Laporan Praktikum Algoritma & Struktur Data

Jobsheet 11 : Linkded List

Nama: Aqil Rahmat Alifiandi

NIM: 2341760107

Kelas: SIB 1E

PROGRAM STUDI SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

Praktikum 1

1. Buat folder baru Praktikum09
2. Tambahkan class-class berikut: a. Node.java b. LinkedList.java c. 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 Node05 {
    int data;
    Node05 next;

    public Node05(int data, Node05 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 LinkedList05 {
    Node05 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()) {
        Node05 currentNode = head;
        System.out.print(s:"Isi Linked List : \t");

        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) {
    Node05 newNode = new Node05(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) {
    Node05 newNode = new Node05(input, next:null);

    if (isEmpty()) {
        head = newNode;

    } else {
        Node05 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)

```java
public void insertAfter (int key, int input) {
    Node05 newNode = new Node05(input, next:null);

        if (!isEmpty()) {
            Node05 currentNode = head;

        do {
            if (currentNode.data == key) {
                newNode.next = currentNode.next;
                currentNode.next = newNode;
            break;
        }
        currentNode = currentNode.next;
    } while (currentNode!= null);
} else {
    System.out.print(s:"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()

```java
import java.util.Scanner;

public class SLLMain05 {

    Run | Debug
    public static void main(String[] args) {
        LinkedList05 myLinkedList05 = new LinkedList05();
        myLinkedList05.print();
        myLinkedList05.addFirst(input:800);
        myLinkedList05.print();
        myLinkedList05.addFirst(input:700);
        myLinkedList05.print();
        myLinkedList05.addLast(input:500);
        myLinkedList05.print();
        myLinkedList05.insertAfter(key:700, input:300);
        myLinkedList05.print();
    }
}
```

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

1. Mengapa class LinkedList tidak memerlukan method isFull() seperti halnya Stack dan Queue?
   Jawaban: LinkedList tidak memerlukan method isFull() karena sifatnya yang dinamis dan tidak memiliki batasan kapasitas maksimum.

2. Mengapa class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama? Bagaimana informasi node kedua dan lainnya diakses?
   Jawaban: Class LinkedList hanya memiliki atribut head karena sifatnya yang dinamis dan struktur data tertautnya. Informasi node selanjutnya dapat diakses melalui pointer next yang ada di setiap node.
3. Pada langkah, jelaskan kegunaan kode berikut

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

   Jawaban: bagian dari implementasi class LinkedList yang digunakan untuk menyisipkan node baru dengan nilai yang sama dengan kunci yang dicari ke dalam LinkedList.
4. Implementasikan method insertAt(int index, int key) dari tugas mata kuliah ASD (Teori)

```
public void insertAt (int index, int key) {
    if (index < 0) {
        System.out.println(x:"Index salah");
    } else if (index == 0) {
        addFirst(key);
    }else {
        Node05  currentNode = head;
        for (int i = 0; i < index -1; i ++) {
        }
        currentNode.next = new Node05(key, currentNode.next);
        if (currentNode.next.next==null) currentNode = currentNode.next;
    }
```

Praktikum 2

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

```
public int gedData (int index){
    Node05 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

```java
public int indexOf (int key){
    Node05 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

```java
public void removeFirst(){
    if(! isEmpty()){
        head = head.next;
    }else {
        System.out.println(x:"Linked List masih Kosong");
    }
}
```

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

```java
public void removeLast(){
    if(isEmpty()){
        System.out.println(x:"Linked List masih Kosong");
    }else if(head.next == null){
        head = null;
    }else{
        Node05 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

```java
public void remove (int key) {
    if(isEmpty()){
        System.out.println(x:"Linked List masih Kosong");
    }else if (head.data == key) {
        removeFirst();
    } else {
        Node05 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

```java
System.out.println("Data pada indekx ke-1 : " + myLinkedList05.gedData(index:1));
System.out.println("Data 300 berada pada indeks ke:" + myLinkedList05.indexOf(key:300));

myLinkedList05.remove(key:300);
myLinkedList05.print();
myLinkedList05.removeFirst();
myLinkedList05.print();
myLinkedList05.removeLast();
myLinkedList05.print();
```

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
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
```

Pertanyaan:

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

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

Jawaban: Memeriksa apakah data pada node saat ini (currentNode) cocok dengan kunci yang ditentukan (key) dan menghapus node yang sesuai jika cocok.

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

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

Jawaban: If-else block pada method indexOf() secara efektif mengimplementasikan algoritma pencarian linier untuk menemukan indeks elemen yang diberikan dalam daftar tertaut.

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

Jawaban:

```
public int gedData (int index){
    Node05 currentNode = head;
    for(int i = 0; i > index; i++){
        currentNode = currentNode.next;
    }
    return currentNode.data;
}
```

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

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

Jawaban: Pada metode remove(int key), kata kunci break digunakan untuk menghentikan iterasi setelah node yang memiliki nilai key ditemukan dan dihapus. Setelah node tersebut dihapus, tidak perlu lagi melanjutkan iterasi, karena node yang ingin dihapus telah ditemukan dan dihapus dengan sukses

TUGAS

1. Implementasikan method-method berikut pada class LinkedList:
   a. insertBefore() untuk menambahkan node sebelum keyword yang diinginkan
   b. insertAt(int index, int key) untuk menambahkan node pada index tertentu
   c. removeAt(int index) untuk menghapus node pada index tertentu

jawaban:

a. InsertBefore()

```
public void insertBefore(int key, int input){
    Node05 newNode = new Node05(input, next:null);
    Node05 currentNode = head;

    if(currentNode == null) {
        this.addFirst(input);
        return;
    }

    if(currentNode.data == key) {
        this.addFirst(input);
        return;
    }

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

    System.out.println("Node dengan key " + key + " tidak ditemukan dalam linked list");
}
```

b. insertAt(int index, int key)

```
public void insertAt(int index, int key) {
    Node05 newNode = new Node05(key, next:null);

    if (index < 0) {
        System.out.println(x:"Indeks tidak valid");
        return;
    }
    if (index == 0) {
        addFirst(key);
    } else {
        Node05 currentNode = head;
        int currentIndex = 0;

        while (currentNode != null && currentIndex < index - 1) {
            currentNode = currentNode.next;
            currentIndex++;
        }
        if (currentNode == null) {
            System.out.println(x:"Tidak melebihi ukuran Linked List");
        } else {
            newNode.next = currentNode.next;
            currentNode.next = newNode;
        }
    }
}
```

c. removeAt(int index)

```
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;
    }

    Node05 prevNode = null;
    Node05 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;
    }
}
}
```

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.

   Kode Program pada class PointGame05.java dengan memberi beberapa atribut antara lain:
   - Question (int)
   - Answer (String)
   - Next (PointGame01)

   Serta menuliskan konstruktor berparameternya.

```
public class PointGame05 {

    String question;
    String answer;
    PointGame05 next;

    public PointGame05(String question, String answer) {
        this.question = question;
        this.answer = answer;
        this.next = null;
    }
}
```

```java
import java.util.Scanner;

public class ScavengerHunt05 {
    PointGame05 head;

    public ScavengerHunt05() {
        this.head = null;
    }

    public void addPoint(String question, String answer) {
        PointGame05 newPoint = new PointGame05(question, answer);

        if (head == null) {
            head = newPoint;
        } else {
            PointGame05 current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newPoint;
        }
    }

    public void startHunt() {
        PointGame05 current = head;
        Scanner scanner = new Scanner(System.in);

        while (current != null) {
            System.out.println("Pertanyaan: " + current.question);
            System.out.print(s:"Jawaban anda : ");
            String userAnswer = scanner.nextLine();

            if (userAnswer.equalsIgnoreCase(current.answer)) {
                System.out.println();
                System.out.println(x:"Benar! Pindah ke titik berikutnya.");
                current = current.next;
                System.out.println(x:"****************************************");
            } else {
                System.out.println();
                System.out.println(x:"Jawaban salah. Silahkan Coba lagi");
            }
        }
        System.out.println();
        System.out.println(x:"Selamat! Anda telah menemukan harta karunnya!");

        scanner.close();
    }
}
```

```java
public class NodeGameMain05 {
    Run | Debug
    public static void main(String[] args) {
        ScavengerHunt05 hunt = new ScavengerHunt05();

        hunt.addPoint(question:"Brand mobil TOYOTA berasal dari negara mana?",answer:"Jepang");
        hunt.addPoint(question:"Apa kegunaan sistem hybrid pada mobil?", answer:"Efisiensi BBM");
        hunt.addPoint(question:"Apa mall terbesar yang berada di surabaya?", answer:"Pakuwon Mall");
        hunt.addPoint(question:"Sebutkan nama ibu kota Australia!", answer:"Sydney");
        hunt.addPoint(question:"Brand mobil VOLVO berasal dari negara mana?",answer:"Swedia");


        hunt.startHunt();
    }
}
```

Hasil Run Program:

```
Pertanyaan: Brand mobil TOYOTA berasal dari negara mana?
Jawaban anda : Jepang

Benar! Pindah ke titik berikutnya.
**********
Pertanyaan: Apa kegunaan sistem hybrid pada mobil?
Jawaban anda : Efisiensi BBM

Benar! Pindah ke titik berikutnya.
**********
Pertanyaan: Apa mall terbesar yang berada di surabaya?
Jawaban anda : Pakuwon Mall

Benar! Pindah ke titik berikutnya.
**********
Pertanyaan: Sebutkan nama ibu kota Australia!
Jawaban anda : Sydney

Benar! Pindah ke titik berikutnya.
**********
Pertanyaan: Brand mobil VOLVO berasal dari negara mana?
Jawaban anda : Swedia

Benar! Pindah ke titik berikutnya.
**********

Selamat! Anda telah menemukan harta karunnya!
```