Laporan praktikum algoritma struktur data “doublelinkedlist”

Nama = muhamamad ircham

Kelas = D-IV SIB – 1F

NIM = 2341760115

PRAKTIKUM 12

12.2 Percobaan 1

12.2.1 Langkah Langkah percobaan

1. Membuat package dan class serta deklarasi variable dan konstruktornya

```
public class Node20 {
    int  data;
    Node08 prev, next;

    Node08(Node08 prev, int data, Node08 next)
    {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

2. membuat package dan class doublelinkedlists serta deklarasi variable dan konstruktornya

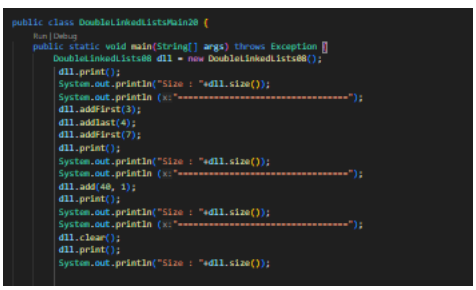
```
public class DoubleLinkedLists20 {
    Node08 head;
    int size;

    public DoubleLinkedLists20() {
        head = null;
        size = 0;
    }
```

3.tambahkan method isempty, addfirst, addlast, add, size, clear, print secara berurutan

```
Linked Lists Kosong
Size : 0
------------------
7       3       4
berhasil diisi
Size : 3
------------------
7       40      3       4
berhasil diisi
Size : 4
------------------
Linked Lists Kosong
Size : 0
```

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

    public void addFirst(int item) {
        if (isEmpty()) {
            head = new Node88(null, item, null);
        } else {
            Node88 newNode = new Node88(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;

    }

    public void addlast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node88 current = head;
            while (current.next != null) {
                current = current.next;
            }

            Node88 newNode = new Node88(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    public void add(int item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception(message:"Nilai indeks di luar batas");
        } else {
            Node88 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }

            if (current.prev == null) {
                Node88 newNode = new Node88(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node88 newNode = new Node88(current.prev, item, current);
                newNode.prev = current.prev;
                newNode.next = current;
                current.prev.next = newNode;
                current.prev = newNode;
            }
        }
        size++;
    }

    public int size() {
        return size;
    }

    public void clear() {
        head = null;
        size = 0;
    }

    public void print() {
        if (!isEmpty()) {
            Node88 tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println(x:"\nberhasil diisi");
        } else {
            System.out.println(x:"Linked Lists Kosong");
        }
    }
```

4.membuat class main dan mengisinya



```
public class DoubleLinkedListsMain20 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        DoubleLinkedLists88 dll = new DoubleLinkedLists88();
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println (x:"--------------------------------");
        dll.addFirst(3);
        dll.addlast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println (x:"--------------------------------");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println (x:"--------------------------------");
        dll.clear();
        dll.print();
        System.out.println("Size : "+dll.size());
```

Hasil kdoe program

PERTANYAAN 12.2.3

1. **Jelaskan perbedaan antara single linked list dengan double linked lists!**

Jika single linked hanya ada satu arah, yaitu dari node awal – akhir, dan hanya memiliki satu node yaitu next.
Double linked memiliki dua arah, yaitu bolak balik antar node, dan memiliki dua node yaitu next dan prev.

2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

Attribute next berfungsi untuk menunjukan node berikutnya, prev berfungsi untuk menunjukan node sebelumnya.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

```
public DoubleLinkedLists() {
    head = null;
    size = 0;
}
```

Head menunjukan pointer node pertama dalam daftar, size menyimpan jumlah node dalam daftar

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

Node newNode = new Node(null, item, head)

AddFirst befungsi untuk menambahkan data pada saat pertama kali. Karena isi daftar node pada awal berisikan 0, tidak ada data. Jadi tidak ada yang bisa untuk dilakukan prev

5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?

Untuk mengatur node yang sebelumnya menjadi head untuk menunjuk ke node baru yang kita tambahkan newNode

6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?

Node newNode = new Node(current, item, null);

AddLast befungsi untuk menambahkan data pada urutan terakhir. Karena akan di tempatkan pada posisi terakhir. Jadi tidak ada yang bisa untuk dilakukan next

7. Pada method add(), terdapat potongan kode program sebagai berikut:

```
while (i < index) {
    current = current.next;
    i++;
}
if (current.prev == null) {
    Node newNode = new Node(null, item, current);
    current.prev = newNode;
    head = newNode;
} else {
    Node newNode = new Node(current.prev, item, current);
    newNode.prev = current.prev;
    newNode.next = current;
    current.prev.next = newNode;
    current.prev = newNode;
}
```

jelaskan maksud dari bagian yang ditandai dengan kotak kuning

untuk mendeteksi apakah posisi sebelumnya null, jika ya maka posisi sebelumnya akan di isi oleh data baru, dan head akan ada pada posisi data yang baru saja ditambahkan

12.3 Percobaan 2

PERTANYAAN 12.3.3

1. **Apakah maksud statement berikut pada method *removeFirst()*?**
   head = head.next;
   head.prev = null;

   head=head.next berfungsi untuk memperbarui pointer head yang menunjukan ke node berikutnya setelah node pertama yang akan dihapus.
   head.prev=null berfungsi untuk menghapus ponter prev karena head berasa pada daftar pertama

2. **Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method** *removeLast()*?

```
Node08 current = head;
while (current.next.next != null) {
    current = current.next;
}
current.next = null;
```

Dengan menggunakan iterasi loop while, mencari current.next hingga ke daftar yang terakhir. Karena daftar terakhir tidak bisa melakukan .next, jadi data tersebut menunjukan data terakhir, dan ponter akan di simpan di variable current

3. **Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah** *remove*!

```
Node tmp = head.next;

head.next=tmp.next;
tmp.next.prev=head;
```

Karena kode program di atas tidak memeriksa kondisi spesifik dari daftar

4. **Jelaskan fungsi kode program berikut ini pada fungsi *remove*!**

```
current.prev.next = current.next;
current.next.prev = current.prev;
```

Kode program pertama berfungsi memperbarui pointer next dari node sebelumnya untuk menunjuk ke node berikutnya
Kode program kedua berfungsi memperbarui ponter prev dari node berikutnya untuk menunjuk ke node sebelumnya

12.4 Percobaan 2

12.4.1 Langkah-langkah Percobaan

1. Menambahkan method getFIrst
2. Menambahkan method getLast
3. Menambahkan method get
4. Menambahkan kode pada main

```
    public int getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception(message:"Linked List kosong");
        }
        return head.data;
    }

    public int getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception(message:"Linked List kosong");
        }
        Node20 tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception(message:"Nilai indeks di luar batas.");
        }
        Node20 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }
}
```

VERIFIKASI HASIL PERCOBAAN 12.4.2

```
Linked Lists Kosong
Size: 0
================================
7       3       4
berhasil diisi
Size: 3
================================
7       40      3       4
berhasil diisi
Size: 4
================================
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

PERTANYAAN 12.4.3

1. **Jelaskan method size() pada class DoubleLinkedLists!**
   Size() mengembalikan nilai size yang telah di atur pada awal menjalankan program
2. **Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1**
   Secara default index umumnya dimulai dari 0. Namun dengan dengan sedikit mengubah pada method add dengan memberikan validasi untuk index seperti dibawah ini

```
(index < 1 || index > size) {
```

3. **Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!**
   Penambahan pada single linked hanya satu arah dari depan ke belakang, tidak bisa menambahkan elemen di tengah daftar
   Penambahan pada double linked memiliki 2 arah, bisa dari depan dan belakang, bisa menyisipkan kode di mana saja