

**LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA**



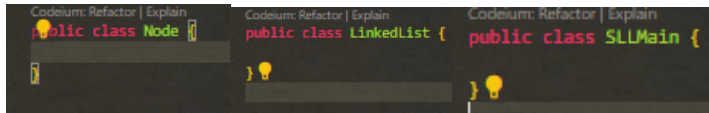
**Oleh:
DZULFIKAR MUHAMMAD AL GHIFARI
NIM. 2341760071
SIB-1F / 08
D-IV SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG**

PRAKTIKUM 1

11.2 Percobaan 1

11.2.1 Langkah Langkah percobaan

1. Membuat class Node, LinkedList, SLLMain

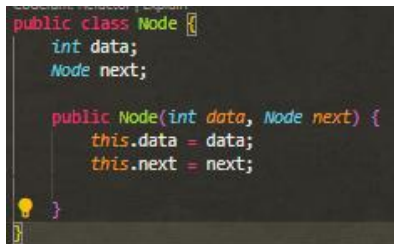


```
Codeium: Refactor | Explain
public class Node {
}

Codeium: Refactor | Explain
public class LinkedList {
}

Codeium: Refactor | Explain
public class SLLMain {
}
```

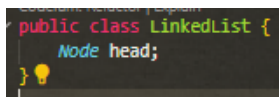
2. Deklarasi attr & construct class Node



```
Codeium: Refactor | Explain
public class Node {
    int data;
    Node next;

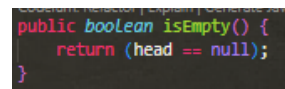
    public Node(int data, Node next) {
        this.data = data;
        this.next = next;
    }
}
```

3. Deklarasi class linkedlists



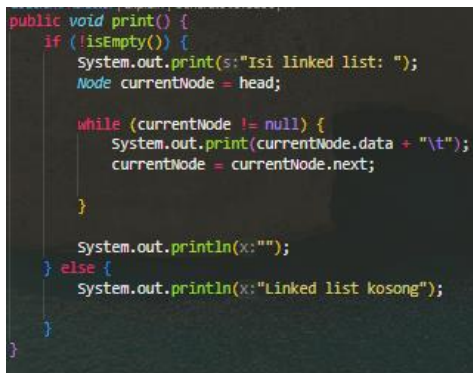
```
Codeium: Refactor | Explain
public class LinkedList {
    Node head;
}
```

4. Menambahkan method empty pada class linkedlist



```
Codeium: Refactor | Explain
public boolean isEmpty() {
    return (head == null);
}
```

5. Menambahkan method print pada class linkedlist

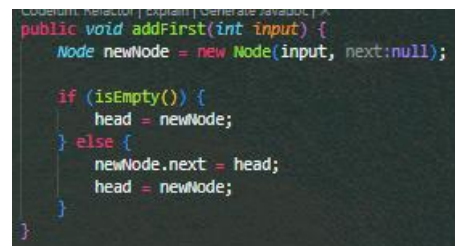


```
Codeium: Refactor | Explain
public void print() {
    if (isEmpty()) {
        System.out.print(s:"Isi linked list: ");
        Node currentNode = head;

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

        System.out.println(x:"");
    } else {
        System.out.println(x:"Linked list kosong");
    }
}
```

6. Menambahkan method addFirst pada class linkedlist



```
Codeium: Refactor | Explain
public void addFirst(int input) {
    Node newNode = new Node(input, next:null);

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

7. Menambahkan method addLast pada class linkedlist

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

    if (isEmpty()) {
        head = newNode;
    } else {
        Node currentNode = head;

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

        currentNode.next = newNode;
    }
}
```

8. Menambahkan method insertAfter pada class linkedlist

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

    if (!isEmpty()) {
        Node 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");
    }
}
```

9. Membuat method main pada class SSLMain

```
public static void main(String[] args) {
    LinkedList myLinkedList = new LinkedList();
    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();
}
```

VERIFIKASI HASIL PERCOBAAN 11.2.2

```
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 E:\1POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```

PERTANYAAN 12.2.3

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

LinkedList, memiliki struktur data yang bersifat linear, yaitu elemen-elemen terhubung satu sama lain dalam urutan tertentu. LinkedList tidak memiliki batasan kapasitas, sehingga tidak memerlukan method isFull()

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

LinkedList memulai dari head untuk menyimpan node pertama, ketika ingin mengakses node kedua, maka akan mengikuti pointer. Node pertama memiliki pointer yang menunjuk ke node kedua. Untuk mengakses ke node selanjutnya, dapat mengikuti pointer dari node saat ini. Pointer dari setiap node menunjuk ke node berikutnya.

3. Pada langkah, jelaskan kegunaan kode berikut

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

Jika data dari node saat ini cocok dengan attr key, maka node baru akan disisipkan setelah key.

4. Implementasikan method insertAt(int index, int key) dari tugas mata kuliah ASD (Teori 12.3 Percobaan 2

```
CodeRunner: Run | Explain | Generate | ...
public void insertAt(int index, int key) {
    Node node = new Node(key, next:null);
    if (index == 0) {
        addFirst(key);
        return;
    }

    Node currentNode = head;
    int counter = 0;
    while (currentNode != null && counter < index - 1) {
        currentNode = currentNode.next;
        counter++;
    }

    if (currentNode.next == null) {
        addLast(key);
        return;
    }

    node.next = currentNode.next;
    currentNode.next = node;
}

public static void main(String[] args) {
    LinkedList myLinkedList = new LinkedList();
    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();
    myLinkedList.insertAt(index:2, key:900);
    myLinkedList.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    300    900    800    500
```

11.3.1 Langkah-langkah Percobaan

1. Menambahkan method getData pada class LinkedList

```
public int getData(int index) {
    Node currentNode = head;

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

    return currentNode.data;
}
```

2. Menambahkan method indexOf pada class LinkedList

```
public int indexOf(int key) {
    Node currentNode = head;
    int index = 0;

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

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

3. Menambahkan method removeFirst pada class LinkedList

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

- Menambahkan method `removeLast` pada class `LinkedList`

```
public void removeLast() {
    if (isEmpty()) {
        System.out.println("Linked list kosong");
    } else if (head.next == null) {
        head = null;
    } else {
        Node currentNode = head;

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

- Menambahkan method `remove` pada class `LinkedList`

```
public void remove(int key) {
    if (isEmpty()) {
        System.out.println("Linked list kosong");
    } else if (head.data == key) {
        removeFirst();
    } else {
        Node currentNode = head;
        while (currentNode.next != null) {
            if (currentNode.next.data == key) {
                currentNode.next = currentNode.next.next;
                break;
            }
            currentNode = currentNode.next;
        }
    }
}
```

- Menambahkan kodebari pada `SLLMain`

```
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();
```

VERIFIKASI HASIL PERCOBAAN 11.3.2

```
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      300      900      800      500
Data pada index ke-1: 300
Data 300 berada pada index ke: 1
Isi linked list: 700      900      800      500
Isi linked list: 900      800      500
Isi linked list: 900      800
PS E:\1POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```


PERTANYAAN 11.3.3

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

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

memeriksa apakah data dari node selanjutnya sama dengan key . Jika sama, maka node yang ingin dihapus telah ditemukan.

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

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

Memeriksa apakah code saat ini bernilai null, jika iya maka akan mengembalikan -1, jika tidak maka akan mengembalikan index saat ini untuk kemudian ditampilkan.

3. Error apa yang muncul jika argumen method getData() lebih besar dari jumlah node pada

linked list? Modifikasi kode program untuk handle hal tersebut.

Akan terjadi error seperti dibawah 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  
Isi linked list: 700    300    900    800    500  
Exception in thread "main" java.lang.NullPointerException: Cannot read field "next" because "currentNode" is null  
    at LinkedList.getData(LinkedList.java:100)  
    at SLIMain.main(SLIMain.java:16)
```

Menambahkan method untuk mengecek apakah data dengan index tersebut ada

```
public boolean checkIndex(int index){  
    Node currentNode = head;  
    for (int i = 0; i < index; i++) {  
        if (currentNode == null){  
            System.out.println(x:"Data tidak ditemukan");  
            return false;  
        }  
        currentNode = currentNode.next;  
    }  
    return true;  
}
```

Mengubah pada main

```
if(myLinkedList.checkIndex(index:10))  
    System.out.println("Data pada index ke-1: " + myLinkedList.getData(index:10));
```

Hasil compile

```
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    300    900    800    500  
Data tidak ditemukan  
Data 300 berada pada index ke: 1  
Isi linked list: 700    900    800    500  
Isi linked list: 900    800    500  
Isi linked list: 900    800  
PS E:\POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```

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

berfungsi untuk menghentikan iterasi loop while lebih awal jika sudah menemukan node yang ingin dihapus

11.4 TUGAS

1. Implementasikan method-method berikut pada class LinkedList:

a. insertBefore() untuk menambahkan node sebelum keyword yang diinginkan

```
myLinkedList.insertBefore(key:2, input:404);  
myLinkedList.print();
```

```
public void insertBefore(int key, int input) {  
    Node newNode = new Node(input, next:null);  
  
    if (!isEmpty()) {  
        Node current = head;  
        while (current.next != null) {  
            newNode.next = current.next;  
            current.next = newNode;  
            return;  
        }  
    } else {  
        System.out.print(s:"Linked list kosong");  
    }  
}
```

```
Data 300 berada pada index ke: 1  
Isi linked list: 700    900    800    500  
Isi linked list: 900    800    500  
Isi linked list: 900    800  
Isi linked list: 900    404    800
```

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

```
myLinkedList.insertAt(index:0, key:120);  
myLinkedList.print();
```

```
public void insertAt(int index, int key) {  
    Node node = new Node(key, next:null);  
    if (index == 0) {  
        addFirst(key);  
        return;  
    }  
  
    Node currentNode = head;  
    int counter = 0;  
    while (currentNode != null && counter < index - 1) {  
        currentNode = currentNode.next;  
        counter++;  
    }  
  
    if (currentNode.next == null) {  
        addLast(key);  
        return;  
    }  
  
    node.next = currentNode.next;  
    currentNode.next = node;  
}
```

```
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    300    900    800    500  
Data tidak ditemukan  
Data 300 berada pada index ke: 1  
Isi linked list: 700    900    800    500  
Isi linked list: 900    800    500  
Isi linked list: 900    800  
Isi linked list: 120    900    800  
PS E:\1POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```

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

```
myLinkedList.removeAt(index:1);  
myLinkedList.print();
```

```
public void removeAt(int index) {  
    if (index == 0) {  
        removeFirst();  
    } else {  
        Node currentNode = head;  
        int currentIndex = 0;  
        while (currentNode != null && currentIndex < index - 1) {  
            currentNode = currentNode.next;  
            currentIndex++;  
        }  
        currentNode.next = currentNode.next.next;  
    }  
}
```

```
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    300    900    800    500  
Data tidak ditemukan  
Data 300 berada pada index ke: 1  
Isi linked list: 700    900    800    500  
Isi linked list: 900    800    500  
Isi linked list: 900    800  
Isi linked list: 120    900    800  
Isi linked list: 120    800  
PS E:\1POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```


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 node

```
Codeium: Refactor | Explain
public class ScavengerHuntNode {
    String soal, kunci;
    ScavengerHuntNode next;

    ScavengerHuntNode() {}

    ScavengerHuntNode(String soal, String kunci, ScavengerHuntNode next) {
        this.soal = soal;
        this.kunci = kunci;
        this.next = next;
    }
}
```

Class linkedlist

```
Codeium: Refactor | Explain | Generate Javadoc | X
public class LinkedList {
    ScavengerHuntNode head;

    Codeium: Refactor | Explain | Generate Javadoc | X
    public boolean isEmpty() {
        return (head == null);
    }

    Codeium: Refactor | Explain | Generate Javadoc | X
    public void print() {
        if (!isEmpty()) {
            System.out.println(x:"=====");
            System.out.println(x:"Daftar Pertanyaan");
            System.out.println(x:"=====");

            ScavengerHuntNode currentNode = head;

            while (currentNode != null) {
                System.out.println("+" + currentNode.soal);
                currentNode = currentNode.next;
            }
            System.out.println(x:"");
        } else {
            System.out.println(x:"Linked List Kosong");
        }
    }

    public void addLast(String soal, String kunci) {
        ScavengerHuntNode newNode = new ScavengerHuntNode(soal, kunci, next:null);
        if (head == null) {
            head = newNode;
        } else {
            ScavengerHuntNode currentNode = head;
            while (currentNode.next != null) {
                currentNode = currentNode.next;
            }
            currentNode.next = newNode;
        }
    }
}
```

```

public void start(){
    Scanner input = new Scanner(System.in);
    ScavengerHuntNode currentNode = head;

    int benar = 0;
    int jumlahSoal = 0;

    while (currentNode != null) {
        System.out.println("Pertanyaan yang harus dijawab\t: " + currentNode.soal);
        System.out.print(s:"Isi jawaban dari pertanyaan\t: ");
        String jawaban = input.nextLine();

        if (jawaban.equalsIgnoreCase(currentNode.kunci)) {
            System.out.println(x:"Jawaban benar! Lanjut ke point berikutnya.");
            System.out.println();
            benar++;
            currentNode = currentNode.next;
        } else {
            System.out.println(x:"Jawaban salah. Coba lagi!");
            System.out.println();
        }
        jumlahSoal++;
    }

    System.out.println(x:"=====");
    System.out.println(x:"SUCCESSFULLY ");
    System.out.println(x:"=====");
    System.out.println(x:"Skor anda :");
    System.out.println("Benar "+benar);
    System.out.println("Salah "+(jumlahSoal-benar));
    System.out.println("Total Percobaan "+(jumlahSoal));
    System.out.println();
    System.out.println();
}

```

Class main

```

public static void menu() {
    System.out.println(x:"=====");
    System.out.println(x:"PILIHAN MENU");
    System.out.println(x:"=====");
    System.out.println(x:"1. Tampilkan pertanyaan");
    System.out.println(x:"2. Jawab pertanyaan");
    System.out.println(x:"3. Exit");
    System.out.println(x:"=====");
}

```

```

public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    LinkedList linkedList = new LinkedList();
    int pil;

    do {
        menu();
        System.out.print(s:"Pilih salah satu (1/2/3) : ");
        pil = input.nextInt();
        switch (pil) {
            case 1:
                linkedList.addLast(soal:"suku yang ramah", kunci:"jawa");
                linkedList.addLast(soal:"poltek identik dengan warna", kunci:"biru");
                linkedList.addLast(soal:"roda mobil ada (jawab dengan kata)",
                    kunci:"empat");
                linkedList.addLast(soal:"roda motor ada (jawab dengan kata)",
                    kunci:"dua");
                linkedList.print();
                break;
            case 2:
                linkedList.start();
                break;
            case 3:
                return;
            default:
                System.out.println(x:"Pilihan tidak tersedia .");
                break;
        }
    } while (pil == 1 || pil == 2 || pil == 3);
}

```

Hasil compile

```
=====
PILIHAN MENU
=====
1. Tampilkan pertanyaan
2. Jawab pertanyaan
3. Exit
=====
Pilih salah satu (1/2/3) : 1
=====
Daftar Pertanyaan
=====
+suku yang ramah
+poltek identik dengan warna
+roda mobil ada (jawab dengan kata)
+roda motor ada (jawab dengan kata)

=====
Pilih salah satu (1/2/3) : 2
Pertanyaan yang harus dijawab : suku yang ramah
Isi jawaban dari pertanyaan : jawa
Jawaban benar! Lanjut ke point berikutnya.

Pertanyaan yang harus dijawab : poltek identik dengan warna
Isi jawaban dari pertanyaan : biru
Jawaban benar! Lanjut ke point berikutnya.

Pertanyaan yang harus dijawab : roda mobil ada (jawab dengan kata)
Isi jawaban dari pertanyaan : empat
Jawaban benar! Lanjut ke point berikutnya.

Pertanyaan yang harus dijawab : roda motor ada (jawab dengan kata)
Isi jawaban dari pertanyaan : dua
Jawaban benar! Lanjut ke point berikutnya.

=====
SUCCESSFULLY
=====
Skor anda :
Benar 4
Salah 0
Total Percobaan 4
=====
PILIHAN MENU
=====
1. Tampilkan pertanyaan
2. Jawab pertanyaan
3. Exit
=====
Pilih salah satu (1/2/3) : 3
PS E:\1POLINEMA\2Genap 2023-2024\PraktikumAlgoritma\pertemuan11>
```