void getFirst() {
    if (isEmpty()) {
        System.out.println(x:"List kosong.");
    } else {
        System.out.println(x:"Data pertama:");
        head.data.tampil();
    }
}

void getLast() {
    if (isEmpty()) {
        System.out.println(x:"List kosong.");
    } else {
        System.out.println(x:"Data terakhir:");
        tail.data.tampil();
    }
}
```

5.
```
int size() {
    int count = 0;
    Node24 current = head;
    while (current != null) {
        count++;
        current = current.next;
    }
    return count;
}
```