

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package LinkedList;

/**
 *
 * @author Vikki
 */
import java.util.Iterator;

public class ListNode11<T> {

    private T data;
    private ListNode next;

    public ListNode11(T data, ListNode next) {
        this.data = data;
        this.next = next;
    }

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public ListNode getNext() {
        return next;
    }

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

    @Override
    public String toString() {
        return data + " ---> ";
    }
}

class LinkedList11<T> {

    private ListNode head;
    protected int size = 0;

    LinkedList11() {
        head = null;
    }

    public void add(T a) {
        ListNode newnode = new ListNode(a, null);
        ListNode current = head;

        if (head == null) {
            head = newnode;
        } else {
            while (current.getNext() != null) {
                current = current.getNext();
            }
        }
    }
}

```

```

        }
        current.setNext(newnode);
    }
}

public void showlist() {
    ListNode current = head;
    while (current != null) {
        System.out.print(current.toString());
        current = current.getNext();
    }
}

public void deleteback() {
    ListNode currentNode = head;
    ListNode previousNode = head;
    if (head != null) {
        if (currentNode.getNext() == null) {
            head = null;
        } else {
            while (currentNode.getNext() != null) {
                previousNode = currentNode;
                currentNode = currentNode.getNext();
            }
            previousNode.setNext(null);
        }
    } else {
        System.out.println("The list is empty.");
    }
}

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

public void deleteNodeByPosition(int index) {
    if (index == 0) {
        deletefront();
    } else if (index == length() - 1) {
        deleteback();
    } else if (index >= length()) {
        System.out.println("Invalid index. No node deleted");
    } else {
        ListNode currentNode = head;
        for (int i = 1; i < index; i++) {
            currentNode = currentNode.getNext();
        }
        ListNode tempNode = currentNode.getNext();
        currentNode.setNext(tempNode.getNext());
        tempNode = null;
    }
}

public void set(T t, int index) {
    if (index == 0) {
        setFront(t);
    } else if (index == length() - 1) {

```

```

        setBack(t);
    } else if (index >= length()) {
        System.out.println("Invalid index. Update failed.");
    } else {
        ListNode currentNode = head;
        for (int i = 1; i <= index; i++) {
            currentNode = currentNode.getNext();
        }
        currentNode.setData(t);
    }
}

public void setFront(T t) {
    if (head != null) {
        head.setData(t);
    } else {
        System.out.println("Invalid update. List is empty");
    }
}

public void setBack(T t) {
    if (head != null) {
        ListNode currentNode = head;
        while (currentNode.getNext() != null) {
            currentNode = currentNode.getNext();
        }
        currentNode.setData(t);
    } else {
        System.out.println("Invalid update. List is empty");
    }
}

public T get(int index) {
    if (index == 0) {
        return getFront();
    } else if (index == length() - 1) {
        return getBack();
    } else if (index >= length()) {
        return null;
    } else {
        ListNode currentNode = head;
        for (int i = 1; i <= index; i++) {
            currentNode = currentNode.getNext();
        }
        return (T) currentNode.getData();
    }
}

public T getFront() {
    if (head != null) {
        return (T) head.getData();
    } else {
        return null;
    }
}

public T getBack() {
    if (head != null) {
        ListNode currentNode = head;
        while (currentNode.getNext() != null) {
            currentNode = currentNode.getNext();
        }
        return (T) currentNode.getData();
    } else {
        return null;
    }
}

```

```

        return null;
    }
}

public int length() {
    int count = 0;
    ListNode currentNode = head;
    while (currentNode != null) {
        currentNode = currentNode.getNext();
        count++;
    }
    return count;
}

public Iterator<T> iterator() {
    return new LinkedListIterator<T>();
}

class LinkedListIterator<T> implements Iterator<T> {

    private int index;
    protected int count = 0;

    public LinkedListIterator() {
        index = 0;
    }

    @Override
    public boolean hasNext() {
        return index < length();
    }

    @Override
    public T next() {
        count++;
        return (T) get(index++);
    }

    public void remove() {
        deleteNodeByPosition(count-1);
    }

}

}

class masdam {

    public static void main(String[] args) {
        LinkedList11<String> a = new LinkedList11<>();

        a.add("ARS");
        a.add("CHE");
        a.add("LEI");
        a.add("MAN");
        a.add("LIV");
        a.add("TOT");
        a.showlist();
        System.out.println("");

        for (Iterator<String> it = a.iterator(); it.hasNext();) {

            String i = it.next();
            if (i.contains("A") == true) {

```

```
        it.remove();
    }

    }
    System.out.println("Remove all the word that consists of the A character
using iterator");
    System.out.print("The updated list consists of ");
    a.showlist();
}
}
```

This study resource was  
shared via CourseHero.com