JOBSHEET XII
Double Linked Lists

12.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. Memahami algoritma double linked lists;
2. Membuat dan mendeklarasikan struktur algoritma double linked lists;
3. Menerapkan algoritma double linked lists dalam beberapa *study case*.

12.2 Kegiatan Praktikum 1

12.2.1 Percobaan 1

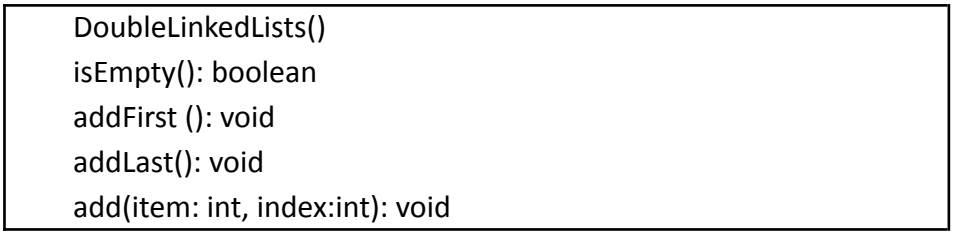
Pada percobaan 1 ini akan dibuat class data, class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan dan belakang linked list)

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
public class Mahasiswa14 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa14(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);
        System.out.println("Nama  : " + nama);
        System.out.println("Kelas : " + kelas);
        System.out.println("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
public class Node14 {
    Mahasiswa14 data;
    Node14 prev;
    Node14 next;

    public Node14(Node14 prev, Mahasiswa14 data, Node14 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

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.

```java
public class DoubleLinkedLists {
    Node14 head;
    Node14 tail;
```

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

```
public DoubleLinkedLists() {
    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(Mahasiswa14 item) {
    Node14 newNode = new Node14(null, item, head);
    if (isEmpty()) {
        head = tail = newNode;
    } else {
        head.prev = newNode;
        head = newNode;
    }
}
```

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

```
public void addLast(Mahasiswa14 item) {
    Node14 newNode = new Node14(tail, item, null);
    if (isEmpty()) {
        head = tail = newNode;
    } else {
        tail.next = newNode;
        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, Mahasiswa01 data) {
     Node01 current = head;

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

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

     Node01 newNode = new Node01(data);

     // Jika current adalah tail, cukup 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.

```java
public void print() {
    if (!isEmpty()) {
        Node14 tmp = head;
        while (tmp != null) {
            tmp.data.tampil();
            System.out.println("------------------");
            tmp = tmp.next;
        }
    } else {
        System.out.println("Linked list kosong");
    }
}
```

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

```java
import java.util.Scanner;

public class DoubleLinkedListsMain {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedLists dll = new DoubleLinkedLists();
        int pilih;
```

13. Buatlah menu pilihan pada class main

```
do {
    System.out.println(x:"============================");
    System.out.println(x:"Menu:");
    System.out.println(x:"1. Tambah data di awal");
    System.out.println(x:"2. Tambah data di akhir");
    System.out.println(x:"3. Cetak data");
    System.out.println(x:"4. Hapus data pertama");
    System.out.println(x:"5. Hapus data terakhir");
    System.out.println(x:"7. Cari data berdasarkan NIM");
    System.out.println(x:"0. Keluar");
    System.out.print(s:"Pilih: ");
    pilihan = sc.nextInt();
```

14. Tambahkan switch case untuk menjalankan menu pilihan di atas

```
    switch (pilihan) {
     case 1 -> {
         Mahasiswa01 mhs = inputMahasiswa(scan);
         list.addFirst(mhs);
     }
     case 2 -> {
         Mahasiswa01 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();
         Node01 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!");
}

} while (pilih != 0);

sc.close();
```

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

```
} while (pilih != 0);

sc.close();
```

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

```
public static Mahasiswa14 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 Mahasiswa14(nim, nama, kelas, ipk);
}
```

12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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
```

```
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
```

12.2.3 Pertanyaan Percobaan

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?

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

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

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

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.

Jawaban Pertanyaan

1. single lingked list memiliki pointer next saja sedangkan double lingked list memiliki pointer next dan prev
2. pointer next digunakan untuk node menunjuk ke node selanjutnya sedangkan pointer prev adalah menunjuk node sebelumnya
3. head = null dan tail = null adalah set default pada double linked list yang masih kosong, jadi head dan tailnya di set ke null terlebih dahulu
4. maksudnya adalah jika linked list masih kosong maka head dan tailnya akan langsung di set ke newnode
5. pointer prev dari node yang saat ini merupakan head (yaitu, node yang sebelumnya berada di posisi pertama) akan diatur untuk menunjuk ke newNode. Ini menciptakan tautan balik (backward link) dari node lama ke newNode yang baru saja ditambahkan di posisi paling depan
6.

```
public void print() {
    if (!isEmpty()) {
        Node14 tmp = head;
        while (tmp != null) {
            tmp.data.tampil();
            System.out.println("------------------");
            tmp = tmp.next;
        }
    } else {
        System.out.println("Linked list kosong");
    }
}
```

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.

12.3 Kegiatan Praktikum 2

12.3.1 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("Data yang dihapus:");
        head.data.tampil();
        head = head.next;
        if (head != null) {
            head.prev = null;
        } else {
            tail = null;
        }
    } else {
        System.out.println("Linked list kosong");
    }
}
```

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

```java
public void removeLast() {
    if (!isEmpty()) {
        System.out.println("Data yang dihapus:");
        tail.data.tampil();
        tail = tail.prev;
        if (tail != null) {
            tail.next = null;
        } else {
            head = null;
        }
    } else {
        System.out.println("Linked list kosong");
    }
}
```

12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

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

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

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

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

12.3.3 Pertanyaan Percobaan

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 Pertanyaan

1. posisi head akan dipindah ke node setelah node yang merupakan posisi head saat ini, sedangkan pointer prev dari head yang telah dipindah itu akan di nullkan sehingga menghapus posisi pertama karena tidak akan bisa diakses

2.
```java
public void removeFirst() {
    if (!isEmpty()) {
        System.out.println(x:"Data yang dihapus:");
        head.data.tampil();
        head = head.next;
        if (head != null) {
            head.prev = null;
        } else {
            tail = null;
        }
    } else {
        System.out.println(x:"Linked list kosong");
    }
}
```

12.5 Tugas Praktikum

1. Tambahkan fungsi add() pada kelas DoubleLinkedList untuk menambahkan node pada indeks tertentu
2. Tambahkan removeAfter() pada kelas DoubleLinkedList untuk menghapus node setelah data *key.*
3. Tambahkan fungsi remove() pada kelas DoubleLinkedList untuk menghapus node pada indeks tertentu.
4. Tambahkan fungsi getFirst(), getLast() dan getIndex() untuk menampilkan data pada node head, node tail dan node pada indeks tertentu.
5. tambahkan kode program dan fungsi agar dapat membaca size/ jumlah data pada Double Linked List

Jawaban Tugas

SAYA TAMBAH VARIABEL BARU PADA DoubleLinkedList.java

```java
int size;
```

1.
```java
public void add(int index, Mahasiswa14 item) {
    if (index < 0 || index > size) {
        System.out.println(x:"Index tidak valid.");
        return;
    }

    if (index == 0) {
        addFirst(item);
    } else if (index == size) {
        addLast(item);
    } else {
        Node14 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }

        Node14 newNode = new Node14(current.prev, item, current);
        current.prev.next = newNode;
        current.prev = newNode;
        size++;
    }
}
```

2.
```java
public void removeAfter(String keyNim) {
    Node14 current = head;

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

    if (current == null || current.next == null) {
        System.out.println("Node setelah NIM " + keyNim + " tidak ditemukan atau tidak ada.");
        return;
    }

    Node14 toDelete = current.next;
    current.next = toDelete.next;

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

    size--;
    System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus.");
}
```

3.

```
public void remove(int index) {
    if (index < 0 || index >= size) {
        System.out.println(x:"Index tidak valid.");
        return;
    }

    if (index == 0) {
        removeFirst();
    } else if (index == size - 1) {
        removeLast();
    } else {
        Node14 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(x:"Data yang dihapus:");
        current.data.tampil();
        size--;
    }
}
```

4.
```
public void getFirst() {
    if (!isEmpty()) {
        System.out.println(x:"Data pertama:");
        head.data.tampil();
    } else {
        System.out.println(x:"Linked list kosong.");
    }
}
 public void getLast() {
    if (!isEmpty()) {
        System.out.println(x:"Data terakhir:");
        tail.data.tampil();
    } else {
        System.out.println(x:"Linked list kosong.");
    }
}
public void getIndex(int index) {
    if (index < 0 || index >= size) {
        System.out.println(x:"Index tidak valid.");
        return;
    }

    Node14 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }

    System.out.println("Data pada index ke-" + index + ":");
    current.data.tampil();
}
```

5.
```
public int getSize() {
    return size;
}
```

--- *** ---