Data Structure and Algorithm Practicum
Double Linked Lists

Name
Muhammad Baihaqi Aulia Asy'ari

NIM
2241720145

Class
1I

Department
Information Technology

Study Program
D4 Informatics Engineering

1.1 Learning Objective

After learning this lab activity, students will be able to:

1. Understand Double Linked List algorithm
2. Create and declare double linked list algorithm
3. Implement double linked list algorithm in various case studies

1.2 Lab Activities 1

In this lab activity, we will create Node class and DoubleLinkedList class that has operations to insert data in multiple way. (from the beginning or the tail of the list)

1.2.1 Steps

1. Take this class diagram as your reference for creating the **DoubleLinkedList class**

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. Create a new package named **DoubleLinkedList**
3. Create a new class in that package named **Node**

4. In that class, declare the attributes as described in the class diagram

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

5. Next, add the default constructor in Node class

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

6. Create a new class named **DoubleLinkedList** in the same package with the node as following image:

```
package LabActivities;

public class DoubleLinkedList {

}
```

7. Next, we add the attributes

```
Node head;
int size;
```

8. Then, add the constructor in class **DoubleLinkedList**

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

9. Create method isEmpty(), this method will be used to check whether the linked list is empty or not

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

10. Then add method **addFirst()**. This method will be executed when we want to add data in the beginning of the 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. Let's not forget about adding the data in the end of the list. We can do it after adding these lines of code in **addLast()** method

```
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. If we want to add a data that specified by a certain index, we will need to provide additional method to do so. It can be done by creating the **add()** method

```
public void add(int item, int index) throws Exception{
    if (isEmpty()) {
        addFirst(item);
    } else if (index < 0 || index > size) {
        throw new Exception("Index out of bound");
    } else {
        Node current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
```

```
            if (current.next == 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. We want to make our list has an easy access to retrieve the length of the list. That's why we create method **size()**

```
public int size() {
    return size;
}
```

14. We create a method **clear()** to remove all the data that are exist in linked lists

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

15. Next up, to print the whole data in the list, we need to create a method print().

```
public void print() {
    if (!isEmpty()) {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data + "\t");
            temp = temp.next;
        }
        System.out.println("\n successfully added");
    } else {
        System.out.println("Linked list is empty");
    }
}
```

16. After creating the blueprint classes, we will need one main class so that all of that can be included in the program. Create **DoubleLinkedListMain** class to do so

    ```java
    package LabActivities;

    public class DoubleLinkedListMain {
        public static void main(String[] args) throws Exception {

        }
    }
    ```

17. Instantiate an object from **DoubleLinkedList** class in the main method. Then apply these program code

    ```java
    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);
    ```

1.2.2 Result

Compile the program and see if the result matches with following image

```
PS D:\Kuliah\Smt 2\Algoritma dan Struktur Data\Praktikum\Week
 ↪  12\Double Linked Lists>  d:; cd 'd:\Kuliah\Smt 2\Algoritma dan
 ↪  Struktur Data\Praktikum\Week 12\Double Linked Lists'; &
 ↪  'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe'
 ↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Kuliah\Smt
 ↪  2\Algoritma dan Struktur Data\Praktikum\Week 12\Double Linked
 ↪  Lists\bin' 'LabActivities.DoubleLinkedListMain'
Linked list is empty
Size: 0
================================================
7       3       4
successfully added
Size: 3
================================================
7       40      3       4
successfully added
Size: 4
================================================
Linked list is empty
Size: 0
```

1.2.3 Questions

1. What's the difference between single linked list and double linked list?
2. In **Node class**, what is the usage of attribute next and prev ?
3. In constructor of **DoubleLinkedList** class. What's the purpose of head and size attribute in this following code?
4. In method **addFirst()**, why do we initialize the value of Node object to be null at first? Node newNode = new Node(null, item, head);
5. In method **addLast()**, what's the purpose of creating a node object by passing the **prev** parameter with **current** and **next** with **null** ? Node newNode = new Node(current, item, null);

1.3 Lab Activities 2

In this lab activity, we have added some methods from our 1st lab activity. Now, we added some ways for the users to remove a data in the beginning of the list, the tail, or with specified index. For more details, pay attention to this class diagram:

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**

1.3.1 Steps

1. Create method **removeFirst()** in class **DoubleLinkedList**

```java
public void removeFirst() throws Exception{
    if (isEmpty()) {
        throw new Exception("Linked list is still empty, cannot
         ↪  remove data");
    } else if (size == 1) {
        removeLast();
    } else {
        head = head.next;
        head.prev = null;
        size--;
    }
}
```

2. Create method **removeLast()** in class **DoubleLinkedList**

```java
public void removeLast() throws Exception{
    if (isEmpty()) {
        throw new Exception("Linked list is still empty, cannot
         ↪  remove data");
    } 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. Create method **remove()** in class **DoubleLinkedList**, alongside with its parameter

```
public void remove(int index) throws Exception{
    if (isEmpty() || index >= size) {
        throw new Exception("Index value is out of bound");
    } else if (size == 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. To execute additional codes we've just added, also make addition in the main class as well

```
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);
```

1.3.2 Result

Compile the program and see if the result matches with following image

```
PS D:\Kuliah\Smt 2\Algoritma dan Struktur Data\Praktikum\Week
↪  12\Double Linked Lists>  d:; cd 'd:\Kuliah\Smt 2\Algoritma dan
↪  Struktur Data\Praktikum\Week 12\Double Linked Lists'; &
↪  'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Kuliah\Smt
↪  2\Algoritma dan Struktur Data\Praktikum\Week 12\Double Linked
↪  Lists\bin' 'LabActivities.DoubleLinkedListMain'
50      40      10      20
successfully added
Size: 4
=================================================
40      10      20
successfully added
Size: 3
=================================================
40      10
successfully added
Size: 2
=================================================
40
successfully added
Size: 1
```

1.3.3 Questions

1. What's the meaning of these statements in **removeFirst()** method?
2. How do we detect the position of the data that are in the last index in method **removeLast()**?
3. Explain why this program code is not suitable if we include it in **remove** command!
4. Explain what's the function of this program code in method **remove**!

1.4 Lab Activities 3

In this 3rd lab activity, we will test if we can retrieve a data in linked list in various needs. The first is we can get a data in the beginning of the list, at the end of the list, or in specified index of the list. We will create 3 methods to realize the idea. For more details, feel free to check this class diagram

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**

1.4.1 Steps

1. Create a method **getFirst()** in class **DoubleLinkedList** to retrieve the first data in the list

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

2. Create a method **getLast()** in class **DoubleLinkedList** to retrieve the data in the list

```
public int getLast() throws Exception{
    if (isEmpty()) {
        throw new Exception("Linked list still empty");
    }
    Node temp = head;
    while (temp.next != null) {
        temp = temp.next;
    }
    return temp.data;
}
```

3. Create a method **get(int index)** in class **DoubleLinkedList** to retrieve the data in specified index of the list

```
public int get(int index) throws Exception{
    if (isEmpty()) {
        throw new Exception("Linked list still empty");
    }
    Node temp = head;
    for (int i = 0; i < index; i++) {
        temp = temp.next;
    }
    return temp.data;
}
```

4. In the main class, add the program code as follows and see the result

```
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 in the head of the linked list is : " +
     ↪  dll.getFirst());
    System.out.println("Data in the tail of the linked list is : " +
     ↪  dll.getLast());
    System.out.println("Data in the 1st index of the linked list is :
     ↪  " + dll.get(1));
```

1.4.2 Result

Compile the program and see if the result matches with following image

```
PS D:\Kuliah\Smt 2\Algoritma dan Struktur Data\Praktikum\Week
 ↪  12\Double Linked Lists>  d:; cd 'd:\Kuliah\Smt 2\Algoritma dan
 ↪  Struktur Data\Praktikum\Week 12\Double Linked Lists'; &
 ↪  'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe'
 ↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'D:\Kuliah\Smt
 ↪  2\Algoritma dan Struktur Data\Praktikum\Week 12\Double Linked
 ↪  Lists\bin' 'LabActivities.DoubleLinkedListMain'
Linked list is empty
Size: 0
================================================
7       3       4
successfully added
Size: 3
================================================
7       40      3       4
successfully added
Size: 4
================================================
Data in the head of the linked list is : 7
Data in the tail of the linked list is : 4
Data in the 1st index of the linked list is : 40
```

1.4.3 Questions

1. What is the function of method **size()** in **DoubleLinkedList** class ?

2. How do we set the index in double linked list so that it starts from 1st index instead of 0th index?

3. Please explain the difference between method **Add()** in double linked list and single linked list !

4. What's the logic difference of these 2 following codes?

1.5 Assignment

1. Create a program with double linked list implementation that allows user to choose a menu as following image! The searching uses sequential search approach and the program should be able to sort the data in descending order. You may any choose sorting approach you prefer (bubble sort, selection sort, insertion sort, or merge sort) **Adding a data Add data in specified index and display the result Search Data Sorting Data** 2. We are required to create a program which Implement Stack using double linked list. The features are described in following illustrations: **Initial menu and add Data (push) Print All Data See the data on top of the stack Pop the data from the top of the stack** 3. Create a program that helps vaccination process by having a queue algorithm alongside with double linked list as follows **(the amount left of queue length in menu print(3) and recent vaccinated person in menu Remove data (2) should be displayed) Initial menu and adding a data Print data (notice the highlighted red in the result) Remove Data (the highlighted red must displayed in the console too)** 4. Create a program implementation that list students score. Each student's data consist of their nim, name, and gpa. The program should implement double linked list and should be able to search based on NIM and sort the GPA in descending order. **Students class must be implemented in this program Initial menu and adding data Printing data Searching data Sorting data**