FAKULTI TEKNOLOGI KEJURUTERAAN KELAUTAN DAN INFORMATIK

2022/2023

DATA STRUCTURE & ALGORITHM

Lab 2: Singly Linked list

Name: OMAR ISMAIL ABDJALEEL ALOMORY

Matric Number: S63955

Lab: MP3

Date: 8/11/2022

VERSION 1

`

STUDENT INFORMATION

PLEASE FILL IN YOUR PERSONAL DETAILS:

NAME: OAMR ISMAIL ABDJALEEL ALOMORY

MATRIC NUMBER:S63955

GROUP:K2

LAB:MP3

DATE:8/11/2022

`

TABLE OF CONTENTS

`

INSTRUCTIONS

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Fakulti Teknologi Kejuruteraan Kelautan dan Informatik, Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual.

This laboratory manual is for use by the students of the Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu (UMT) only. It is not permissible to print and distribute this manual without the official authorisation of the author.

Please follow step by step as described in the manual.

`

TASK 1: IMPLEMENTATION OF SINGLY LINKED LIST

OBJECTIVE

In this lab, we will learn the following topics:

- A linked list
- Implementation of linked list using java
- Insert Node at the beginning of the list
- Traverse List
- Delete Node from a list

TASK DESCRIPTION

In each of the topics, students should implement the tasks step by steps in order to have better understanding of singly linked list.

ESTIMATED TIME

[60 Minutes]

DEFINITION OF LINKED LIST

A linked list is just a chain of nodes, with each subsequent node being a child of the previous one. Many programs rely on linked lists for their storage because these don't have any evident restrictions. There is no limit (other than the amount of memory) on the number of elements they can store.

SINGLY LINKED LISTS

Basics:

- A singly linked list is a concrete data structure consisting of a sequence of nodes
- It has a head node pointer indicating the first node in list.
- It could have optionally a tail pointer node indication the last node in list.
- Each node stores
    - Element (data)
    - Link to the next node

`

Operations:

The common operations of Singly linked list are:

1. Insertion (or Add):
   - Add first
   - Add last
   - Add middle (after existing node); example:

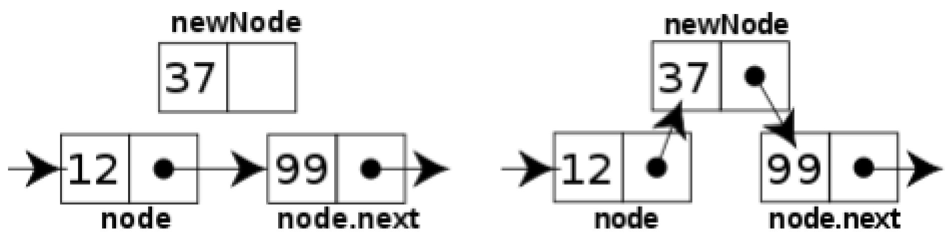


Figure 1: An example of insertion a new node in the middle of linked list

2. Deletion (or Remove):
   - Delete first
   - Delete last
   - Delete after existing node; example:

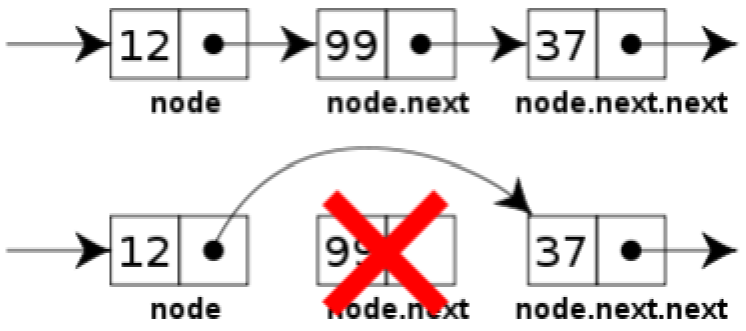


Figure 2: An example of insertion a new node in the middle of linked list

`

SOME COMMON HANDLING METHODS OF SINGLY/DOUBLY LINKED LIST ARE:

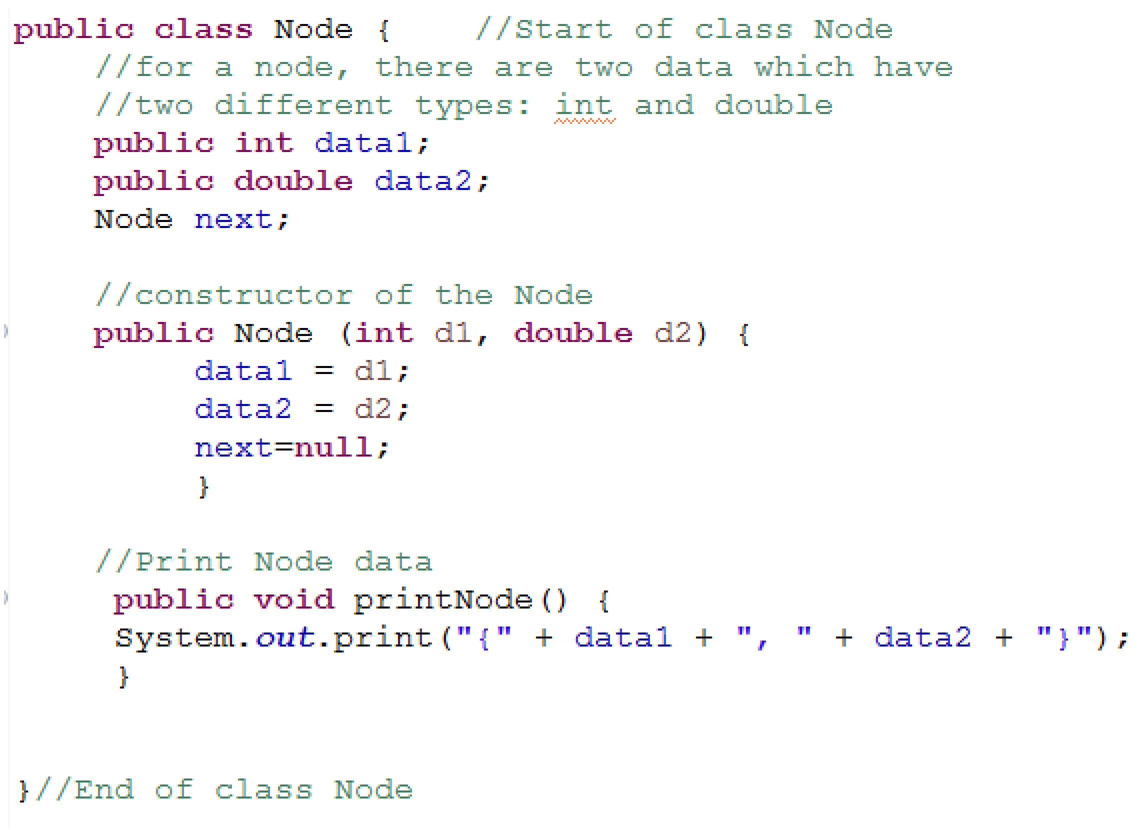
1. Print (Or Show):
2. Print all list elements
3. Print certain node
4. Search for an element
5. Find list size (if no size variable in list class)
6. Reverse the linked list [step 2]

ACTIVITIES

Activity 1:

Apply and test the Linked List implementation bellow:

// the code below is a simple example of a linked list that inserts a new link at the beginning of the list, deletes from the beginning of the list and loops through the list to print the links contained in it.



```java
public class Node {     //Start of class Node
    //for a node, there are two data which have
    //two different types: int and double
    public int data1;
    public double data2;
    Node next;

    //constructor of the Node
    public Node (int d1, double d2) {
         data1 = d1;
         data2 = d2;
         next=null;
         }

    //Print Node data
     public void printNode() {
     System.out.print("{" + data1 + ", " + data2 + "}");
     }


}//End of class Node
```

`

```
public class LinkList { //Start of class LinkList

    private Node first;

    //LinkList constructor
    public LinkList() {
        first = null;
    }

    //Returns true if the linked list is empty
    public boolean isEmpty() {
        return first == null;
    }

    //Inserts a new node at the first of the linked list
     public void addFirst(int d1, double d2) {
         Node node = new Node(d1, d2);
         node.next = first;
         first = node;
     }

  //Inserts a new node at the first of the linked list
   public void addFirst(int d1, double d2) {
       Node node = new Node(d1, d2);
       node.next = first;
       first = node;
   }
   //Deletes the node at the first of the linked list
   public Node deleteFirst() {
       Node temp = first;
       first = first.next;
       return temp;
  }

  //Prints the linked list data
   public void printList() {
       Node currentNode = first;
       System.out.print("List: ");
       while(currentNode!= null) {
              currentNode. printNode();
              currentNode = currentNode.next;
       }
       System.out.println("");
     }
  } //End of class LinkList
```

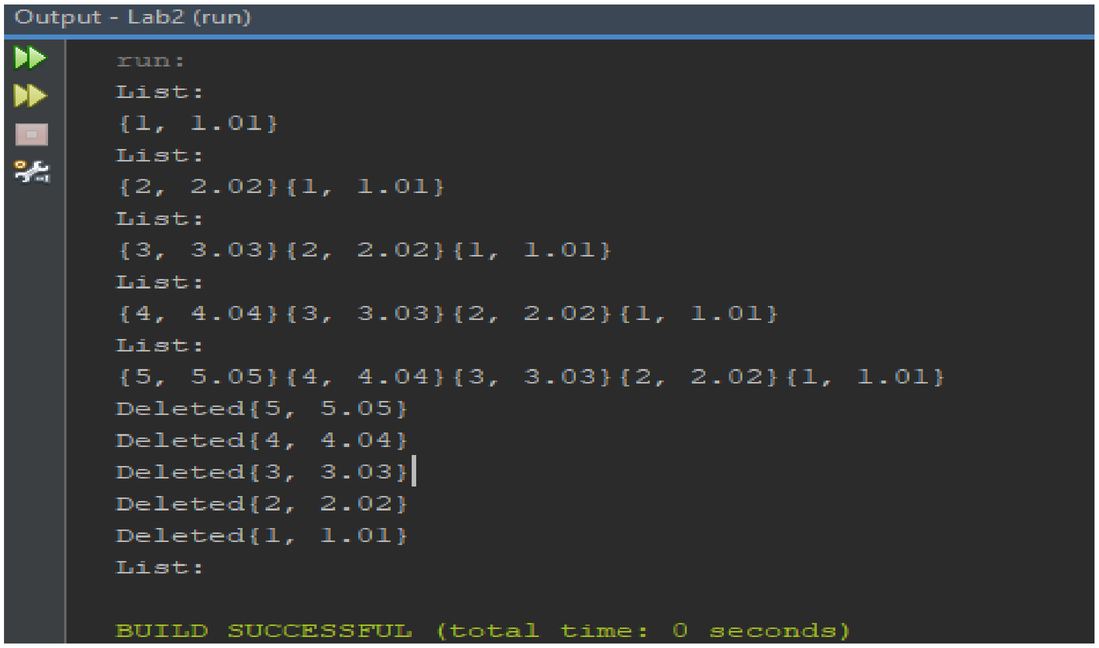
`

```java
public class LinkListTest { //Start of class LinkListTest
    public static void main(String[] args) {
         LinkList list = new LinkList();
         list.addFirst(1, 1.01);
         list.printList();
         list.addFirst(2, 2.02);
         list.printList();
         list.addFirst(3, 3.03);
         list.printList();
         list.addFirst(4, 4.04);
         list.printList();
         list.addFirst(5, 5.05);
         list.printList();
         while(!list.isEmpty()) {
             Node deletedLink = list.deleteFirst();
             System.out.print("deleted: ");
             deletedLink.printNode();
             System.out.println("");
         }
        list.printList();
    }
} //End of class LinkListTest
```

Activity 2: Compile the program. Execute the program and record the results.

Upload the screenshot using the control box provided below:

Answer:


Output - Lab2 (run)

```
run:
List:
{1, 1.01}
List:
{2, 2.02}{1, 1.01}
List:
{3, 3.03}{2, 2.02}{1, 1.01}
List:
{4, 4.04}{3, 3.03}{2, 2.02}{1, 1.01}
List:
{5, 5.05}{4, 4.04}{3, 3.03}{2, 2.02}{1, 1.01}
Deleted{5, 5.05}
Deleted{4, 4.04}
Deleted{3, 3.03}
Deleted{2, 2.02}
Deleted{1, 1.01}
List:

BUILD SUCCESSFUL (total time: 0 seconds)
```

`

TASK 2: APPLICATION OF SINGLY LINKED LIST

TASK DESCRIPTION

Based on the previous code, you may use it to complete the following tasks.

ESTIMATED TIME

[120 Minutes]

---

1. Write the Node class consists of two components of a node (i.e.: element, next), with a default construct and a constructor that accepts an item assigned to the initially declared element variable.

2. Write a class called **`MyLinkedList`**. The class should have the following:
   a. Default constructor
   b. Nodes for head and tail

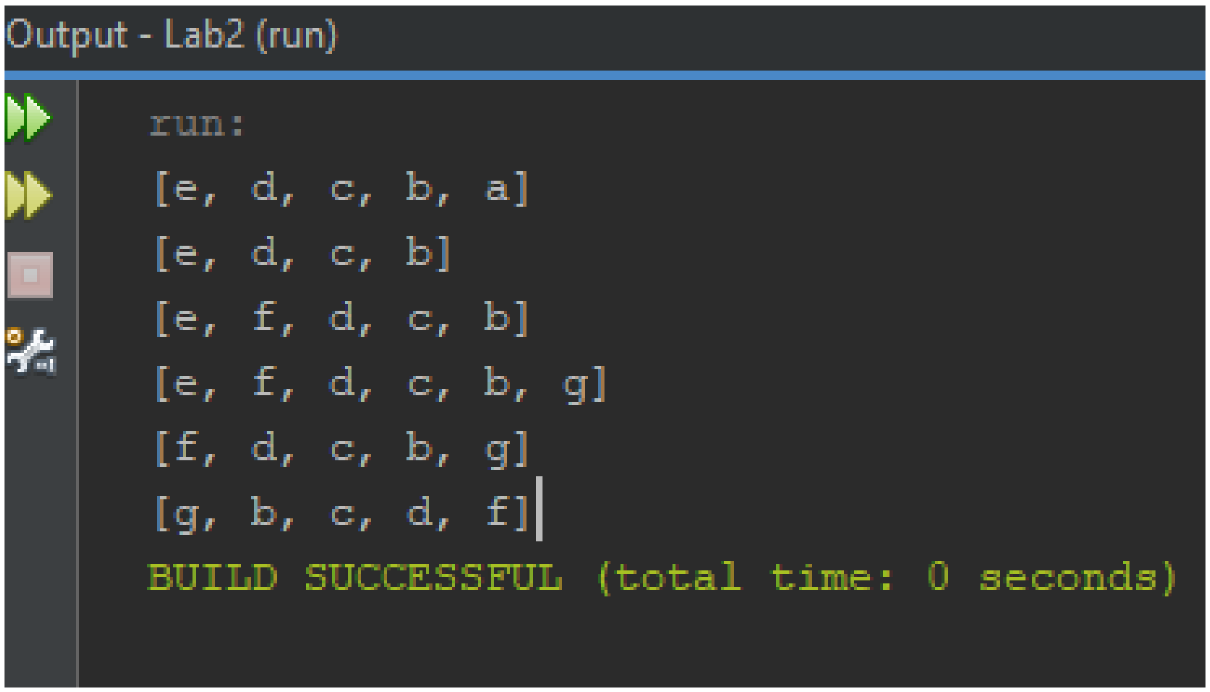
3. Implement the following methods (some methods had already been shown in Task 1)
   ```
   a. public void addFirst(Node e)
   b. public void addLast(Node e)
   c. public void add(int index, Node e)
   d. public Node removeFirst()
   e. public Node removeLast()
   f. public void printList()
   g. public void reverse()
   ```

4. Write a test program called `TestLinkedList` that creates a list from `MyLinkedList` class. Using the methods in (3), do the following:
   a. Add these elements to the linked list using `addFirst()` method according to the order: `a, b, c, d, e`
   b. Print all the elements in the list.
   c. Delete the last value.
   d. Print current list.
   e. Add `'f'` at the second position in the linked list.
   f. Print current list.
   g. Add `'g'` at the end of the list.
   h. Print current list.

`

i. Delete the first element in the current list.
j. Print current list.
k. Reverse the list.
l. Print the list.

5. Compile the program. Execute the program and record the results.

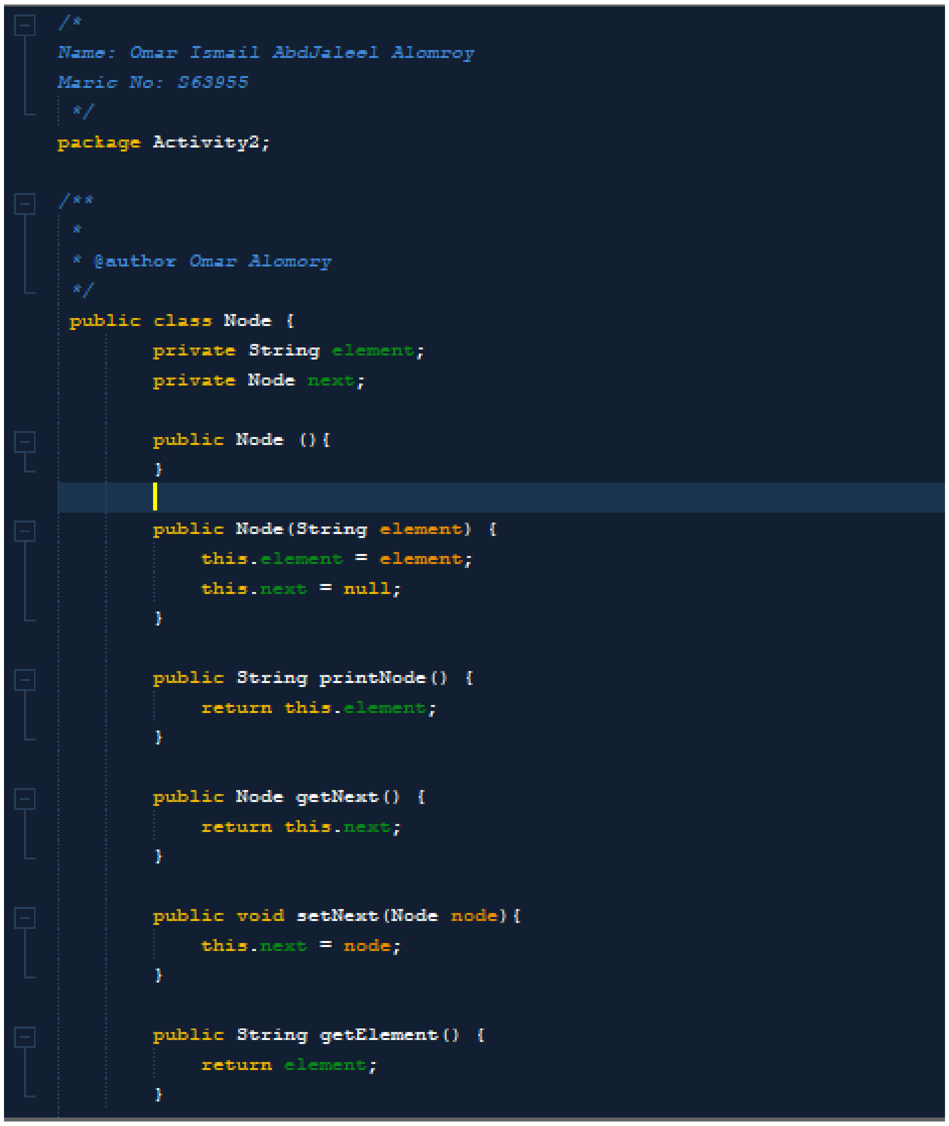


```
run:
[e, d, c, b, a]
[e, d, c, b]
[e, f, d, c, b]
[e, f, d, c, b, g]
[f, d, c, b, g]
[g, b, c, d, f]
BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Submit your codes at epembelajaran. Please ensure your codes are submitted to the correct group.

`

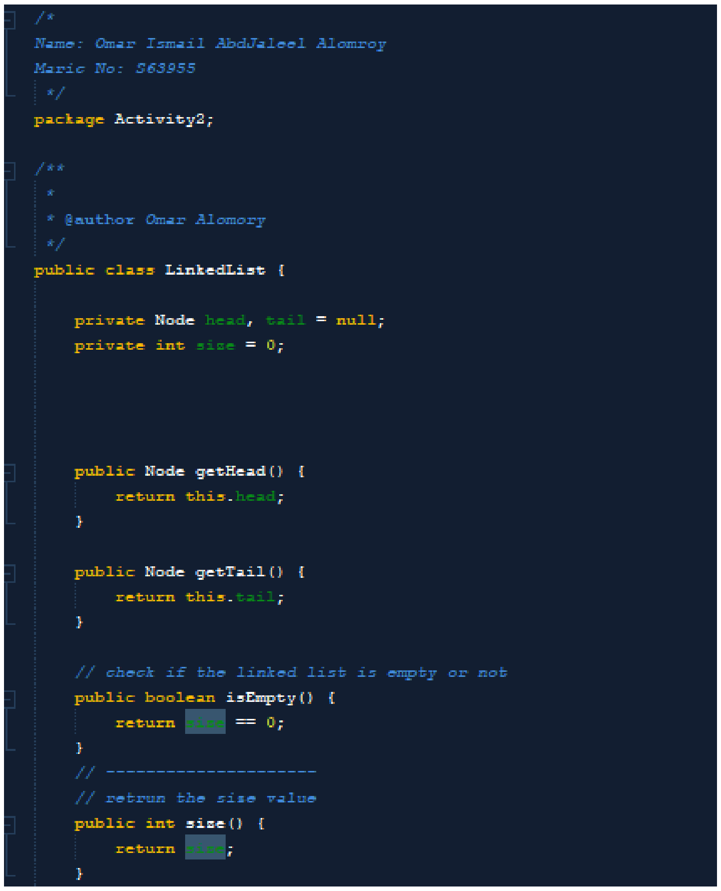
Node class



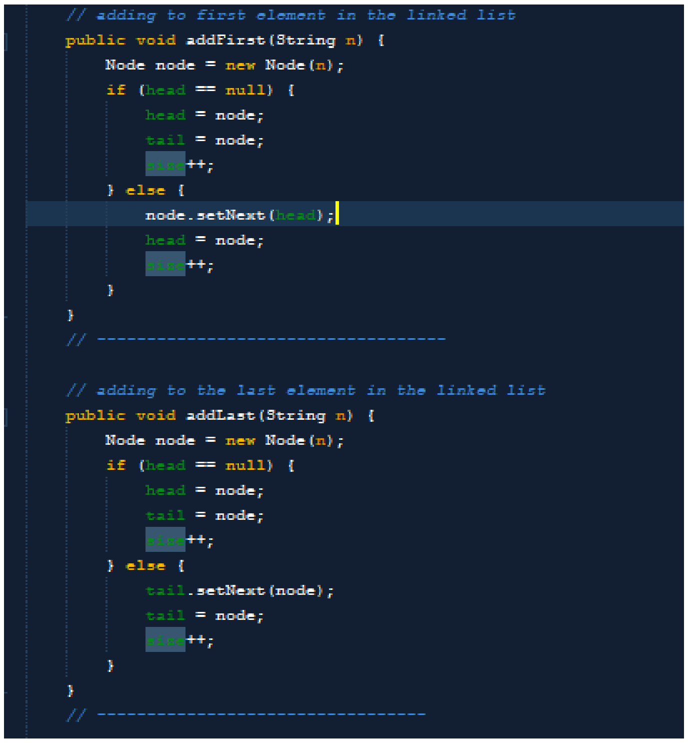
```
/*
Name: Omar Ismail AbdJaleel Alomroy
Maric No: S63955
 */
package Activity2;

/**
 *
 * @author Omar Alomory
 */
 public class Node {
        private String element;
        private Node next;

        public Node (){
        }
        
        public Node(String element) {
            this.element = element;
            this.next = null;
        }

        public String printNode() {
            return this.element;
        }

        public Node getNext() {
            return this.next;
        }

        public void setNext(Node node){
            this.next = node;
        }

        public String getElement() {
            return element;
        }
```

`

LinkedList class

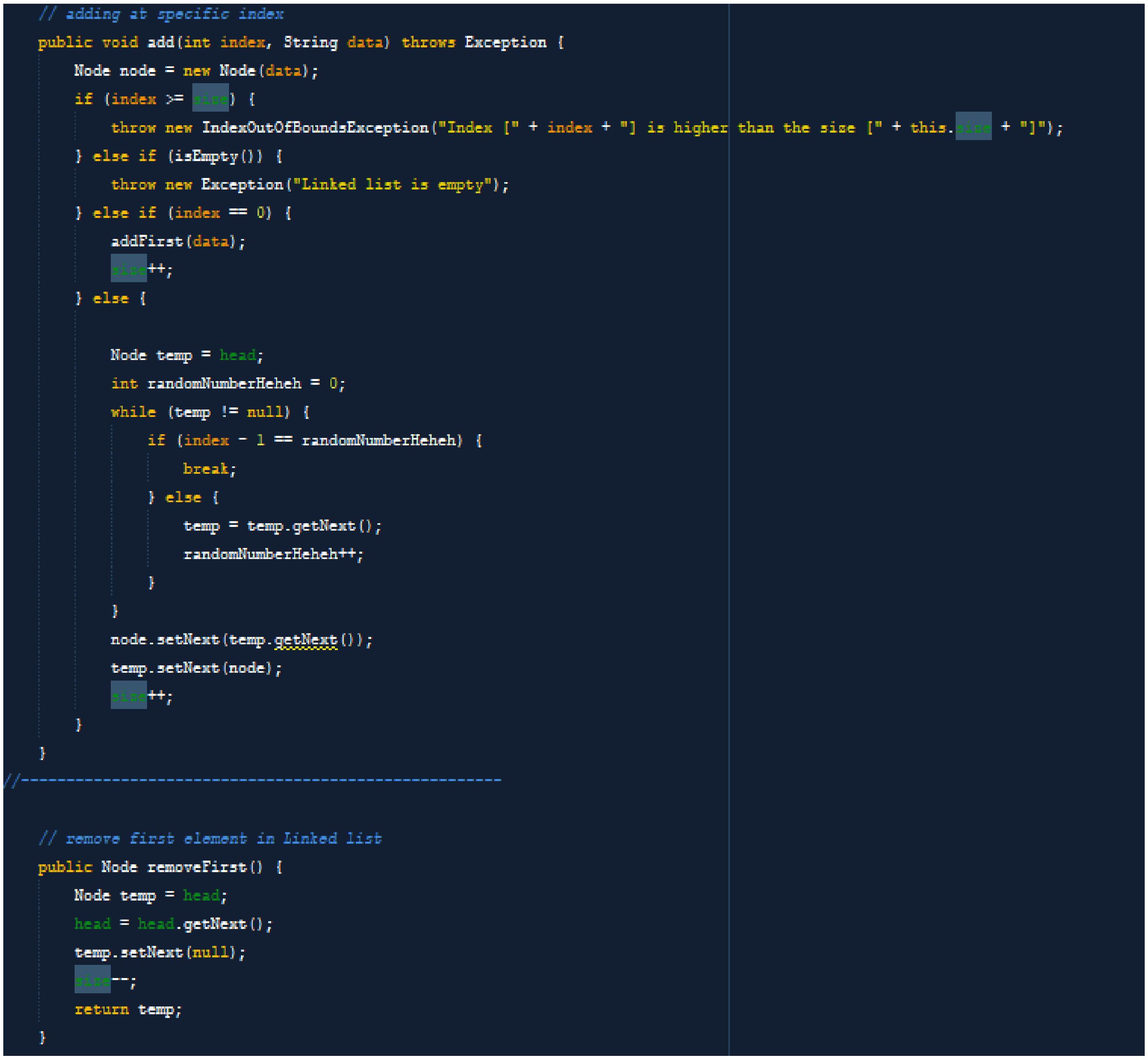


```java
/*
Name: Omar Ismail AbdJaleel Alomroy
Maric No: S63955
 */
package Activity2;

/**
 *
 * @author Omar Alomory
 */
public class LinkedList {

    private Node head, tail = null;
    private int size = 0;



    public Node getHead() {
        return this.head;
    }

    public Node getTail() {
        return this.tail;
    }

    // check if the linked list is empty or not
    public boolean isEmpty() {
        return size == 0;
    }
    // ---------------------
    // retrun the size value
    public int size() {
        return size;
    }
```



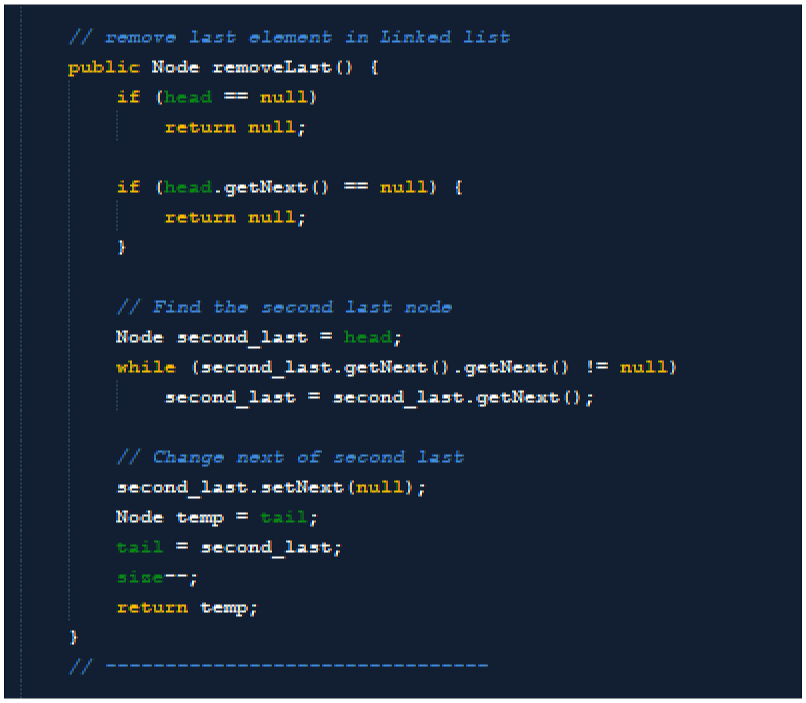
```java
    // adding to first element in the linked list
    public void addFirst(String n) {
        Node node = new Node(n);
        if (head == null) {
            head = node;
            tail = node;
            size++;
        } else {
            node.setNext(head);
            head = node;
            size++;
        }
    }
    // -----------------------------------

    // adding to the last element in the linked list
    public void addLast(String n) {
        Node node = new Node(n);
        if (head == null) {
            head = node;
            tail = node;
            size++;
        } else {
            tail.setNext(node);
            tail = node;
            size++;
        }
    }
    // --------------------------------
```

`

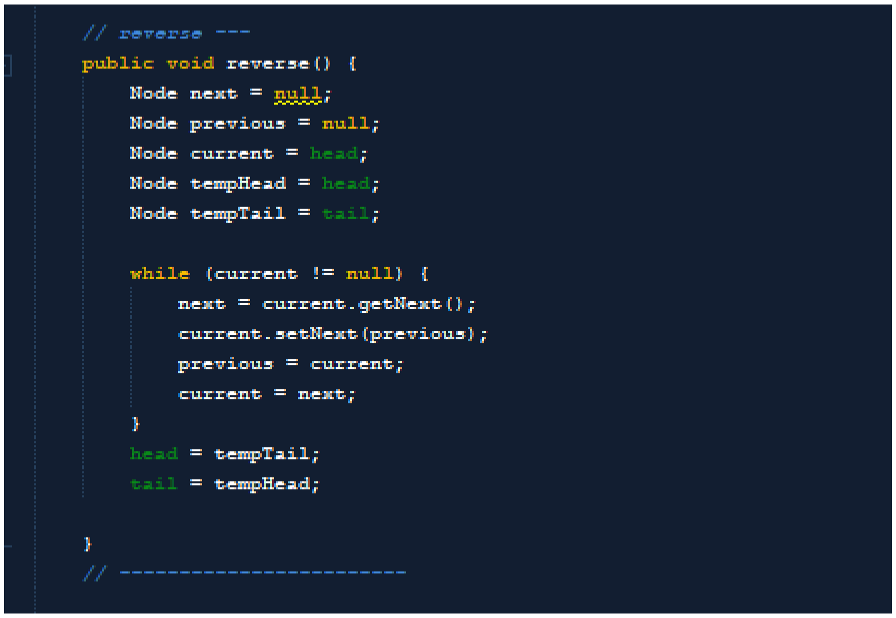


```java
    // adding at specific index
    public void add(int index, String data) throws Exception {
        Node node = new Node(data);
        if (index >= size) {
            throw new IndexOutOfBoundsException("Index [" + index + "] is higher than the size [" + this.size + "]");
        } else if (isEmpty()) {
            throw new Exception("Linked list is empty");
        } else if (index == 0) {
            addFirst(data);
            size++;
        } else {

            Node temp = head;
            int randomNumberHeheh = 0;
            while (temp != null) {
                if (index - 1 == randomNumberHeheh) {
                    break;
                } else {
                    temp = temp.getNext();
                    randomNumberHeheh++;
                }
            }
            node.setNext(temp.getNext());
            temp.setNext(node);
            size++;
        }
    }
//-----------------------------------------------------

    // remove first element in Linked list
    public Node removeFirst() {
        Node temp = head;
        head = head.getNext();
        temp.setNext(null);
        size--;
        return temp;
    }
```

`

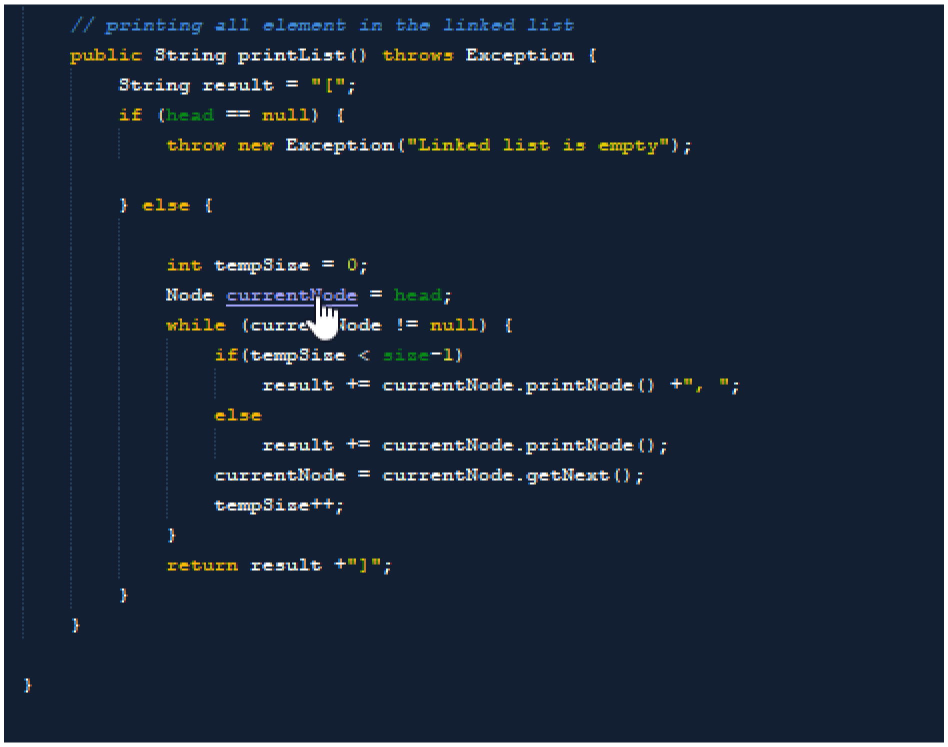


```
// remove last element in Linked list
public Node removeLast() {
    if (head == null)
        return null;

    if (head.getNext() == null) {
        return null;
    }

    // Find the second last node
    Node second_last = head;
    while (second_last.getNext().getNext() != null)
        second_last = second_last.getNext();

    // Change next of second last
    second_last.setNext(null);
    Node temp = tail;
    tail = second_last;
    size--;
    return temp;
}
// --------------------------------
```



```
// reverse ---
public void reverse() {
    Node next = null;
    Node previous = null;
    Node current = head;
    Node tempHead = head;
    Node tempTail = tail;

    while (current != null) {
        next = current.getNext();
        current.setNext(previous);
        previous = current;
        current = next;
    }
    head = tempTail;
    tail = tempHead;

}
// ------------------------
```

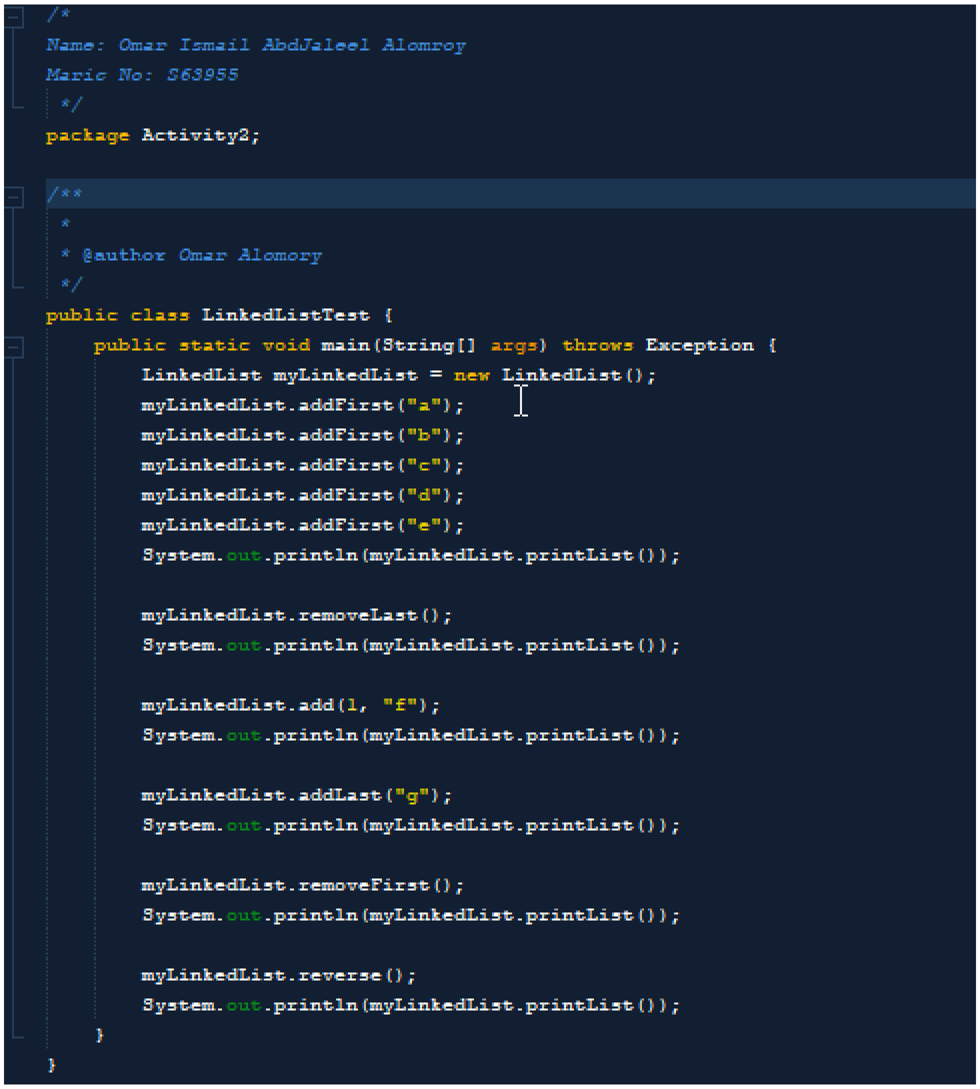
`



```
    // printing all element in the linked list
    public String printList() throws Exception {
        String result = "[";
        if (head == null) {
            throw new Exception("Linked list is empty");

        } else {

            int tempSize = 0;
            Node currentNode = head;
            while (currentNode != null) {
                if(tempSize < size-1)
                    result += currentNode.printNode() +", ";
                else
                    result += currentNode.printNode();
                currentNode = currentNode.getNext();
                tempSize++;
            }
            return result +"]";
        }
    }

}
```

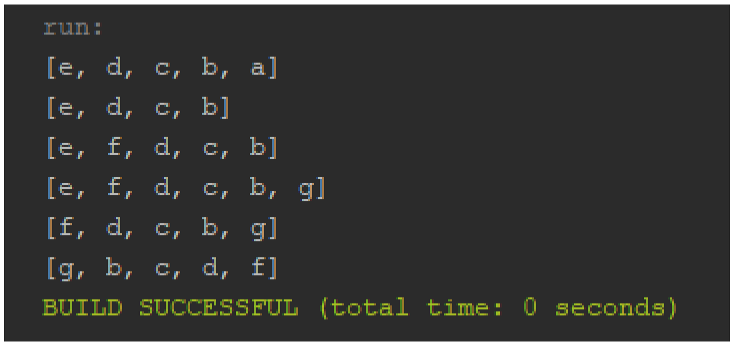
`

LinkedListTest class



```java
/*
Name: Omar Ismail AbdJaleel Alomroy
Maric No: S63955
 */
package Activity2;

/**
 *
 * @author Omar Alomory
 */
public class LinkedListTest {
    public static void main(String[] args) throws Exception {
        LinkedList myLinkedList = new LinkedList();
        myLinkedList.addFirst("a");
        myLinkedList.addFirst("b");
        myLinkedList.addFirst("c");
        myLinkedList.addFirst("d");
        myLinkedList.addFirst("e");
        System.out.println(myLinkedList.printList());

        myLinkedList.removeLast();
        System.out.println(myLinkedList.printList());

        myLinkedList.add(1, "f");
        System.out.println(myLinkedList.printList());

        myLinkedList.addLast("g");
        System.out.println(myLinkedList.printList());

        myLinkedList.removeFirst();
        System.out.println(myLinkedList.printList());

        myLinkedList.reverse();
        System.out.println(myLinkedList.printList());
    }
}
```

Output:



```
run:
[e, d, c, b, a]
[e, d, c, b]
[e, f, d, c, b]
[e, f, d, c, b, g]
[f, d, c, b, g]
[g, b, c, d, f]
BUILD SUCCESSFUL (total time: 0 seconds)
```