Name:	Evangelista, Alyanna Ricci T.	Date:	03/04/25
	Tesnado, Johncen Jekk P.	Section:	1CSC

Code:

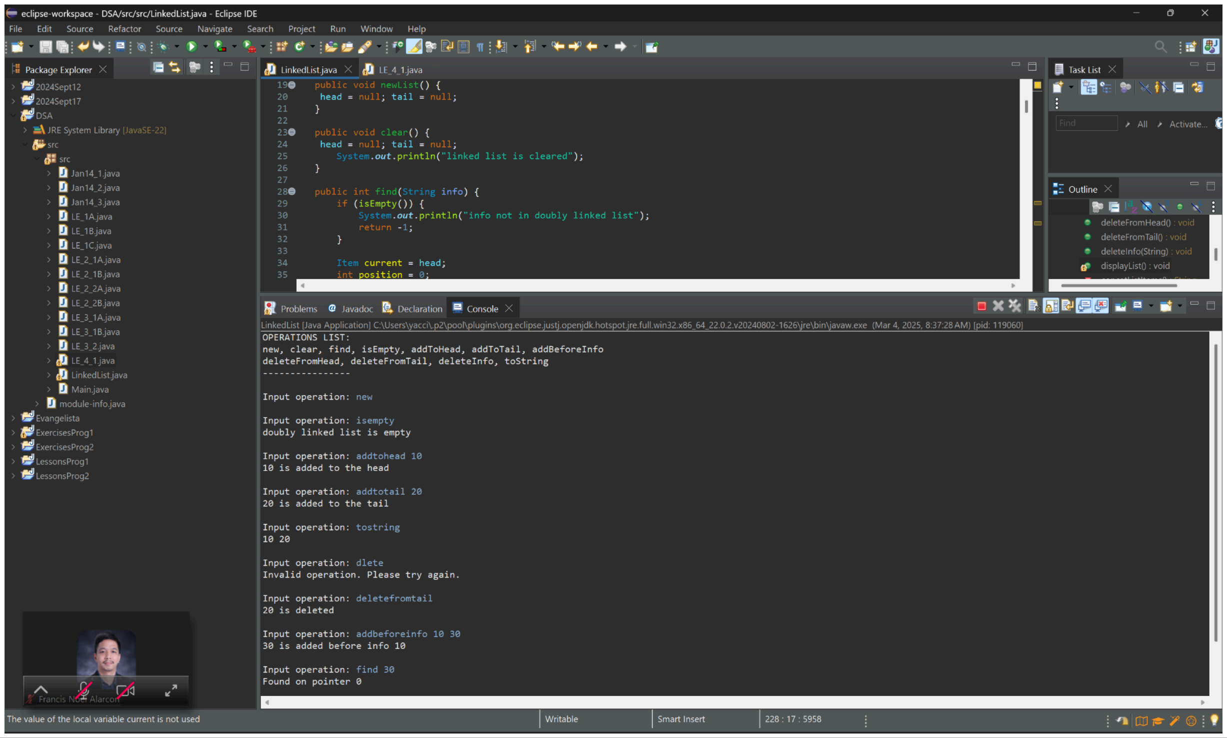
```
package src;
import java.util.Scanner;
public class LinkedList {
  private class Item {
      String info; Item next; Item prev;

      public Item(String info) {
          this.info = info; this.next = null; this.prev = null;
      }
  }
   private Item head; private Item tail;
   // Constructor
  public LinkedList() {
      head = null; tail = null;
  }
   public void newList() {
      head = null; tail = null;
  }
   public void clear() {
      head = null; tail = null;
      System.out.println("linked list is cleared");
  }
   public int find(String info) {
      if (isEmpty()) {
          System.out.println("info not in doubly linked list");
          return -1;
      }

      Item current = head;
      int position = 0;

      while (current != null) {
          if (current.info.equals(info)) {
              System.out.println("Found on pointer " + position);
              return position;
          }
          current = current.next;
          position++;
      }

      System.out.println("info not in doubly linked list");
      return -1;
  }
  public boolean isEmpty() {
      return head == null;
  }
  public void addToHead(String info) {
      Item newItem = new Item(info);

      if (isEmpty()) {
            head = newItem; tail = newItem;
        } else {
            newItem.next = head;
            head.prev = newItem;
            head = newItem;
        }

        System.out.println(info + " is added to the head");
    }
    public void addToTail(String info) {
        Item newItem = new Item(info);

        if (isEmpty()) {
            head = newItem; tail = newItem;
        } else {
            tail.next = newItem;
            newItem.prev = tail;
            tail = newItem;
        }

        System.out.println(info + " is added to the tail");
    }
    private Item getItemWithInfo(String info) {
        Item current = head;

        while (current != null) {
            if (current.info.equals(info)) {
                return current;
            }
            current = current.next;
        }

        return null;
    }
    public void addBeforeInfo(String targetInfo, String newInfo) {
        if (isEmpty()) {
            System.out.println("info not in doubly linked list");
            return;
        }

        Item targetItem = getItemWithInfo(targetInfo);

        if (targetItem == null) {
            System.out.println("info not in doubly linked list");
            return;
        }

        Item newItem = new Item(newInfo);

        if (targetItem == head) {
            newItem.next = head;
            head.prev = newItem;
            head = newItem;
        } else {
            newItem.prev = targetItem.prev;
            newItem.next = targetItem;
            targetItem.prev.next = newItem;
        targetItem.prev = newItem;
    }

    System.out.println(newInfo + " is added before info " + targetInfo);
}
public void addAfterInfo(String targetInfo, String newInfo) {
    if (isEmpty()) {
        System.out.println("info not in doubly linked list");
        return;
    }

    Item targetItem = getItemWithInfo(targetInfo);

    if (targetItem == null) {
        System.out.println("info not in doubly linked list");
        return;
    }

    Item newItem = new Item(newInfo);

    if (targetItem == tail) {
        newItem.prev = tail;
        tail.next = newItem;
        tail = newItem;
    } else {
        newItem.next = targetItem.next;
        newItem.prev = targetItem;
        targetItem.next.prev = newItem;
        targetItem.next = newItem;
    }

    System.out.println(newInfo + " is added after info " + targetInfo);
}
public void deleteFromHead() {
    if (isEmpty()) {
        System.out.println("info not in doubly linked list");
        return;
    }

    String deletedInfo = head.info;

    if (head == tail) {
        head = null;
        tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }

    System.out.println(deletedInfo + " is deleted");
}
public void deleteFromTail() {
    if (isEmpty()) {
        System.out.println("info not in doubly linked list");
        return;
    }
```

```java
    String deletedInfo = tail.info;

    if (head == tail) {
        head = null;
        tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }

    System.out.println(deletedInfo + " is deleted");
}
 public void deleteInfo(String info) {
    if (isEmpty()) {
        System.out.println("info not in doubly linked list");
        return;
    }

    // if info in head
    if (head.info.equals(info)) {
        deleteFromHead();
        return;
    }

    // if info in taill
    if (tail.info.equals(info)) {
        deleteFromTail();
        return;
    }

    Item current = head.next;

    while (current != null && current != tail) {
        if (current.info.equals(info)) {
            current.prev.next = current.next;
            current.next.prev = current.prev;

            System.out.println(info + " is deleted");
            return;
        }
        current = current.next;
    }

    System.out.println("info not in doubly linked list");
}
public void displayList() {
    if (isEmpty()) {
        System.out.println("linked list is empty");
        return;
    }

    Item current = head;
    String result = concatListItems();

    System.out.println(result);
}
 private String concatListItems() {
      if (isEmpty()) {
          return "linked list is empty";
      }

      String[] items = new String[countItems()];
      Item current = head;
      int index = 0;

      while (current != null) {
          items[index++] = current.info;
          current = current.next;
      }

      return String.join(" ", items);
  }
   private int countItems() {
      int count = 0;
      Item current = head;

      while (current != null) {
          count++;
          current = current.next;
      }

      return count;
  }
   public String toString() {
      return concatListItems();
  }
   public static void main(String[] args) {
      LinkedList list = new LinkedList();
      Scanner scanner = new Scanner(System.in);
      System.out.println("OPERATIONS LIST: "
              + "\nnew, clear, find, isEmpty, addToHead, addToTail,
addBeforeInfo"
              + "\ndeleteFromHead, deleteFromTail, deleteInfo, toString"
              + "\n-----------------");

      while (true) {
          System.out.println();
          System.out.print("Input operation: ");
          String line = scanner.nextLine().trim();
          String[] parts = line.split("\\s+");
          String operation = parts[0].toLowerCase();

          switch (operation) {
              case "new":
                  list.newList();
                  break;
              case "clear":
                  list.clear();
                  break;
              case "find":
                  if (parts.length > 1) {
                      list.find(parts[1]);
                  } else {
                      System.out.println("Please provide an info to find");
```

```java
                  }
                  break;
              case "isempty":
                  if (list.isEmpty()) {
                      System.out.println("doubly linked list is empty");
                  } else {
                      System.out.println("doubly linked list is not empty");
                  }
                  break;
              case "addtohead":
                  if (parts.length > 1) {
                      list.addToHead(parts[1]);
                  } else {
                      System.out.println("Please provide an info to add");
                  }
                  break;
              case "addtotail":
                  if (parts.length > 1) {
                      list.addToTail(parts[1]);
                  } else {
                      System.out.println("Please provide an info to add");
                  }
                  break;
              case "addbeforeinfo":
                  if (parts.length > 2) {
                      list.addBeforeInfo(parts[1], parts[2]);
                  } else {
                      System.out.println("Please provide target info and new
info");
                  }
                  break;
              case "addafterinfo":
                  if (parts.length > 2) {
                      list.addAfterInfo(parts[1], parts[2]);
                  } else {
                      System.out.println("Please provide target info and new
info");
                  }
                  break;
              case "deletefromhead":
                  list.deleteFromHead();
                  break;
              case "deletefromtail":
                  list.deleteFromTail();
                  break;
              case "deleteinfo":
                  if (parts.length > 1) {
                      list.deleteInfo(parts[1]);
                  } else {
                      System.out.println("Please provide an info to delete");
                  }
                  break;
              case "tostring":
                  list.displayList();
                  break;
              default:
                  System.out.println("Invalid operation. Please try again.");
```

```
                    }
                }
            }
}
```

Screenshots:



eclipse-workspace - DSA/src/src/LinkedList.java - Eclipse IDE

```java
    public void newList() {
     head = null; tail = null;
    }

    public void clear() {
     head = null; tail = null;
        System.out.println("linked list is cleared");
    }

    public int find(String info) {
        if (isEmpty()) {
            System.out.println("info not in doubly linked list");
            return -1;
        }

        Item current = head;
        int position = 0;
```

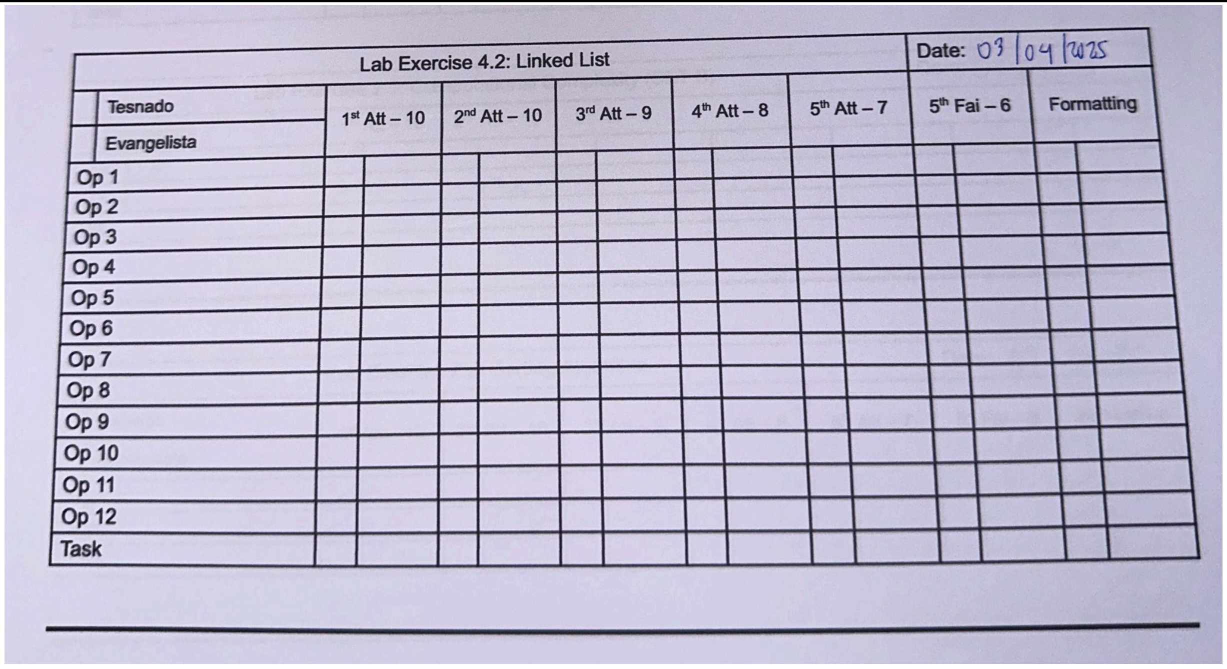
LinkedList [Java Application] C:\Users\yacci\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (Mar 4, 2025, 8:37:28 AM) [pid: 119060]

```
OPERATIONS LIST:
new, clear, find, isEmpty, addToHead, addToTail, addBeforeInfo
deleteFromHead, deleteFromTail, deleteInfo, toString
----------------

Input operation: new

Input operation: isempty
doubly linked list is empty

Input operation: addtohead 10
10 is added to the head

Input operation: addtotail 20
20 is added to the tail

Input operation: tostring
10 20

Input operation: dlete
Invalid operation. Please try again.

Input operation: deletefromtail
20 is deleted

Input operation: addbeforeinfo 10 30
30 is added before info 10

Input operation: find 30
Found on pointer 0
```

The value of the local variable current is not used

Writable | Smart Insert | 228 : 17 : 5958

Scoring Sheet:



Lab Exercise 4.2: Linked List											Date: 03/04/2025			
Tesnado	1st Att – 10		2nd Att – 10		3rd Att – 9		4th Att – 8		5th Att – 7		5th Fai – 6		Formatting	
Evangelista														
Op 1														
Op 2														
Op 3														
Op 4														
Op 5														
Op 6														
Op 7														
Op 8														
Op 9														
Op 10														
Op 11														
Op 12														
Task														