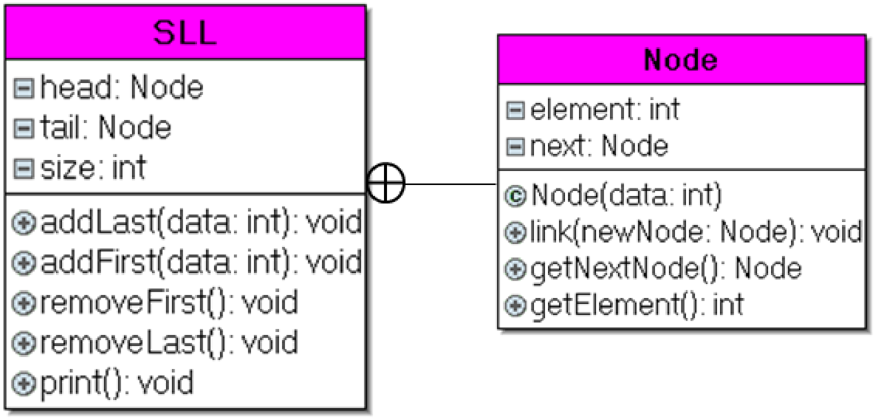
Data Structures and Algorithms Laboratory		
Laboratory 3: Linked Lists	**School of Information Technology**	
Name:Sai Hae Naing Lay	**ID:6531501208**	**Section:4**
Date: 10 Feb 2023	**Due date:on the LMS**	

Objective

- To create, access and modify linked lists
- To implement singly linked lists

Exercise 1:(In-class)From the given class diagram, create a singly linked list and complete the program to get the results as shown. Assume that elements in all nodes are integer.



Expected result

```
1->2->3->null
2->3->null
2->null
```

Complete the codes below.

```
package lab3;
class SLL {
      // ----------------------- Node --------------------
      private class Node {
            private int element;
            private Node next;

            // constructor
            public Node(int data) {
                  element = data;
                  next = null;
            }

            // link a new node to this node
```

```
          public void link(Node newNode) {
                next = newNode;
          }

          // return next node
          public Node getNextNode() {
                return next;
          }

          // return element of this node
          public int getElement() {
                return element;
          }
    }
    // --------------------- End Node --------------------

    // SLL properties and methods
    private Node head = null;
    private Node tail = null;
    private int size = 0; // SLL's size

    public void addLast(int data) {
          // create new node
          Node newNode = new Node(data);
          if (size == 0) {
                head = newNode;
          } else {
                tail.link(newNode);
          }
          tail = newNode;
          size++;
    }

    public void addFirst(int data) {

          Node newNode = new Node(data);
           if(size == 0){
           tail = newNode;
           }else{
           newNode.link(head);
           }
           head = newNode;
           size++;
    }

    public void removeFirst() {

          if(size == 1){
                 head = null;
                 tail = null;
```

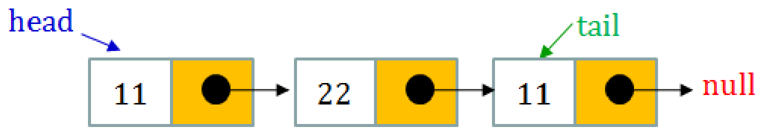
```
                   size--;
                  }
            else{
                   head = head.getNextNode();
                   size--;
                   }
      }

      public void removeLast() {

            if(size == 1){
                   head = null;
                   tail = null;
                   size--;
                   }
            else{
                   Node p = head;
                   while(p.getNextNode() != tail){
                   p = p.getNextNode();
                   }
                   tail = p;
                   tail.link(null);
                   size--;
                   }

      }

      public void print() {
            if (size == 0) {
                  System.out.println("Empty linked list");
            } else {
                  for (Node p = head; p != null; p = p.getNextNode()) {
                        System.out.print(p.getElement() + "->");
                  }
                  System.out.println("null");
            }
      }
}

//======================== Main Class ========================
public classMainSLL {
      public static void main(String[] args) {
      SLL sll = new SLL();
      sll.addFirst(2);
      sll.addLast(3);
      sll.addFirst(1);
      sll.print();
      sll.removeFirst();
      sll.print();
      sll.removeLast();
```

```
        sll.print();
        }
}
```

Exercise 2:(In-class)From the previous exercise, create four new methods as follows:

1. *int getSize()* → return size of SLL
2. *booleanisEmpty()* → return true if SLL is empty
3. *element get(index)* → *return element at specified position*
4. void clear() → remove all elements

For example, if we have a SLL.



Expected result

```
=== Empty SLL ===
Size = 0
Empty = true

=== After adding elements ===
Size = 3
Empty = false
Element 0 = 11
Element 1 = 22
Element 2 = 11

=== After clearing elements ===
Size = 0
Empty = true
```

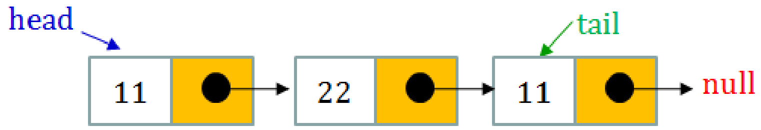
```
package lab3;
class SLL {
      // ----------------------- Node --------------------
      private class Node {
            private int element;
            private Node next;

            // constructor
            public Node(int data) {
                  element = data;
                  next = null;
            }

            // link a new node to this node
            public void link(Node newNode) {
                  next = newNode;
            }

            // return next node
```

```
          public Node getNextNode() {
                return next;
          }

          // return element of this node
          public int getElement() {
                return element;
          }

    }
    // -------------------- End Node --------------------

    // SLL properties and methods
    private Node head = null;
    private Node tail = null;
    private int size = 0; // SLL's size

    public int size() {
          return size;
    }
    public boolean isEmpty() {
          boolean isEmpty=false;
          if(size==0) {
                isEmpty= true;
          }
          return isEmpty;
    }
    public int get(int index) {
          Node p = head;
          for(int i=0;i<index;i++) {
                 p = p.getNextNode();
          }
                 return p.getElement();


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

    public void addLast(int data) {
          // create new node
          Node newNode = new Node(data);
          if (size == 0) {
                head = newNode;
          } else {
                tail.link(newNode);
          }
          tail = newNode;
          size++;
    }

    public void addFirst(int data) {

          Node newNode = new Node(data);
           if(size == 0){
           tail = newNode;
           }else{
           newNode.link(head);
           }
           head = newNode;
           size++;
    }

    public void removeFirst() {

          if(size == 1){
                 head = null;
                 tail = null;
                 size--;
                }
          else{
                 head = head.getNextNode();
                 size--;
                 }
    }

    public void removeLast() {

          if(size == 1){
                 head = null;
                 tail = null;
                 size--;
                 }
          else{
                 Node p = head;
                 while(p.getNextNode() != tail){
                 p = p.getNextNode();
                 }
                 tail = p;
                 tail.link(null);
                 size--;
                 }

    }

    public void print() {
          if (size == 0) {
                  System.out.println("Empty linked list");
            } else {
                  for (Node p = head; p != null; p = p.getNextNode()) {
                        System.out.print(p.getElement() + "->");
                  }
                  System.out.println("null");
            }
      }
}
public class MainSLL{
public static void main(String[] args){
        SLL sll = new SLL();
System.out.println("=== Empty SLL ===");
System.out.println("Size = "+ sll.size());
System.out.println("Empty = " + sll.isEmpty());
System.out.println();

//add elements
sll.addLast(11);
sll.addLast(22);
sll.addLast(11);
System.out.println("=== After adding elements ===");
System.out.println("Size = "+ sll.size());
System.out.println("Empty = " + sll.isEmpty());
System.out.println("Element 0 = " + sll.get(0));
System.out.println("Element 1 = " + sll.get(1));
System.out.println("Element 2 = " + sll.get(2));
System.out.println();

//clear
sll.clear();
System.out.println("=== After clearing elements ===");
System.out.println("Size = "+ sll.size());
System.out.println("Empty = " + sll.isEmpty());
    }
}
```

Exercise 3(Homework): From the previous exercise, create three new methods as follows:

1. **findElement(element)** to find whether a number is in a SLL and return **true** or **false**when found and not found respectively
2. **countElement(element)** to count nodes whose elements are equal to a number
3. **sumElement()** to find summation of all numeric elements of nodes

For example, if we have a SLL.



- findElement(11) will return true but findElement(33) will return false
- countElement(11) will return 2, countElement(22) will return 1 and countElement(33) will return 0
- sumElement() will return 44

Expected result

```
11->22->11->null

Find element 11: true
Find element 22: true
Find element 33: false
Count element 11: 2
Count element 22: 1
Count element 33: 0
Sum of all elements: 44
```

```
package lab3;
class SLL {
      // ----------------------- Node --------------------
      private class Node {
            private int element;
            private Node next;

            // constructor
            public Node(int data) {
                  element = data;
                  next = null;
            }

            // link a new node to this node
            public void link(Node newNode) {
                  next = newNode;
            }
```

```
        // return next node
        public Node getNextNode() {
            return next;
        }

        // return element of this node
        public int getElement() {
            return element;
        }

    }
    // -------------------- End Node --------------------

    // SLL properties and methods
    private Node head = null;
    private Node tail = null;
    private int size = 0; // SLL's size

    public int size() {
        return size;
    }
    public boolean isEmpty() {
        boolean isEmpty=false;
        if(size==0) {
            isEmpty= true;
        }
        return isEmpty;
    }
    public boolean findElement(int fe) {
        boolean findElement=true;
        Node p =head;
         while(p.getNextNode() != tail){
             p = p.getNextNode();
             if (p.getElement() == fe) {
                 findElement=true;
             }
             else {
                 findElement=false;
             }
             }
         if (head.getElement() == fe || tail.getElement() == fe) {
             findElement=true;
         }
         return findElement;
    }
    public int countElement (int ce) {
        int countElement=0;
        Node p =head;
         while(p.getNextNode() != null){
                 if (p.getElement() == ce) {
                       countElement++;
                 }
                 p = p.getNextNode();
                 }
           if ( tail.getElement() == ce) {
                 countElement++;
           }
           return countElement;
    }

    public int sumElement () {
          int sumElement=0;
          Node p =head;
           while(p.getNextNode() != null){
                 sumElement+=p.getElement();
                 p = p.getNextNode();
                 }
          sumElement+=tail.getElement();
           return sumElement;
    }

    public int get(int index) {
          Node p = head;
          for(int i=0;i<index;i++) {
                 p = p.getNextNode();
          }
                 return p.getElement();


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

    public void addLast(int data) {
          // create new node
          Node newNode = new Node(data);
          if (size == 0) {
                head = newNode;
          } else {
                tail.link(newNode);
          }
          tail = newNode;
          size++;
    }

    public void addFirst(int data) {
          Node newNode = new Node(data);
           if(size == 0){
           tail = newNode;
           }else{
           newNode.link(head);
           }
           head = newNode;
           size++;
    }

    public void removeFirst() {

          if(size == 1){
                 head = null;
                 tail = null;
                 size--;
                }
          else{
                 head = head.getNextNode();
                 size--;
                 }
    }

    public void removeLast() {

          if(size == 1){
                 head = null;
                 tail = null;
                 size--;
                 }
          else{
                 Node p = head;
                 while(p.getNextNode() != tail){
                 p = p.getNextNode();
                 }
                 tail = p;
                 tail.link(null);
                 size--;
                 }

    }

    public void print() {
          if (size == 0) {
                System.out.println("Empty linked list");
          } else {
                for (Node p = head; p != null; p = p.getNextNode()) {
                      System.out.print(p.getElement() + "->");
                }
```

```
                  System.out.println("null");
            }
      }
}
 public class MainSLL{
 public static void main(String[] args){
        SLL sll = new SLL();
sll.addFirst(11);
sll.addLast(22);
sll.addLast(11);
sll.print();

System.out.println();
System.out.println("Find element 11: "+sll.findElement(11));
System.out.println("Find element 22: "+sll.findElement(22));
System.out.println("Find element 33: "+sll.findElement(33));
System.out.println("Count element 11: "+sll.countElement(11));
System.out.println("Count element 22: "+sll.countElement(22));
System.out.println("Count element 33: "+sll.countElement(33));
System.out.println("Sum of all elements: "+sll.sumElement());
    }
}
```

Example (Optional): Implementation of SLL using ArrayList

JAVA provides a class ArrayList which is one type of SLL with several built-in methods. Try to observe the following codes.

Be careful that the data to store in ArrayList must be objects not the primitive variables.

```
importjava.util.ArrayList;

publicclassMainArrayList {

      publicstaticvoid main(String[] args) {
            //create an Integer ArrayList with default size of 10
ArrayList<Integer>sll = newArrayList<Integer>();

//add first
sll.add(0, new Integer(2));     //[2]
//add last
sll.add(new Integer(3));        //[2,3]
//add first
sll.add(0, new Integer(1));     //[1,2,3]
//print
System.out.println(sll);

//remove first
sll.remove(0);     //[2,3]
//remove last
sll.remove(sll.size() - 1);   //[2]
//print
System.out.println(sll);

//find element
System.out.println(sll.contains(new Integer(2)));      //true
System.out.println(sll.contains(new Integer(1)));      //false
      }
}
```

To see more details and functionality of ArrayList, please visit https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Exercise 4 (Optional): Use the ArrayList and its functions above to get the following result.

Assume that we want to create a back-end of a shopping system for the administrator. This system allows to add, delete and show products. Suppose that we use two SLL to store product name and product price.

Expected result

```
=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...1
        --- Product ---
        No product

=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...2
        Enter product name: Shirt
        Enter product price: 199

=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...2
        Enter product name: Shoes
        Enter product price: 299

=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...1
        --- Product ---
        1. Shirt 199 baht
        2. Shoes 299 baht
```

```
=== Shopping Backend ===
1. Show product
2. Add product
3. Delete produc t
4. Exit
Choose 1-4...3
        1. Shirt 199 baht
        2. Shoes 299 baht
        Enter the number of product to delete...1

=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...1
        --- Product ---
        1. Shoes 299 baht

=== Shopping Backend ===
1. Show product
2. Add product
3. Delete product
4. Exit
Choose 1-4...4
        Good bye
```

Paste your code here.

```
package asdf;

import java.util.ArrayList;

import java.util.Scanner;

public class Optional {

public static void main(String[] args) {

// TODO Auto-generated method stub

int choice;

int delete;

int i = 0;

String productname ="";
```

```
int productprice;

ArrayList<String> name = new ArrayList<String>();

ArrayList<Integer> price = new ArrayList<Integer>();

Scanner input = new Scanner(System.in);

while(true) {

System.out.println("=== Shopping Backend === \n1.Show Product\n2.Add
Product\n3.Delete Product\n4.Exit");

System.out.print("Choose 1-4...");

choice = input.nextInt();

if(choice == 1) {

System.out.println("---Product---");

if(name.isEmpty()) {

System.out.println("No Product");

}

else {

for(int j=0; j<name.size();j++) {

System.out.println((j+1)+ ". " + name.get(j)+ " "+ price.get(j)+"
baht");

}

}

}

else if(choice == 2) {

System.out.print("Enter product name: ");

productname = input.next();

name.add(productname);

System.out.print("Enter the product price: ");
productprice = input.nextInt();
price.add(productprice);
i++;
}
else if(choice == 3) {
for(int j=0; j<name.size();j++) {
System.out.println((j+1)+ ". " + name.get(j)+ " "+ price.get(j)+" baht");
}
System.out.print("Enter the number of product to delete...");
delete = input.nextInt();
if (delete <= name.size()) {
name.remove(delete - 1);
price.remove(delete -1);
i--;
}
}
else if (choice == 4){
System.out.println("Good Bye");
break;
}
}
}
}
```