

LAB # 08

Doubly Linked List implementation of the list ADT

Objective: Implementing doubly linked list, associated operations and LRU technique.

Lab Tasks:

1. Write a program that can insert the records of employees in a link list. The record includes employee's name, designation, department and company name. The program should be able to insert the record as first, last and as middle node in the list and search any record.

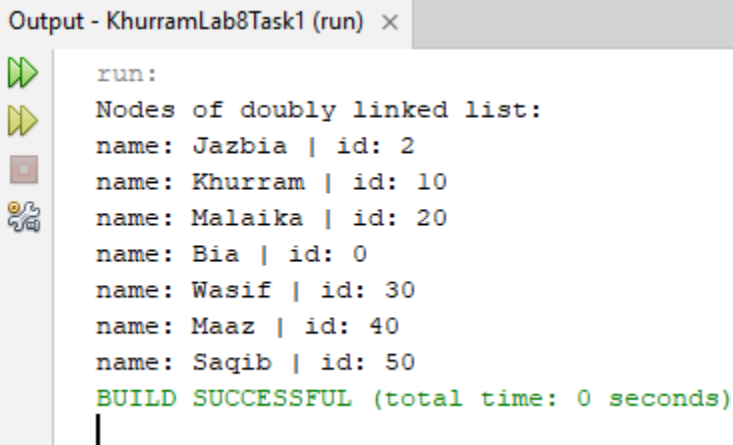
Source Code:

```
public class KhurramLab8Task1 {
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.addNode(10, "Khurram");
        list.addNode(20, "Malaika");
        list.addNode(30, "Wasif");
        list.addNode(40, "Maaz");
        list.addNode(50, "Saqib");
        list.addNodeStart(2, "Jazbia");
        list.insertAt(4, 0, "Bia");
        list.printNodes();
    }
    static class Node {
        int id;
        String name;
        Node previous, next;
        Node(int id, String name) { this.id = id; this.name = name; }
    }
    static class LinkedList {
        Node head, tail;
        void addNode(int id, String name) {
            Node newNode = new Node(id, name);
            if (tail == null) head = tail = newNode;
            else {
                tail.next = newNode;
                newNode.previous = tail;
                tail = newNode;
            }
        }
        void addNodeStart(int id, String name) {
            Node newNode = new Node(id, name);
            if (head == null) head = tail = newNode;
            else {
                newNode.next = head;
                head.previous = newNode;
                head = newNode;
            }
        }
        void insertAt(int pos, int id, String name) {
```

```

    if (pos < 1) return;
    Node newNode = new Node(id, name);
    if (pos == 1) {
        newNode.next = head;
        if (head != null) head.previous = newNode;
        head = newNode;
        return;
    }
    Node temp = head;
    for (int i = 1; temp != null && i < pos - 1; i++) temp = temp.next;
    if (temp != null) {
        newNode.next = temp.next;
        newNode.previous = temp;
        if (temp.next != null) temp.next.previous = newNode;
        temp.next = newNode;
    }
    void printNodes() {
        if (head == null) {
            System.out.println("Doubly linked list is empty.");
            return;
        }
        System.out.println("Nodes of doubly linked list:");
        for (Node current = head; current != null; current = current.next)
            System.out.println("name: " + current.name + " | id: " + current.id);
    }
}

```

Output:


```

Output - KhurramLab8Task1 (run) x
run:
Nodes of doubly linked list:
name: Jazbia | id: 2
name: Khurram | id: 10
name: Malaika | id: 20
name: Bia | id: 0
name: Wasif | id: 30
name: Maaz | id: 40
name: Saqib | id: 50
BUILD SUCCESSFUL (total time: 0 seconds)

```

- Write a program to insert the records of students in a Doubly linked list and insert elements at first and last node using Deque.

Source Code:

```

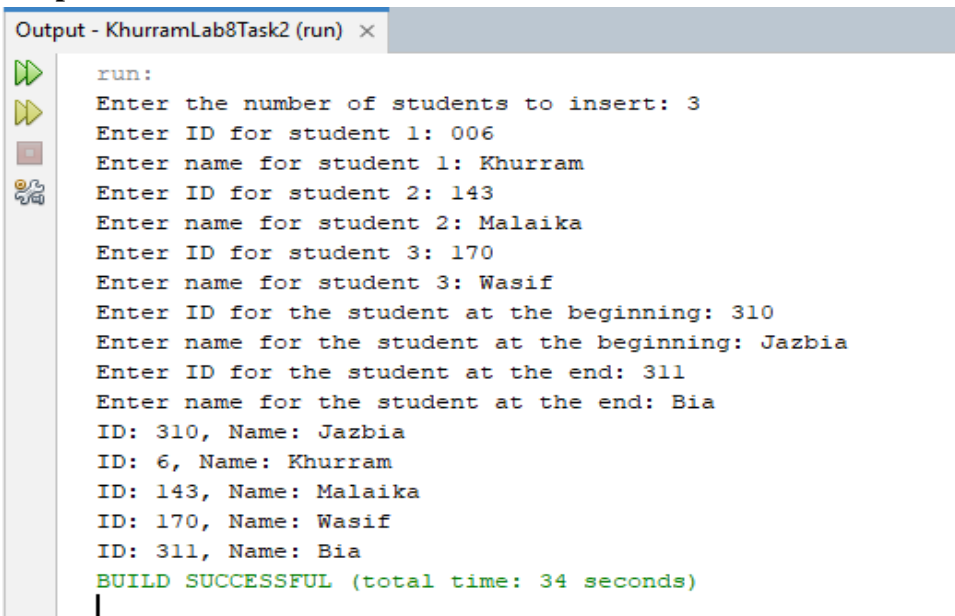
package khurramlab8task2;
import java.util.Deque;
import java.util.LinkedList;
import java.util.Scanner;
public class KhurramLab8Task2 {
    static class Student {
        int id;
        String name;
    }
}

```

```

Student(int id, String name) {
    this.id = id;
    this.name = name;}
@Override
public String toString() {
    return "ID: " + id + ", Name: " + name;}}
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    Deque<Student> studentDeque = new LinkedList<>();
    System.out.print("Enter the number of students to insert: ");
    int numStudents = scanner.nextInt();
    scanner.nextLine();
    for (int i = 0; i < numStudents; i++) {
        System.out.print("Enter ID for student " + (i + 1) + ": ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter name for student " + (i + 1) + ": ");
        studentDeque.addLast(new Student(id, scanner.nextLine()));}
    System.out.print("Enter ID for the student at the beginning: ");
    int firstId = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter name for the student at the beginning: ");
    studentDeque.addFirst(new Student(firstId, scanner.nextLine()));
    System.out.print("Enter ID for the student at the end: ");
    int lastId = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter name for the student at the end: ");
    studentDeque.addLast(new Student(lastId, scanner.nextLine()));
    studentDeque.forEach(System.out::println);
    scanner.close();}}

```

Output:


```

Output - KhurramLab8Task2 (run) x
run:
Enter the number of students to insert: 3
Enter ID for student 1: 006
Enter name for student 1: Khurram
Enter ID for student 2: 143
Enter name for student 2: Malaika
Enter ID for student 3: 170
Enter name for student 3: Wasif
Enter ID for the student at the beginning: 310
Enter name for the student at the beginning: Jazbia
Enter ID for the student at the end: 311
Enter name for the student at the end: Bia
ID: 310, Name: Jazbia
ID: 6, Name: Khurram
ID: 143, Name: Malaika
ID: 170, Name: Wasif
ID: 311, Name: Bia
BUILD SUCCESSFUL (total time: 34 seconds)

```

3. Write a program to create Doubly LinkedList and perform any five operations on it.

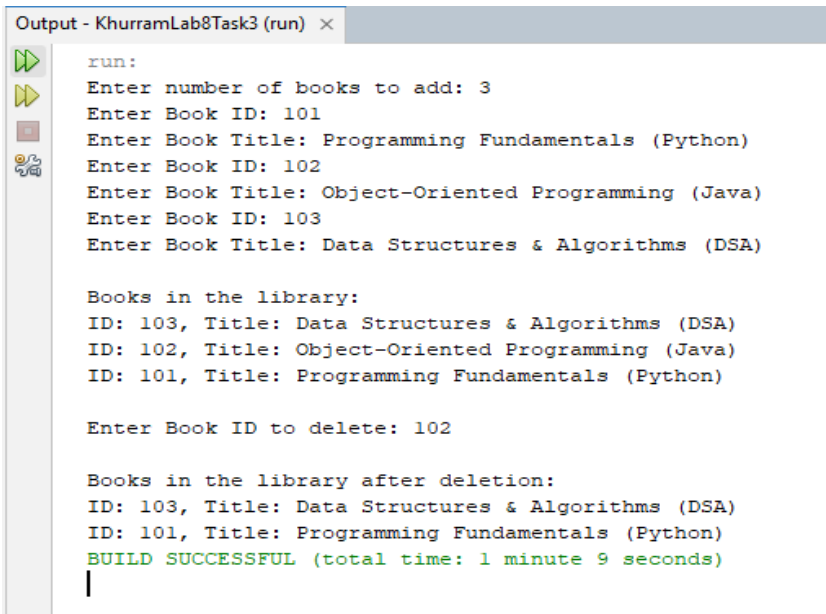
Source Code:

```
package khurramlab8task3;
import java.util.Scanner;
public class KhurramLab8Task3 {
    static class Book {
        int id;
        String title;
        Book next, prev;
        Book(int id, String title) {
            this.id = id;
            this.title = title;}}
    static class DoublyLinkedList {
        Book head, tail;
        void insertAtFirst(int id, String title) {
            Book newBook = new Book(id, title);
            if (head == null) head = tail = newBook;
            else {
                newBook.next = head;
                head.prev = newBook;
                head = newBook;}}
        void display() {
            Book current = head;
            while (current != null) {
                System.out.println("ID: " + current.id + ", Title: " + current.title);
                current = current.next;}}
        void deleteById(int id) {
            Book current = head;
            while (current != null) {
                if (current.id == id) {
                    if (current.prev != null) current.prev.next = current.next;
                    if (current.next != null) current.next.prev = current.prev;
                    if (current == head) head = current.next;
                    if (current == tail) tail = current.prev;
                    return;}
                current = current.next;}
            System.out.println("Book with ID " + id + " not found.");}}
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DoublyLinkedList library = new DoublyLinkedList();
        System.out.print("Enter number of books to add: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        for (int i = 0; i < n; i++) {
            System.out.print("Enter Book ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
```

```

        System.out.print("Enter Book Title: ");
        String title = scanner.nextLine();
        library.insertAtFirst(id, title);}
    System.out.println("\nBooks in the library:");
    library.display();
    System.out.print("\nEnter Book ID to delete: ");
    int idToDelete = scanner.nextInt();
    library.deleteById(idToDelete);
    System.out.println("\nBooks in the library after deletion:");
    library.display();
    scanner.close();}}

```

Output:


```

Output - KhurramLab8Task3 (run) x
run:
Enter number of books to add: 3
Enter Book ID: 101
Enter Book Title: Programming Fundamentals (Python)
Enter Book ID: 102
Enter Book Title: Object-Oriented Programming (Java)
Enter Book ID: 103
Enter Book Title: Data Structures & Algorithms (DSA)

Books in the library:
ID: 103, Title: Data Structures & Algorithms (DSA)
ID: 102, Title: Object-Oriented Programming (Java)
ID: 101, Title: Programming Fundamentals (Python)

Enter Book ID to delete: 102

Books in the library after deletion:
ID: 103, Title: Data Structures & Algorithms (DSA)
ID: 101, Title: Programming Fundamentals (Python)
BUILD SUCCESSFUL (total time: 1 minute 9 seconds)
|

```

Home Task

1. Write a program to create Unsorted LinkedList as well as Sorted LinkedList.

Source Code:

```

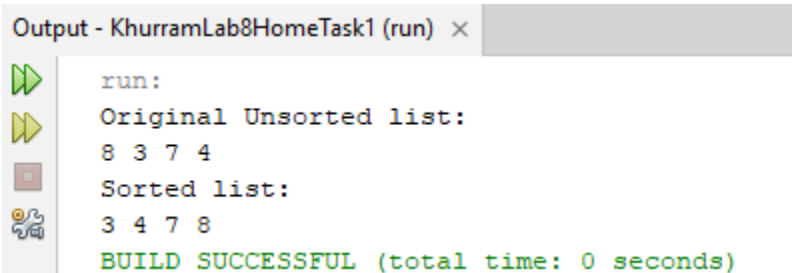
package khurramlab8hometask1;
public class KhurramLab8HomeTask1 {
    static class SortList {
        static class Node {
            int data;
            Node next;
            Node(int data) { this.data = data; }}
        Node head, tail;
        void addNode(int data) {
            Node newNode = new Node(data);
            if (head == null) head = tail = newNode;
            else { tail.next = newNode; tail = newNode; }}
        void sortList() {

```

```
if (head == null) return;
Node current = head;
while (current != null) {
    Node index = current.next;
    while (index != null) {
        if (current.data > index.data) {
            int temp = current.data;
            current.data = index.data;
            index.data = temp;
        }
        index = index.next;
    }
    current = current.next;
}

void display() {
    if (head == null) { System.out.println("List is empty"); return; }
    Node current = head;
    while (current != null) {
        System.out.print(current.data + " ");
        current = current.next;
    }
    System.out.println();
}

public static void main(String[] args) {
    SortList sList = new SortList();
    sList.addNode(8);
    sList.addNode(3);
    sList.addNode(7);
    sList.addNode(4);
    System.out.println("Original Unsorted list: ");
    sList.display();
    sList.sortList();
    System.out.println("Sorted list: ");
    sList.display();
}
```

Output:

```
Output - KhurramLab8HomeTask1 (run) ×
run:
Original Unsorted list:
8 3 7 4
Sorted list:
3 4 7 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

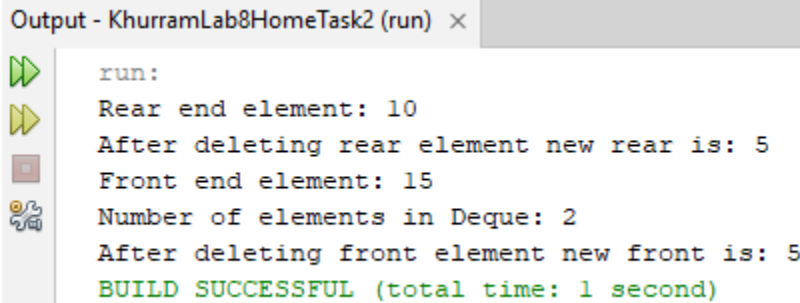
2. Write a program to create LinkedList using Deque and apply any five methods of Deque interface.

Source Code:

```
package khurramlab8hometask2;
import java.util.*;
public class KhurramLab8HomeTask2 {
    static class Deque {
```

```
static class Node {
    int data;
    Node prev, next;
    Node(int data) { this.data = data; }}
Node front, rear;
int size;
Deque() { front = rear = null; size = 0; }
boolean isEmpty() { return front == null; }
int size() { return size; }
void insertFront(int data) {
    Node newNode = new Node(data);
    if (isEmpty()) front = rear = newNode;
    else {
        newNode.next = front;
        front.prev = newNode;
        front = newNode; }
    size++; }
void insertRear(int data) {
    Node newNode = new Node(data);
    if (isEmpty()) front = rear = newNode;
    else {
        newNode.prev = rear;
        rear.next = newNode;
        rear = newNode; }
    size++; }
void deleteFront() {
    if (isEmpty()) System.out.println("UnderFlow");
    else {
        front = front.next;
        if (front == null) rear = null;
        else front.prev = null;
        size--; } }
void deleteRear() {
    if (isEmpty()) System.out.println("UnderFlow");
    else {
        rear = rear.prev;
        if (rear == null) front = null;
        else rear.next = null;
        size--; } }
int getFront() { return isEmpty() ? -1 : front.data; }
int getRear() { return isEmpty() ? -1 : rear.data; }
void erase() { front = rear = null; size = 0; }
public static void main(String[] args) {
    Deque dq = new Deque();
    dq.insertRear(5);
    dq.insertRear(10);
    System.out.println("Rear end element: " + dq.getRear());
    dq.deleteRear();
    System.out.println("After deleting rear element new rear is: " + dq.getRear());
}
```

```
dq.insertFront(15);
System.out.println("Front end element: " + dq.getFront());
System.out.println("Number of elements in Deque: " + dq.size());
dq.deleteFront();
System.out.println("After deleting front element new front is: " + dq.getFront());}}
```

Output:

```
Output - KhurramLab8HomeTask2 (run) ×
run:
Rear end element: 10
After deleting rear element new rear is: 5
Front end element: 15
Number of elements in Deque: 2
After deleting front element new front is: 5
BUILD SUCCESSFUL (total time: 1 second)
```

3. You are managing a **playlist system** where each song is represented by a node in a **doubly linked list**. Each node contains:

- **Song ID** (integer)
- **Song Title** (string)

Your task is to:

- a) **Insert** a song at the **end** of the playlist.
- b) **Display** the playlist from **start to end**.
- c) **Reverse** the playlist and display it again.

Source Code:

```
package khurramlab8hometask3;

public class KhurramLab8HomeTask3 {

    static class DoublyLinkedList {

        static class Node {

            int songID;

            String songTitle;

            Node prev, next;

            Node(int id, String title) {

                songID = id;
```



```
        songTitle = title;}}

private Node head, tail;

public void insertAtEnd(int id, String title) {

    Node newNode = new Node(id, title);

    if (tail == null) head = tail = newNode;

    else {

        tail.next = newNode;

        newNode.prev = tail;

        tail = newNode;}}

public void displayPlaylist() {

    for (Node current = head; current != null; current = current.next)

        System.out.println("ID: " + current.songID + ", Title: " + current.songTitle);}

public void reverseAndDisplayPlaylist() {

    for (Node current = tail; current != null; current = current.prev)

        System.out.println("ID: " + current.songID + ", Title: " + current.songTitle);}}

public static void main(String[] args) {

    DoublyLinkedList playlist = new DoublyLinkedList();

    playlist.insertAtEnd(1, "All The Stars");

    playlist.insertAtEnd(2, "Dream On");

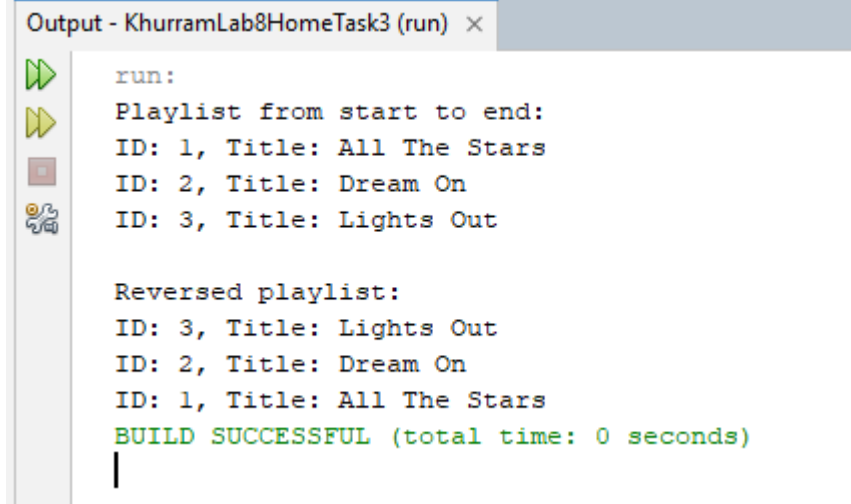
    playlist.insertAtEnd(3, "Lights Out");

    System.out.println("Playlist from start to end:");

    playlist.displayPlaylist();

    System.out.println("\nReversed playlist:");

    playlist.reverseAndDisplayPlaylist();}}
```

Output:

```
Output - KhurramLab8HomeTask3 (run) ×
run:
Playlist from start to end:
ID: 1, Title: All The Stars
ID: 2, Title: Dream On
ID: 3, Title: Lights Out

Reversed playlist:
ID: 3, Title: Lights Out
ID: 2, Title: Dream On
ID: 1, Title: All The Stars
BUILD SUCCESSFUL (total time: 0 seconds)
|
```