Laporan Praktikum ASD
Pertemuan 11

Nama: Gegas Anugrah Derajat
Kelas: SIB-1F
NIM: 2341760140

Percobaan 2.1

Mendeklarasikan class node dan membuat konstruktor berparameter

```
1   public class Node11 {
2       int data;
3       Node11 next;
4
5       public Node11(int data, Node11 next) {
6           this.data = data;
7           this.next = next;
8       }
9   }
```

Mendeklarasikan class LinkedList

```
public class LinkedList11 {
    Node11 head;
```

Menambahkan method isEmpty dan print

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

    public void print() {
        if (!isEmpty()) {
            System.out.print(s:"Isi Linked List: ");
            Node11 currentNode11 = head;

            while (currentNode11 != null) {
                System.out.print(currentNode11.data + "\t");
                currentNode11 = currentNode11.next;
            }

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

Menambahkan method addFirst dan addLast

```
public void addFirst(int input) {
    Node11 newNode11 = new Node11(input, next:null);

    if (isEmpty()) {
        head = newNode11;
    } else {
        newNode11.next = head;
        head = newNode11;
    }
}

public void addLast(int input) {
    Node11 newNode11 = new Node11(input, next:null);

    if (isEmpty()) {
        head = newNode11;
    } else {
        Node11 currentNode11 = head;

        while (currentNode11.next != null) {
            currentNode11 = currentNode11.next;
        }

        currentNode11.next = newNode11;

    }
}
```

Menambahkan method insertAfter()

```
public void insertAfter(int key, int input) {
    Node11 newNode11 = new Node11(input, next:null);

    if (!isEmpty()) {
        Node11 currentNode11 = head;

        do {
            if (currentNode11.data == key) {
                newNode11.next = currentNode11.next;
                currentNode11.next = newNode11;
                break;
            }

            currentNode11 = currentNode11.next;
        } while (currentNode11 != null);
    } else {
        System.out.println(x:"Linked List Kosong!");
    }
```

Membuat fungsi SLLMain dan membuat objek mylinkedlist dan memanggil method print

```
public class SLLMain {
    Run | Debug
    public static void main(String[] args) {
        LinkedList11 myLinkedList = new LinkedList11();
        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();
    }
}
```

Hasil

```
2a675ce2061d\redhat.java\jdt_ws\Pertemuan11_
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?
2. Mengapa class LinkedList hanya memiliki atribut head yang menyimpan informasi node pertama? Bagaimana informasi node kedua dan lainnya diakses?
3. Pada langkah, jelaskan kegunaan kode berikut

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

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

Jawaban:

1. Karena pada Linked List tidak terdapat batasan kapasitas.

2. Linked list memulai dari head untuk menyimpan node pertama, untuk mengakses node kedua, maka akan mengikuti pointer. Untuk mengakses node berikutnya dapat mengikuti pointer yang ada pada node saat ini.

3. Jika data node saat ini sama dengan key, maka node baru akan disisipkan setelah key.

4.

```
public void insertAt(int index, int key) {
    Node11 node = new Node11(key, next:null, prev:null);
    if (index == 0) {
        addFirst(key);
        return;
    }

    Node11 currentNode11 = head;
    int counter = 0;
    while (currentNode11 != null && counter < index - 1) {
        currentNode11 = currentNode11.next;
        counter++;
    }

    if (currentNode11.next == null) {
        addLast(key);
        return;
    }

    node.next = currentNode11.next;
    currentNode11.next = node;

}
```

```
myLinkedList.insertAt(index:3, key:250);
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     800     250     500
Data pada index ke-1: 300
Data 300 berada pada index ke: 1
```

Percobaan 2.2

Menambahkan method getData dan indexOf pada class LinkedList

```
public int getData(int index) {
    Node11 currentNode11 = head;

    for (int i = 0; i < index; i++) {
        currentNode11 = currentNode11.next;
    }

    return currentNode11.data;

}

public int indexOf(int key) {
    Node11 currentNode11 = head;
    int index = 0;

    while (currentNode11 != null && currentNode11.data != key) {
        currentNode11 = currentNode11.next;
        index++;
    }

    if (currentNode11 == null) {
        return -1;
    } else {
        return index;
    }
}
```

Menambahkan method removeFirst dan removeLast pada class LinkedList

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

public void removeLast() {
    if (isEmpty()) {
        System.out.println(x:"Linked list kosong!");
    } else if (head.next == null) {
        head = null;
    } else {
        Node11 curretNode11 = head;

        while (curretNode11.next != null) {
            if (curretNode11.next.next == null) {
                curretNode11.next = null;
                break;
            }

            curretNode11 = curretNode11.next;

        }
    }
}
```

Menambahkan method remove() untuk menghapus elemen tertentu

```
public void remove(int key) {
    if (isEmpty()) {
        System.out.println(x:"Linked list kosong!");
    } else if (head.data == key) {
        removeFirst();
    } else {
        Node11 currentNode11 = head;

        while (currentNode11.next != null) {
            if (currentNode11.next.data == key) {
                currentNode11.next = currentNode11.next.next;
                break;
            }

            currentNode11 = currentNode11.next;

        }
    }
}
```

Megakses dan menghapus data pada class 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();
```

Hasill

```
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
PS C:\Users\Pongo\Documents\Kuliah\smstr2\al
```

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;
}
```

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

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

3. Error apa yang muncul jika argumen method getData() lebih besar dari jumlah node pada linked list? Modifikasi kode program untuk menghandle hal tersebut.
4. Apa fungsi keyword break pada method remove()? Bagaimana efeknya jika baris tersebut dihapus?

Jawaban:

1. Memeriksa apakah data dari node selanjutnya sama dengan key. Jika sama, maka node yang ditemukan akan dihapus.

2. Jika node saat ini bernilai null, maka akan mengmbalikan nilai -1. Jika tidak, maka akan mengembalikan index untuk ditmpilkan.

3. Error yang muncul adalah

```
Exception in thread "main" java.lang.NullPointerException: Cannot read field "next" because "currentNode11" is null
        at LinkedList11.getData(LinkedList11.java:78)
        at SLLMain.main(SLLMain.java:14)
```

Kode untuk menghandle

```
public boolean cekIndex(int index) {
    Node11 currentNode11 = head;
    for (int i = 0; i < index; i++) {
        if (currentNode11 == null) {
            System.out.println(x:"Data tidak ditemukan");
            return false;
        }
        currentNode11 = currentNode11.next;
    }
    return true;
}
```

```
if (myLinkedList.cekIndex(index:7)) {
    System.out.println("Data pada index ke-1: " + myLinkedList.getData(index:7));
}
```

```
Isi Linked List: 700    800
Isi Linked List: 700    800     500
Isi Linked List: 700    300     800     500
Isi Linked List: 700    300     800     250     500
Data tidak ditemukan
Data 300 berada pada index ke: 1
```

4. Berfungsi untuk menghentikan perulangan while jika sudah menemukan node yang ingin dihapus. jika dihapus maka akan terus berulang.

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

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.

Jawaban:

1.

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

```
public void insertBefore(int key, int input) {
    Node11 newNode11 = new Node11(input, next:null, prev:null);

    if (!isEmpty()) {
        Node11 currentNode11 = head;

        do {
            if (currentNode11.data == key) {
                newNode11.next = currentNode11;
                newNode11.prev = currentNode11.prev;
                break;
            }

            if (currentNode11.prev != null) {
                currentNode11.prev.next = newNode11;
            }

            currentNode11 = currentNode11.prev;
        } while (currentNode11 != null);
    } else {
        System.out.println(x:"Linked List Kosong!");
    }
}
```

```
myLinkedList.insertBefore(key:500, input:450);
myLinkedList.print();
```

```
Data 300 berada pada index ke: 1
Isi Linked List: 700    800    250    450    500
```

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

```
public void insertAt(int index, int key) {
    Node11 node = new Node11(key, next:null, prev:null);
    if (index == 0) {
        addFirst(key);
        return;
    }

    Node11 currentNode11 = head;
    int counter = 0;
    while (currentNode11 != null && counter < index - 1) {
        currentNode11 = currentNode11.next;
        counter++;
    }

    if (currentNode11.next == null) {
        addLast(key);
        return;
    }

    node.next = currentNode11.next;
    currentNode11.next = node;

}
```

```
myLinkedList.insertAt(index:3, key:250);
myLinkedList.print();
Isi Linked List: 700    300    800    250    500
```

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

```
public void removeAt(int index) {
    if (index == 0) {
        removeFirst();
    } else {
        Node11 currentNode11 = head;
        int currentIndex = 0;
        while (currentNode11 != null && currentIndex < index - 1) {
            currentNode11 = currentNode11.next;
            currentIndex++;
        }
        currentNode11.next = currentNode11.next.next;
    }
}
```

```
myLinkedList.removeAt(index:0);
myLinkedList.print();
Isi Linked List: 800    250    450    500
Isi Linked List: 800    250    450
Isi Linked List: 250    450
```

2.

Class Node

```
public class ScavengerHunt {
    String soal, kunci;
    ScavengerHunt next;

    ScavengerHunt() {

    }

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

Class linkedList

```
import java.util.Scanner;

public class LinkedList {

    ScavengerHunt head;

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

    public void print() {
        if (!isEmpty()) {
            System.out.println(x:"------------------------");
            System.out.println(x:"Daftar Pertanyaan");
            System.out.println(x:"------------------------");

            ScavengerHunt 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) {
        ScavengerHunt newNode = new ScavengerHunt(soal, kunci, next:null);
        if (head == null) {
            head = newNode;
        } else {
            ScavengerHunt currentNode = head;
            while (currentNode.next != null) {
                currentNode = currentNode.next;
            }
            currentNode.next = newNode;
        }
    }
```

```
public void start() {
    Scanner gs = new Scanner(System.in);
    ScavengerHunt currentNode = head;

    int benar = 0;
    int jmlSoal = 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 = gs.next();

        if (jawaban.equalsIgnoreCase(currentNode.kunci)) {
            System.out.println(x:"Jawaban benar, lanjut ke soal berikutnya");
            System.out.println();
            benar++;
            currentNode = currentNode.next;

        } else {
            System.out.println(x:"Jawaban salah, coba lagi!");
            System.out.println();
        }
        jmlSoal++;
    }
```

```
    System.out.println(x:"-----------");
    System.out.println(x:"SUKSES");
    System.out.println(x:"-----------");
    System.out.println(x:"Skor anda: ");
    System.out.println("Benar: " + benar);
    System.out.println("Salah: " + (jmlSoal - benar));
    System.out.println("Total percobaan: " + (jmlSoal));
    System.out.println();
    System.out.println();
}
```

Class Main

```
import java.util.Scanner;

public class ScavMain {
    public static void menu() {
        System.out.println(x:"----------------------");
        System.out.println(x:"Pilih Menu");
        System.out.println(x:"----------------------");
        System.out.println(x:"1. Tampilkan pertanyaan");
        System.out.println(x:"2. Menjawab pertanyaan");
        System.out.println(x:"3. Keluar");
        System.out.println(x:"----------------------");
    }
```

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

    LinkedList linkedList = new LinkedList();
    int pilihan;

    do {
        menu();
        System.out.print(s:"Pilih 1/2/3: ");
        pilihan = gs.nextInt();
        switch (pilihan) {
            case 1:
                linkedList.addLast(soal:"Nasi yang digoreng?", kunci:"nasigoreng");
                linkedList.addLast(soal:"Kucing berkaki? (jawab dengan kata)", kunci:"empat");
                linkedList.addLast(soal:"Ketan yang hitam?", kunci:"ketanhitam");
                linkedList.print();
                break;
            case 2:
                linkedList.start();
                break;
            case 3:
                return;

            default:
                System.out.println(x:"Pilihan tidak tersedia!");
                break;
        }
    } while (pilihan == 1 || pilihan == 2 || pilihan == 3);
}
```

Hasil

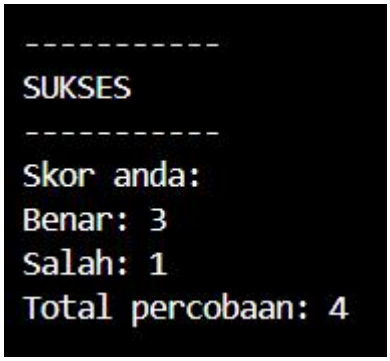
```
----------------------
Pilih Menu
----------------------
1. Tampilkan pertanyaan
2. Menjawab pertanyaan
3. Keluar
----------------------
Pilih 1/2/3: 1
-------------------------
Daftar Pertanyaan
-------------------------
-->Nasi yang digoreng?
-->Kucing berkaki? (jawab dengan kata)
-->Ketan yang hitam?
```

```
----------------------
Pilih Menu
----------------------
1. Tampilkan pertanyaan
2. Menjawab pertanyaan
3. Keluar
----------------------
Pilih 1/2/3: 2
Pertanyaan yang harus dijawab   : Nasi yang digoreng?
Isi jawaban dari pertanyaan     : nasigorang
Jawaban salah, coba lagi!

Pertanyaan yang harus dijawab   : Nasi yang digoreng?
Isi jawaban dari pertanyaan     : nasigoreng
Jawaban benar, lanjut ke soal berikutnya

Pertanyaan yang harus dijawab   : Kucing berkaki? (jawab dengan kata)
Isi jawaban dari pertanyaan     : empat
Jawaban benar, lanjut ke soal berikutnya

Pertanyaan yang harus dijawab   : Ketan yang hitam?
Isi jawaban dari pertanyaan     : ketanhitam
Jawaban benar, lanjut ke soal berikutnya
```



SUKSES

Skor anda:
Benar: 3
Salah: 1
Total percobaan: 4