ALGORITMA STRUKTUR DATA

Praktikum – Linked List

Lavina 2341760062

Praktikum 1: Pembuatan Linked List

Node.java

```java
package Praktikum9;

public class Node {
  int data;
  Node next;

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

LinkedList.java

```java
package Praktikum9;

public class LinkedList {
  Node head;

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

  public void print() {
    if (!isEmpty()) {
      System.out.print("Isi linked list: ");
      Node currentNode = head;

```

```
15        while (currentNode != null) {
16          System.out.print(currentNode.data + "\t");
17          currentNode = currentNode.next;
18        }
19
20        System.out.println("");
21      } else {
22        System.out.println("Linked list kosong");
23      }
24    }
25
26    public void addFirst(int input) {
27      Node newNode = new Node(input, null);
28
29      if (isEmpty()) {
30        head = newNode;
31      } else {
32        newNode.next = head;
33        head = newNode;
34      }
35    }
36
37    public void addLast(int input) {
38      Node newNode = new Node(input, null);
39
40      if (isEmpty()) {
41        head = newNode;
42      } else {
43        Node currentNode = head;
44
45        while (currentNode.next != null) {
46          currentNode = currentNode.next;
47        }
48
49        currentNode.next = newNode;
50      }
51    }
```

```
53    public void insertAfter(int key, int input) {
54      Node newNode = new Node(input, null);
55
56      if (!isEmpty()) {
57        Node currentNode = head;
58
59        do {
60          if (currentNode.data == key) {
61            newNode.next = currentNode.next;
62            currentNode.next = newNode;
63          }
64          currentNode = currentNode.next;
65        } while (currentNode != null);
66      } else {
67        System.out.print("Linked list kosong");
68      }
69    }
70  }
```

SLLMain.java

```
1   package Praktikum9;
2
3   public class SLLMain {
      Run | Debug
4     public static void main(String[] args) {
5       LinkedList myLinkedList = new LinkedList();
6
7       myLinkedList.print();
8       myLinkedList.addFirst(input:800);
9       myLinkedList.print();
10      myLinkedList.addFirst(input:700);
11      myLinkedList.print();
12      myLinkedList.addLast(input:500);
13      myLinkedList.print();
14      myLinkedList.insertAfter(key:700, input:300);
15      myLinkedList.print();
16    }
17  }
```

Output :

```
Praktikum9.SLLMain
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?

   Jawab : Karena linked list bersifat dinamis.

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

   Jawab : Karena datanya yang dinamis sehingga kita hanya perlu mengetahui informasi data yang pertama yang selanjutnya untuk mengetahui data-data selanjutnya bisa menggunakan next dari head.

3. Pada langkah, jelaskan kegunaan kode berikut

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

   Jawab : Pertama kode tersebut memeriksa apakah node saat ini sama dengan key, jika true lalu nilai yang baru akan disipkan di node berikutnya dari node saat ini, terakhir mengarahkan pointer next dari node saat ini ke node yang baru.

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

   Method insertAt() :

```
74    public void insertAt(int index, int key) {
75      Node newnNode = new Node(key, next:null);
76
77      if (index == 0) {
78        newnNode.next = head;
79        head = newnNode;
80        return;
81      }
```

```
83        Node currentNode = head;
84        int count = 0;
85        while (currentNode != null && count < index - 1) {
86          currentNode = currentNode.next;
87          count++;
88        }
89
90        newnNode.next = currentNode.next;
91        currentNode.next = newnNode;
92      }
```

Output :

```
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    200     300     800     500
```

Praktikum 2: Mengakses dan menghapus node pada Linked List

LinkedList.java

```
1   public int getData(int index) {
2     Node currentNode = head;
3
4     for (int i = 0; i < index; i++) {
5       currentNode = currentNode.next;
6     }
7
8     return currentNode.data;
9   }
10
11  public int indexOf(int key) {
12    Node currentNode = head;
13    int index = 0;
14
15    while (currentNode != null && currentNode.data != key) {
16      currentNode = currentNode.next;
17      index++;
18    }
19
```

```
20    if (currentNode == null) {
21      return -1;
22    } else {
23      return index;
24    }
25  }
26
27  public void removeFirst() {
28    if (!isEmpty()) {
29      head = head.next;
30    } else {
31      System.out.println("Linked list kosong");
32    }
33  }
34
35  public void removeLast() {
36    if (isEmpty()) {
37      System.out.println("Linked list kosong");
38    } else if (head.next == null) {
39      head = null;
40    } else {
41      Node currentNode = head;
42
43      while (currentNode.next != null) {
44        if (currentNode.next.next == null) {
45          currentNode.next = null;
46          break;
47        }
48        currentNode = currentNode.next;
49      }
50    }
51  }
52
53  public void remove(int key) {
54    if (isEmpty()) {
55      System.out.println("Linked list kosong");
56    } else if (head.data == key) {
57      removeFirst();
58    } else {
59      Node currentNode = head;
60
61      while (currentNode.next != null) {
62        if (currentNode.next.data == key) {
63          currentNode.next = currentNode.next.next;
64          break;
65        }
```

```
67          currentNode = currentNode.next;
68      }
69    }
70  }
```

SLLMain.java

```
System.out.println("Data 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();
```

Output :

```
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

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

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

Jawab : Kode tersebut awalnya memeriksa apakah data node setelah node saat ini (currentNode) sama dengan key, jika true maka nilai dari node setelah node saat ini akan diisi dengan nilai dari node selanjutnya dan selanjutnya lagi sehingga node tersebut akan terhapus.

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

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

Jawab : Maksudnya jika node saat ini null maka beri return index -1 yang berarti kosong, jika false maka mereturn index yang dicari.

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 : Terjadi error NullPointerException karena index yang dicari tidak ada. Method dibawah ini akan memberikan output -1 apabila index yang dicari melebihi jumlah data pada linked list.

```
public int getData(int index) {
  if (index < 0) {
    System.out.println(x:"Index tidak valid !");
  }

  int count = 0;
  Node currentNode = head;

  while (currentNode != null && count <= index) {
    if (count == index) {
      return currentNode.data;
    }
    count++;
    currentNode = currentNode.next;
  }

  return -1;
}
```

   Output :

```
Data pada index ke-7: -1
```

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

   Jawab : Fungsinya adalah untuk menghentikan loop ketika data setelah dan setelahnya lagi dari node saat ini sama dengan key, jika tidak diberi break maka loop akan terus berjalan selama node setelah node saat ini tidak sama dengan null.