UTP

UNIVERSITI TEKNOLOGI PETRONAS

LAB

JAN 2023

TEB1113

Algorithm & Data Structure

Lab 4

NO.	NAME	STUDENT ID	PROGRAM (IT / IS / CS / BM)
1.	**CHENG PIN-JIE**	**21000548**	**CS**

Assoc. Prof. Dr Manzoor Ahmed Hashmani

Madam Maryam Omar Abdullah Sawad

Homework 1

Methods to remove last node:

```
/* First, assign a pointer 'temp' at the head position. Then check
* whether the linked list is empty. After that, use for loop to
* reach the very end of the linked list based on the length of the
list.
* When reach, assign the pointer as the temp data and make the temp
tail become null.
*/
public void removeLastNode(){
            Node temp = head;
            if(head == null){
                System.out.println("The linked list is empty. ");
            }else{
                for (int i = 1; i < length(); i++){
                temp = temp.next;
                }
                tail = temp;
                temp.next = null;
                System.out.println("Last node has been removed. ");
            }

        }
```

Full Code:

```
import java.util.*;
class LinkedListOfInt
{
      class Node
      {
            int data;
            Node next;

            public Node(int initialData)
```

```
    {
            data= initialData;
            next=null;
    }
}

public Node head = null;
public Node tail = null;

public void addNodeToEnd(int newEntry)
{
    Node newNode = new Node(newEntry);
    if(head==null)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        tail.next = newNode;
        tail = newNode;
    }
}

public void addNodeToFront(int newEntry)
{
    Node newNode = new Node(newEntry);
    if(head==null)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        newNode.next = head;
```

```
                head = newNode;
        }
}

public Node findNode(int intNode)
{
        Node temp = head;
        while(temp.data != intNode && temp.next!=null)
        {
                temp = temp.next;
        }
        if(temp.data==intNode)
                return(temp);
        else
        {
                temp = head;
                return (temp);
        }
}

public void addNodeAfter(int newEntry, int intNode)
{
        Node newNode = new Node(newEntry);

        if(head==null)
        {
                head = newNode;
                tail = newNode;
        }
        else
        {
                Node temp = findNode (intNode);
                if (temp.data ==intNode)
                {
                        newNode.next = temp.next;
```

```
                          temp.next=newNode;
                    }
                    else
                          System.out.println("Can not add after " + intNode
+ " because it is not in the list");
             }
       }

       public void printLL()
       {
             Node current = head;
             if(head==null)
             {
                    System.out.println("List is empty");
                    return;
             }
             System.out.println("Nodes of singly linked list: ");
             while(current !=null)
             {
                    System.out.print(current.data + " ");
                    current = current.next;
             }
             System.out.println();
       }

        public int length(){
            int counter = 0;
            Node temp = head;
            if (head == null){
                return 0;
            }
            else if(head!=null){
                while(temp.next != null){
                    temp = temp.next;
                    counter++;
                }
            }
            return counter;
        }

      public void removeFirstNode(){
            if (head != null){
                head = head.next;
            }else{
                head = null;
                System.out.println("The linked list is empty." );
            }
        }

      /* First, assign a pointer 'temp' at the head position. Then check
         * whether the linked list is empty. After that, use for loop to
         * reach the very end of the linked list based on the length of the
list.
         * When reach, assign the pointer as the temp data and make the
temp tail
         * become null.
         */
        public void removeLastNode(){
            Node temp = head;
            if(head == null){
                System.out.println("The linked list is empty. ");
            }else{
                for (int i = 1; i < length(); i++){
                temp = temp.next;
                }
                tail = temp;
                temp.next = null;
                System.out.println("Last node has been removed. ");
            }

        }
        public void addNodeAt(int newEntry, int intLoc)
       {
             Node newNode = new Node(newEntry);

             if(head==null)
             {
                   head = newNode;
                   tail = newNode;
             }
             else
                {
                    if (intLoc <= length()){
                        Node temp = head;
                    for (int i = 1; i < intLoc-1; i++){
                            temp = temp.next;
                        }
                    newNode.next = temp.next;
                    temp.next=newNode;
                    }
                    else{
                        System.out.println("The location is out of
bound.");
                    }
             }
       }

       public static void main(String args[])
       {
             LinkedListOfInt ls= new LinkedListOfInt();
             Scanner sc= new Scanner(System.in);
             int ch;
             while (true)
             {
                   System.out.println("Choose one option from the
following list: ");
```

```
                    System.out.println("1: Add node to the front. ");

                    System.out.println("2: Add node to the end. ");

                    System.out.println("3: Add node after specific element.
");

                    System.out.println("4: Print out the elements of the
linked list. ");

            System.out.println("5: Add node at specific location. ");

            System.out.println("6: Remove the first node from linked list.
");

            System.out.println("7: Remove the last node from the linked
list. ");

                    System.out.println("8: Exit. ");

                    System.out.print("Enter your option: ");

                    ch= sc.nextInt();

                    if(ch == 1)

                    {

                          int input;

                          System.out.println("Enter the number that you
want to add to the linked list: ");

                          input=sc.nextInt();

                          ls.addNodeToFront(input);

                    }

                    else if(ch == 2)

                    {

                          int input;

                          System.out.println("Enter the number that you
want to add to the linked list: ");

                          input=sc.nextInt();

                          ls.addNodeToEnd(input);

                    }

                    else if(ch== 3)

                    {

                          int input, afterValue;

                          System.out.println("Enter the number that you
want to add to the linked list: ");

                          input=sc.nextInt();

                          System.out.println("Enter the number that you
want to add after it: ");
```

```
                          afterValue=sc.nextInt();

                          ls.addNodeAfter(input, afterValue);

                    }

                    else if(ch== 4)

                    {

                          ls.printLL();

                    }

                        else if(ch == 5){

                            int input, location;

                            System.out.println("Enter the number that you
want to add to the linked list: ");

                            input = sc.nextInt();

                            System.out.println("Enter the location that you
want to add the number: ");

                            location = sc.nextInt();

                            ls.addNodeAt(input, location);

                        }

                    else if(ch == 6)

                    {

                          ls.removeFirstNode();

                                System.out.println("First node has been
removed. ");

                    }

                        else if (ch == 7){

                            ls.removeLastNode();

                        }

                        else if (ch == 8){

                            break;

                        }

                    else

                    {

                          System.out.println("Invalid Input.");

                    }

             }

       }

}
```

Homework 2

Methods to remove node with specific information:

```
/* First, we check whether the linked list is empty, then we find
the location of the node that users want. When the ‘temp' pointer
* reach the location and the program assigns the pointer 'next' to
* null which means the last node data become null now.
*/
public void removeNodeAfter(int intNode)
      {
            if(head==null)
            {
                  System.out.println("The linked list is empty");
            }
            else
            {
                  Node temp = findNode (intNode);
                  if (temp.data ==intNode)
                  {
                        temp.next = null;
                  }
                  else
                        System.out.println("Cannot remove after " +
intNode + " because it is not in the list");
            }
      }
```

Full Code:

```
import java.util.*;
class LinkedListOfInt
{
      class Node
      {
            int data;
            Node next;
```

```
    public Node(int initialData)
    {
            data= initialData;
            next=null;
    }
}

public Node head = null;
public Node tail = null;

public void addNodeToEnd(int newEntry)
{
    Node newNode = new Node(newEntry);
    if(head==null)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        tail.next = newNode;
        tail = newNode;
    }
}

public void addNodeToFront(int newEntry)
{
    Node newNode = new Node(newEntry);
    if(head==null)
    {
        head = newNode;
        tail = newNode;
    }
    else
```

```
        {
                newNode.next = head;
                head = newNode;
        }
}

public Node findNode(int intNode)
{
        Node temp = head;
        while(temp.data != intNode && temp.next!=null)
        {
                temp = temp.next;
        }
        if(temp.data==intNode)
                return(temp);
        else
        {
                temp = head;
                return (temp);
        }
}

public void addNodeAfter(int newEntry, int intNode)
{
        Node newNode = new Node(newEntry);

        if(head==null)
        {
                head = newNode;
                tail = newNode;
        }
        else
        {
                Node temp = findNode (intNode);
                if (temp.data ==intNode)
```

```
                    {
                          newNode.next = temp.next;
                          temp.next=newNode;
                    }
                    else
                          System.out.println("Can not add after " + intNode
+ " because it is not in the list");
             }
      }

      public void printLL()
      {
             Node current = head;
             if(head==null)
             {
                    System.out.println("List is empty");
                    return;
             }
             System.out.println("Nodes of singly linked list: ");
             while(current !=null)
             {
                    System.out.print(current.data + " ");
                    current = current.next;
             }
             System.out.println();
      }

        public int length(){
            int counter = 0;
            Node temp = head;
            if (head == null){
                return 0;
            }
            else if(head!=null){
                while(temp.next != null){
```

```
              temp = temp.next;
              counter++;
          }
      }
      return counter;
  }

public void removeFirstNode(){
      if (head != null){
          head = head.next;
      }else{
          head = null;
          System.out.println("The linked list is empty." );
      }
  }

  public void removeLastNode(){
      Node temp = head;
      if(head == null){
          System.out.println("The linked list is empty. ");
      }else{
          for (int i = 1; i < length(); i++){
          temp = temp.next;
          }
          tail = temp;
          temp.next = null;
          System.out.println("Last node has been removed. ");
      }

  }

  public void addNodeAt(int newEntry, int intLoc)
{
       Node newNode = new Node(newEntry);
```

```
             if(head==null)
             {
                    head = newNode;
                    tail = newNode;
             }
             else
                {
                    if (intLoc <= length()){
                        Node temp = head;
                    for (int i = 1; i < intLoc-1; i++){
                            temp = temp.next;
                        }
                    newNode.next = temp.next;
                    temp.next=newNode;
                    }
                    else{
                        System.out.println("The location is out of
bound.");
                    }
             }
      }


      /* First, we check whether the linked list is empty, then we find
the
         * the location of the node that users want. When the ‘temp'
pointer
         * reach the location and the program assigns the pointer 'next' to
         * null which means the last node data become null now.
         */
        public void removeNodeAfter(int intNode)
      {
             if(head==null)
             {
                    System.out.println("The linked list is empty");
             }
             else
             {
                   Node temp = findNode (intNode);
                   if (temp.data ==intNode)
                   {
                         temp.next = null;
                   }
                   else
                         System.out.println("Can not add after " + intNode
+ " because it is not in the list");
             }
       }


       public static void main(String args[])
       {
             LinkedListOfInt ls= new LinkedListOfInt();
             Scanner sc= new Scanner(System.in);
             int ch;
             while (true)
             {
                   System.out.println("Choose one option from the
following list: ");
                   System.out.println("1: Add node to the front. ");
                   System.out.println("2: Add node to the end. ");
                   System.out.println("3: Add node after specific element.
");
                   System.out.println("4: Print out the elements of the
linked list. ");
                       System.out.println("5: Add node at specific
location. ");
                       System.out.println("6: Remove the first node from
linked list. ");
                       System.out.println("7: Remove the last node from
the linked list. ");
                       System.out.println("8: Remove node after specific
element.");
                   System.out.println("9: Exit. ");
                   System.out.print("Enter your option: ");
                   ch= sc.nextInt();
```

```
                    if(ch == 1)

                    {

                           int input;

                           System.out.println("Enter the number that you
want to add to the linked list: ");

                           input=sc.nextInt();

                           ls.addNodeToFront(input);

                    }

                    else if(ch == 2)

                    {

                           int input;

                           System.out.println("Enter the number that you
want to add to the linked list: ");

                           input=sc.nextInt();

                           ls.addNodeToEnd(input);

                    }

                    else if(ch== 3)

                    {

                           int input, afterValue;

                           System.out.println("Enter the number that you
want to add to the linked list: ");

                           input=sc.nextInt();

                           System.out.println("Enter the number that you
want to add after it: ");

                           afterValue=sc.nextInt();

                           ls.addNodeAfter(input, afterValue);

                    }

                    else if(ch== 4)

                    {

                           ls.printLL();

                    }

                        else if(ch == 5){

                            int input, location;

                            System.out.println("Enter the number that you
want to add to the linked list: ");

                            input = sc.nextInt();
                            System.out.println("Enter the location that you want to add the number: ");
                            location = sc.nextInt();
                            ls.addNodeAt(input, location);
                        }
                    else if(ch == 6)
                    {
                           ls.removeFirstNode();
                                 System.out.println("First node has been removed. ");
                    }
                        else if (ch == 7){
                            ls.removeLastNode();
                        }
                        else if (ch == 8){
                            int num;
                            System.out.println("Enter the number that you want to remove the node after this number: ");
                            num = sc.nextInt();
                            ls.removeNodeAfter(num);
                        }
                        else if (ch == 9){
                            break;
                        }
                    else
                    {
                           System.out.println("Invalid Input.");
                    }
              }
       }
}
```

Full Code Link:
https://onlinegdb.com/9afmVr9R9