



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

Percobaan 1

```
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
```

```
1 package p12;
2
3 public class DoubleLinkedListMain {
4     public static void main(String[] args) throws Exception {
5         DoubleLinkedList dll = new DoubleLinkedList();
6         dll.print();
7         System.out.println("Size : " + dll.size());
8         System.out.println("=====");
9         dll.addFirst(3);
10        dll.addLast(4);
11        dll.addFirst(7);
12        dll.print();
13
14        System.out.println("Size : " + dll.size());
15        System.out.println("=====");
16        dll.add(40, 1);
17        dll.print();
18        System.out.println("Size : " + dll.size());
19        System.out.println("=====");
20        dll.clear();
21        dll.print();
22        System.out.println("Size : " + dll.size());
23
24    }
25 }
26
```



NAMA : Roy Wijaya

NIM : 2341720120

KELAS : 1-G

MATERI : Double Linked List (Jobsheet 10)

```
1 package P12;
2
3 public class DoubleLinkedList {
4     Node head;
5     int size;
6
7     DoubleLinkedList(Node head, int size) {
8         head = null;
9         size = 0;
10    }
11
12    DoubleLinkedList() {
13    }
14
15
16    boolean isEmpty() {
17        return head == null;
18    }
19
20    void addFirst(int item) {
21        if (isEmpty()) {
22            head = new Node(null, item, null);
23        } else {
24            Node newNode = new Node(null, item, head);
25            head.prev = newNode;
26            head = newNode;
27        }
28        size++;
29    }
30
31    void addLast(int item) {
32        if (isEmpty()) {
33            addFirst(item);
34        } else {
35            Node current = head;
36            while (current.next != null) {
37                current = current.next;
38            }
39            Node newNode = new Node(current, item, null);
40            current.next = newNode;
41            size++;
42        }
43    }
44
45    void add(int item, int index) throws Exception {
46        if (isEmpty()) {
47            addFirst(item);
48        } else if (index < 0 || index > size) {
49            throw new Exception("Nilai indeks di luar batas");
50        } else {
51            Node current = head;
52            int i = 0;
53            while (i < index) {
54                current = current.next;
55                i++;
56            }
57            if (current.prev == null) {
58                Node newNode = new Node(null, item, current);
59                current.prev = newNode;
60                head = newNode;
61            } else {
62                Node newNode = new Node(current.prev, item, current);
63                current.prev.next = newNode;
64                current.prev = newNode;
65            }
66        }
67        size++;
68    }
69
70    public int size() {
71        return size;
72    }
73
74    public void clear() {
75        head = null;
76        size = 0;
77    }
78
79    public void print() {
80        if (!isEmpty()) {
81            Node tmp = head;
82            while (tmp != null) {
83                System.out.print(tmp.data + " ");
84                tmp = tmp.next;
85            }
86            System.out.println("\nberhasil diisi");
87        } else {
88            System.out.println("Linked Lists Kosong");
89        }
90    }
91 }
92
```

```
1 package P12;
2
3 public class Node {
4     int data;
5     Node prev;
6     Node next;
7
8     Node(Node prev, int data, Node next) {
9         this.prev = prev;
10        this.data = data;
11        this.next = next;
12    }
13 }
14
15
```



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

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

Jawab:

Double linked list mempunyai 2 pointer yaitu next dan prev sedangkan linked list hanya mempunyai 1 pointer yaitu next.

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

Jawab:

Atribut next digunakan untuk menunjuk node selanjutnya sedangkan atribut prev digunakan untuk menunjuk 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;  
}
```

Jawab:

Untuk mendeklarasikan linked list diawal bahwa nilai head = null, size = 0;

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:

Ketika pembuatan node baru dengan menggunakan method addFirst, node baru tersebut akan menggantikan head, karena secara konsep nilai prev dari head adalah null.



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

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

Jawab:

Arti statement tersebut adalah nilai `prev` dari `head` mengarah ke `newNode` atau node baru tersebut. Dan node baru tersebut akan menjadi `head`.

6. Perhatikan isi method `addLast()`, apa arti dari pembuatan object `Node` dengan mengisi parameter `prev` dengan `current`, dan `next` dengan `null`? `Node newNode = new Node(current, item, null);`

Jawab:

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

Kode diatas adalah melakukan traverse, sehingga didapatkan nilai `current` adalah node terakhir. Lalu nilai `next` diisi dengan `null` karena new node tersebut akan menjadi tail, dimana konsep tail adalah nilai `nextnya` adalah `null`.

Nilai `prev` diisi dengan `current` dan `next` diisi dengan `null`, karena agar new node tersebut mengarah ke nilai node sebelumnya dan mengarah ke `null` karena new node tersebut menjadi tail



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

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

jawab:

pada perulangan while dilakukan traverse untuk menentukan nilai current. Kemudian dilakukan pengecekan kondisi apakah `current.prev == null`. Pengecekan kondisi tersebut akan terpenuhi ketika nilai current adalah head karena nilai prev dari head adalah null. Lalu dilakukan pembuatan `newNode` dengan nilai prev adalah null, next adalah current. Pada baris ketiga yaitu `current.prev = newNode` agar nilai current atau head mengarah ke node yang baru. Kemudian pada baris terakhir terjadi inisialisasi `head = newNode`, agar node baru tersebut menjadi head.



NAMA : Roy Wijaya

NIM : 2341720120

KELAS : 1-G

MATERI : Double Linked List (Jobsheet 10)

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

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

```
1 package p12;
2
3 public class DoubleLinkedListMain {
4     public static void main(String[] args) throws Exception {
5         DoubleLinkedList dll = new DoubleLinkedList();
6         dll.print();
7         System.out.println("Size : " + dll.size());
8         System.out.println("=====");
9         dll.addFirst(3);
10        dll.addLast(4);
11        dll.addFirst(7);
12        dll.print();
13
14        System.out.println("Size : " + dll.size());
15        System.out.println("=====");
16        dll.add(40, 1);
17        dll.print();
18        System.out.println("Size : " + dll.size());
19        System.out.println("=====");
20        dll.clear();
21        dll.print();
22        System.out.println("Size : " + dll.size());
23
24        dll.addLast(50);
25        dll.addLast(40);
26        dll.addLast(10);
27        dll.addLast(20);
28        dll.print();
29        System.out.println("Size : " + dll.size());
30        System.out.println("=====");
31        dll.removeFirst();
32        dll.print();
33        System.out.println("Size : " + dll.size());
34        System.out.println("=====");
35        dll.removeLast();
36        dll.print();
37        System.out.println("Size : " + dll.size());
38        System.out.println("=====");
39        dll.remove(1);
40        dll.print();
41        System.out.println("Size : " + dll.size());
42
43    }
44 }
45
```

```
1 package p12;
2
3 public class Node {
4     int data;
5     Node prev;
6     Node next;
7
8     Node(Node prev, int data, Node next) {
9         this.prev = prev;
10        this.data = data;
11        this.next = next;
12    }
13 }
14 }
15
```



NAMA : Roy Wijaya

NIM : 2341720120

KELAS : 1-G

MATERI : Double Linked List (Jobsheet 10)

```
1 package El1;
2
3 public class DoubleLinkedList {
4     Node head;
5     int size;
6
7     DoubleLinkedList() {
8         head = null;
9         size = 0;
10    }
11
12    boolean isEmpty() {
13        return head == null;
14    }
15
16    void addFirst(int item) {
17        if (isEmpty()) {
18            head = new Node(null, item, null);
19        } else {
20            Node newNode = new Node(null, item, head);
21            head.prev = newNode;
22            head = newNode;
23        }
24        size++;
25    }
26
27    void addLast(int item) {
28        if (isEmpty()) {
29            addFirst(item);
30        } else {
31            Node current = head;
32            while (current.next != null) {
33                current = current.next;
34            }
35            Node newNode = new Node(current, item, null);
36            current.next = newNode;
37            size++;
38        }
39    }
40
41    void add(int item, int index) throws Exception {
42        if (isEmpty()) {
43            addFirst(item);
44        } else if (index < 0 || index > size) {
45            throw new Exception("Nilai indeks di luar batas");
46        } else {
47            Node current = head;
48            int i = 0;
49            while (i < index) {
50                current = current.next;
51                i++;
52            }
53            if (current.prev == null) {
54                Node newNode = new Node(null, item, current);
55                current.prev = newNode;
56                head = newNode;
57            } else {
58                Node newNode = new Node(current.prev, item, current);
59                current.prev.next = newNode;
60                current.prev = newNode;
61            }
62            size++;
63        }
64    }
65
66    public int size() {
67        return size;
68    }
69
70    public void clear() {
71        head = null;
72        size = 0;
73    }
74
75    public void print() {
76        if (isEmpty()) {
77            Node tmp = head;
78            while (tmp != null) {
79                System.out.print(tmp.data + " ");
80                tmp = tmp.next;
81            }
82            System.out.println("\nBerhasil diisi");
83        } else {
84            System.out.println("Linked Lists Kosong");
85        }
86    }
87
88    public void removeFirst() throws Exception {
89        if (isEmpty()) {
90            throw new Exception("Linked List masih kosong, tidak dapat dihapus");
91        } else if (size == 1) {
92            removeLast();
93        } else {
94            head = head.next;
95            head.prev = null;
96            size--;
97        }
98    }
99
100    public void removeLast() throws Exception {
101        if (isEmpty()) {
102            throw new Exception("Linked list masih kosong, tidak dapat dihapus");
103        } else if (head.next == null) {
104            head = null;
105            size--;
106            return;
107        }
108        Node current = head;
109        while (current.next.next != null) {
110            current = current.next;
111        }
112        current.next = null;
113        size--;
114    }
115
116    public void remove(int index) throws Exception {
117        if (isEmpty() || index > size) {
118            throw new Exception("Nilai indeks di luar batas");
119        } else if (index == 0) {
120            removeFirst();
121        } else {
122            Node current = head;
123            int i = 0;
124            while (i < index) {
125                current = current.next;
126                i++;
127            }
128            if (current.next == null) {
129                current.prev.next = null;
130            } else if (current.prev == null) {
131                current = current.next;
132                current.prev = null;
133                head = current;
134            } else {
135                current.prev.next = current.next;
136                current.next.prev = current.prev;
137            }
138            size--;
139        }
140    }
141 }
142
```



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

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

Jawab:

Head = head.next => pointer head akan diatur ke node berikutnya

Head.prev = null => pointer prev dari node baru head diatur menjadi null dan menjadi node pertama dalam list dan tidak memiliki node sebelumnya.

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

Jawab:

Untuk mendeteksi posisi data pada bagian akhir adalah dengan iterasi(traverse).

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

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:

Jika node yang dihapus adalah node terakhir akan menyebabkan nullpointerexception



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

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

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

Jawab:

Menghubungkan node sebelum dan sesudah node yang akan dihapus dalam double linked list, sehingga daftar tetap terhubung dengan benar setelah node di tengah-tengah dihapus



NAMA : Roy Wijaya

NIM : 2341720120

KELAS : 1-G

MATERI : Double Linked List (Jobsheet 10)

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

```
1 package p12;  
2  
3 public class Node {  
4     int data;  
5     Node prev;  
6     Node next;  
7  
8     Node(Node prev, int data, Node next) {  
9         this.prev = prev;  
10        this.data = data;  
11        this.next = next;  
12    }  
13 }  
14 }  
15
```

```
1 package p12;  
2  
3 public class DoubleLinkedListMain {  
4     public static void main(String[] args) throws Exception {  
5         DoubleLinkedList dll = new DoubleLinkedList();  
6  
7         dll.print();  
8         System.out.println("Size " + dll.size());  
9         System.out.println("=====");  
10        dll.addFirst(3);  
11        dll.addLast(4);  
12        dll.addFirst(7);  
13        dll.print();  
14        System.out.println("Size: " + dll.size());  
15        System.out.println("=====");  
16        dll.add(40, 1);  
17        dll.print();  
18        System.out.println("Size: " + dll.size());  
19        System.out.println("=====");  
20        System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());  
21        System.out.println("Data akhir pada Linked Lists adalah: " + dll.getLast());  
22        System.out.println("Data indeks ke-1 pada Linked Lists adalah: " + dll.get(1));  
23  
24    }  
25 }  
26
```



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

```
1 package MD;
2
3 public class DoubleLinkedList {
4     Node head;
5     int size;
6
7     DoubleLinkedList() {
8         head = null;
9         size = 0;
10    }
11
12    boolean isEmpty() {
13        return head == null;
14    }
15
16    void addFirst(int item) {
17        if (isEmpty()) {
18            head = new Node(null, item, null);
19        } else {
20            Node newnode = new Node(null, item, head);
21            head.prev = newnode;
22            head = newnode;
23        }
24        size++;
25    }
26
27    void addLast(int item) {
28        if (isEmpty()) {
29            addFirst(item);
30        } else {
31            Node current = head;
32            while (current.next != null) {
33                current = current.next;
34            }
35            Node newnode = new Node(current, item, null);
36            current.next = newnode;
37            size++;
38        }
39    }
40
41    void addAt(int item, int index) throws Exception {
42        if (isEmpty()) {
43            addFirst(item);
44        } else if (index < 0 || index > size) {
45            throw new Exception("Nilai index di luar batas");
46        } else {
47            Node current = head;
48            for (int i = 0; i < index; i++) {
49                current = current.next;
50            }
51            if (current.prev == null) {
52                Node newnode = new Node(null, item, current);
53                current.prev = newnode;
54                head = newnode;
55            } else {
56                Node newnode = new Node(current.prev, item, current);
57                current.prev.next = newnode;
58                current.prev = newnode;
59            }
60            size++;
61        }
62    }
63
64    public int size() {
65        return size;
66    }
67
68    public void clear() {
69        head = null;
70        size = 0;
71    }
72
73    public void print() {
74        if (isEmpty()) {
75            Node tmp = head;
76            while (tmp != null) {
77                System.out.print(tmp.data + " ");
78                tmp = tmp.next;
79            }
80            System.out.println("\nBerhasil di print");
81        } else {
82            System.out.println("Isi dari list kosong");
83        }
84    }
85
86    public void removeFirst() throws Exception {
87        if (isEmpty()) {
88            throw new Exception("Linked List masih kosong. Tidak dapat dihapus");
89        } else if (size == 1) {
90            removeLast();
91        } else {
92            head = head.next;
93            head.prev = null;
94            size--;
95        }
96    }
97
98    public void removeLast() throws Exception {
99        if (isEmpty()) {
100            throw new Exception("Linked List masih kosong. Tidak dapat dihapus");
101        } else if (head.next == null) {
102            head = null;
103            size = 0;
104            return;
105        } else {
106            Node current = head;
107            while (current.next.next != null) {
108                current = current.next;
109            }
110            current.next = null;
111            size--;
112        }
113    }
114
115    public void remove(int index) throws Exception {
116        if (isEmpty()) || index == size() {
117            throw new Exception("Nilai index di luar batas");
118        } else if (index == 0) {
119            removeFirst();
120        } else {
121            Node current = head;
122            for (int i = 0; i < index; i++) {
123                current = current.next;
124            }
125            if (current.next == null) {
126                current.prev.next = null;
127            } else if (current.prev == null) {
128                current = current.next;
129                current.prev = null;
130            } else {
131                current.prev.next = current.next;
132                current.next.prev = current.prev;
133            }
134            size--;
135        }
136    }
137
138    public int getFirst() throws Exception {
139        if (isEmpty()) {
140            throw new Exception("Linked List kosong");
141        }
142        return head.data;
143    }
144
145    public int getLast() throws Exception {
146        if (isEmpty()) {
147            throw new Exception("Linked List kosong");
148        }
149        Node tmp = head;
150        while (tmp.next != null) {
151            tmp = tmp.next;
152        }
153        return tmp.data;
154    }
155
156    public int get(int index) throws Exception {
157        if (isEmpty()) || index > size() {
158            throw new Exception("Nilai index di luar batas");
159        }
160        Node tmp = head;
161        for (int i = 0; i < index; i++) {
162            tmp = tmp.next;
163        }
164        return tmp.data;
165    }
166 }
```



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

PERTANYAAN

1. Jelaskan method size() pada class DoubleLinkedLists!

Jawab:

Untuk mendapatkan nilai size atau method size tersebut mengembalikan nilai size.

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!

Jawab:

Untuk mengatur indeks pada double linked list (DLL) agar dimulai dari indeks ke-1, perlu dilakukan penyesuaian pada setiap metode yang mengacu pada indeks, terutama metode add(int item, int index), remove(int index), dan get(int index). Saat ini, indeks pada DLL Anda dimulai dari 0, jadi perlu menambahkan atau mengurangi 1 pada parameter indeks saat digunakan.

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

Jawab:

DLL: Memungkinkan traversal dua arah, sehingga lebih fleksibel untuk operasi yang membutuhkan akses ke elemen sebelumnya.

SLL: Hanya memungkinkan traversal satu arah, sehingga beberapa operasi mungkin memerlukan traversal tambahan dari awal untuk mengakses elemen sebelumnya.



NAMA : Roy Wijaya
NIM : 2341720120
KELAS : 1-G
MATERI : Double Linked List (Jobsheet 10)

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:

kode a memeriksa menggunakan size apakah nilai size 0 jika benar mengembalikan nilai true jika salah maka sebaliknya yaitu false dengan size adalah ukuran linked list itu sendiri. Kode b memeriksa apakah head bernilai null jika benar fungsi tersebut mengembalikan nilai true dan linked list masih kosong.

LINK GITHUB:

https://github.com/RoyW12/AlgoritmaStrukturData_1G_28