Panji Iman Baskoro
171111023 / TI
Praktikum Pemprograman Dasar 2

LinkedListNode.java

```java
public class LinkedListNode {
    LinkedListNode next;
    LinkedListNode prev;
    int data;

    /* Constructor
    * set this.data into new_data
    * set this.prev into null
    * set this.next into null
    */
    LinkedListNode(int new_data) {
        this.data = new_data;
        this.prev = null;
        this.next = null;
    }

    /* set this.prev into other
    * if other is not null, set other.next into this
    */
    void set_prev(LinkedListNode other) {
        this.prev = other;
        if (other != null) {
            other.next = this;
        }
    }

    /* set this.next into other
    * if other is not null, set other.prev into this
    */
    void set_next(LinkedListNode other) {
        this.next = other;
        if (other != null) {
            other.prev = this;
        }
    }
}
```

LinkedList.java (quick sort)

```java
import java.util.Scanner;
public class LinkedList {

    LinkedListNode head;
    LinkedListNode tail;

    LinkedList() {
        this.head = null;
        this.tail = null;
    }


    /* First set a Node named current into head
    * while current is not null, print current.data, set current = cur
    * print end of line
     */
    void print() {
        LinkedListNode current = this.head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
        System.out.println("");
    }

    /* if LinkedList is empty, set new_node as head and tail
    * if LinkedList is not empty, set tail.next into new_node, set
      new_node.prev into tail, and make new_node a new tail
     */
    void push(LinkedListNode new_node) {
        if (this.head == null) {
            this.head = new_node;
            this.tail = new_node;
        } else {
            if (find_node_by_data(new_node.data) == null) {
                this.tail.set_next(new_node);
                this.tail = new_node;
            }

        }
    }

    /* if linked list is empty, set new_node as head and tail
    * if new_node < head, make it a new head
    * if new_node > tail, make it a new tail
    * otherwise traverse to the current position, and put new_node the
     */
    void insert(LinkedListNode new_node) {
        if (this.head == null) {
            this.head = new_node;
            this.tail = new_node;
        } else if (new_node.data <= this.head.data) {
            this.head.set_prev(new_node);
            this.head = new_node;

        } else if (new_node.data >= this.tail.data) {
            this.tail.set_next(new_node);
            this.tail = new_node;

        } else {
            LinkedListNode position = head;
            while (position.data < new_node.data) {
                position = position.next;

            }
            LinkedListNode previous_position = position.prev;
            new_node.set_prev(previous_position);
            new_node.set_next(position);

        }
    }

```

LinkedList.java (Quick sort)

```
    LinkedListNode find_node_by_data(int data) {
        LinkedListNode current = this.head;
        while (current != null) {
            if (current.data == data) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
    LinkedListNode lastNode(LinkedListNode node)
    {
        while (node.next != null ) {
            node = node.next;
        }
//        System.out.println("last : " + node.data);
        return node;
    }

    void delete(LinkedListNode deleted) {
        if (deleted != null && this.head != null) {
            if (this.head == this.tail && deleted == this.head) {
                this.head = null;
                this.tail = null;
            } else if (deleted == this.head) {
                LinkedListNode new_head = this.head.next;
                this.head.set_next(null);
                new_head.set_prev(null);
                this.head = new_head;
            } else if (deleted == this.tail) {
                LinkedListNode new_tail = this.tail.prev;
                this.tail.set_prev(null);
                new_tail.set_next(null);
                this.tail = new_tail;
            } else {
                LinkedListNode deleted_prev = deleted.prev;
                LinkedListNode deleted_next = deleted.next;
                deleted.set_prev(null);
                deleted.set_next(null);
                deleted_prev.set_next(deleted_next);
            }
        }

    }
    //mengecek nilai pertama dan terakhir
    public void quickSort(LinkedListNode node)
    {
        LinkedListNode last = lastNode(node);

        _quickSort(node, last);
    }

    void _quickSort(LinkedListNode l, LinkedListNode h)
    {
        if(h != null && l != h && l != h.next)
        {
            LinkedListNode temp = partition(l, h);
            _quickSort(l, temp.prev);
            _quickSort(temp.next, h);
        }
    }

    LinkedListNode partition(LinkedListNode l, LinkedListNode h)
    {
        int x = h.data;
        LinkedListNode i = l.prev;

        for (LinkedListNode j=l; j != h; j=j.next) {
            if (j.data <= x) {
                i = (i == null) ? l : i.next;
                int temp = i.data;
                i.data = j.data;
                j.data = temp;
            }
        }
        i = (i==null) ? l : i.next;
        int temp = i.data;
        i.data = h.data;
        h.data = temp;
        return i;
    }

```

LinkedList.java (Quick Sort)

```
155.
156.     public static void main(String[] args) {
157.        Scanner input = new Scanner(System.in);
158.         LinkedList a = new LinkedList();
159.
160.         System.out.print("berapa data yang mau diinput?  : ");
161.         int kolo = input.nextInt();
162.         input.nextLine();
163.
164.         for(int i=1; i<=kolo; i++)
165.         {
166.             System.out.print("Data ke "+i + " : ");
167.             int dato = input.nextInt();
168.             a.push(new LinkedListNode(dato));
169.         }
170.         a.print();
171.         a.quickSort(a.head);
172.         a.print();
173.     }
174. }
175.
```

LinkedList.java output

```
budosen@budosen-pc:/mnt/b2c7efbf-ef52-437d-8ca7-e46ea581cbba
sar 2/Pertemuan 3/tugas$ java LinkedList
berapa data yang mau diinput?  : 3
Data ke 1 : 9
Data ke 2 : 2
Data ke 3 : 4
9 2 4
2 4 9
budosen@budosen-pc:/mnt/b2c7efbf-ef52-437d-8ca7-e46ea581cbba
sar 2/Pertemuan 3/tugas$
```

bublesort.java

```
01. import java.util.Scanner;
02. class angka {
03.     public angka next;
04.     public int node;
05. }
06.
07. public class bubblesort {
08.
09.     static angka head;
10.     static int size = 0;
11.
12.     public static void print() {
13.         angka current = head;
14.         while (current != null) {
15.             System.out.print(current.node + " ");
16.             current = current.next;
17.         }
18.         System.out.println("");
19.     }
20.
21.     public static void insert(int new_node) {
22.         angka nilai = new angka();
23.         nilai.node = new_node;
24.         if (head != null) {
25.             angka datax = head;
26.             while (datax.next != null) {
27.                 datax = datax.next;
28.             }
29.             datax.next = nilai;
30.         } else {
31.             head = nilai;
32.         }
33.         size++;
34.     }
```

bublesort.java

```
    public static void bubsort() {
        if (size > 1) {
            boolean berubah;
            do {
                angka current = head;
                angka previous = null;
                angka next = head.next;
                berubah = false;
                while (next != null) {
                    if (current.node > next.node) {
                        berubah = true;
                        if (previous != null) {
                            angka sig = next.next;
                            previous.next = next;
                            next.next = current;
                            current.next = sig;
                        } else {
                            angka anon = next.next;
                            head = next;
                            next.next = current;
                            current.next = anon;
                        }
                        previous = next;
                        next = current.next;
                    } else {
                        previous = current;
                        current = next;
                        next = next.next;
                    }
                }
            } while (berubah);
        }
    }

    public static void main(String[] args) {
        Scanner masuk = new Scanner(System.in);
        System.out.print("Berapa data yang diinput? : ");
        int tero = masuk.nextInt();
        for (int i = 1; i <= tero; i++) {
            System.out.print("data ke "+i + " : ");
            int isidata = masuk.nextInt();
            insert(isidata);
        }
        System.out.println("Bublesort");
        print();
        bubsort();
        print();
    }
}
```

bublesort.java output

```
bubblesort.class  LinkedList.class  LinkedListNo
budosen@budosen-pc:/mnt/b2c7efbf-ef52-437d-8ca7-
sar 2/Pertemuan 3/tugas$ java bubblesort
Berapa data yang diinput? : 4
data ke 1 : 10
data ke 2 : 7
data ke 3 : 1
data ke 4 : 5
Bublesort
10 7 1 5
1 5 7 10
budosen@budosen-pc:/mnt/b2c7efbf-ef52-437d-8ca7-
sar 2/Pertemuan 3/tugas$
```