LAB 08

Doubly Linked List

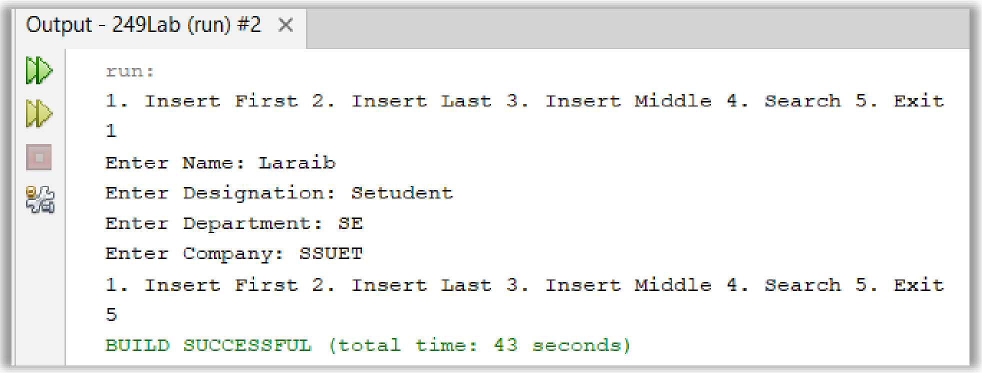
Lab Tasks

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 :

```
import java.util.LinkedList;
class Employee {
    String name, designation, department, company;

    Employee(String name, String designation, String department, String
company) {
        this.name = name;
        this.designation = designation;
        this.department = department;
        this.company = company;
    }public String toString() {
   return name + ", " + designation + ", " + department + ", " + company;
    }
}
public class lab249 {
    public static void main(String[] args) {
        LinkedList<Employee> employees = new LinkedList<>();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("1. Insert First 2. Insert Last 3. Insert
Middle 4. Search 5. Exit");
            int choice = sc.nextInt();
            sc.nextLine();
            if (choice == 5) break;

            if (choice >= 1 && choice <= 3) {
                System.out.print("Enter Name: ");
                String name = sc.nextLine();
                System.out.print("Enter Designation: ");
                String designation = sc.nextLine();
                System.out.print("Enter Department: ");
                String department = sc.nextLine();
                System.out.print("Enter Company: ");
                String company = sc.nextLine();
     Employee emp = new Employee(name, designation, department, company);
                if (choice == 1) employees.addFirst(emp);
                else if (choice == 2) employees.addLast(emp);
                else {
                    System.out.print("Enter Position: ");
                    int pos = sc.nextInt();
    if (pos >= 0 && pos < employees.size()) employees.add(pos, emp);
                    else System.out.println("Invalid position");
                }
            } else if (choice == 4) {
                System.out.print("Enter Name to Search: ");
                String searchName = sc.nextLine();
                boolean found = false;
                for (Employee e : employees) {
                    if (e.name.equalsIgnoreCase(searchName)) {
                        System.out.println("Found: " + e);
                        found = true;
                        break;
                    }
                }
                if (!found) System.out.println("Record not found");
            }
        }
    }
}
```

Output :



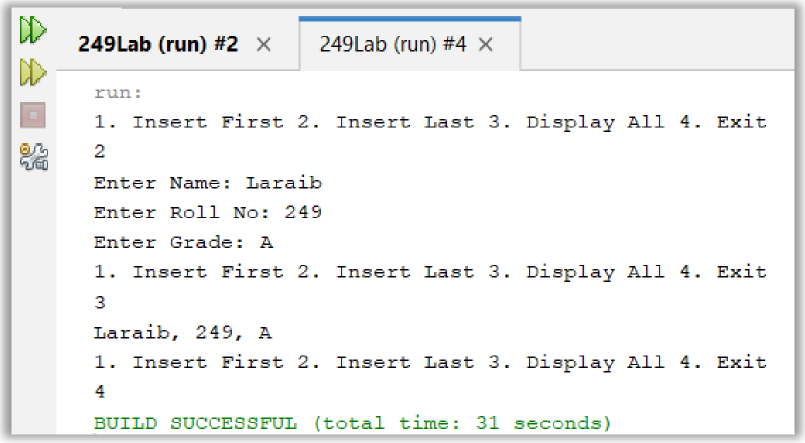
```
run:
1. Insert First 2. Insert Last 3. Insert Middle 4. Search 5. Exit
1
Enter Name: Laraib
Enter Designation: Setudent
Enter Department: SE
Enter Company: SSUET
1. Insert First 2. Insert Last 3. Insert Middle 4. Search 5. Exit
5
BUILD SUCCESSFUL (total time: 43 seconds)
```

2. 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 :

```
import java.util.Deque;
import java.util.LinkedList;
import java.util.Scanner;
class Student {
    String name, rollNo, grade;

    Student(String name, String rollNo, String grade) {
        this.name = name;
        this.rollNo = rollNo;
        this.grade = grade;
    }public String toString() {
        return name + ", " + rollNo + ", " + grade;
    }
}
public class DoublyLinkedListStudents {
    public static void main(String[] args) {
        Deque<Student> students = new LinkedList<>();
        Scanner sc = new Scanner(System.in);
        while (true) {
System.out.println("1. Insert First 2. Insert Last 3. Display All 4.
Exit");
            int choice = sc.nextInt();
            sc.nextLine();
            if (choice == 4) break;
            if (choice == 1 || choice == 2) {
                System.out.print("Enter Name: ");
                String name = sc.nextLine();
                System.out.print("Enter Roll No: ");
                String rollNo = sc.nextLine();
                System.out.print("Enter Grade: ");
                String grade = sc.nextLine();
                Student student = new Student(name, rollNo, grade);

                if (choice == 1) students.addFirst(student);
                else students.addLast(student);
            } else if (choice == 3) {
         if (students.isEmpty()) System.out.println("No records found.");
                else for (Student s : students) System.out.println(s);
            }
        }
    }
}
```

Output :



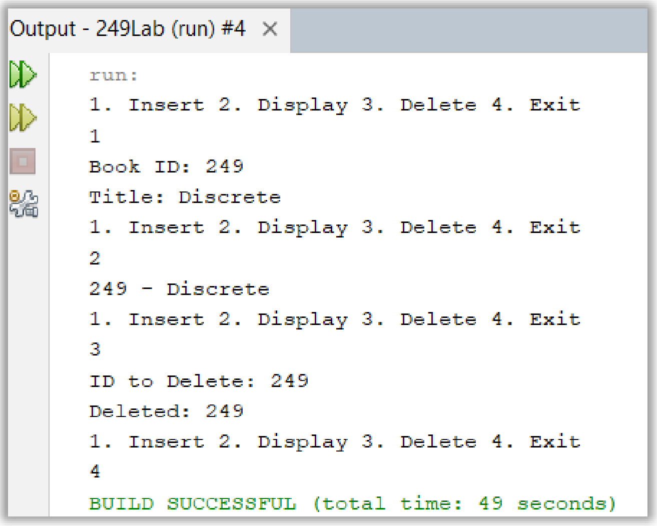
```
run:
1. Insert First 2. Insert Last 3. Display All 4. Exit
2
Enter Name: Laraib
Enter Roll No: 249
Enter Grade: A
1. Insert First 2. Insert Last 3. Display All 4. Exit
3
Laraib, 249, A
1. Insert First 2. Insert Last 3. Display All 4. Exit
4
BUILD SUCCESSFUL (total time: 31 seconds)
```

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

Source Code :

```
class Book {
    int bookID;
    String title;
    Book prev, next;
    Book(int bookID, String title) {
        this.bookID = bookID;
        this.title = title;
    }}public class LibrarySystem {
    private Book head;

    public void insertAtBeginning(int bookID, String title) {
        Book newBook = new Book(bookID, title);
        if (head != null) {
            newBook.next = head;
            head.prev = newBook;
        }head = newBook;
    }public void displayBooks() {
        Book temp = head;
        while (temp != null) {
            System.out.println(temp.bookID + " - " + temp.title);
            temp = temp.next;
        }
    }public void deleteByBookID(int bookID) {
        Book temp = head;
        while (temp != null && temp.bookID != bookID) temp = temp.next;
        if (temp != null) {
            if (temp.prev != null) temp.prev.next = temp.next;
            if (temp.next != null) temp.next.prev = temp.prev;
            if (temp == head) head = temp.next;
            System.out.println("Deleted: " + bookID);
        } else System.out.println("Book not found.");
    }public static void main(String[] args) {
        LibrarySystem library = new LibrarySystem();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("1. Insert 2. Display 3. Delete 4. Exit");
            int choice = sc.nextInt();
            if (choice == 4) break;
            if (choice == 1) {
                System.out.print("Book ID: ");
                int id = sc.nextInt();
                sc.nextLine();
                System.out.print("Title: ");
                String title = sc.nextLine();
                library.insertAtBeginning(id, title);
            } else if (choice == 2) library.displayBooks();
            else if (choice == 3) {
                System.out.print("ID to Delete: ");
                library.deleteByBookID(sc.nextInt());
            }
        }
           }
}
```

Output :



Output - 249Lab (run) #4

```
run:
1. Insert 2. Display 3. Delete 4. Exit
1
Book ID: 249
Title: Discrete
1. Insert 2. Display 3. Delete 4. Exit
2
249 - Discrete
1. Insert 2. Display 3. Delete 4. Exit
3
ID to Delete: 249
Deleted: 249
1. Insert 2. Display 3. Delete 4. Exit
4
BUILD SUCCESSFUL (total time: 49 seconds)
```

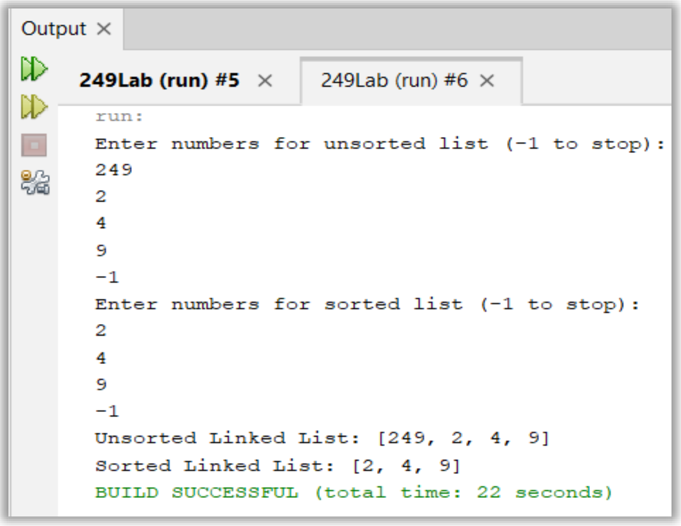
Home Task

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

Source Code :

```
public class lab249 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<Integer> unsortedList = new LinkedList<>();
        LinkedList<Integer> sortedList = new LinkedList<>();

        System.out.println("Enter numbers for unsorted list (-1 to stop):");
        int num;
        while ((num = sc.nextInt()) != -1) {
            unsortedList.add(num);
        }
        System.out.println("Enter numbers for sorted list (-1 to stop):");
        while ((num = sc.nextInt()) != -1) {
            int i = 0;
            while (i < sortedList.size() && sortedList.get(i) < num) i++;
            sortedList.add(i, num);
        }
        System.out.println("Unsorted Linked List: " + unsortedList);
        System.out.println("Sorted Linked List: " + sortedList);
        sc.close();
    }
}
```

Output :



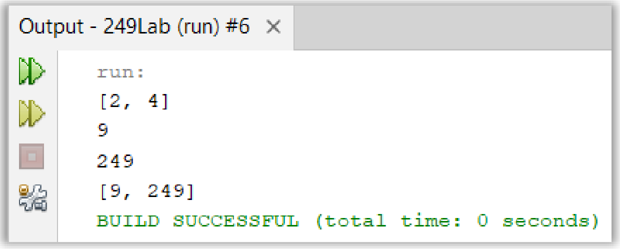
```
run:
Enter numbers for unsorted list (-1 to stop):
249
2
4
9
-1
Enter numbers for sorted list (-1 to stop):
2
4
9
-1
Unsorted Linked List: [249, 2, 4, 9]
Sorted Linked List: [2, 4, 9]
BUILD SUCCESSFUL (total time: 22 seconds)
```

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

Source Code :

```
public class lab249 {
    public static void main(String[] args) {
        Deque<Integer> deque = new LinkedList<>();
        deque.addFirst(2);
        deque.addLast(4);
        System.out.println(deque);

        deque.removeFirst();
        deque.removeLast();

        deque.addFirst(9);
        deque.addLast(249);

        System.out.println(deque.peekFirst());
        System.out.println(deque.peekLast());

        System.out.println(deque);
    }
}
```

Output :



Output - 249Lab (run) #6

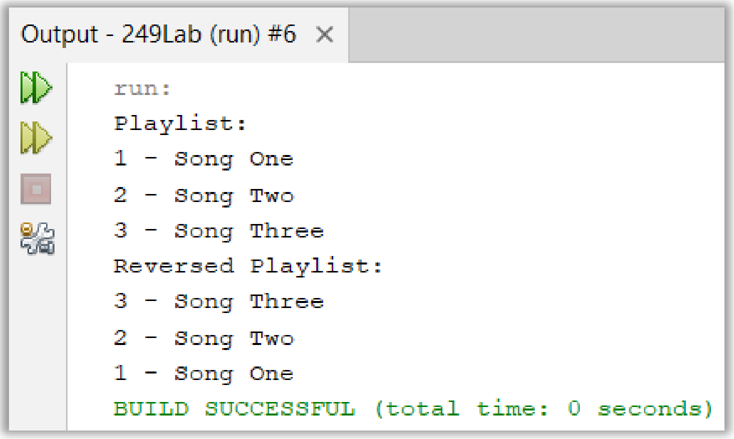
```
run:
[2, 4]
9
249
[9, 249]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

Source Code :

```
    class Song {
     int songID;
     String title;
     Song prev, next;

     Song(int songID, String title) {
         this.songID = songID;
         this.title = title;
     }
}
public class lab249 {
     private Song head, tail;

     public void insertAtEnd(int songID, String title) {
         Song newSong = new Song(songID, title);
         if (tail == null) head = tail = newSong;
         else { tail.next = newSong; newSong.prev = tail; tail = newSong; }
     }

     public void display() {
         for (Song temp = head; temp != null; temp = temp.next)
System.out.println(temp.songID + " - " + temp.title);
     }
     public void reverse() {
         Song temp = null, current = head;
         while (current != null) { temp = current.prev; current.prev = 
current.next; current.next = temp; current = current.prev; }
         if (temp != null) head = temp.prev;
     }

     public static void main(String[] args) {
         lab249 p = new lab249();
         p.insertAtEnd(1, "Song One");
         p.insertAtEnd(2, "Song Two");
         p.insertAtEnd(3, "Song Three");

         System.out.println("Playlist:");
         p.display();

         p.reverse();

         System.out.println("Reversed Playlist:");
         p.display();
     }
}
```

Output :



```
run:
Playlist:
1 - Song One
2 - Song Two
3 - Song Three
Reversed Playlist:
3 - Song Three
2 - Song Two
1 - Song One
BUILD SUCCESSFUL (total time: 0 seconds)
```