LAPORAN PRAKTIKUM

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

PERTEMUAN 11 : LINKED LIST

KAYLA RACHMAUDINA SATITI PUTRI

2341760103

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI POLITEKNIK

NEGERI MALANG

2024

JOBSHEET IX
LINKED LIST

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Membuat struktur data linked list
2. Membuat linked list pada program
3. Membedakan permasalahan apa yang dapat diselesaikan menggunakan linked list

2. Praktikum

2.1 Pembuatan Linked List

Waktu percobaan: 50 menit

Didalam praktikum ini, akan dilakukan implementasi pembuatan linked list menggunakan array dan penambahan node ke dalam linked list

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

```java
public class Node15 {
    int data;
    Node15 next;

    public Node15(int data, Node15 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 LinkedList15 {
    Node15 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.println(x:"Isi linked list: ");
            Node15 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) {
        Node15 newNode = new Node15(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

```java
public void addLast(int input) {
    Node15 newNode = new Node15(input, next:null);

    if (isEmpty()) {
        head = newNode;
    } else {
        Node15 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) {
    Node15 newNode = new Node15(input, next:null);

    if (!isEmpty()) {
        Node15 currentNode = head;

        do {
            if (currentNode.data == key) {
                newNode.next = newNode;
                break;
            }
            currentNode = currentNode.next;
        } while (currentNode != null);
    } else {
        System.out.println(x:"Linked list kosong");
    }
```

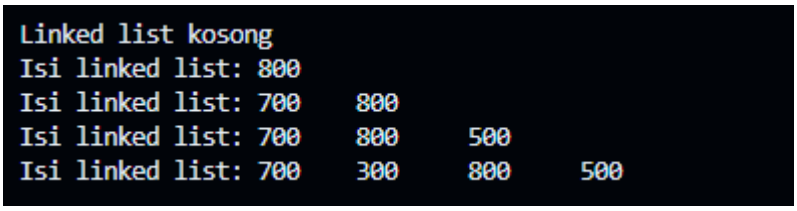
11. Pada class SLLMain, buatlah fungsi **main**, kemudian buat object myLinkedList bertipe LinkedList. Lakukan penambahan beberapa data. Untuk melihat efeknya terhadap object myLinkedList, panggil method print()

```java
public class SLLMain15 {
    Run | Debug
    public static void main(String[] args) {
        LinkedList15 myLinkedList = new LinkedList15();
        myLinkedList.print();
        myLinkedList.addFirst(input:800);
        myLinkedList.print();
        myLinkedList.addFirst(input:700);
        myLinkedList.print();
        myLinkedList.addLast(input:500);
        myLinkedList.print();
        myLinkedList.insertAfter(key:700, input:300);
        myLinkedList.print();
```

2.1.1 Verifikasi Hasil Percobaan

Cocokkan hasil run program Anda dengan output berikut ini.

```
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
```



2.1.2 Pertanyaan

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

   Class LinkedList tidak memerlukan method isFull() karena sifatnya yang dinamis dan tidak memiliki batasan kapasitas. Method isEmpty() dan size() dapat digunakan sebagai alternatif untuk mengetahui status LinkedList.

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

   Karena struktur datanya yang dinamis dan terhubung. Mengakses node dilakukan dengan menggunakan pointer next.

3. Pada langkah, jelaskan kegunaan kode berikut

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

   Memasukkan node baru dengan data yang diberikan setelah node pertama di dalam linked list yang memiliki nilai data sama dengan key.

2.2 Mengakses dan menghapus node pada Linked List

Waktu percobaan: 50 menit

Didalam praktikum ini, kita akan mengimplementasikan method untuk melakukan pengaksesan dan penghapusan data pada linked list

2.2.1 Langkah-langkah Percobaan

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

```
69      // percobaan 2
70      public int getData(int index) {
71          Node15 currentNode = head;
72
73          for (int i = 0; i < index; i++) {
74              currentNode = currentNode.next;
75          }
76
77          return currentNode.data;
78      }
```

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

```
80      public int indexOf(int key) {
81          Node15 currentNode = head;
82          int index = 0;
83
84          while (currentNode != null && currentNode.data != key) {
85              currentNode = currentNode.next;
86              index++;
87          }
88
89          if (currentNode == null) {
90              return -1;
91          } else {
92              return index;
93          }
```

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

```
96      public void removeFirst() {
97          if (!isEmpty()) {
98              head = head.next;
99          } else {
100             System.out.println(x:"Linked list kosong");
101         }
102     }
```

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

```
104     public void removeLast() {
105         if (isEmpty()) {
106             System.out.println(x:"Linked list kosong");
107         } else if (head.next == null) {
108             head = null;
109         } else {
110             Node15 currentNode = head;
111 
112             while (currentNode.next != null) {
113                 if (currentNode.next.next == null) {
114                     currentNode.next = null;
115                     break;
116                 }
117                 currentNode = currentNode.next;
118             }
119         }
120     }
```

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

```
122     public void remove(int key) {
123         if (isEmpty()) {
124             System.out.println(x:"Linked list kosong");
125         } else if (head.data == key) {
126             removeFirst();;
127         } else {
128             Node15 currentNode = head;
129 
130             while (currentNode.next != null) {
131                 if (currentNode.next.data == key) {
132                     currentNode.next = currentNode.next.next;
133                     break;
134                 }
135                 currentNode = currentNode.next;
136             }
```

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

```
// percobaan 2
System.out.println("\nData pada index ke-1          : " + myLinkedList.getData(index:1));
System.out.println("Data 300 berada pada index ke : " + myLinkedList.indexOf(key:300));

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

7. Compile dan run program kemudian amati hasilnya

2.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil run program dengan output berikut ini.

```
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
```

```
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
```

2.2.3 Pertanyaan

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

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

   Menerima key sbg input dan iterasi melalui linked list untuk melihat apakah data dari node mana pun cocok dengan key. Jika ditemukan kecocokan, fungsi ini mengembalikan true, jika tidak maka akan mengembalikan false.

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

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

   Menentukan hasil dari operasi pencarian dan mengembalikan nilai yang sesuai berdasarkan apakah nilai tersebut ditemukan atau tidak.

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

Kesalahan akan muncul jika index yang diberikan lebih besar dari jumlah node pada linked list. Hal ini terjadi karena method getData() akan mencoba mengakses node yang tidak ada, sehingga menyebabkan exception out-of-bounds.

```
70      public int getData(int index) {
71          Node15 currentNode = head;
72          int currentIndex = 0;
73
74          while (currentNode != null && currentIndex < index) {
75              currentNode = currentNode.next;
76              currentIndex++;
77          }
78
79          if (currentNode == null) {
80              throw new IndexOutOfBoundsException("Index " + index + " is out of bounds");
81          }
82          // for (int i = 0; i < index; i++) {
83          //     currentNode = currentNode.next;
84          // }
85
86          return currentNode.data;
87      }
```

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

   Break memastikan bahwa loop berhenti setelah menghapus node yang cocok pertama, sehingga mencegah masalah-masalah yang bisa terjadi. Tanpa break, loop while akan terus beriterasi bahkan setelah node yang cocok dihapus.

3. Tugas

Waktu pengerjaan: 50 menit

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

```java
    public void insertBefore(int key, int input) {
        Node15 newNode = new Node15(input, next:null);

        if (isEmpty()) {
            System.out.println(x:"Linked list kosong");
            return;
        }
        if (head.data == key) {
            addFirst(input);
            return;
        }
        // Traverse
        Node15 currentNode = head;
        while (currentNode.next != null && currentNode.next.data != key) {
            currentNode = currentNode.next;
        }
        if (currentNode.next == null) {
            System.out.println("Keyword " + key + " tidak ditemukan");
            return;
        }
        newNode.next = currentNode.next;
        currentNode.next = newNode;
    }
```

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

```java
// b. insertAt(int index, int key) method
public void insertAt(int index, int input) {
    Node15 newNode = new Node15(input, next:null);

    if (index < 0) {
        throw new IndexOutOfBoundsException(s:"Index cannot be negative");
    }

    if (index == 0) {
        addFirst(input);
    } else {
        Node15 currentNode = head;
        int currentIndex = 0;

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

        if (currentNode == null) {
            throw new IndexOutOfBoundsException("Index " + index + " is out of bounds");
        }

        newNode.next = currentNode.next;
        currentNode.next = newNode;
    }
}
```

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

```java
// c. removeAt(int index) method
public void removeAt(int index) {
    if (index < 0) {
        throw new IndexOutOfBoundsException(s:"Index cannot be negative");
    }

    if (index == 0) {
        removeFirst();
    } else {
        Node15 currentNode = head;
        int currentIndex = 0;

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

        if (currentNode == null || currentNode.next == null) {
            throw new IndexOutOfBoundsException("Index " + index + " is out of bounds");
        }

        currentNode.next = currentNode.next.next;
    }
}
```

Main classnya :

```java
        myLinkedList.insertBefore(key:800, input:450);
        myLinkedList.print();
        myLinkedList.insertAt(index:1, input:400);
        myLinkedList.print();
        myLinkedList.removeAt(index:1);
        myLinkedList.print();
```

Outputnya :

```
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
Isi linked list: 450    800
Isi linked list: 450    400     800
Isi linked list: 450    800
```

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.

   Class gameNode15

```java
public class gameNode15 {
    int pointId;
    String question;
    String answer;
    int nextPointId;
    gameNode15 next;

    public gameNode15(int pointId, String question, String answer, int nextPointId, gameNode15 next) {
        this.pointId = pointId;
        this.question = question;
        this.answer = answer;
        this.nextPointId = nextPointId;
        this.next = next;
    }
}
```

   Class game15

```java
import java.util.Scanner;

public class game15 {
    gameNode15 head;

    public void addPoint(int pointId, String question, String answer, int nextPointId) {
        gameNode15 newNode = new gameNode15(pointId, question, answer, nextPointId, null);

        if (head == null) {
            head = newNode;
        } else {
            gameNode15 currentNode = head;
            while (currentNode.next != null) {
                currentNode = currentNode.next;
            }
            currentNode.next = newNode;
        }
    }

        public void startGame() {
        Scanner scanner = new Scanner(System.in);
        gameNode15 currentNode = head;

        while (currentNode != null) {
            System.out.println("============================================================");
            System.out.println("Point " + currentNode.pointId);
            System.out.println("Pertanyaan: " + currentNode.question);

            System.out.print("Jawaban: ");
            String userAnswer = scanner.nextLine();

            if (userAnswer.equalsIgnoreCase(currentNode.answer)) {
                System.out.println("Jawaban benar!");
                currentNode = currentNode.next;
            } else {
                System.out.println("Jawaban salah. Coba lagi.");
            }
        }
```

```
        if (currentNode == null) {
            System.out.println("\nSelamat! Anda telah mendapatkan hadiahhh!");
            System.out.println("Hubungi nomor berikut untuk klaim hadiah xxxxx");
        }
    }
}
```

Class gameMain15

```
public class gameMain15 {
    public static void main(String[] args) {
        game15 game = new game15();

        // Add points
        game.addPoint(1, "Sungai apa yang paling panjang di dunia?", "sungai nil", 2);
        game.addPoint(2, "232 x 219 = ", "50808", 3);
        game.addPoint(3, "Selalu bertambah tapi tidak pernah berkurang?", "usia", 4);
        game.addPoint(4, "Selalu datang tapi tidak pernah sampai?", "Besok", 5);
        game.addPoint(5, "Rumus gaya?", "m x a", 0);

        game.startGame();
    }
}
```

Output :

```
============================================================
Point 1
Pertanyaan: Sungai apa yang paling panjang di dunia?
Jawaban: sungai nil
Jawaban benar!
============================================================
Point 2
Pertanyaan: 232 x 219 =
Jawaban: 50808
Jawaban benar!
============================================================
Point 3
Pertanyaan: Selalu bertambah tapi tidak pernah berkurang?
Jawaban: usia
Jawaban benar!
============================================================
Point 4
Pertanyaan: Selalu datang tapi tidak pernah sampai?
Jawaban: besok
Jawaban benar!
============================================================
Point 5
Pertanyaan: Rumus gaya?
Jawaban: m x a
Jawaban benar!

Selamat! Anda telah mendapatkan hadiahhh!
Hubungi nomor berikut untuk klaim hadiah xxxxx
```