Week 11

Subject	Data Structure and Algorithm
Lecturer	Imam Fahrur Rozi ST. MT.
Type	Assignment
Semester	Semester 2
Time	@May 12, 2023
Files & Media	

Jobsheet 9

Activities #1

Code

```
package JB11.Prac;

public class Node
{
    int data;
    Node next;

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

```
package JB11.Prac;

public class SingleLinkedList
{
    Node head;
    Node tail;
```

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

void print()
{
    if(!isEmpty())
    {
        Node tmp = head;
        System.out.print("Linked list content: \t");
        while (tmp != null)
        {
            System.out.print(tmp.data + "\t");
            tmp = tmp.next;
        }
        System.out.println();
    }
    else System.out.println("Linked list is empty!");
}

void addFirst(int input)
{
    Node ndInput = new Node(input, null);
    if (isEmpty())
    {
        head = ndInput;
        tail = ndInput;
    }
    else
    {
        ndInput.next = head;
        head = ndInput;
    }
}

void addLast(int input)
{
    Node ndInput = new Node(input, null);
    if(isEmpty())
    {
        head = ndInput;
        tail = ndInput;
    }
    else
    {
        tail.next = ndInput;
        tail = ndInput;
    }
}

void insertAfter(int key, int input)
{
        Node ndInput = new Node (input, null);
        Node temp = head;
        do
        {
            if (temp.data == key)
            {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if(ndInput.next == null) tail = ndInput;
                break;
            }
            temp = temp.next;
        }
        while (temp != null);
    }

    void insertAt(int index, int input)
    {
        if(index < 0) System.out.println("Wrong index");
        else if (index == 0) addFirst(input);
        else
        {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) temp = temp.next;
            temp.next = new Node(input, temp.next);
            if(temp.next.next == null) tail = temp.next;
        }
    }
}
```

```
package JB11.Prac;

public class SLLMain
{
    public static void main(String[] args)
    {
        SingleLinkedList singLL = new SingleLinkedList();
        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.insertAfter(700, 999);
        singLL.print();
        singLL.insertAt(3, 833);
        singLL.print();
    }
}
```

Questions #1

1. because we haven't inputted any list, and we tried to print the list
2. `ndInput.next` means that we choose the next node of the last list, then we place it to the `temp.next`, after that we change the `temp.next` to the one that we have inputted
3. it used to change the tail if the 3rd list is `null` to the one that we saved on `temp.next`

Activities #2

Code

```
package JB11.Prac;

public class SingleLinkedList
{
    Node head;
    Node tail;

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

    void print()
    {
        if(!isEmpty())
        {
            Node tmp = head;
            System.out.print("Linked list content: \t");
            while (tmp != null)
            {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println();
        }
        else System.out.println("Linked list is empty!");
    }

    void addFirst(int input)
    {
        Node ndInput = new Node(input, null);
        if (isEmpty())
        {
            head = ndInput;
            tail = ndInput;
        }
        else
        {
            ndInput.next = head;
            head = ndInput;
        }
    }

    void addLast(int input)
    {
        Node ndInput = new Node(input, null);
        if(isEmpty())
        {
            head = ndInput;
            tail = ndInput;
        }
        else
        {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    void insertAfter(int key, int input)
    {
        Node ndInput = new Node (input, null);
        Node temp = head;
        do
        {
            if (temp.data == key)
            {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if(ndInput.next == null) tail = ndInput;
                break;
            }
            temp = temp.next;
        }
        while (temp != null);
    }

    void insertAt(int index, int input)
    {
        if(index < 0) System.out.println("Wrong index");
        else if (index == 0) addFirst(input);
        else
        {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) temp = temp.next;
            temp.next = new Node(input, temp.next);
            if(temp.next.next == null) tail = temp.next;
        }
    }
```

```
//here's where activities 2 started
int getData(int index)
{
    Node tmp = head;
    for (int i = 0; i < index; i++) tmp = tmp.next;
    return tmp.data;
}

int indexOf(int key)
{
    Node tmp = head;
    int index = 0;
    while (tmp != null && tmp.data != key)
    {
        tmp = tmp.next;
        index++;
    }

    if (tmp == null) return -1;
    else return index;
}

void removeFirst()
{
    if (isEmpty()) System.out.println("Linked list is empty, cannot remove a data");
    else if (head == tail) head = tail = null;
    else head = head.next;
}

void removeLast()
{
    if(isEmpty()) System.out.println("Linked list is empty, cannot remove a data");
    else if(head == tail) head = tail = null;
    else
    {
        Node temp = head;
        while (temp.next != tail) temp = temp.next;
        temp.next = null;
        tail = temp;
    }
}

void remove(int key)
{
    if(isEmpty()) System.out.println("Linked list is empty, cannot remove a data");
    else
    {
        Node temp = head;
        while (temp != null)
        {
            if((temp.data == key) && (temp == head))
            {
                this.removeFirst();
```

```
                    break;
                }
                else if (temp.next.data == key)
                {
                    temp.next = temp.next.next;
                    if(temp.next == null) tail = temp;
                    break;
                }
                temp = temp.next;
            }
        }
    }

    void removeAt(int index)
    {
        if(index == 0) removeFirst();
        else
        {
            Node temp = head;
            for (int i = 0; i < index - 1; i++) temp = temp.next;
            temp.next = temp.next.next;
            if (temp.next == null) tail = temp;
        }
    }
}
```

```
package JB11.Prac;

public class SLLMain
{
    public static void main(String[] args)
    {
        SingleLinkedList singLL = new SingleLinkedList();
        singLL.print();
        singLL.addFirst(890);
        singLL.print();
        singLL.addLast(760);
        singLL.print();
        singLL.addFirst(700);
        singLL.print();
        singLL.insertAfter(700, 999);
        singLL.print();
        singLL.insertAt(3, 833);
        singLL.print();

        //activities number 2 start from here
        System.out.println("=========================");

        System.out.println("Data in 1st index : " + singLL.getData(1));
        System.out.println("Data 3 is in index : " + singLL.indexOf(760));
        System.out.println("Index of 890: " + singLL.indexOf(890));
        singLL.remove(999);
        singLL.print();
        singLL.removeAt(0);
        singLL.print();
        singLL.removeFirst();
        singLL.print();
        singLL.removeLast();
        singLL.print();
    }
}
```

Questions #2

1. because if we have the same data after the temporary data that we wanted to use, it will also remove the same data
2. because, if we want to remove a `data` from a linked list, we have to skip the `data` that's same with the `key` that we inputted
3. the output is where the index of an inputted number, if there isn't any number within the inputted number, the output will be -1

Assignment

1. `inserBefore()`

```
public void insertBefore(int key, int input) {
        if (isEmpty()) {
            head = tail = new Node(input, null);
            return;
        }
        if (head == tail) {
            head = new Node(input, head);
        }
    }
```

2. based on image

```
package JB11.Asg;

public class ThisIsASSignment<TData> {
    Node<TData> head;
    Node<TData> tail;

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

    public void print() {
        if (!isEmpty()) {
            Node<TData> tmp = head;
            System.out.println("Linked List Content: \t");
            while (tmp != null) {
                System.out.print(tmp.value + "\t");
                tmp = tmp.next;
            }
            System.out.println();
        } else System.out.println("Linked list is empty");
    }

    public void addFirst(TData input) {
        Node<TData> ndInput = new Node<TData>(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    public void addLast(TData input) {
        Node<TData> ndInput = new Node<TData>(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    public TData getData(int index) {
        Node<TData> tmp = head;
        for (int i = 0; i < index; i++) tmp = tmp.next;
        return tmp.value;
    }

    public void removeFirst() {
        if (isEmpty()) System.out.println("Linked list is empty");
        else if (head == tail) head = tail = null;
        else head = head.next;
    }

    public void removeLast() {
        if (isEmpty()) System.out.println("Linked list is empty");
        else if (head == tail) head = tail = null;
        else {
            Node<TData> temp = head;
            while (temp.next == null) temp = temp.next;
            temp.next = null;
            tail = temp;
        }
    }

    public void insertBefore(int key, int input) {
        if (isEmpty()) {
            head = tail = new Node(input, null);
            return;
        }
        if (head == tail) {
            head = new Node(input, head);
        }
    }
}
```

3. stack

```
package JB11.AsgNo3;

public class StackLinkedList
{
    int size, top;
    Node head, tail;

    StackLinkedList(int size)
    {
        this.size = size;
        top = -1;
    }

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


    void push(String input)
    {
        Node ndInput = new Node(input, null);
        if (isEmpty())
        {
            top++;
            head = ndInput;
            tail = ndInput;
        }
        else
        {
            ndInput.next = head;
            head = ndInput;
        }
    }

    void print()
    {
        System.out.println("Stack content: ");
        if(!isEmpty())
        {
            Node tmp = head;
            while (tmp != null)
            {
                System.out.println(tmp.data);
                tmp = tmp.next;
            }
            System.out.println();
        }
        else System.out.println("Linked list is empty!");
    }
}
```

```
package JB11.AsgNo3;

public class Main
{
    public static void main(String[] args)
    {
        StackLinkedList stk = new StackLinkedList(8);
        stk.push("Bahasa");
        stk.push("Android");
        stk.push("Komputer");
        stk.push("Basis Data");
        stk.push("Matematika");
        stk.push("Algoritma");
        stk.push("Statistika");
        stk.push("Multimedia");
        stk.print();
    }
}
```

4. this also placed on number 5
5. queue

```
package JB11.AsgNo4and5;
```

```
public class Customer
{
    String name, address, customerAccountNumber;

    Customer (String name, String address, String customerAccountNumber)
    {
        this.name = name;
        this.address = address;
        this.customerAccountNumber = customerAccountNumber;
    }

    public void print(){
        System.out.println("Name : " + name);
        System.out.println("Address : " + address);
        System.out.println("Account Number : " + customerAccountNumber);
        System.out.println("=======================");
    }
}
```

```
package JB11.AsgNo4and5;

public class LinkedList
{
    Node head, tail;

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

    void print()
    {
        if(!isEmpty())
        {
            Node tmp = head;
            System.out.print("Linked list content: \t");
            while (tmp != null)
            {
                System.out.print(tmp.data + "\t");
                tmp = tmp.next;
            }
            System.out.println();
        }
        else System.out.println("Linked list is empty!");
    }

    void addLast(Customer input)
    {
        Node ndInput = new Node(input, null);
        if(isEmpty())
        {
            head = ndInput;
            tail = ndInput;
        }
        else
        {
            tail.next = ndInput;
            tail = ndInput;
        }
    }

    void removeFirst()
    {
        if (isEmpty()) System.out.println("Linked list is empty, cannot remove a data");
        else if (head == tail) head = tail = null;
        else head = head.next;
    }

    Customer getData(int index)
    {
        Node tmp = head;
        for (int i = 0; i < index; i++) tmp = tmp.next;
        return tmp.data;
    }
}
```

```
package JB11.AsgNo4and5;

public class Node
{
    Customer data;
    Node next;
    int can;

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

```
package JB11.AsgNo4and5;

public class Queue
{
    private LinkedList LL = new LinkedList();
    void enqueue(Customer data)
    {
        LL.addLast(data);
    }
    Customer dequeue()
    {
        Customer data = LL.getData(0);
        LL.removeFirst();
        return data;
    }

    void print()
    {
        Node temp = LL.head;
        while (temp != null)
        {
            temp.data.print();
            temp = temp.next;
        }
    }
}
```

```
package JB11.AsgNo4and5;
import java.util.Scanner;

public class Main
{
    static Scanner sc = new Scanner(System.in);
    public static Customer newCustomer()
    {
        System.out.print("Name: ");
        String nm = sc.next();
        sc.nextLine();
        System.out.print("Address: ");
        String addr = sc.next();
        sc.nextLine();
        System.out.print("Customer Account Number: ");
        String can = sc.next();
        sc.nextLine();

        return new Customer(nm, addr, can);
    }
    public static void menu()
    {
        System.out.println("Choose menu: ");
        System.out.println("1. Queue");
        System.out.println("2. Dequeue");
        System.out.println("3. Check all queue");
        System.out.println("===========================");
    }

    public static void main(String[] args)
    {
        Queue q = new Queue();
        int choose;
        do
        {
            menu();
            choose = sc.nextInt();
            switch (choose)
            {
                case 1:
                    q.enqueue(newCustomer());
                    break;
                case 2:
                    q.dequeue();
                    break;
                case 3:
                    q.print();
                    break;
            }
        }
        while(choose <= 3 && choose >= 1);
    }
}
```