NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

PERTEMUAN MINGGU 12 - DOUBLE LINKED LIST

Kegiatan Praktikum 1
Waktu : 90 Menit

```java
package Jobsheet_10.DoubleLinkedList;

public class Node07 {
    int data;
    Node07 next, prev;

    Node07(Node07 prev, int data, Node07 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }

}
```

```java
package Jobsheet_10.DoubleLinkedList;

public class DoubleLinkedList {
    Node07 head;
    int size;

    public DoubleLinkedList() {
        head = null;
        size = 0;
    }
    public boolean isEmpty() {
        return head == null;
    }
    public void addFirst(int item) {
        if (isEmpty()) {
            head = new Node07(null, item, null);
        } else {
            Node07 newNode = new Node07(null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }
    public void addLast(int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node07 current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node07 newNode = new Node07(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("Nilai indeks di luar batas");
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

```
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        } else {
            Node07 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.prev == null) {
                Node07 newNode = new Node07(null, item, current);
                current.prev = newNode;
                head = newNode;
            } else {
                Node07 newNode = new Node07(current.prev, item, 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()) {
            Node07 tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println("\nBerhasil diisi");
        } else {
            System.out.println("Linked Lists Kosong");
        }
    }
}
```

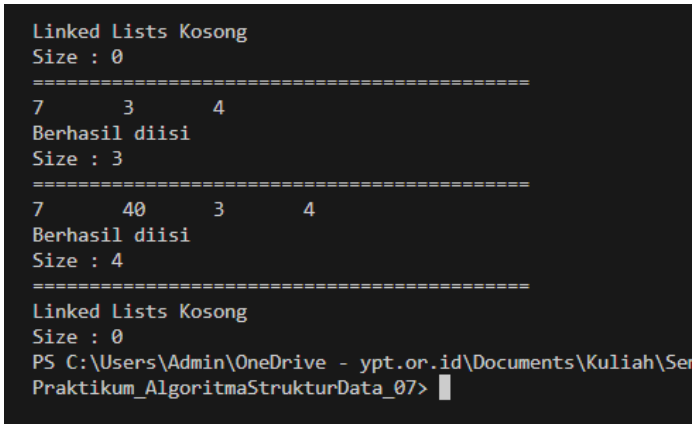
```
package Jobsheet_10.DoubleLinkedList;

public class DoubleLinkedListMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();

        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================================");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================================");
        dll.clear();
        dll.print();
        System.out.println("Size : " + dll.size());
    }
}
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

Verifikasi Hasil Percobaan



Pertanyaan Percobaan 1

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

Jawab :

- Single list hanya memiliki satu pointer yang menunjuk ke node berikutnya yaitu next. Single list hanya bisa dijalankan searah (maju), sesuai dengan pointer next atau nilai sesudahnya.
- Double linked list memiliki dua buah pointer yaitu next dan prev. Pointer next menunjuk ke node berikutnya, sedangkan pointer prev menunjuk ke node sebelumnya. Double linked list dapat mengakses node sebelumnya dan sesudahnya secara langsung karena dapat dijalankan dengan dua arah (maju / mundur).

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

Jawab : Atribut next digunakan untuk menyimpan referensi ke node berikutnya dalam linked list. Sedangkan atribut prev digunakan untuk menyimpan referensi ke node sebelumnya dalam linked list.

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

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

Jawab : head = null, untuk mengatur double linked list pada saat pertama kali dibuat tidak memiliki node (kosong). Sedangkan size = 0 untuk mengatur karena pada awalnya ukuran /

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

jumlah node pada double linked list adalah 0 / kosong karena belum ada node yang ditambahkan.

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

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

Jawab : Node prev dianggap = null karena node yang baru ditambahkan akan menjadi head (node pertama dari linked list). Karena node pertama tidak memiliki node sebelumnya maka Node prev akan bernilai null.

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

Jawab : Karena posisi awal head.prev = null, kemudian ketika ingin menambahkan data sebelum head, maka head.prev yang awalnya null kemudian diubah menunjuk ke node baru yang telah ditambahkan. Statement ini dilakukan agar node baru menjadi node paling depan (head) sehingga saling berkaitan.

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**);

Jawab : Ketika ingin menambahkan node di posisi terakhir setelah current, maka prev dari newNode akan menunjuk ke current tersebut karena current awalnya adalah node yang paling terakhir. Parameter prev = current bertujuan agar node baru terhubung dengan node sebelumnya (current). Kemudian karena ingin menambahkan node di posisi terakhir, maka next dari newNode berisi null yang artinya tidak menunjuk data setelahnya lagi.

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.

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

Jawab : Current digunakan untuk menunjuk node sekarang. jika current.prev = null, ini artinya current sedang berada di posisi head karena prev tidak menunjuk ke data apapun.

Kemudian menambahkan node baru di posisi index 0 / di posisi sebelum current. Kemudian kode dalam if akan dijalankan, kode tersebut membuat node baru dengan prev = null (karena akan menjadi node paling depan), data = item, dan next = current (node head sebelumnya).

Kemudian, pointer prev dari current (node head sebelumnya) diubah untuk menunjuk ke newNode, dan head diubah menjadi newNode (node baru menjadi node paling depan).

Kegiatan Praktikum 2
Waktu : 60 Menit

```java
public void removeFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
    } else if (size == 1) {
        removeLast();
    } else {
        head = head.next;
        head.prev = null;
        size--;
    }
}
public void removeLast() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
    } else if (head.next == null) {
        head = null;
        size--;
        return;
    }
    Node07 current = head;
    while (current.next.next != null) {
        current = current.next;
    }
    current.next = null;
    size--;
}
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

```java
public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
    } else if (index == 0) {
        removeFirst();
    } else {
        Node07 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.next == null) {
            current.prev.next = null;
        } else if (current.prev == null) {
            current = current.next;
            current.prev = null;
            head = current;
        } else {
            current.prev.next = current.next;
            current.next.prev = current.prev;
        }
        size--;
    }
}
```

```java
package Jobsheet_10.DoubleLinkedList;

public class DoubleLinkedListMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();

        dll.addLast(50);
        dll.addLast(40);
        dll.addLast(10);
        dll.addLast(20);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("=========================================");
        dll.removeFirst();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("=========================================");
        dll.removeLast();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("=========================================");
        dll.remove(1);;
        dll.print();
        System.out.println("Size : " + dll.size());
    }
}
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

Verifikasi Hasil Percobaan

```
50      40      10      20
Berhasil diisi
Size : 4
=========================================
40      10      20
Berhasil diisi
Size : 3
=========================================
40      10
Berhasil diisi
Size : 2
=========================================
40
Berhasil diisi
Size : 1
PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Ku
Praktikum_AlgoritmaStrukturData_07>
```

Pertanyaan Percobaan 2

1. Apakah maksud statement berikut pada method **removeFirst()**?

    head = head.next;
    head.prev = null;

Jawab :

- head = head.next, berfungsi untuk mengubah node head ke node berikutnya dalam linked list. Sehingga node yang sebelumnya menjadi head akan dihapus, dan node berikutnya menjadi head yang baru.
- head.prev = null, Setelah head diubah ke node berikutnya, kode ini memastikan agar node pertama tidak memiliki prev (node sebelumya). Karena head adalah node paling depan, maka prev dari head harus diatur menjadi null.

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

Jawab :

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

Perulangan while tersebut akan berjalan selama current.next.next tidak bernilai null. Sehingga, perulangan akan berhenti ketika current.next adalah node terakhir (karena current.next.next akan menjadi null). Kemudian current akan menunjuk ke node sebelum node terakhir, dan current.next akan menunjuk ke node terakhir yang ingin dihapus.

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

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;
```

Jawab : Potongan kode program di bawah ini tidak cocok untuk perintah remove karena tidak melihat kasus bagaimana jika node yang dihapus berada di posisi pertama (head) atau terakhir (tail) dari linked list. Kode tersebut hanya dapat dijalankan untuk menghapus node bagian tengah pada linked list. Tapi, jika node yang dihapus adalah head atau tail, kode tersebut akan menyebabkan kesalahan.

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

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

Jawab :
kode tersebut untuk memperbarui pointer next dan prev dari node sebelum dan sesudah node yang akan dihapus (current), sehingga double linked list tetap terhubung setelah penghapusan

- current.prev.next = current.next, Mengubah pointer next dari node sebelum current menunjuk ke node setelah current, sehingga node sebelum current terhubung ke node setelah current.
- current.next.prev = current.prev, Mengubah pointer prev dari node setelah current menunjuk ke node sebelum current, sehingga node setelah current terhubung ke node sebelum current.

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

Kegiatan Praktikum 3
Waktu : 50 Menit

```java
public int getFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List kosong");
    }
    return head.data;
}
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List kosong");
    }
    Node07 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("Nilai indeks di luar batas.");
    }
    Node07 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    } return tmp.data;
}
```

```java
dll.print();
System.out.println("Size : " + dll.size());
System.out.println("=========================================");
dll.addFirst(3);
dll.addLast(4);
dll.addFirst(7);
dll.print();
System.out.println("Size : " + dll.size());
System.out.println("=========================================");
dll.add(40, 1);
dll.print();
System.out.println("Size : " + dll.size());
System.out.println("=========================================");
System.out.println("Data awal pada Linked Lists adalah : " + dll.getFirst());
System.out.println("Data akhir pada Linked Lists adalah : " + dll.getLast());
System.out.println("Data indeks ke-1 pada Linked Lists adalah : " + dll.get(1));
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET X - DOUBLE LINKED LIST

Verifikasi Hasil Percobaan

```
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
PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Kul
Praktikum_AlgoritmaStrukturData_07>
```

Pertanyaan Percobaan 3

1. Jelaskan method **size()** pada class DoubleLinkedLists!
Jawab : Method ini digunakan untuk mengembalikan nilai dari atribut size pada objek DoubleLinkedList. Atribut size menyimpan jumlah node saat ini. Setiap ada penambahan atau penghapusan node, nilai size akan diperbarui. Method size() digunakan untuk mendapatkan jumlah node yang ada di dalam double linked list saat ini.

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!
Jawab : Melakukan modifikasi pada method add() dengan mengurangi 1 dari indeks yang diterima agar dapat dimulai dari indeks ke-1, yaitu (else if (index < 1 || index > size + 1)). Atau setiap kali mengakses atau mengubah node menggunakan index, mengurangi nilai indeks yang diterima dengan 1.

3. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
Jawab :

- Fungsi add pada DoubleLinkedList lebih fleksibel karena setiap node memiliki pointer ke node sebelumnya (prev) dan node setelahnya (next). Lebih mudah mengakses node sebelum dan sesudah posisi yang diinginkan melalui pointer prev dan next sehingga memudahkan operasi penambahan di tengah, addFirst, addLast.
- SingleLinkedList lebih terbatas karena node hanya memiliki pointer ke node setelahnya (next). Untuk menambahkan node baru di tengah, harus melakukan traversal dari awal (head) sampai menemukan posisi. addFirst hanya perlu mengubah

head, sedangkan addLast membutuhkan traversal dari awal sampai mencapai node terakhir.

4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

```
public boolean isEmpty(){
    if(size ==0){
        return true;
    } else{
        return false;
    }
}
```

(a)

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

(b)

Jawab :

a. Menggunakan variabel size untuk menentukan DoubleLinkedList kosong / tidak. Jika size bernilai 0, nilai true yang artinya kosong akan di return. Jika size tidak bernilai 0, nilai false yang artinya tidak kosong akan di return
b. Menggunakan pointer head untuk menentukan apakah double linked list kosong atau tidak. Jika head bernilai null, maka double linked list tidak memiliki node sama sekali

Link GitHub Pribadi :
https://github.com/ayleenrq/Prak_AlgoritmaStrukturData_07.git