LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 12

RIFO ANGGI BARBARA DANUARTA
244107020063
TI_1E
PROGRAM STUDI D_IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

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.
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

```
package Jobsheet12;

public class Mahasiswa26 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa26(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

```
package Jobsheet12;

public class Node26 {
    Mahasiswa26 data;
    Node26 prev;
    Node26 next;

    public Node26(Mahasiswa26 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 DoubleLinkedList {
    Node26 head;
    Node26 tail;
```

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

```
    public DoubleLinkedList() {
        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(Mahasiswa26 data) {
        Node26 newNode = new Node26(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(Mahasiswa26 data) {
        Node26 newNode = new Node26(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, Mahasiswa26 data) {
    Node26 current = head;

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

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

    Node26 newNode = new Node26(data);

    // jika current adalah tail, tambahkan di akhir
    if (current == tail) {
        current.next = newNode;
        newNode.prev = current;
        tail = newNode;
    } else {
        // sisipkan di tengah
        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() {
        Node26 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 class DLLMain {
    public static void main(String[] args) {
        DoubleLinkedList list = new DoubleLinkedList();
        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 di Awal");
                System.out.println("2. Tambah di Akhir");
                System.out.println("3. Hapus di Awal");
                System.out.println("4. Hapus di Akhir");
                System.out.println("5. Tampilkan Data");
                System.out.println("6. 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 -> {
                    Mahasiswa26 mhs = inputMahasiswa(scan);
                    list.addFirst(mhs);
                }
                case 2 -> {
                    Mahasiswa26 mhs = inputMahasiswa(scan);
                    list.addLast(mhs);
                }
                case 3 -> list.removeFirst();
                case 4 -> list.removeLast();
                case 5 -> list.print();
                case 6 -> {
                    System.out.print("Masukkan NIM yang dicari:
");
                    String nim = scan.nextLine();
                    Node26 found = list.search(nim);
                    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");
            }
```

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.
    Yang pertama yaitu kurang method inputMahasiswa26

```
public static Mahasiswa26 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();
    scan.nextLine();
    return new Mahasiswa26(nim, nama, kelas, ipk);
```

    Yang kedua kurang method removeFirst,removeLast dan search

```
public void removeFirst() {
    if (head == null) {
        System.out.println("List kosong.");
        return;
    }

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

public void removeLast() {
    if (tail == null) {
        System.out.println("List kosong.");
        return;
    }

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

public Node26 search(String nim) {
    Node26 current = head;
    while (current != null) {
        if (current.data.nim.equals(nim)) {
            return current;
        }
        current = current.next;
    }
    return null;
}
```

17. Verifikasi Hasil Percobaan

```
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: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
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
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.0
```

Pertanyaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?
4. Pada method addFirst(), apa maksud dari kode berikut?
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
6. Modifikasi code pada fungsi print() agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.
7. Pada insertAfter(), apa maksud dari kode berikut ?
   current.next.prev = newNode;
8. Modifikasi menu pilihan dan switch-case agar fungsi insertAfter() masuk ke dalam menu pilihan dan dapat berjalan dengan baik.

Jawaban

1. single linked list hanya bisa next sedangkan double linked list bisa next dan prev lalu print juga untuk single linked list hanya bisa bergerak kekanan jika data nya ada 1000+ dan front didepan lalu yang mau diprint di tail maka akan memakan waktu dan double linked list bisa ngeprint dari depan dan dari belakang.

2. Kedua atribut ini memegang peran utama dalam bagaimana node saling terhubung satu sama lain di dalam list. Atribut next berfungsi untuk menunjuk ke node berikutnya, yaitu node yang berada setelah node saat ini. Sementara itu, atribut prev berfungsi sebaliknya, yaitu menunjuk ke node sebelumnya, atau node yang berada sebelum node saat ini. Dengan adanya kedua arah ini maju (next) dan mundur (prev) maka struktur data yang dibentuk disebut double linked list (linked list ganda). Hal ini berbeda dengan single linked list yang hanya memiliki satu arah, yaitu maju saja.

3. Fungsi utama konstruktor adalah untuk menginisialisasi atribut-atribut awal dari objek yang baru saja dibuat. Pada class DoubleLinkedLists, biasanya konstruktor digunakan untuk mengatur nilai awal dari pointer head dan tail menjadi null. Kenapa begitu? Karena saat pertama kali list dibuat, belum ada node apa pun di dalamnya — alias linked list masih kosong. Dengan menyetel head = null dan tail = null, kita memberitahu program bahwa: Belum ada node pertama (head), Belum ada node terakhir (tail), Struktur list-nya siap digunakan, tapi masih kosong
4. Yaitu pemilihan if dengan kondisi isEmpety yaitu untuk memastikan apakah linked list masih kosong dan jika iya maka variable sementara (newCode) disimpan di head dan tail, dan jika tidak akan lanjut ke pemilihan selanjutnya
5. yaitu saat kita menambahkan node baru di awal (depan) linked list yang sudah berisi node sebelumnya. Pada kondisi ini: newNode akan menjadi node baru di depan (sebagai head baru) head saat ini masih menunjuk ke node lama yang sebelumnya menjadi kepala Maka kita perlu menghubungkan node lama (head saat ini) ke newNode sebagai pendahulunya
6. modif print

```
    public void print() {
        if (head == null) {
            System.out.println("List kosong.");
        } else {
            Node26 current = head;
            while (current != null) {
                current.data.tampil();
                current = current.next;
            }
        }
    }
```

7. tujuannya adalah untuk menyisipkan node baru setelah node tertentu dalam double linked list.
   Misalnya linked list seperti ini:
   [A] <-> [B] <-> [C]
   Dan kamu ingin menyisipkan node [X] setelah [B], maka hasil akhirnya akan jadi:
   [A] <-> [B] <-> [X] <-> [C]
   Untuk membuat ini berhasil, harus:
   Menghubungkan [B].next ke [X] Menghubungkan [X].prev ke [B] Menghubungkan [X].next ke [C] Dan mengubah [C].prev agar menunjuk ke [X], bukan ke [B] lagi
8. di menu

```
                System.out.println("7. Sisipkan Setelah NIM
Tertentu");
```

   di switch

```
                case 7 -> {
                    System.out.print("Masukkan NIM untuk sisipkan
setelah: ");
                    String keyNim = scan.nextLine();
                    Mahasiswa26 mhs = inputMahasiswa(scan);
                    list.insertAfter(keyNim, mhs);
                }
```

Percobaan 2

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("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("List kosong, tidak bisa
dihapus.");
            return;
        }

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

Verifikasi Hasil Percobaan

```
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. Sisipkan Setelah NIM Tertentu
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 di Awal
2. Tambah di Akhir
3. Hapus di Awal
4. Hapus di Akhir
5. Tampilkan Data
6. Cari Mahasiswa Berdasarkan NIM
7. Sisipkan Setelah NIM Tertentu
0. Keluar
Pilih menu: 3
```

Pertanyaan

1. Apakah maksud statement berikut pada method removeFirst()?
   head = head.next;
   head.prev = null;
2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “

Jawaban

1. Head = head.next yaitu menggeser head ke elemen selanjutnya
   Head.prev = null yaitu kan head sudah digeser maka elemen sebelumnya akan bernilai null atau data nya dihapus.
2. Method removeFirst

```
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa
dihapus.");
            return;
        }
        System.out.println("Data sudah berhasil dihapus. Data yang
dihapus adalah: ");
        head.data.tampil();

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

Tugas

Pada method DoubleLinkedList

```
    public void add(Mahasiswa26 data, int index) {
        Node26 newNode = new Node26(data);
        if (index < 0 || index >= size()) {
            System.out.println("Index di luar jangkauan.");
            return;
        }

        if (index == 0) {
            addFirst(data);
            System.out.println("Data berhasil ditambahkan di index " +
index);
            return;
        } else if (index == size()) {
            addLast(data);
            System.out.println("Data berhasil ditambahkan di index " +
index);
            return;
        } else {
            Node26 current = head;
            for (int i = 0; i < index - 1; i++) {
                current = current.next;
            }

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

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

            System.out.println("Data berhasil ditambahkan di index " +
index);
        }
        size++;
    }

    public void removeAfter(String key) {
        Node26 current = head;

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

        if (current == null || current.next == null) {
            System.out.println("Tidak ada data setelah NIM " + key);
            return;
        }

        Node26 toRemove = current.next;
        System.out.println("Data yang dihapus: ");
        toRemove.data.tampil();
        current.next = toRemove.next;

        if (toRemove == tail) {
            tail = current;
            tail.next = null;
        } else {
            current.next = toRemove.next;
            toRemove.next.prev = current;
        }
        size--;
    }
```

```
    public void remove(int index) {
        if (index < 0 || index >= size()) {
            System.out.println("Index di luar jangkauan.");
            return;
        }

        if (index == 0) {
            removeFirst();
            return;
        } else if (index == size() - 1) {
            removeLast();
            return;
        } else {
            Node26 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("Data yang dihapus: ");
            current.data.tampil();

            size--;
        }
    }

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

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

    public void getIndex(int index) {
        if (index < 0 || index >= size()) {
            System.out.println("Index di luar jangkauan.");
            return;
        }

        Node26 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        System.out.println("Data pada index " + index + ": ");
        current.data.tampil();
    }

    public int size() {
        int count = 0;
        Node26 current = head;
        while (current != null) {
            count++;
            current = current.next;
        }
        return count;
    }
```

Pada method DLLMain

```
                case 8 -> {
                    System.out.print("Masukkan index untuk menambahkan
data: ");
                    int index = scan.nextInt(); scan.nextLine();
                    Mahasiswa26 mhs = inputMahasiswa(scan);
                    list.add(mhs, index);
                }
                case 9 -> {
                    System.out.print("Masukkan NIM Mahasiswa yang ingin
dihapus: ");
                    String keyNim = scan.nextLine();
                    list.removeAfter(keyNim);
                }
                case 10 -> {
                    System.out.print("Masukkan index yang ingin dihapus:
");
                    int index = scan.nextInt(); scan.nextLine();
                    list.remove(index);
                }
                case 11 -> list.getFirst();
                case 12 -> list.getLast();
                case 13 -> {
                    System.out.print("Masukkan index yang ingin
ditampilkan: ");
                    int index = scan.nextInt(); scan.nextLine();
                    list.getIndex(index);
                }
                case 14 -> System.out.println("Jumlah data mahasiswa: "
+ list.size());
```