Nama : Zaskia Maulidina Mutiara Hati

NIM : 244107060056

Kelas : SIB 1A

# JOBSHEET 12 PRAKTIKUM ALGORITMA STRUKTUR DATA

## Praktikum 1

- **Class Mahasiswa01**

jobsheet_12 > Mahasiswa01.java > Mahasiswa01

```java
package jobsheet_12;

public class Mahasiswa01 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa01(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);
    }
}
```

- **Class Node01**

jobsheet_12 > Node01.java > Node01

```java
package jobsheet_12;

public class Node01 {
    Mahasiswa01 data;
    Node01 prev;
    Node01 next;

    public Node01(Mahasiswa01 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

- **Class DoubleLinkedList01**

jobsheet_12 > DoubleLinkedList01.java > DoubleLinkedList01

```java
package jobsheet_12;

public class DoubleLinkedList01 {
    Node01 head;
    Node01 tail;

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

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

    public void addFirst(Mahasiswa01 data) {
        Node01 newNode = new Node01(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }
```

```java
    public void addLast(Mahasiswa01 data) {
        Node01 newNode = new Node01(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }
```

```java
    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);
    }
```

```java
    public void print() {
        Node01 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

    public Node01 search(String nim) {
        Node01 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
```

- **Class DLLMain**

jobsheet_12 > DLLMain.java > DLLMain

```java
package jobsheet_12;
import java.util.Scanner;
public class DLLMain {
    Run | Debug
    public static void main(String[] args) {
        DoubleLinkedList01 list = new DoubleLinkedList01();
        Scanner scan = 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. Hapus dari awal");
            System.out.println("4. Hapus dari 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();

            switch (pilihan) {
                case 1 : {
                    Mahasiswa01 mhs = inputMahasiswa(scan);
                    list.addFirst(mhs);
                    break;
                }
                case 2 : {
                    Mahasiswa01 mhs = inputMahasiswa(scan);
                    list.addLast(mhs);
                    break;
                }
                //case 3 : list.removeFirst(); break;
                //case 4 : list.removeLast(); break;
                case 5 : list.print(); break;
                case 6 : {
                    System.out.print("Masukkan NIM yang dicari: ");
                    String nim = scan.nextLine();
                    Node01 found = list.search(nim);
                    if (found != null) {
                        System.out.println("Data ditemukan:");
                        found.data.tampil();
                    } else {
                        System.out.println("Data tidak ditemukan.");
                    }
                    break;
                }
                case 0 : System.out.println("Keluar dari program."); break;
                default: System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 0);
        scan.close();
    }

    public static Mahasiswa01 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 Mahasiswa01(nim, nama, kelas, ipk);
    }
}
```

- **Output**

```
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. 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 dari awal
4. Hapus dari 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**

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
   - Singel Linked List : Hanya memiliki 1 pointer (next) dan arah datanya cuman bisa satu arah.
   - Double Linked List : Memiliki 2 pointer (prev dan next) dan arah datanya dua arah (maju mundur).

2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   - Next : Untuk menyimpan Alamat node selanjutnya.
     Prev : Untuk menyimpan Alamat node sebelumnya.

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

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

   - Konstruktor tersebut digunakan untuk menginisialisasi linked list dalam keadaan kosong, dengan menyetel head dan tail ke null, yang berarti belum ada node didalam list.

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

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

   - Jika double linked list masih kosong maka node yang baru ditambahkan (newNode) akan menjadi head dan juga menjadi tail.

5. Perhatikan pada method **addFirst()**. Apakah arti statement head.prev = newNode ?
   - Node yang sebelumnya menjadi head, menunjuk kembali ke node yang baru lewat pointer prev.

6. Modifikasi code pada fungsi print() agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.
   -
```
    public void print() {
        if (isEmpty()) {
            System.out.println("List Masih kosong.");
            return;
        }
        Node01 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }
```

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

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

   - Baris current.next.prev = newNode; digunakan untuk memperbarui node **setelah current** supaya menunjuk kembali ke newNode melalui pointer prev.

8. Modifikasi menu pilihan dan switch-case agar fungsi insertAfter() masuk ke dalam menu pilihan dan dapat berjalan dengan baik.
   -
```
        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 dari awal");
            System.out.println("4. Hapus dari akhir");
            System.out.println("5. Tampilkan data");
            System.out.println("6. Cari Mahasiswa berdasarkan NIM");
            System.out.println("7. Tambah setelah data tertentu");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = scan.nextInt();
            scan.nextLine();
```

```
                case 7 : {
                    System.out.print("Masukkan NIM sebelum data yang akan diinputkan: ");
                    String keyNim = scan.nextLine();
                    Mahasiswa01 mhs = inputMahasiswa(scan);
                    list.insertAfter(keyNim, mhs);
                    break;
```

```
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. Cari Mahasiswa berdasarkan NIM
7. Tambah setelah data tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 1234567
Masukkan Nama: Iak
Masukkan Kelas: 1A
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. Cari Mahasiswa berdasarkan NIM
7. Tambah setelah data tertentu
0. Keluar
Pilih menu: 7
Masukkan NIM sebelum data yang akan diinputkan: 1234567
Masukkan NIM: 7654321
Masukkan Nama: Rara
Masukkan Kelas: 1C
Masukkan IPK: 4,0
Node berhasil disisipkan setelah NIM 1234567
```

```
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. Cari Mahasiswa berdasarkan NIM
7. Tambah setelah data tertentu
0. Keluar
Pilih menu: 5
NIM: 1234567, Nama: Iak, Kelas: 1A, IPK: 4.0
NIM: 7654321, Nama: Rara, Kelas: 1C, IPK: 4.0
```

**Praktikum 2**

```
88      public void removeFirst() {
89          if (isEmpty()) {
90              System.out.println(x:"List kosong, tidak bisa dihapus.");
91              return;
92          }
93          if (head == tail) {
94              head = tail = null;
95          } else {
96              head = head.next;
97              head.prev = null;
98          }
99      }
100
101     public void removeLast() {
102         if (isEmpty()) {
103             System.out.println(x:"List kosong, tidak bisa dihapus.");
104             return;
105         }
106         if (head == tail) {
107             head = tail = null;
108         } else {
109             tail = tail.prev;
110             tail.next = null;
111         }
112     }
113 }
```

- **Output**

```
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. 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. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 3
```

**Pertanyaa 2**

1. Apakah maksud statement berikut pada method removeFirst()?
   head = head.next;
   head.prev = null;
   - Head = head.next berarti memindahkan head ke node berikutnya, kemudian head.prev = null memastikan bahwa node yang menjadi head sekarang tidak menunjuk ke node sebelumnya.

2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “
   -
```
92      public void removeFirst() {
93          if (isEmpty()) {
94              System.out.println(x:"List kosong, tidak bisa dihapus.");
95              return;
96          }
97
98          Mahasiswa01 removedData = head.data;
99
100         if (head == tail) {
101             head = tail = null;
102         } else {
103             head = head.next;
104             head.prev = null;
105         }
106
107         System.out.println(x:"Data sudah berhasil dihapus. Data yang dihapus adalah: " );
108         removedData.tampil();
109     }
```

```
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. Cari Mahasiswa berdasarkan NIM
7. Tambah setelah data tertentu
0. Keluar
Pilih menu: 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Griffyndor
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. Cari Mahasiswa berdasarkan NIM
7. Tambah setelah data tertentu
0. Keluar
Pilih menu: 3
Data sudah berhasil dihapus. Data yang dihapus adalah:
NIM: 20304050, Nama: Hermione, Kelas: Griffyndor, IPK: 4.0
```

**Tugas**

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

```
124     public void add(int index, Mahasiswa01 data) {
125         if (index < 0) {
126         System.out.println(x:"Index tidak boleh negatif.");
127         return;
128         }
129         if (index == 0) {
130         addFirst(data);
131         return;
132         }
133 
134         Node01 current = head;
135         int i = 0;
136         while (current != null && i < index - 1) {
137         current = current.next;
138         i++;
139         }
140         if (current == null) {
141         System.out.println(x:"Index melebihi panjang list.");
142         return;
143         }
144         if (current == tail) {
145         addLast(data);
146         return;
147         }
148 
149         Node01 newNode = new Node01(data);
150         newNode.next = current.next;
151         newNode.prev = current;
152         current.next.prev = newNode;
153         current.next = newNode;
154     }
```

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

```
248     public void removeAfter(String keyNim) {
249         if (isEmpty()) {
250         System.out.println(x:"List kosong, tidak bisa dihapus.");
251         return;
252         }
253 
254         Node01 current = head;
255 
256         while (current != null && !current.data.nim.equals(keyNim)) {
257         current = current.next;
258         }
259 
260         if (current == null) {
261         System.out.println("Node dengan NIM " + keyNim + " tidak ditemukan.");
262         return;
263         }
264 
265         if (current.next == null) {
266         System.out.println("Tidak ada node setelah NIM " + keyNim + " yang bisa dihapus.");
267         return;
268         }
269 
270         Node01 toRemove = current.next;
271 
272         if (toRemove == tail) {
273         tail = current;
274         current.next = null;
275         } else {
276         current.next = toRemove.next;
277         toRemove.next.prev = current;
278         }
280         System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus. Data yang dihapus:");
281         toRemove.data.tampil();
282         size--;
283     }
```

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

```
156     public void remove(int index) {
157         if (isEmpty()) {
158         System.out.println(x:"List kosong, tidak bisa dihapus.");
159         return;
160         }
161         if (index < 0) {
162         System.out.println(x:"Index tidak boleh negatif.");
163         return;
164         }
165         if (index == 0) {
166         removeFirst();
167         return;
168         }
169 
170         Node01 current = head;
171         int i = 0;
172 
173         while (current != null && i < index) {
174         current = current.next;
175         i++;
176         }
177         if (current == null) {
178         System.out.println(x:"Index melebihi panjang list.");
179         return;
180         }
181         if (current == tail) {
182         removeLast();
183         return;
184         }
185 
186         current.prev.next = current.next;
187         current.next.prev = current.prev;
188 
189         System.out.println(x:"Data sudah berhasil dihapus. Data yang dihapus adalah:");
190         current.data.tampil();
191     }
```

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

```
193     public void getFirst() {
194         if (isEmpty()) {
195         System.out.println(x:"List kosong.");
196         } else {
197         System.out.println(x:"Data pada node head:");
198         head.data.tampil();
199         }
200     }
201 
202     public void getLast() {
203         if (isEmpty()) {
204         System.out.println(x:"List kosong.");
205         } else {
206         System.out.println(x:"Data pada node tail:");
207         tail.data.tampil();
208         }
209     }
```

```
211     public void getIndex(int index) {
212         if (isEmpty()) {
213         System.out.println(x:"List kosong.");
214         return;
215         }
216         if (index < 0) {
217         System.out.println(x:"Index tidak boleh negatif.");
218         return;
219         }
220 
221         Node01 current = head;
222         int i = 0;
223 
224         while (current != null && i < index) {
225         current = current.next;
226         i++;
227         }
228         if (current == null) {
229         System.out.println(x:"Index melebihi panjang list.");
230         } else {
231         System.out.println("Data pada index ke-" + index + ":");
232         current.data.tampil();
233         }
234     }
235 }
```

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

```
3   public class DoubleLinkedList01 {
4       Node01 head;
5       Node01 tail;
6       int size = 0;
7   
8       public DoubleLinkedList01() {
9           head = null;
10          tail = null;
11      }
12  
13      public int getSize() {
14          return size;
15      }
```

- 
- Dan menambahkan size++ dan size - -, disetiap method.

https://github.com/iakmorales/ASD.git