Modul

3

Linked List

Tujuan

Mahasiswa diharapkan dapat menerapkan penggunaan linkedlist pada suatu kasus. Secara khusus program yang mengandung instruksi-instruksi yang menerapkan bagian dari linkedlist. Mahasiswa juga dapat menggunakan object pada penggunaan class sesuai dengan kasus yang akan diselesaikan. Di akhir praktikum ini mahasiswa dapat:

- Menjelaskan konsep dasar linkedlist
- Memahami bagaimana penerapan node pada linkedlist
- Memahami bagaimana menggunakan linkedlist pada suatu kasus

Pendahuluan

Praktikum ini mengasumsikan bahwa mahasiswa telah dapat mengoperasikan Netbeans. Mahasiswa juga diharapkan sudah memahami dengan baik materi-materi yang telah diberikan sebelumnya (pemrograman dasar I). Agar mahasiswa dapat mencapai tujuan dalam pertemuan praktikum yang pertama ini, mahasiswa harus memiliki pengetahuan (minimal telah membaca) penggunaan konsep dasar class dan object. Bagian dari class dan object yaitu: variabel, constructor dan method. Pemahaman tentang penggunaan tipe data dan variabel juga dibutuhkan dalam praktikum ini.

Proses

Praktikan membaca buku/diktat yang dilakukan selama 20 menit termasuk membuat ringkasan penting, kemudian berdiskusi sesuai dengan panduan aktifitas yang dilakukan selama 20 menit, sedangkan waktu untuk mengerjakan *project* (membuat program/*coding*) secara individual harus diselesaikan di dalam laboratorium dalam waktu 30 menit. Berikutnya untuk sesi latihan, ada soal tentang pengembangan program yang juga harus diselesaikan di laboratorium dalam waktu 50 menit. Terakhir adalah bagian tugas, yaitu *project* yang dikerjakan dirumah dan wajib dikumpulkan pada pertemuan berikutnya.

Aktifitas

1. Mahasiswa membaca buku ajar (jika ada)/diktat kuliah/materi dari sumber lain tentang konsep dasar class dan object sebagai materi bab pertama. Temukan bagian penting dalam topik class dan object ini, kemudian tulis sebagai ringkasan hasil belajar.

2. Mahasiswa berdiskusi tentang definisi linkedlist, tujuan adanya node, head dan tail sebagai bahan diskusi

a. Lengkapilah bagian yang kosong pada class LinkedListNode di bawah ini sesuai dengan perintah / command yang tertera.

```
  public class LinkedListNode{
     LinkedListNode next;
     LinkedListNode prev;
     int data;

    /* Constructor
     * set this.data into new_data
     * set this.prev into null
     * set this.next into null
     */
     LinkedListNode(int new_data){
        ……
        ……
        ……

    }

    /* set this.prev into other
     * if other is not null, set other.next into this
     */
     void set_prev(LinkedListNode other){
        ……
        if( ……){
                    ……
        }
    }

    /* set this.next into other
     * if other is not null, set other.prev into this
     */
     void set_next(LinkedListNode other){
            ……
        if( ……){
             ……

        }
    }
}
```

b. Lengkapilah bagian yang kosong pada class LinkedList di bawah ini sesuai dengan perintah / command yang tertera.

```
  public class LinkedList{
    LinkedListNode head;
    LinkedListNode tail;
```

```
    LinkedList(){
        this.head = null;
        this.tail = null;
    }

    /* First set a Node named current into head
     * while current is not null, print current.data, set current =
current.next
     * print end of line
     */
    void print(){
        ……..
        while(……..){
            System.out.print(……..+ " ");
      ……
        }
        System.out.println();
    }

    /* if LinkedList is empty, set new_node as head and tail
     * if LinkedList is not empty, set tail.next into new_node, set
new_node.prev into tail, and make new_node a new tail
     */
    void push(LinkedListNode new_node){
        if(……){
      …….
      …….
        }else{
           ……(new_node);
            ……..;
        }
    }
```

```
/* if linked list is empty, set new_node as head and tail
     * if new_node < head, make it a new head
     * if new_node > tail, make it a new tail
     * otherwise traverse to the current position, and put new_node
there
     */
    void insert(LinkedListNode new_node){
        if(……..){
            ……..
           ……
        }else if(new_node.data <= this.head.data){
            …….(new_node);
            ……..
        }else if(new_node.data >= this.tail.data){
      …….(new_node);
      …….
        }else{
            ……
            while(……){
                ……..
            }
            LinkedListNode previous_position = position.prev;
            ……..(previous_position);
            ……..(position);
        }
    }
```

```
    LinkedListNode find_node_by_data(int data){
        LinkedListNode current = this.head;
        while(current != null){
            if(current.data == data){
                return current;
            }
        }
        return null;
    }
```

```
   void delete(LinkedListNode deleted){
        if(deleted != null && this.head != null){
            if(this.head == this.tail && deleted == this.head){
                this.head = null;
                this.tail = null;
            }else if(deleted == this.head){
                LinkedListNode new_head = this.head.next;
                this.head.set_next(null);
                new_head.set_prev(null);
                this.head = new_head;
            }else if(deleted == this.tail){
                LinkedListNode new_tail = this.tail.prev;
                this.tail.set_prev(null);
                new_tail.set_next(null);
                this.tail = new_tail;
            }else{
                LinkedListNode deleted_prev = deleted.prev;
                LinkedListNode deleted_next = deleted.next;
                deleted.set_prev(null);
                deleted.set_next(null);
                deleted_prev.set_next(deleted_next);
            } } } }
```

c. Simpan class LinkedListNode.java dan LinkedList.java pada satu package yang sama. Selanjutnya buat class baru pada package yang sama ketikkan class Test.java di bawah ini dan jalankan.

```
public class Test{
    public static void main(String Args[]){
        LinkedList a = new LinkedList();
        LinkedList a = new LinkedList();
        a.print(); // should show nothing
        a.insert(new LinkedListNode(5));
        a.insert(new LinkedListNode(4));
        a.insert(new LinkedListNode(7));
        a.insert(new LinkedListNode(6));
        a.print(); // should show 4 5 6 7
        a.delete(a.head);
        a.print(); // should show 5 6 7
        a.delete(a.tail.prev);
        a.print(); // should show 5 7
        a.delete(a.tail);
        a.print(); // should show 5
        a.delete(a.head);
        a.print(); // should show nothing
}}
```

```

```

d. Setelah dijalankan akan muncul output seperti di bawah ini :

```
4 5 6 7
5 6 7
5 7
5
```

Kesimpulan

1. Buatlah interface dari program diatas !
2. Tambahkan proses searching pada program di atas!

Penutup

Tugas

1. Buatlah suatu program menggunakan linkedlist dan lakukan sorting menggunakan bubble sort!
2. Buatlah suatu program menggunakan linkedlist dan lakukan sorting menggunakan quick sort!