Nama : Deanissa Sherly Sabilla
Kelas / Absen : 1B SIB / 06

JOBSHEET 12
Double Linked Lists

12.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami algoritma double linked lists;
2. membuat dan mendeklarasikan struktur algoritma double linked lists;
3. menerapkan algoritma double linked lists dalam beberapa *study case*.

12.2 Kegiatan Praktikum 1

12.2.1 Percobaan 1

Pada percobaan 1 ini akan dibuat class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan linked list, belakang ataupun indeks tertentu pada linked list).

1. Perhatikan diagram class Node dan class DoublelinkedLists di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.

Node
data: int prev: Node next: Node
Node(prev: Node, data:int, next:Node)

DoubleLinkedLists
head: Node size : int

DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void

2. Buat paket baru dengan nama **doublelinkedlists**
3. Buat class di dalam paket tersebut dengan nama **Node**

```
package doublelinkedlists;

/**...4 lines */
public class Node {

}
```

4. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
4       int data;
5       Node prev, next;
```

5. Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.

```
7       Node(Node prev, int data, Node next){
8           this.prev=prev;
9           this.data=data;
10          this.next=next;
11      }
12  }
```

6. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:

```
package doublelinkedlists;

/**...4 lines */
public class DoubleLinkedLists {

}
```

7. Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
8       Node head;
9       int size;
```

8. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

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

9. Buat method **isEmpty()**. Method ini digunakan untuk memastikan kondisi linked list kosong.

```
16      public boolean isEmpty(){
17          return head == null;
18      }
```

10. Kemudian, buat method **addFirst().** Method ini akan menjalankan penambahan data di bagian depan linked list.

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

11. Selain itu pembuatan method **addLast()** akan menambahkan data pada bagian belakang linked list.

```
public void addLast(int item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
        Node current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node newNode = new Node(current, item, null);
        current.next = newNode;
        size++;
    }
}
```

12. Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method **add(int item, int index)**

```
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");
    } else {
        Node current = head;
        int i = 0;
        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;
        }
    }
    size++;
}
```

13. Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis, sehingga dapat dibuat method **size()** untuk mendapatkan nilai dari size.

```
138     public int size(){
139         return size;
140     }
```

14. Selanjutnya dibuat method **clear()** untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong.

```
141     public void clear(){
142         head = null;
143         size = 0;
144     }
```

15. Untuk mencetak isi dari linked lists dibuat method **print().** Method ini akan mencetak isi linked lists berapapun size-nya. Jika kosong akan dimunculkan suatu pemberitahuan bahwa linked lists dalam kondisi kosong.

```
public void print() {
    if (!isEmpty()) {
        Node 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");
    }
}
```

16. Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.

```
package doublelinkedlists;

/**...4 lines */
public class DoubleLinkedListsMain {
    public static void main(String[] args) {

    }
}
```

17. Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

```
19          doubleLinkedList dll = new doubleLinkedList();
20          dll.print();
21          System.out.println("Size : "+dll.size());
22          System.out.println("=====================================");
23          dll.addFirst(3);
24          dll.addLast(4);
25          dll.addFirst(7);
26          dll.print();
27          System.out.println("Size : "+dll.size());
28          System.out.println("=====================================");
29          dll.add(40, 1);
30          dll.print();
31          System.out.println("Size : "+dll.size());
32          System.out.println("=====================================");
33          dll.clear();
34          dll.print();
35          System.out.println("Size : "+dll.size());
```

12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec
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
========================
------------------------------------
BUILD SUCCESS
------------------------------------
```

CLASS NODE

doublelinkedlists > Node06.java > {} doublelinkedlists

```java
package doublelinkedlists;
/**
 * Node06
 */
public class Node06 {

    int data;
    Node06 prev, next;

    Node06 (Node06 prev, int data, Node06 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

CLASS DOUBLE LINKED LIST

```java
package doublelinkedlists;
public class DoubleLinkedLists06 {
    Node06 head;
    int size;

    public DoubleLinkedLists06 () {
        head = null;
        size = 0;
    }
    public boolean isEmpty() {
        return head == null;
    }
    public void addFirst(int item) {
        if (isEmpty()) {
            head = new Node06 (prev:null, item, next:null);
        } else {
            Node06 newNode = new Node06 (prev:null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }
```

```java
    public void addLast (int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node06 current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node06 newNode = new Node06 (current, item, next:null);
            current.next = newNode;
            size++;
        }
    }
```

```java
    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 diluar batas");
        } else {
            Node06 current = head;
            int i = 0;
            while (i < index - 1) {
                current = current.next;
                i++;
            }
            if (current == null) {
                addLast(item); // If index points beyond the last element, add at the end
            } else {
                Node06 newNode = new Node06(current, item, current.next);
                if (current.next != null) {
                    current.next.prev = newNode;
                }
                current.next = newNode;
            }
            size++;
        }
    }
```

```java
    public int size() {
        return size;
    }
    public void clear() {
        head = null;
        size = 0;
    }
    public void print() {
        if (!isEmpty()) {
            Node06 tmp = head;
            while (tmp != null) {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println(x:"\n berhasil diisi");
        } else {
            System.out.println(x:"Linked Lists Kosong");
        }
    }
}
```

CLASS MAIN

```
5  public class DoubleLinkedListsMain {
6
       Run | Debug
7      public static void main(String[] args) throws Exception {
8          DoubleLinkedLists06 dll = new  DoubleLinkedLists06();
9          dll.print();
10         System.out.println("Size : "+dll.size());
11         System.out.println(x:"=======================================");
12         dll.addFirst(item:3);
13         dll.addLast(item:4);
14         dll.addFirst(item:7);
15         dll.print();
16         System.out.println("Size : "+dll.size());
17         System.out.println(x:"=======================================");
18         dll.add(item:40,index:1);
19         dll.print();
20         System.out.println("Size : "+dll.size());
21         System.out.println(x:"=======================================");
22         dll.clear();
23         dll.print();
24         System.out.println("Size : "+dll.size());
25     }
26 }
```

HASIL OUTPUT

```
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
PS C:\Users\TOSHIBA\Semester 2\Jobsheet12>
```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
   - **Single linked list** adalah struktur data dimana setiap elemen dalam list hanya memiliki referensi ke elemen berikutnya dalam urutan, Sedangkan **Double linked list** adalah struktur data dimana setiap elemen dalam list memiliki dua referensi: satu ke elemen sebelumnya dan satu ke elemen berikutnya.
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   - Atribut **next** dan **prev** digunakan untuk menyimpan referensi ke node berikutnya (next) dan sebelumnya (prev) 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;
}
```

- Inisialisasi atribut **head** dan **size** dilakukan untuk menetapkan keadaan awal dari linked list yang akan dibuat.
  - **head** digunakan untuk menunjukkan elemen pertama (atau awal) dari double linked list.
  - **size** digunakan untuk melacak jumlah elemen dalam double linked list.

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

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

- Karena menambahkan elemen baru di awal linked list. Elemen baru akan menjadi elemen pertama dalam linked list, sehingga tidak ada elemen sebelumnya yang harus ditunjuk oleh prev.

5. Perhatikan pada method **addFirst().** Apakah arti statement head.prev = newNode ?
   - Artinya elemen sebelumnya dari elemen pertama (yang saat ini adalah head) menunjuk ke node baru yang ditambahkan sebagai elemen pertama dalam linked list.
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);
```

   - Dalam membuat objek **Node** dengan mengatur `prev` menjadi `current` dan `next` menjadi `null` berarti menambahkan elemen baru di akhir linked list. Node baru ini menjadi elemen terakhir dalam linked list, sehingga `prev`-nya adalah node yang saat ini adalah elemen terakhir, sedangkan tidak ada node berikutnya setelahnya, maka `next`-nya diatur menjadi `null`.
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.

- Yaitu menangani kasus ketika ingin menambahkan elemen baru di posisi awal linked list (indeks 0), di mana saat ini tidak ada elemen sebelumnya.

12.3 Kegiatan Praktikum 2

12.3.1 Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists. Method tambahan tersebut akan ditambahkan sesuai pada diagram class berikut ini.

DoubleLinkedLists
head: Node size : int
DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void **removeFirst(): void removeLast(): void remove(index:int):void**

1. Buatlah method **removeFirst()** di dalam class **DoubleLinkedLists.**

```
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--;
    }
}
```

2. Tambahkan method **removeLast()** di dalam class **DoubleLinkedLists.**

```
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;
    }
    Node current = head;
    while (current.next.next != null) {
        current = current.next;
    }
    current.next = null;
    size--;
}
```

3. Tambahkan pula method **remove(int index)** pada class **DoubleLinkedLists** dan amati hasilnya.

```
public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
        Node 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--;
    }
}
```

4. Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada **main class.**

```
42              dll.addLast(50);
43              dll.addLast(40);
44              dll.addLast(10);
45              dll.addLast(20);
46              dll.print();
47              System.out.println("Size : "+dll.size());
48              System.out.println("==================================");
49              dll.removeFirst();
50              dll.print();
51              System.out.println("Size : "+dll.size());
52              System.out.println("==================================");
53              dll.removeLast();
54              dll.print();
55              System.out.println("Size : "+dll.size());
56              System.out.println("==================================");
57              dll.remove(1);
58              dll.print();
59              System.out.println("Size : "+dll.size());
```

12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec
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
------------------------------------
BUILD SUCCESS
------------------------------------
```

CLASS DOUBLE LINKED LIST (TAMBAHAN)

```java
    public void removeFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception (message:"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 (message:"Linked list masih kosong, tidak dapat dihapus!");
        } else if (head.next == null) {
            head = null;
            size--;
            return;
        }
        Node06 current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        current.next = null;
        size--;
    }
    public void remove (int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception (message:"Nilai indeks diluar batas");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node06 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--;
        }
```

CLASS MAIN

```java
    public static void main(String[] args) throws Exception {
        DoubleLinkedLists06 dll = new  DoubleLinkedLists06();
        dll.addLast(item:50);
        dll.addLast(item:40);
        dll.addLast(item:10);
        dll.addLast(item:20);
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println(x:"=======================================");
        dll.removeFirst();
        //dll.addFirst(3);
        //dll.addLast(4);
        //dll.addFirst(7);
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println(x:"=======================================");
        dll.removeLast();
        //dll.add(40,1);
        dll.print();
        System.out.println("Size : "+dll.size());
        System.out.println(x:"=======================================");
        dll.remove(index:1);
        //dll.clear();
        dll.print();
        System.out.println("Size : "+dll.size());
```

HASIL OUTPUT

```
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\TOSHIBA\Semester 2\Jobsheet12>
```

12.3.3 Pertanyaan Percobaan

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

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

   - Yaitu untuk menghapus elemen pertama, **head = head.next;** Statement ini menetapkan head ke elemen berikutnya dari elemen pertama (yang akan dihapus). **head.prev = null;** Setelah menetapkan head ke elemen kedua, memastikan bahwa prev dari elemen kedua (yang sebelumnya adalah elemen pertama) diatur menjadi null

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method **removeLast()**?
   - Menggunakan pendekatan iteratif untuk mencari elemen terakhir dalam linked list. Berikut adalah langkah-langkah untuk mendeteksi posisi data di bagian akhir:
     1. Menginisialisasi sebuah variabel current yang menunjuk ke elemen pertama (head) dari linked list.
     2. Selanjutnya, lakukan iterasi melalui linked list dengan menggunakan loop while. Ini akan terus maju ke elemen berikutnya selama elemen berikutnya (current.next) tidak null. Dalam konteks ini, saat sampai pada elemen terakhir, current.next akan null.
     3. Ketika current.next.next menjadi null, ini menunjukkan bahwa current saat ini adalah elemen kedua terakhir dalam linked list.
     4. Setelah menemukan elemen terakhir, atur current.next menjadi null, sehingga elemen terakhir dihapus dari 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;
```

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

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

- **(Jawaban no 3)**
  Karena tidak memperhitungkan kasus-kasus khusus yang mungkin terjadi saat menghapus elemen dari linked list, terutama ketika menghapus elemen pertama atau terakhir.
- **(Jawaban no 4)**
  Untuk menghapus node yang ditunjuk oleh current dari linked list dengan memperbarui referensi dari node sebelumnya dan sesudahnya.

12.4 Kegiatan Praktikum 3

12.4.1 Tahapan Percobaan

Pada praktikum 3 ini dilakukan uji coba untuk mengambil data pada linked list dalam 3 kondisi, yaitu mengambil data paling awal, paling akhir dan data pada indeks tertentu dalam linked list. Method mengambil data dinamakan dengan **get**. Ada 3 method get yang dibuat pada praktikum ini sesuai dengan diagram class DoubleLinkedLists.

DoubleLinkedLists
head: Node size : int
DoubleLinkedLists() isEmpty(): boolean addFirst (): void addLast(): void add(item: int, index:int): void size(): int clear(): void print(): void removeFirst(): void removeLast(): void remove(index:int):void **getFirst(): int getLast() : int get(index:int): int**

1. Buatlah method **getFirst()** di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists.

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

2. Selanjutnya, buatlah method **getLast()** untuk mendapat data pada akhir linked lists.

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

3. Method **get(int index)** dibuat untuk mendapatkan data pada indeks tertentu

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

4. Pada main class tambahkan potongan program berikut dan amati hasilnya!

```
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));
```

12.4.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec (default-cli)
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
------------------------------------------------
BUILD SUCCESS
------------------------------------------------
```

CLASS DOUBLE LINKED LIST (TAMBAHAN)

```
130     public int getFirst() throws Exception {
131         if (isEmpty()) {
132             throw new Exception (message:"Linked List kosong");
133         }
134         return head.data;
135     }
136     public int getLast() throws Exception {
137         if (isEmpty()) {
138             throw new Exception (message:"Linked List kosong");
139         }
140         Node06 tmp = head;
141         while (tmp.next != null) {
142             tmp = tmp.next;
143         }
144         return tmp.data;
145     }
146     public int get (int index) throws Exception {
147         if (isEmpty() || index >= size) {
148             throw new Exception (message:"Nilai indeks di luar batas.");
149         }
150         Node06 tmp = head;
151         for (int i = 0; i < index; i++) {
152             tmp = tmp.next;
153         }
154         return tmp.data;
```

CLASS MAIN

```
5   public class DoubleLinkedListsMain {
7       public static void main(String[] args) throws Exception {
8           DoubleLinkedLists06 dll = new  DoubleLinkedLists06();
9           //dll.addLast(50);
10          //dll.addLast(40);
11          //dll.addLast(10);
12          //dll.addLast(20);
13          dll.print();
14          System.out.println("Size : "+dll.size());
15          System.out.println(x:"=====================================");
16          //dll.removeFirst();
17          dll.addFirst(item:3);
18          dll.addLast(item:4);
19          dll.addFirst(item:7);
20          dll.print();
21          System.out.println("Size : "+dll.size());
22          System.out.println(x:"=====================================");
23          //dll.removeLast();
24          dll.add(item:40,index:1);
25          dll.print();
26          System.out.println("Size : "+dll.size());
27          System.out.println(x:"=====================================");
28          System.out.println("Data awal pada linked adalah : " +dll.getFirst());
29          System.out.println("Data akhir pada linked adalah : " +dll.getLast());
30          System.out.println("Data awal pada linked adalah : " +dll.get(index:1));
```

HASIL OUTPUT

```
Linked Lists Kosong
Size : 0
=========================================
7       3       4
 berhasil diisi
Size : 3
=========================================
7       40      3       4
 berhasil diisi
Size : 4
=========================================
Data awal pada linked adalah : 7
Data akhir pada linked adalah : 4
Data awal pada linked adalah : 40
PS C:\Users\TOSHIBA\Semester 2\Jobsheet12>
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method **size()** pada class DoubleLinkedLists!
   - Untuk mengembalikan jumlah elemen yang ada dalam linked list.
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!
   - Untuk mengatur indeks pada double linked list agar dimulai dari indeks ke-1, bisa menggunakan pendekatan sederhana dengan menambahkan sebuah node dummy di awal linked list.
3. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
   - **Fungsi Add dalam Double Linked List :** Penambahan node baru di tengah linked list dengan lebih efisien karena setiap node memiliki dua referensi, satu ke node sebelumnya dan satu ke node berikutnya.
   - **Fungsi Add dalam Single Linked List :** penambahan di tengah linked list karena setiap node hanya memiliki satu referensi ke node berikutnya, sehingga operasi penambahan di tengah linked list memerlukan pencarian dari awal linked list untuk menemukan node sebelumnya.
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)

- Pada gambar a : terdapat if else yang dimana jika size = 0 maka true sedangkan jika tidak maka false.
- Pada gambar b : tidak menggunakan if else hanya langsung apakah head bernilai null jika benar maka kondisi nya adalah true