Laporan Praktikum Alogaritma dan Struktur Data

Jobsheet 11 : Linked List

Alogaritma dan Struktur Data

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

Eka Putri Natalya Kabelen

2341760107

SIB 1E

PROGRAM STUDI SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

2. Praktikum

2.1 Pembuatan Single Linked List

1. Tambahkan class-class berikut:
   a. Node.java
   b. LinkedList.java
   c. SLLMain.java

2. 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.

```java
/**
 * Node11
 */
public class Node {
    int data;
    Node next;

    public Node(int data, Node next){
        this.data = data;
        this.next = next;
    }
}
```

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

```java
public class LinkedList {
    Node head; //posisi awal linked list
```

4. Sebagai langkah berikutnya, akan diimplementasikan method-method yang terdapat pada class LinkedList. Dan Tambahkan method isEmpty().

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

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

```
7       public void print () {
8           if (!isEmpty()) {
9               Node currentNode = head;
10              System.out.print(s:"Isi Linked List : \t");
11
12              while (currentNode != null) {
13                  System.out.print(currentNode.data + "\t");
14                  currentNode = currentNode.next;
15              }
16              System.out.println(x:"");
17          } else {
18              System.out.println(x:"Linked List Kosong");
19          }
20      }
21      public void addFirst(int input) {
```

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

```
21      public void addFirst(int input) {
22          Node newNode = new Node(input, berikutnya:null);
23
24          if (isEmpty()) {
25              head = newNode;
26          } else {
27              newNode.next = head;
28              head = newNode;
29          }
30      }
```

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

```
31      public void addLast (int input) {
32          Node newNode = new Node(input, berikutnya:null);
33          if (isEmpty()) {
34              head = newNode;
35
36          } else {
37              Node currentNode = head;
38
39              while (currentNode != null) {
40                  currentNode = currentNode.next;
41              }
42              currentNode.next = newNode;
43          }
44      }
```

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

```
45      public void insertAfter (int key, int input) {
46          Node  newNode  = new Node(input, berikutnya:null);
47
48              if (!isEmpty()) {
49                  Node currentNode = head;
50
51              do {
52                  if (currentNode.data == key) {
53                      newNode.next = currentNode.next;
54                      currentNode.next = newNode;
55                  break;
56              }
57              currentNode = currentNode.next;
58          } while (currentNode!= null);
59      } else {
60          System.out.print(s:"Linked List Kosong");
61      }
62  }
```

9. 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().

```java
import java.util.Scanner;
/**
 * SLLMain
 */
public class SLLMain {

    Run | Debug
    public static void main(String[] args) {
        LinkedList singLL = new LinkedList();
        singLL.print();
        singLL.addFirst(input:800);
        singLL.print();
        singLL.addFirst(input:700);
        singLL.print();
        singLL.addLast(input:500);
        singLL.print();
        singLL.insertAfter(key:700, input:300);
        singLL.print();
    }
}
```

10. 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
PS C:\Users\ASUS\Documents\SEMESTER 2\Alogaritma&StrukturData\JOBSHEET 11>
```

2.1.2 Pertanyaan

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

   Jawab : Class LinkedList tidak memerlukan method isFull() karena struktur data LinkedList tidak memiliki batasan maksimum pada jumlah elemen yang dapat disimpan di dalamnya.

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

   Jawab : Class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama karena LinkedList dibangun dengan cara beruntun (berurutan) dari node-node yang saling terhubung. Setiap node dalam LinkedList memiliki dua bagian utama yaitu data dan referensi ke node berikutnya dalam LinkedList. Ketika kita ingin mengakses informasi node kedua atau node lainnya dalam LinkedList, kita mulai dari node pertama (yang disimpan dalam atribut head). Kemudian, kita menggunakan referensi yang disimpan di setiap node untuk melanjutkan ke node berikutnya secara berurutan.

3. Pada langkah, jelaskan kegunaan kode berikut

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

   Jawab :

   Kode yang ditunjukkan dalam gambar adalah bagian dari algoritme untuk menyisipkan elemen baru ke dalam daftar tertaut. Algoritme ini mengasumsikan bahwa daftar tertaut tersebut tidak kosong dan tidak mengandung elemen duplikat.

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

   Jawab :

```java
public void insertAt (int index, int input) {
    if (index < 0) {
        System.out.println(x:"Index salah");
    } else if (index == 0) {
        addFirst(input);
    }else {
        Node  currentNode = head;
        for (int i = 0; i < index -1; i ++) {
        }
        currentNode.next = new Node(input, currentNode.next);
        if (currentNode.next.next==null) currentNode = currentNode.next;
    }
}
```

```java
        singLL.insertAt(3, 400);
        singLL.print();
```

```
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
Isi Linked List :       700     400     300     800     500
Data pada indekx ke-1 : 400
Data 300 berada pada indeks ke:2
```

2.2 Mengakses dan menghapus node pada Linked List

2.2.1 Langkah-langkah Percobaan

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

```java
public int gedData (int index){
    Node  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

```
72  public int indexOf (int key){
73      Node currentNode = head;
74      int index = 0;
75
76      while (currentNode != null && currentNode.data != key){
77          currentNode = currentNode.next;
78          index++;
79      }
80      if(currentNode == null){
81          return -1;
82      }else{
83          return index;
84      }
85  }
```

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

```
86  public void removeFirst(){
87      if(! isEmpty()){
88          head = head.next;
89      }else {
90          System.out.println(x:"Linked List masih Kosong");
91      }
92  }
```

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

```
4   public void removeLast(){
5       if(isEmpty()){
6           System.out.println(x:"Linked List masih Kosong");
7       }else if(head.next == null){
8           head = null;
9       }else{
0           Node currentNode = head;
1
2           while(currentNode.next != null){
3               if (currentNode.next.next!= null) {
4                   currentNode.next = null;
5                   break;
6               }
7               currentNode = currentNode.next;
8       }
9   }
0   }
```

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

```
public void remove (int key) {
    if(isEmpty()){
        System.out.println(x:"Linked List masih Kosong");
    }else if (head.data == key) {
        removeFirst();
    } else {
        Node currentNode = head;

        while(currentNode != 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 indekx ke-1 : " + singLL.gedData(index:1));
        System.out.println("Data 300 berada pada indeks ke:" + singLL.indexOf(key:300));

        singLL.remove(key:300);
        singLL.print();
        singLL.removeFirst();
        singLL.print();
        singLL.removeLast();
        singLL.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 indekx ke-1 : 300
Data 300 berada pada indeks ke:1
Isi Linked List :       700     800     500
Isi Linked List :       800     500
Isi Linked List :       800     500
PS C:\Users\ASUS\Documents\SEMESTER 2\Alogaritma&StrukturData\JOBSHEET 11>
```

2.2.2 Pertanyaan

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

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

Jawab :

Potongan kode yang ditunjukkan dalam gambar merupakan bagian dari method remove() yang digunakan untuk menghapus kunci (key) dari sebuah string. Kode ini bekerja dengan cara memeriksa kunci berikutnya pada linked list, menghapus node berikutnya jika kunci ditemukan, dan keluar dari loop setelah kunci dihapus.

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

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

Jawab :

Yang dimaksud Blok if-else dalam method indexOf() ini yaitu digunakan untuk menentukan apakah elemen yang dicari ada dalam daftar tertaut atau tidak. Jika elemen ditemukan, method mengembalikan posisinya. Jika elemen tidak ditemukan, method mengembalikan nilai -1.

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

Jawab :

Error yang terjadi muncul jika argumen method getData() lebih besar dari jumlah node pada linked list yaitu Exception in thread "main" java.lang.NullPointerException: Cannot read field "next" because "currentNode" is null, berikut modifikasi kode program untuk menghandle hal tersebut.

```
public int getData (int index){
    Node14  currentNode = head;
    for(int i=0; i > index; i++){
        currentNode = currentNode.next;
    }
    return currentNode.data;
}
```

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

   Keyword break digunakan dalam sebuah pernyataan switch atau loop seperti for atau while untuk menghentikan eksekusi dari loop atau switch statement tersebut. Dalam konteks method remove() pada suatu linked list, penggunaan break mungkin terjadi dalam sebuah loop yang digunakan untuk mencari elemen yang ingin dihapus. Jika baris break dihapus dari metode remove(), maka loop akan terus berjalan bahkan setelah node yang sesuai telah dihapus. Ini akan menyebabkan loop untuk melanjutkan pencarian node dengan nilai data yang sama, bahkan setelah itu telah dihapus. Hal ini dapat mengakibatkan kesalahan dalam operasi penghapusan atau bahkan dapat menyebabkan program mengalami kegagalan saat mencoba mengakses node yang telah dihapus. Dengan kata lain, menghapus baris break dapat menyebabkan loop menjadi tidak efektif dalam konteks penghapusan node.

3. Tugas

1. Implementasikan method-method berikut pada class LinkedList:
   a. insertBefore() untuk menambahkan node sebelum keyword yang diinginkan

```
64  public void insertBefore(int key, int input){
65      Node newNode = new Node(input, next:null);
66      Node currentNode = head;
67
68      if(currentNode == null) { // Linked list kosong, node baru akan menjadi node pertama
69          this.addFirst(input);
70          return;
71      }
72
73      if(currentNode.data == key) { // Jika key ada pada node pertama, sisipkan node baru sebelumnya
74          this.addFirst(input);
75          return;
76      }
77
78      while(currentNode.next != null){
79          if(currentNode.next.data == key){ // Jika key ada pada node selanjutnya
80              newNode.next = currentNode.next;
81              currentNode.next = newNode;
82              return;
83          }
84          currentNode = currentNode.next;
85      }
86
87      System.out.println("Node dengan key " + key + " tidak ditemukan dalam linked list");
88  }
```

   b. insertAt(int index, int key) untuk menambahkan node pada index tertentu

```
82      public void insertAt (int index, int key) {
83          if (index < 0) {
84              System.out.println(x:"Index salah");
85          } else if (index == 0) {
86              addFirst(key);
87          }else {
88              Node  currentNode = head;
89              for (int i = 0; i < index -1; i ++) {
90              }
91              currentNode.next = new Node(key, currentNode.next);
92              if (currentNode.next.next==null) currentNode = currentNode.next;
93          }
94      }
```

c. removeAt(int index) untuk menghapus node pada index tertentu

```
143     public void removeAt(int index){
144         if (index == 0){
145             removeFirst();
146         }else{
147             Node temp = head;
148             for(int i=0; i<index-1; i++){
149                 temp = temp.next;
150             }
151             temp.next = temp.next.next;
152             if(temp.next == null){
153             }
154         }
155         }
```

ssl main

```
singLL.insertAt(index:3, key:400);
singLL.print();
singLL.insertBefore(key:890, input:100);
singLL.print();
```

Hasil run

```
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
Isi Linked List :       700     400     300     800     500
Node dengan key 890 tidak ditemukan dalam linked list
Isi Linked List :       700     400     300     800     500
Data pada indekx ke-1 : 400
Data 300 berada pada indeks ke:2
Isi Linked List :       700     400     800     500
Isi Linked List :       400     800     500
Isi Linked List :       800     500
Isi Linked List :       800     500
PS C:\Users\ASUS\Documents\SEMESTER 2\Alogaritma&StrukturData\JOBSHEET 11>
```

2. Dalam suatu game scavenger hunt, terdapat beberapa point yang harus dilalui pesertauntuk menemukan harta karun. Setiap point memiliki soal yang harus dijawab, kunci jawaban, dan pointer ke point selanjutnya. Buatlah implementasi game tersebut dengan linked list.

```java
import java.util.Scanner;
import org.w3c.dom.Node;
public class ScavengerHuntGame11 {


    private NodeGame11 head;

    public ScavengerHuntGame11() {
        head = null;
    }

    public void addNode(String question, String answer) {
        if (head == null) {
            head = new NodeGame11(question, answer);
        } else {
            NodeGame11 current = head;
            while (current.getNext() != null) {
                current = current.getNext();
            }
            current.setNext(new NodeGame11(question, answer));
        }
    }

    public void play() {
        Scanner scanner = new Scanner(System.in);
        NodeGame11 current = head;
        while (current != null) {
            System.out.println(current.getQuestion());
            String answer = scanner.nextLine();
            if (current.getAnswer().equalsIgnoreCase(answer)) {
                System.out.println("Benar! Lanjut ke pertanyaan selanjutnya...");
                current = current.getNext();
            } else {
                System.out.println("SALAH! Game over.");
                break;
            }
        }
        if (current == null) {
            System.out.println("Selamat! Anda telah menjawab seluruh pertanyaan");
        }
        scanner.close();
    }
}

```

```
Apa makanan khas padang ?
Rendang
Benar! Lanjut ke pertanyaan selanjutnya...
 Dimana letak candi borobudur?
Yogya
Benar! Lanjut ke pertanyaan selanjutnya...
 siapa nama presiden ke 7 RI?
Jokowi
Benar! Lanjut ke pertanyaan selanjutnya...
Selamat! Anda telah menjawab seluruh pertanyaan
PS D:\Prak Algoritma dan Struktur Data\jobsheet 11\jobsheet-11>
```