Nama : Siti Mutmainah

Kelas : TI-1B

NIM : 244107020143/21

Percobaan 1

Pada percobaan 1 ini akan dibuat class data, class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan dan belakang linked list)

1. Perhatikan diagram class **Mahasiswa01**, **Node01** dan class **DoublelinkedLists** di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.
   Ganti **01** sesuai dengan nomor absen Anda.

Mahasiswa**01**
nim: String nama: String kelas: String ipk: Double
Mahasiswa01(String nim, String nama, String kelas, Double IPK) tampil()

Node**01**
data: Mahasiswa01 prev: Node01 next: Node01
Node01(prev:null, data: Mahasiswa01 data, next:null)

DoubleLinkedLists
head: Node01 tail : Node01
DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void

print(): void

removeFirst():

void removeLast():

void search(): null

insertAfter: void

2. Pada Project yang sudah dibuat pada Minggu sebelumnya, 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

```java
Mahasiswa21.java > Mahasiswa21
public class Mahasiswa21{
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa21(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

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

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

```java
Node21.java > ...
public class Node21 {
    Mahasiswa21 data;
    Node21 prev;
    Node21 next;

    public Node21 (Mahasiswa21 data){
        this.data=data;
        this.prev=null;
        this.next=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.

```
DoubleLinkedList21.java > DoubleLinkedList21 > search(String)
1   public class DoubleLinkedList21 {
2       Node21 head;
3       Node21 tail;
4
```

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

```
public DoubleLinkedList21(){
    head = null;
    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(Mahasiswa21 data) {
    Node21 newNode = new Node21(data);
    if (isEmpty()) {
        head = tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
    System.out.println(x:"Data berhasil ditambahkan di awal.");
}
```

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

```
public void addLast (Mahasiswa21 data){
    Node21 newNode = new Node21(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, Mahasiswa21 data){
    Node21 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;
    }

    Node21 newNode = new Node21(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);
}
```

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

```
public void print(){
    Node21 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.

```
DLLMain21.java > ...
1   import java.util.Scanner;
2
3   public class DLLMain21 {
        Run | Debug
4       public static void main(String[] args) {
5           DoubleLinkedList21 list = new DoubleLinkedList21();
6           Scanner scan = new Scanner(System.in);
7           int pilihan;
```

13. Buatlah menu pilihan pada class main

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

14. Tambahkan switch case untuk menjalankan menu pilihan di atas

```
    switch (pilihan) {
        case 1 -> {
            Mahasiswa21 mhs = inputMahasiswa(scan);
            list.addFirst(mhs);
        }
        case 2 -> {
            Mahasiswa21 mhs = inputMahasiswa(scan);
            list.addLast(mhs);
        }
        case 3 -> list.removeFirst();
        case 4 -> list.removeLast();
        case 5 -> list.print();
        case 6 -> {
            System.out.print(s:"Masukkan NIM yang dicari: ");
            String nim = scan.nextLine();
            Node21 found = list.search(nim);
            if (found != null) {
                System.out.println(x:"Data ditemukan:");
                found.data.tampil();
            } else {
                System.out.println(x:"Data tidak ditemukan");
            }
        }
        case 0 -> System.out.println(x:"Keluar dari program");
        default -> System.out.println(x:"Pilihan tidak valid!");
    }
```

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.

- DLLMain21

```
    public static Mahasiswa21 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();
     scan.nextLine();

     return new Mahasiswa21(nim, nama, kelas, ipk);
}
}
```

- DoubleLinkedList21

```
public Node21 search(String nim) {
    Node21 current = head;
    while (current != null) {
        if (current.data.nim.equalsIgnoreCase(nim)) {
            return current;
        }
        current = current.next;
    }
    return null;
}
```

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

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

}
```

Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
PS C:\Semester 2\Praktikum ASD\Jobsheet\Jobsheet12>  & 'C:\Program
jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExce
es' '-cp' 'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage
15a59a1d50d939407bed92\redhat.java\jdt_ws\Jobsheet12_cab07d28\bin'


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
0. Keluar
Pilih menu: 1
Masukkan NIM: 12345
Masukkan Nama: Hermione
Masukkan Kelas: 1B
Masukkan IPK: 4.0
Data berhasil ditambahkan di awal.

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
0. Keluar
Pilih menu: 5
NIM: 12345, Nama: Hermione, Kelas: 1B, IPK: 4.0
```

Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!

   Jawab: Jika single list hanya satu arah, sedangkan double linked list memiliki dua arah ada next dan prev.

2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

   Jawab: Atribut next dan prev berguna untuk menghubungkan node satu dengan lainnya, next untuk setelahnya, prev untuk sebelumnya.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?

```
public DoubleLinkedList01() {
    head = null;
    tail = null;
}
```

```
if (isEmpty()) {
    head = tail = newNode;
```

   Jawab: Konstruktor berfungsi untuk inisialisasi keadaan awal dari linked list. Mengatur agar head dan tail bernilai null. Saat objek pertama kali dibuat linked list masih kosong, belum ada node di dalamnya. Dengan ini program bisa mudah mengecek apakah list kosong atau tidak. Seperti metode **isEmpty** jika kosong, maka node pertama ditambahkan, head dan tail akan langsung menunjuk node tersebut.

4. Pada method **addFirst()**, apa maksud dari kode berikut?

```
public void addFirst(Mahasiswa21 data) {
    Node21 newNode = new Node21(data);
    if (isEmpty()) {
        head = tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
    System.out.println(x:"Data berhasil ditambahkan di awal.");
}
```

   Jawab: Kode ini bertujuan untuk menambah node baru di awal linked list, **if isEmpty** mengecek apakah linked masi kosong. Jika kosong, artinya belum ada node, maka head dan tail menunjuk ke newNode. **Else** jika sudah terisi **newNode.next = head** node baru akan menunjuk ke node yang sekarang menjadi head. **head.prev = newNode** node yang sebelumnya menjadi head, menunjuk ke node baru sebagai node sebelumnya. **head = newNode** node baru resmi menjadi head baru.

5. Perhatikan pada method **addFirst().** Apakah arti statement head.prev = newNode ?

Jawab: node yang sebelumnya menjadi head sekarang tau bahwa ada node baru di depannya

6. Modifikasi code pada fungsi **print()** agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.

   Jawab:

```
    public void print() {
    if (head == null) {
        System.out.println(x:"Linked list masih kosong.");
    } else {
        Node21 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }
}
```

7. Pada insertAfter(), apa maksud dari kode berikut ?

   current.next.prev = newNode;

   Jawab: Kode ini digunakan untuk menjaga agar struktur tetap terhubung dua arah setelah menyiapkan node baru di tengah-tengah list. Node setelah current menunjuk ke newNode sebagai node sebelumnya.

8. Modifikasi menu pilihan dan switch-case agar fungsi **insertAfter()** masuk ke dalam menu pilihan dan dapat berjalan dengan baik.

   Jawab:

```
case 7 -> {
    System.out.print(s:"Masukkan NIM yang dicari: ");
    String cariNIM = scan.nextLine();
    System.out.print(s:"Masukkan NIM baru: ");
    String nimBaru = scan.nextLine();
    System.out.print(s:"Masukkan nama: ");
    String namaBaru = scan.nextLine();
    System.out.print(s:"Masukkan kelas: ");
    String kelasBaru = scan.nextLine();
    System.out.print(s:"Masukkan ipk: ");
    double ipkBaru = scan.nextDouble();
    Mahasiswa21 mhsBaru = new Mahasiswa21(nimBaru, namaBaru, kelasBaru, ipkBaru);
    list.insertAfter(cariNIM, mhsBaru);
}
```

Kegiatan Praktikum 2

Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists.

1. Buatlah method **removeFirst()** di dalam class **DoubleLinkedLists.**

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

2. Tambahkan method **removeLast()** di dalam class **DoubleLinkedLists.**

```
public 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;
    }
}
```

Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
PS C:\Semester 2\Praktikum ASD\Jobsheet\Jobsheet12>  & 'C:\Program
jdk-24\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExc
es' '-cp' 'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorag
15a59a1d50d939407bed92\redhat.java\jdt_ws\Jobsheet12_cab07d28\bin'


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 data
0. Keluar
Pilih menu: 2
Masukkan NIM: 20304050
Masukkan Nama: iin
Masukkan Kelas: 1B
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 data
0. Keluar
Pilih menu: 3

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
```

Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method **removeFirst()**?

   head = head.next; head.prev = null;

   Jawab: **head = head.next** artinya head sekarang dipindah ke node berikutnya, node yang kedua.

   head.prev = null artinya setelah head pindah memutuskan hubungan dengan nide sebelumnya yang sudah dihapus, diset null agar tidak menunjuk ke node lama yang sudah tidak digunakan

2. Modifikasi kode program untuk menampilkan pesan "Data sudah berhasil dihapus. Data yang terhapus adalah … "

   Jawab:

```
public void removeFirst() {
    if (head == null) {
        System.out.println(x:"Linked list kosong. Tidak bisa dihapus");
    } else {
        Mahasiswa21 dataTerhapus = head.data;
        head = head.next;
        if (head != null) {
            head.prev = null;
        }
        System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah " + dataTerhapus);
    }
}

public void removeLast() {
    if (tail == null) {
        System.out.println(x:"Linked list kosong. Tidak bisa dihapus");
    } else {
        Mahasiswa21 dataTerhapus = tail.data;
        tail = tail.prev;
        if (tail != null) {
            tail.next = null;
        } else {
            head = null;
        }
        System.out.println("Data sudah berhasil dihapus dari belakang. Data yang terhapus adalah: " + dataTerhapus);
    }
}
```

Tugas Praktikum

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

```
public void add(int index, Mahasiswa21 data) {
    if (index < 0) {
        System.out.println(x:"Index tidak boleh negatif.");
        return;
    }

    Node21 newNode = new Node21(data);

    if (isEmpty()) {
        if (index == 0) {
            head = tail = newNode;
            System.out.println("Data berhasil ditambahkan pada indeks " + index);
        } else {
            System.out.println(x:"Index out of bounds.");
        }
        return;
    }

    if (index == 0) {
        // Tambah di awal
        addFirst(data);
        return;
    }
```

```
// Cari ukuran list untuk validasi indeks
int size = 0;
Node21 temp = head;
while (temp != null) {
    size++;
    temp = temp.next;
}

if (index > size) {
    System.out.println(x:"Index out of bounds.");
    return;
}

if (index == size) {
    // Tambah di akhir
    addLast(data);
    System.out.println("Data berhasil ditambahkan pada indeks " + index);
    return;
}

// Tambah di tengah
Node21 current = head;
for (int i = 0; i < index; i++) {
    current = current.next;
}

Node21 prevNode = current.prev;

prevNode.next = newNode;
newNode.prev = prevNode;
newNode.next = current;
current.prev = newNode;

System.out.println("Data berhasil ditambahkan pada indeks " + index);
```

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

```
public void removeAfter(String keyNim) {
if (isEmpty()) {
    System.out.println(x:"Linked list kosong. Tidak ada yang dihapus.");
    return;
}

Node21 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;
}

if (current.next == null) {
    System.out.println("Tidak ada node setelah NIM " + keyNim + " untuk dihapus.");
    return;
}
Node21 nodeToRemove = current.next;

current.next = nodeToRemove.next;
if (nodeToRemove.next != null) {
    nodeToRemove.next.prev = current;
} else {
    tail = current;
}

System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus. Data yang dihapus: " + nodeToRemove.data);
```

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

```
public void remove(int index) {
if (isEmpty()) {
    System.out.println(x:"Linked list kosong. Tidak bisa dihapus.");
    return;
}

if (index < 0) {
    System.out.println(x:"Index tidak boleh negatif.");
    return;
}

// Hitung size list
int size = 0;
Node21 temp = head;
while (temp != null) {
    size++;
    temp = temp.next;
}

if (index >= size) {
    System.out.println(x:"Index out of bounds.");
    return;
}
```

```
if (index == 0) {
    // Hapus node pertama
    removeFirst();
    return;
}

if (index == size - 1) {
    // Hapus node terakhir
    removeLast();
    return;
}

// Hapus node di tengah
Node21 current = head;
for (int i = 0; i < index; i++) {
    current = current.next;
}

current.prev.next = current.next;
current.next.prev = current.prev;

System.out.println("Node pada indeks " + index + " berhasil dihapus. Data yang dihapus: " + current.data);
```

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

```
public Mahasiswa21 getFirst() {
    if (isEmpty()) {
        System.out.println(x:"Linked list kosong.");
        return null;
    }
    return head.data;
}

public Mahasiswa21 getLast() {
    if (isEmpty()) {
        System.out.println(x:"Linked list kosong.");
        return null;
    }
    return tail.data;
}

public Mahasiswa21 getIndex(int index) {
    if (isEmpty()) {
        System.out.println(x:"Linked list kosong.");
        return null;
    }
    if (index < 0 || index >= size) {
        System.out.println(x:"Index tidak valid.");
        return null;
    }

    Node21 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    return current.data;
}
```

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

```
public int size() {
    return size;
}
```

```
public class DoubleLinkedList21 {
    Node21 head;
    Node21 tail;
    int size;

    public DoubleLinkedList21() {
        head = null;
        tail = null;
        size = 0;
    }
```

```
DoubleLinkedList21 list = new DoubleLinkedList21();
System.out.println("Jumlah data dalam linked list: " + list.size());
list.print();
```