

MATERI

: DOUBLE LINKED LIST

### Percobaan 1

```
poblic class DoubleLinkedList {
    ModeIs head;
    int size;

public boolean isEmpty() {
    head = null;
    size = 0;
}

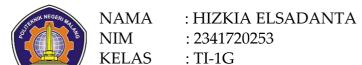
public void addFirst(int item) {
    if (isEmpty()) {
     head = new NodeIs(prev:null, item, next:null);
    } else {
    NodeIs newNode = new NodeIs(prev:null, item, head);
    head = newNode;
    head = newNode;
    head = newNode;
    }

public void addLast(int item) {
    if (isEmpty()) {
      head.prev = newNode;
    head.prev = newNode;
    head = newNode;
    }

public void addLast(int item) {
    if (isEmpty()) {
      addFirst(item);
    }
    else {
      NodeIs newNode = new NodeIs(prev:null, item, head);
    head.prev = newNode;
    }

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

NodeIs newNode = new NodeIs(current, item, next:null);
    current.next = newNode;
    size++;
    }
}
```



MATERI: DOUBLE LINKED LIST

### Pertanyaan

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

Jawab: Single linked list merupakan linked list yang hanya mempunyai satu variabel pointer yaitu pointer yang menunjuk node selanjutnya, sedangkan double linked list merupakan linked list yang mempunyai dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya.

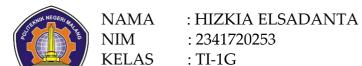
- 2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut? Jawab : Atribut next menunjuk ke node berikutnya dalam double linked list, memungkinkan traversal maju, dan atribut prev menunjuk ke node sebelumnya dalam double linked list, memungkinkan traversal mundur.
- 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: Fungsi dari menginisialisasi head dengan nilai null menandakan bahwa linked list yang baru dibuat masih kosong dan belum ada node di dalamnya, dan fungsi menginisialisasi size dengan nilai 0 menandakan bahwa jumlah elemen dalam linked list adalah nol. Oleh karena itu, inisialisasi atribut head dan size dalam konstruktor kelas DoubleLinkedList bertujuan untuk memastikan bahwa linked list baru dimulai dalam keadaan kosong dan siap untuk digunakan tanpa error, menjadikan linked list ini mudah dipelihara dan dioperasikan.

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 : Pada method addFirst() atribut prev diatur ke null dalam konstruktor class Node ketika menambahkan node baru di awal linked list karena node tersebut akan menjadi node pertama dalam list. Sebagai node pertama, ia tidak memiliki node sebelumnya, sehingga prev harus null. Ini menjaga



MATERI : DOUBLE LINKED LIST

konsistensi struktur double linked list, di mana node pertama selalu memiliki prev yang menunjuk ke null.

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

Jawab : Pada metode addFirst(), pernyataan head.prev = newNode digunakan untuk menetapkan referensi node sebelumnya dari node pertama dalam linked list ke node baru yang baru saja ditambahkan.

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 : Pada method addLast() pembuatan object Node dengan mengisikan parameter prev dengan

current, dan next dengan null digunakan untuk membuat node baru yang akan ditambahkan di akhir linked list dengan prev menunjuk ke node terakhir saat ini (current) dan next diatur ke null karena node

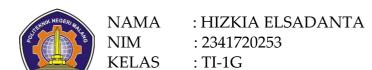
baru ini akan menjadi node terakhir dalam list.

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;
}</pre>
```

Jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

Jawab : Kode tersebut menambahkan node baru di awal double linked list. Jika current adalah node pertama (head) dari linked list, maka node baru (newNode) dibuat dan ditempatkan di depan current. Setelah itu, newNode diatur sebagai head baru dari linked list.



: DOUBLE LINKED LIST

### Percobaan 2

MATERI

```
void removeFirst() throws Exception {
              if (isEmpty()) {
                  throw new Exception(message: "LinkedList masih kosong, tidak dapat dihapus");
              } else if (size == 1) {
                 removeLast():
              } else {
                  head = head.next;
                  head.prev = null;
                 size--;
          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;
              Node15 current = head;
              while (current.next.next != null) {
109
                  current = current.next;
              current.next = null;
              size--;
```

```
void remove(int index) throws Exception {
   if (isEmpty() || index >= size) {
       throw new Exception(message:"Nilai indeks di luar batas");
   } else if (index == 0) {
       removeFirst();
   } else {
       Node15 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--;
```



: HIZKIA ELSADANTA

: 2341720253

: TI-1G

MATERI : DOUBLE LINKED LIST

```
40
                10
                        20
Berhasil diisi
Size: 4
        10
                20
Berhasil diisi
Size : 3
        10
Berhasil diisi
Size : 2
40
Berhasil diisi
Size: 1
PS D:\Kuliah\Semester 2\Praktek Algoritma Struktur Data\Algoritma_Struktur_Data_1G_15>
```

# Pertanyaan

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

head = head.next;

head.prev = null;

jawab : Pada method removeFirst() dalam kelas DoubleLinkedList, pernyataan head = head.next dan head.prev = null digunakan untuk menghapus node pertama dari linked list, dimana pernyataan head = head.next digunakan untuk menggeser head ke node kedua dalam linked list setelah node pertama dihapus, sedangkan pernyataan head.prev = null digunakan untuk memastikan bahwa prev dari node baru yang menjadi head diatur ke null untuk menjaga konsistensi struktur double linked list

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

Jawab: Untuk mendeteksi posisi data pada bagian akhir dalam metode removeLast(), kita menggunakan iterasi melalui linked list sampai kita mencapai node kedua terakhir. Setelah kita menemukan node kedua terakhir, kita mengatur next dari node tersebut menjadi null, sehingga node terakhir dihapus dari linked list.

Memeriksa apakah linked list kosong



NAMA : HIZKIA ELSADANTA

NIM : 2341720253

KELAS : TI-1G

MATERI : DOUBLE LINKED LIST

```
if (isEmpty()) {
    throw new Exception(message:"Linked List masih kosong, tidak dapat dihapus");
} else if (head newt == pull){
```

Memeriksa apakah head adalah node terakhir

```
} else if (head.next == null){
    head = null;
    size--;
    return;
}
```

Iterasi untuk menemukan node terakhir sebelum node terakhir

```
Node15 current = head;
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: Potongan kode program tersebut tidak cocok untuk perintah remove karena potongan kode tersebut tidak memperhatikan situasi dimana node yang akan dihapus adalah node terakhir dalam linked list sehingga dapat menyebabkan error atau kesalahan dalam manipulasi linked list.

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

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

Jawab : Potongan kode tersebut digunakan untuk menghapus node yang terletak di posisi index dari linked list. Perintah current.prev.next = current.next; menghubungkan node sebelumnya dengan node setelahnya, dan current.next.prev = current.prev; menghubungkan node setelahnya dengan node sebelumnya. Dengan demikian, node current dihapus dari linked list.



: HIZKIA ELSADANTA

: 2341720253

: TI-1G

MATERI : DOUBLE LINKED LIST

### Percobaan 3

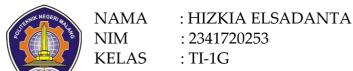
```
int getFirst() throws Exception {
                if (isEmpty()) {
                     throw new Exception(message:"Linked List kosong");
                return head.data:
             int getLast() throws Exception {
                if (isEmpty()) {
                     throw new Exception(message:"Linked List kosong");
                Node15 tmp = head;
                while (tmp.next != null) {
                     tmp = tmp.next;
                return tmp.data;
159
            int get(int index) throws Exception {
   if (isEmpty() || index >= size) {
      throw new Exception(message:"Nilai indeks di luar batas.");
                Node15 tmp = head;
                for (int i = 0; i < index; i++) {
                tmp = tmp.next;
} return tmp.data;
```

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

## Pertanyaan

1. Jelaskan method size() pada class DoubleLinkedLists!

Jawab : Method size() pada kelas DoubleLinkedList mengembalikan jumlah elemen dalam linked list, yang disimpan dalam variabel size. Ini berguna untuk mengetahui ukuran linked list pada saat tertentu.



MATERI : DOUBLE LINKED LIST

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

Jawab: Untuk memulai indeks pada double linked lists dari 1, kita dapat mengubah implementasi metode add(int item, int index) dengan menambahkan 1 pada nilai index yang diterima sebagai argumen. Dengan ini, indeks akan dimulai dari 1 saat diakses, meskipun secara internal dimulai dari 0.

3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists! Jawab: Pada double linked lists, saat menambahkan node baru, baik node sebelumnya maupun node berikutnya akan diperbarui, memungkinkan akses ke node-node dalam kedua arah. Sementara pada single linked lists, hanya node berikutnya yang diperbarui, membatasi kemampuan akses ke depan saja dalam linked list.

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

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

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

Jawab: Kode program (a) memeriksa apakah variabel size sama dengan 0 untuk menentukan apakah struktur data kosong, sedangkan kode program (b) memeriksa apakah variabel head adalah null. Kode program (b) lebih langsung dan jelas dalam menentukan kekosongan struktur data.

Link github:

https://github.com/Kiaakk/Algoritma\_Struktur\_Data\_1G\_15.git