LAPORAN PRAKTIKUM
MATA KULIAH ALGORITMA DAN STRUKTUR DATA

Dosen Pengampu : Triana Fatmawati, S.T, M.T

PERTEMUAN - 11 - Linked List

Nama : **M. Zidna Billah Faza**

NIM : **2341760030**

Prodi : **D-IV Sistem Informasi Bisnis**

JURUSAN TEKNOLOGI INFORMASI

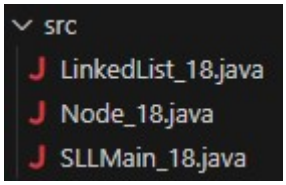
POLITEKNIK NEGERI MALANG

2024

Percobaan 1

1) Buat folder baru Jobsheet 11

2) Tambahkan class-class berikut:
   - Node.java
   - LinkedList.java
   - SLLMain.java



3) Deklarasikan class Node yang memiliki atribut data untuk menyimpan elemen dan atribut next bertipe Node untuk menyimpan node berikutnya. Tambahkan constructor berparameter untuk mempermudah inisialisasi

```
public class Node_18 {

    int data;
    Node_18 next;

    public Node_18(int data, Node_18 next) {

        this.data = data;
        this.next = next;

    }
}
```

4) Deklarasikan class LinkedList yang memiliki atribut head. Atribut head menyimpan node pertama pada linked list

```
public class LinkedList_18 {

    Node_18 head;
```

5) Sebagai langkah berikutnya, akan diimplementasikan method-method yang terdapat pada class LinkedList.

6) Tambahkan method isEmpty()

```
public boolean IsEmpty() {
    return (head == null);
}
```

7) Implementasi method print() untuk mencetak dengan menggunakan proses traverse.

```
public void Print() {
    if (!IsEmpty()) {

        System.out.print(s:"Isi Linked List     : ");
        Node_18 currentNode = head;

        while (currentNode != null) {
            System.out.print(currentNode.data + "\t");
            currentNode = currentNode.next;
        }

        System.out.println(x:"");

    } else {

        System.out.println(x:"Linked List Kosong");

    }
}
```

8) Implementasikan method addFirst() untuk menambahkan node baru di awal linked list

```
public void AddFirst(int input) {

    Node_18 newNode = new Node_18(input, next:null);

    if (IsEmpty()) {
        head = newNode;
    } else {
        newNode.next = head;
        head = newNode;
    }

}
```

9) Implementasikan method addLast() untuk menambahkan node baru di akhir linked list

```
public void AddLast(int input) {

    Node_18 newNode = new Node_18(input, next:null);

    if (IsEmpty()) {
        head = newNode;
    } else {

        Node_18 currentNode = head;

        while (currentNode.next != null) {
            currentNode = currentNode.next;
        }

        currentNode.next = newNode;

    }
}
```

10) Implementasikan method insertAfter() menambahkan node baru pada posisi setelah node yang berisi data tertentu (key)

```
public void InsertAfter(int key, int input) {

    Node_18 newNode = new Node_18(input, next:null);

    if (!IsEmpty()) {

        Node_18 currentNode = head;

        do {

            if (currentNode.data == key) {
                newNode.next = currentNode.next;
                currentNode.next = newNode;
                break;
            }

            currentNode = currentNode.next;

        } while (currentNode != null);

    } else {

        System.out.println(x:"Linked List Kosong");

    }

}
```

11) Pada class SLLMain, buatlah fungsi main, kemudian buat object myLinkedList bertipe LinkedList. Lakukan penambahan beberapa data. Untuk melihat efeknya terhadap object myLinkedList, panggil method print()

```
public class SLLMain_18 {

    Run | Debug
    public static void main(String[] args) {

        LinkedList_18 myLinkedList_18 = new LinkedList_18();

        myLinkedList_18.Print();
        myLinkedList_18.AddFirst(input:800);
        myLinkedList_18.Print();
        myLinkedList_18.AddFirst(input:700);
        myLinkedList_18.Print();
        myLinkedList_18.AddLast(input:500);
        myLinkedList_18.Print();
        myLinkedList_18.InsertAfter(key:700, input:300);
        myLinkedList_18.Print();
```

12) Verifikasi hasil percobaan

```
Linked List Kosong
Isi Linked List     : 800
Isi Linked List     : 700       800
Isi Linked List     : 700       800     500
Isi Linked List     : 700       300     800     500
```

Pertanyaan Percobaan 1

1) Mengapa class LinkedList tidak memerlukan method isFull() seperti halnya Stack dan Queue?

Karena pada class LinkedList tidak memiliki batasan ukuran tertentu karena bersifat dinamis sehingga node-node baru daoat ditambahkan selama memori masih ada

2) Mengapa class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama? Bagaimana informasi node kedua dan lainnya diakses?

Karena head adalah titik awal dari LinkedList. Dari head, kita dapat mengakses seluruh LinkedList karena setiap node memiliki referensi ke node berikutnya (disebut 'next').

3) Pada langkah, jelaskan kegunaan kode berikut

```
if (currentNode.data == key) {
    newNode.next = currentNode.next;
    currentNode.next = newNode;
    break;
}
```

Kode ini digunakan untuk menyisipkan sebuah node baru setelah node yang memiliki nilai data yang sama dengan kunci (key). Jadi, jika dalam LinkedList terdapat node dengan nilai data yang sama dengan kunci (key), maka kode ini akan menyisipkan node baru di antara node tersebut dan node berikutnya.

4) Implementasikan method insertAt(int index, int key) dari tugas mata kuliah ASD (Teori)

```
public void insertAt(int index, int key) {
    Node_18 newNode = new Node_18(key, next:null);

    if (index < 0) {
        System.out.println(x:"Indeks tidak valid");
        return;
    }

    if (index == 0) {
        AddFirst(key);
    } else {
        Node_18 currentNode = head;
        int currentIndex = 0;

        while (currentNode != null && currentIndex < index - 1) {
            currentNode = currentNode.next;
            currentIndex++;
        }

        if (currentNode == null) {
            System.out.println(x:"Indeks melebihi ukuran linked list");
        } else {
            newNode.next = currentNode.next;
            currentNode.next = newNode;
        }
    }
}
```

```
System.out.println(x:"Pertanyaan Percobaan 1 Nomor 4");

myLinkedList_18.InsertAt(index:1,key:100);
myLinkedList_18.Print();
```

```
Pertanyaan Percobaan 1 Nomor 4
Isi Linked List     : 800       100
```

Percobaan 2

1) Tambahkan method getData() untuk mengembalikan nilai elemen di dalam node pada index tertentu

```
public int GetData(int index) {

    Node_18 currentNode = head;

    for (int i = 0; i < index; i++) {
        currentNode = currentNode.next;
    }

    return currentNode.data;

}
```

2) Tambahkan method indexOf() untuk mengetahu index dari node dengan elemen tertentu

```
public int IndexOf(int key) {

    Node_18 currentNode = head;
    int index = 0;

    while (currentNode != null && currentNode.data != key) {
        currentNode = currentNode.next;
        index++;
    }

    if (currentNode == null) {
        return -1;
    } else {
        return index;
    }

}
```

3) Tambahkan method removeFirst() untuk menghapus node pertama pada linked list

```
public void RemoveFirst() {

    if (!IsEmpty()) {
        head = head.next;
    } else {
        System.out.println(x:"Linked List Kosong");
    }

}
```

4) Tambahkan method removeLast() untuk menghapus node terakhir pada linked list

```
public void RemoveLast() {

    if (IsEmpty()) {
        System.out.println(x:"Linked List Kosong");
    } else if (head.next == null) {
        head = null;
    } else {

        Node_18 currentNode = head;

        while (currentNode.next != null) {
            if (currentNode.next.next == null) {
                currentNode.next= null;
                break;
            }

            currentNode = currentNode.next;

        }
    }

}
```

5) Method remove() digunakan untuk mengapus node yang berisi elemen tertentu

```
public void Remove(int key) {

    if (IsEmpty()) {
        System.out.println(x:"Linked List kosong");
    } else if (head.data == key) {
        RemoveFirst();
    } else {

        Node_18 currentNode = head;

        while (currentNode.next != null) {
            if (currentNode.next.data == key) {
                currentNode.next = currentNode.next.next;
                break;
            }

            currentNode = currentNode.next;

        }

    }

}
```

6) Kemudian, coba lakukan pengaksesan dan penghapusan data di method main pada class SLLMain dengan menambahkan kode berikut

```
System.out.println("Data pada index ke-1            : " + myLinkedList_18.GetData(index:1));
System.out.println("Data 300 berada pada index ke   : " + myLinkedList_18.IndexOf(key:300));

myLinkedList_18.Remove(key:300);
myLinkedList_18.Print();
myLinkedList_18.RemoveFirst();
myLinkedList_18.Print();
myLinkedList_18.RemoveLast();
myLinkedList_18.Print();
```

7) Compile dan run program kemudian amati hasilnya

```
Linked List Kosong
Isi Linked List     : 800
Isi Linked List     : 700       800
Isi Linked List     : 700       800     500
Isi Linked List     : 700       300     800     500
Data pada index ke-1             : 300
Data 300 berada pada index ke    : 1
Isi Linked List     : 700       800     500
Isi Linked List     : 800       500
Isi Linked List     : 800
```

Pertanyaan Percobaan 2

1) Jelaskan maksud potongan kode di bawah pada method remove()

```
if (currentNode.next.data == key) {
    currentNode.next = currentNode.next.next;
    break;
}
```

kode tersebut berfungsi untuk menghapus node dari LinkedList yang memiliki nilai data yang sama dengan kunci (key).

2) Jelaskan maksud if-else block pada method indexOf() berikut

```
if (currentNode == null) {
    return -1;
} else {
    return index;
}
```

Pada method indexOf(), blok if-else tersebut bertujuan untuk menentukan hasil pencarian indeks (index) dari node yang memiliki nilai data sesuai dengan kunci (key).

- Jika tidak ditemukan node dengan nilai data yang sama dengan kunci (key), currentNode akan null. Dalam blok if, jika currentNode null, menandakan kunci (key) tidak ada dalam LinkedList, maka -1 dikembalikan.
- Jika ditemukan node dengan nilai data yang sama dengan kunci (key), currentNode tidak null. Pada blok else, nilai index dari node tersebut dikembalikan.

3) Error apa yang muncul jika argumen method getData() lebih besar dari jumlah node pada linked list? Modifikasi kode program untuk menghandle hal tersebut.

Jika argumen method getData() lebih besar dari jumlah node pada linked list, yang muncul biasanya adalah NullPointerException, karena kita mencoba mengakses node yang tidak ada.

```
if (currentNode == null) {
    System.out.println(x:"Indeks melebihi jumlah node dalam linked list.");
    return -1;
} else {
    return currentNode.data;
}
```

4) Apa fungsi keyword break pada method remove()? Bagaimana efeknya jika baris tersebut dihapus?

Ketika node yang memiliki nilai data yang sama dengan kunci (key) ditemukan dan dihapus, break digunakan untuk keluar dari loop while

Jika baris break tersebut dihapus, iterasi akan terus berlanjut meskipun operasi penghapusan sudah dilakukan. Ini bisa mengakibatkan masalah, seperti mencoba mengakses currentNode.next setelah currentNode menjadi null, yang dapat menyebabkan NullPointerException atau iterasi yang tidak terbatas

Tugas Praktikum

1) Implementasikan method-method berikut pada class LinkedList:
   - insertBefore() untuk menambahkan node sebelum keyword yang diinginkan
   - insertAt(int index, int key) untuk menambahkan node pada index tertentu
   - removeAt(int index) untuk menghapus node pada index tertentu
2) Dalam suatu game scavenger hunt, terdapat beberapa point yang harus dilalui peserta untuk menemukan harta karun. Setiap point memiliki soal yang harus dijawab, kunci jawaban, dan pointer ke point selanjutnya. Buatlah implementasi game tersebut dengan linked list.

Jawaban

1a InsertBefore

```
public void InsertBefore(int key, int input) {
    Node_18 newNode = new Node_18(input, next:null);

    if (!IsEmpty()) {
        if (head.data == key) {
            newNode.next = head;
            head = newNode;
        } else {
            Node_18 currentNode = head;

            while (currentNode.next != null) {
                if (currentNode.next.data == key) {
                    newNode.next = currentNode.next;
                    currentNode.next = newNode;
                    break;
                }

                currentNode = currentNode.next;
            }
        }
    } else {
        System.out.println(x:"Linked List Kosong");
    }
}
```

1b InsertAt

```java
public void InsertAt(int index, int key) {
    Node_18 newNode = new Node_18(key, next:null);

    if (index < 0) {
        System.out.println(x:"Indeks tidak valid");
        return;
    }

    if (index == 0) {
        AddFirst(key);
    } else {
        Node_18 currentNode = head;
        int currentIndex = 0;

        while (currentNode != null && currentIndex < index - 1) {
            currentNode = currentNode.next;
            currentIndex++;
        }

        if (currentNode == null) {
            System.out.println(x:"Indeks melebihi ukuran linked list");
        } else {
            newNode.next = currentNode.next;
            currentNode.next = newNode;
        }
    }
}
```

1c RemoveAt

```java
public void RemoveAt(int index){
    if (IsEmpty()) {
        System.out.println(x:"Linked list kosong");
        return;
    }

    if (index < 0 ) {
        System.out.println(x:"Indeks tidak valid");
        return;
    }

    if (index == 0) {
        RemoveFirst();
        return;
    }

    Node_18 prevNode = null;
    Node_18 currentNode = head;
    int currentIndex = 0;

    while (currentNode != null && currentIndex < index) {
        prevNode = currentNode;
        currentNode = currentNode.next;
        currentIndex++;
    }

    if (currentNode == null) {
        System.out.println(x:"Indeks melebihi ukuran linked list");
    }else{
        prevNode.next = currentNode.next;
    }
}
```

1d Output

```
System.out.println(x:"\nInsert Before");
myLinkedList_18.InsertBefore(key:100, input:250);
myLinkedList_18.Print();

System.out.println(x:"\nInsert At");
myLinkedList_18.InsertAt(index:2, key:450);
myLinkedList_18.Print();

System.out.println(x:"\nRemove At");
myLinkedList_18.RemoveAt(index:3);
myLinkedList_18.Print();
```

```
Insert Before
Isi Linked List     : 800       250     100

Insert At
Isi Linked List     : 800       250     450     100

Remove At
Isi Linked List     : 800       250     450
```

2a Program

```
public class ScavengerHuntNode {
    String question;
    String answer;
    ScavengerHuntNode next;

    public ScavengerHuntNode(String question, String answer) {
        this.question = question;
        this.answer = answer;
        this.next = null;
    }
}
```

```java
import java.util.Scanner;

public class ScavengerHunt {
    ScavengerHuntNode head;

    public ScavengerHunt() {
        this.head = null;
    }

    public void addPoint(String question, String answer) {

        ScavengerHuntNode newNode = new ScavengerHuntNode(question, answer);

        if (head == null) {
            head = newNode;
        } else {
            ScavengerHuntNode current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }

    }

    public void startHunt() {

        Scanner scanner = new Scanner(System.in);
        ScavengerHuntNode current = head;

        while (current != null) {

            System.out.println("Pertanyaan  : " + current.question);
            System.out.print("Jawaban     : ");
            String userAnswer = scanner.nextLine();

            if (userAnswer.equalsIgnoreCase(current.answer)) {
                System.out.println("BENARR! Lanjut ke pertanyaan selanjutnya.");
                current = current.next;
            } else {
                System.out.println("Yahhh Jawaban Salah! Coba lagi yahh.");
            }

            System.out.println();

        }

        System.out.println("SELAMATTT! Kamu Berhasil Menemukan Harta Karun!");
        scanner.close();

    }
}

```

```java
public class ScavengerHuntMain {
    Run | Debug
    public static void main(String[] args) {

        ScavengerHunt scavengerHunt = new ScavengerHunt();

        scavengerHunt.addPoint(question:"Kalau hitam dibilang bersih, kalau putih dibilang kotor?", answer:"Papan Tulis");
        scavengerHunt.addPoint(question:"Sabun, sabun apa yang paling genit?", answer:"Sabun Colek");
        scavengerHunt.addPoint(question:"Gajah apa yang paling baik hati?", answer:"Gajahat");

        scavengerHunt.startHunt();

    }
}
```

2b Output

```
Pertanyaan  : Kalau hitam dibilang bersih, kalau putih dibilang kotor?
Jawaban     : papan tulis
BENARR! Lanjut ke pertanyaan selanjutnya.

Pertanyaan  : Sabun, sabun apa yang paling genit?
Jawaban     : sabun colek
BENARR! Lanjut ke pertanyaan selanjutnya.

Pertanyaan  : Gajah apa yang paling baik hati?
Jawaban     : gajahat
BENARR! Lanjut ke pertanyaan selanjutnya.

SELAMATTT! Kamu Berhasil Menemukan Harta Karun!
```

Repository : https://github.com/zidnafaz/Praktikum-Algoritma-Struktur-Data