KUIS 2

Nama: Rizqi Bagus Andrean

Kelas: TI-1D

Absen: 25

Link Github: https://github.com/bagusok/Tugas-Kuliah/tree/main/Semester%202/Praktek%20Algoritma/kuis%202

Kelas Node

```java
public class Node {
    int data;
    Node n;
    Node p;

    Node(){
        data=0;
        n = p = null;
    }

    Node(int data){
        this.data = data;
        n = p = null;
    }

    Node(Node prev, int data, Node next){
        this.data = data;
        this.n = next;
        this.p = prev;
    }

}
```

Kelas DLL

```java
public class DoubleLinkedList {
    Node head, tail;
    int size;

    DoubleLinkedList(){
        head = tail = null;
        size=0;
    }
    boolean isEmpty(){
        return size==0;
```

```
    }
    void addFirst(int data){
        Node nu = new Node(data);
        if(isEmpty()){
            head = tail = nu;
        }else{
            nu.n = head;
            head.p = nu;
            head = nu;
        }
        size++;
    }

    void deleteFirst(){
        head = head.n;
        head.p = null;
        size--;
    }

    void print(){
        Node tmp = head;
        while(tmp!=null){
            System.out.print(""+tmp.data+"-");
            tmp = tmp.n;
        }
        System.out.println("");
    }
    //Soal No 1 Kode Ganjil
    void addLast(int data){
        Node nu = new Node(data);
        if(isEmpty()){
            addFirst(data);
        }else{
            tail.n = nu;
            nu.p = tail;
            tail = nu;
        }
    }
    //Soal No 1 Kode Genap
    void deleteLast(){

    }
    //Soal No 2 Kode Ganjil dan Genap
    void printFromTail(){
        Node tmp = tail;
        while(tmp!=null){
            System.out.print(""+tmp.data+"-");
            tmp = tmp.p;
```

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

    }

}
```

Penjelasan

```
int data;
    Node n;
    Node p;
```

Variabel data digunakan untuk menyimpan data berupa int

Variabel Node n digunakan untuk menyimpan data next dari Node saat ini

Variabel Node p digunakan untuk menyimpan data previous dari Node saat ini

Konstruktor

```
 DoubleLinkedList(){
        head = tail = null;
        size=0;
    }
```

Konstruktor untuk pemberian nilai awal saat menginstansiasi, Dimana head = tail = null

Size = 0 atau berarti linkedlistnya kosong.

Fungsi isEmpty

```
boolean isEmpty(){
        return size==0;
    }
```

Fungsi isEmpty mengecek jika sizeny 0 berati linkedlistnya kosong, dan akan meretun true.

Fungsi addFirst

```
    void addFirst(int data){
        Node nu = new Node(data);
        if(isEmpty()){
            head = tail = nu;
        }else{
            nu.n = head;
            head.p = nu;
            head = nu;
        }
```

```
        size++;
    }
```
Fungsi addFirst digunakan untuk menambah data di element pertama.

Saat dijalankan akan mamanggil fungsi isEmpty, jika kosong maka head = tail = nu atau berarti headnya sama dengan tail atau diisi dengan node saat ini.

Jika tidak kosong maka data akan ditambah ke sebelum head saat ini head.p = nu, dan headnya akan bergeser ke node yang baru ditambah tadi.

Setelah selesai ditambah maka akan menambah size dengan 1

Fungsi deleteFirst

```
    void deleteFirst(){
        head = head.n;
        head.p = null;
        size--;
    }
```
Fungsi delete first digunakan untuk menghapus node di element pertama.

Saat fungsi ini dijalankan maka dia akan mengubah/menggeser head saat ini ke node setelah head.

Lalu membuat node yang sebelum headnya menjadi null.

Fungsi print

```
    void print(){
        Node tmp = head;
        while(tmp!=null){
            System.out.print(""+tmp.data+"-");
            tmp = tmp.n;
        }
        System.out.println("");
    }
```
Fungsi print digunakan untuk menampilkan semua node,

Saat dijalankan dia akan embuat variable temporary san dengan data awal sama dengan head.

Jika head tidak sama dengan null maka dia akan melooping terus dan menampilkan data dati node yang disimpan di variable ttemporary ini, terus dia akan mengganti isi variable temporary dengan node setelahnya sampai tidak tersisa.

Fungsi addLast

```
    void addLast(int data){
        Node nu = new Node(data);
        if(isEmpty()){
            addFirst(data);
```

```
        }else{
            tail.n = nu;
            nu.p = tail;
            tail = nu;
        }
        size++;
    }
```

Fungsi addLast digunakan untuk menambah di bagian akhir linkedlist (tail),

Saat fungsi dijalankan maka akan mengecek dulu apakah isEmpty() jika iya maka dia akan memanggil fungsi addFirst() karena linkedlisntnya masih kosong maka data akan ditambahkan di element pertama.

Jika tidak kosong maka dia akan menambah node baru di setelah tail saat ini, dan menggeser tailnya ke node yang baru ditambahkan tadi.

Setelah selesa maka sizenya akan ditambah 1

Fungsi printFromTail

```
    void printFromTail(){
        Node tmp = tail;
        while(tmp!=null){
            System.out.print(""+tmp.data+"-");
            tmp = tmp.p;
        }
        System.out.println("");

    }
```

Fungsi print from tail digunakan untuk menampilkan linkedlist secara terbalik dari bellakang.

Konsepnya sama dengan fugsi print tetapi kalua fungsi print, saat inisialisai variable temp diisi oleh head, maka ini diisi oleh tail dan akan melakukan looping untuk menampilkan data dan mengganti isi temp ke node setelahnya di setiap loopingnya.

Kelas Main

```
public class Main {
    public static void main(String[] args){
        DoubleLinkedList dll = new DoubleLinkedList();
        dll.addFirst(45);
        dll.addFirst(10);
        dll.addFirst(10);
        dll.addFirst(15);
        dll.addFirst(150);
        dll.print();
        dll.deleteFirst();
```

```
        dll.print();

        System.out.println("Data dari belakang");
        dll.printFromTail();
        //lanjutkan dengan memanggil method addLast, deleteLast, printFromTail
        //lanjutkan dengan memanggil method merge, split

        System.out.println("Data Sebelum ditambah add last");
        dll.print();

        System.out.println("Data Setelah ditambah add last");
        dll.addLast(155);
        dll.addLast(11);
        dll.addLast(13);
        dll.print();

        System.out.println();

        // TEST MERGE
        DoubleLinkedList dll1 = new DoubleLinkedList();
        dll1.addLast(1);
        dll1.addLast(2);
        dll1.addLast(3);
        System.out.println("Data dll1");
        dll1.print();

        DoubleLinkedList dll2 = new DoubleLinkedList();
        dll2.addLast(4);
        dll2.addLast(5);
        dll2.addLast(6);
        System.out.println("Data dll2");
        dll2.print();

        System.out.println("setelah di merge");
        merge(dll1, dll2);



    }
    //No. 3 Kode Ganjil
    public static void merge(DoubleLinkedList dll1,
            DoubleLinkedList dll2){
        dll1.tail.n = dll2.head;
        dll1.print();

    }
    //No. 3 Kode Genap
    public static void split(DoubleLinkedList dll){
        //ex: 2,3,4,34,2,3,45,4 (original list)
```

```
        //list 1-> 2,3,4,34
        //list 2-> 2,3,45,4
    }
}
```

Fungsi Merge

```
 public static void merge(DoubleLinkedList dll1,
            DoubleLinkedList dll2){
        dll1.tail.n = dll2.head;
        dll1.tail = dll2.tail;
        dll1.size += dll2.size;

        dll2.head.p = dll1.tail;
        dll2.head = dll1.head;
        dll2.size = dll1.size;


        System.out.println("Data dll1 dan dll2 setelah di merge, print dari
dll1");
        dll1.print();
        System.out.println("Data dll1 dan dll2 setelah di merge, print dari
dll2");
        dll2.print();

    }
```

Fungsi merge dengan type static karna akan digunakann langsung tanpa instansiasi kelas.

Digunakan untuk menngabungkan 2 linkedlist, dan membutuhkan 2 parameter yaitu linkedlis1 dan 2.

Untuk menggabungkan cukup simple yaitu dengan mengisi node setelah tail dari linkedlist1 denga head linkedlist2 dan menggeser tail dari linkedlist 1 ke tail dari linkedlist 2 dan menambah size linkedlist 1 dengan size linkedlist2.

Untuk linkedlist2 juga dilakukan hal yang mirip, Cuma yang disambung adalah headnya dengan tail linkedlist1 dan menggeser head linkedlist 2 ke linked list1.

Pengujian.

Fungsi main

```
public static void main(String[] args){
        DoubleLinkedList dll = new DoubleLinkedList();
        dll.addFirst(45);
        dll.addFirst(10);
```

```
        dll.addFirst(10);
        dll.addFirst(15);
        dll.addFirst(150);
        dll.print();
        dll.deleteFirst();
        dll.print();

        System.out.println("Data dari belakang");
        dll.printFromTail();
        //lanjutkan dengan memanggil method addLast, deleteLast, printFromTail
        //lanjutkan dengan memanggil method merge, split

        System.out.println("Data Sebelum ditambah add last");
        dll.print();

        System.out.println("Data Setelah ditambah add last");
        dll.addLast(155);
        dll.addLast(11);
        dll.addLast(13);
        dll.print();

        System.out.println();

        // TEST MERGE
        DoubleLinkedList dll1 = new DoubleLinkedList();
        dll1.addLast(1);
        dll1.addLast(2);
        dll1.addLast(3);
        System.out.println("Data dll1");
        dll1.print();

        DoubleLinkedList dll2 = new DoubleLinkedList();
        dll2.addLast(4);
        dll2.addLast(5);
        dll2.addLast(6);
        System.out.println("Data dll2");
        dll2.print();

        System.out.println("setelah di merge");
        merge(dll1, dll2);



    }
```

Pengujian add last

```
dll.addFirst(data:45);
dll.addFirst(data:10);
dll.addFirst(data:10);
dll.addFirst(data:15);
dll.addFirst(data:150);
dll.print();
dll.deleteFirst();
dll.print();

System.out.println(x:"Data dari belakang");
dll.printFromTail();
//lanjutkan dengan memanggil method addLast, deleteLast, print
//lanjutkan dengan memanggil method merge, split

System.out.println(x:"Data Sebelum ditambah add last");
dll.print();

System.out.println(x:"Data Setelah ditambah add last");
dll.addLast(data:155);
dll.addLast(data:11);
dll.addLast(data:13);
dll.print();
```

```
 2_31682476\bin' 'Main'
150-15-10-10-45-
15-10-10-45-
Data dari belakang
45-10-10-15-
Data Sebelum ditambah add last
15-10-10-45-
Data Setelah ditambah add last
15-10-10-45-155-11-13-
```

Pengujian merge

```java
// TEST MERGE
DoubleLinkedList dll1 = new DoubleLinkedList();
dll1.addLast(data:1);
dll1.addLast(data:2);
dll1.addLast(data:3);
System.out.println(x:"Data dll1");
dll1.print();

DoubleLinkedList dll2 = new DoubleLinkedList();
dll2.addLast(data:4);
dll2.addLast(data:5);
dll2.addLast(data:6);
System.out.println(x:"Data dll2");
dll2.print();

System.out.println(x:"setelah di merge");
merge(dll1, dll2);
```

```
    ◦ x The String to be printed.
System.out.println(x:"Data dll1 dan dll2 setelah di merge, print dari dll1");
dll1.print();
System.out.println(x:"Data dll1 dan dll2 setelah di merge, print dari dll2");
dll2.print();
```

```
Data dll1
1-2-3-
Data dll2
4-5-6-
setelah di merge
Data dll1 dan dll2 setelah di merge, print da
1-2-3-4-5-6-
Data dll1 dan dll2 setelah di merge, print da
1-2-3-4-5-6-
PS C:\Users\Acer\Tugas Kuliah\Semester 2\Pral
```

Baik di print dari dll1 atau 2 data tetap sama karna sudah gabung.