Lab 4: Circular Double Linked-List

Objective(s)

1- Create Circular Double Linked-List in Java.
2- Deal with Circular Double Linked-List in case of: insertion, Deletion, searching .

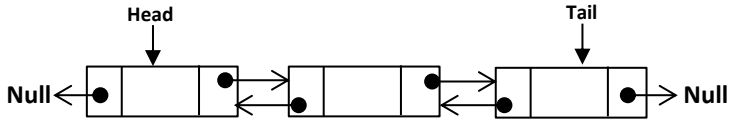
Tool(s)/Software

Java programming language with NetBeans IDE.

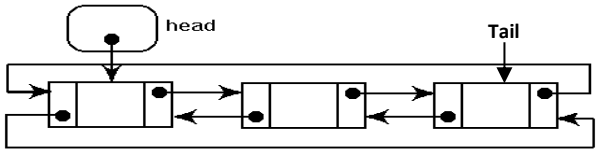
Description:

The **Double Linked List** has the same Node structure but every Node has 2 pointers for the next and previous Node.

- The **Double Linked List** with Head.Prev = null and Tail.next = null are called: **Double Linked-List.**



- The **Double Linked List** with Head.Prev = **tail** and Tail.next = **head** are called: **Circular Double Linked-List.**



A. How to Create Circular Double Linked-List in Java:

There are **3-steps** approach to create Circular Double Linked-List in Java
Step 1: Declare class for the **Node** – forming the structure of the node.

```
private class Node {
    private int courseID;
    private String courseName;
    private Node next;
    private Node prev;
    public Node(int courseID, String courseName, Node next, Node prev)
    {...6 lines }
    public int getCourseID() {...3 lines }
    public String getCourseName() {...3 lines }
    public Node getNext() {...3 lines }
    public Node getPrev() {...3 lines }
    public void setCourseID(int courseID) {...3 lines }
    public void setCourseName(String courseName) {...3 lines }
    public void setNext(Node next) {...3 lines }
    public void setPrev(Node prev) {...3 lines }
```

Step 2: Declare the **CircularDoubleLinkedList** class that includes the Node class.

```
public class CircularDoubleLinkedList {
    private class Node {...38 lines }
    Node head=null;
    Node tail=null;
    int size=0;
    public CircularDoubleLinkedList() {...2 lines }
    public int firstID() {...3 lines }
    public int lastID() {...3 lines }
    public int getSize() {...3 lines }
    public boolean isEmpty() {...3 lines }
    public void forwardTraversing() {...12 lines }
    public void backwardTraversing() {...12 lines }
    public void addFirst(int id, String name) {...15 lines }
    public void addLast(int id, String name) {...18 lines }
    public void find(int id) {...23 lines }
    public void insertAtPos(int id, String name,int pos) {...22 lines }
    public void deleteFirst() {...12 lines }
    public void deleteLast() {...12 lines }
    public void deleteNode(int id) {...38 lines }
```

Step 3: Define the object of **CircularDoubleLinkedList** class:

```
CircularDoubleLinkedList l=new CircularDoubleLinkedLis();
l.addFirst(3,"CS310");
l.addFirst(2,"CS321");
```

B. Circular Double Linked-List Operations:
   1. Traversing Circular Double Linked-List (*Forward*, *Backward*).
   2. Searching in Circular Double Linked-List
   3. Insertion in Circular Double Linked-List (*addFirst, addLast, addAtPos*)
   4. Deletion from Circular Double Linked-List (*deleteFirst, deleteLast, deleteNode*)

Tasks/Assignments(s)

1. Create CircularDoubleLinkedList class. Each node should have course ID and name in the data section. Apply all the following operations:
   - **Display:** *forwardDisplay*, *backwardDisplay.*
   - **Adition:** *addFirst, addLast, addAtPos.*
   - **Deletion:** *deleteFirst, deleteLast, deleteNod.*
   - **FindNode**: to find a node with specific given ID.

2. Add ***findDuplicate*** method to CircularDoubleLinkedList class to find and display courses with DUPLICATE IDs.

   Sample output after adding some nodes to the list and calling ***findDuplicate*** method:

```
run:
ID:333 - Name:Islamic
ID:512 - Name:Math2
ID:616 - Name:Digital Hardware
ID:512 - Name:Data Structures
ID:444 - Name:OOP2
ID:707 - Name:OOP2
ID:121 - Name:OOP1
ID:333 - Name:Math1
ID:444 - Name:Physics
--------------------------
Duplicate ID: 333 For Courses: Islamic , Math1
Duplicate ID: 512 For Courses: Math2 , Data Structures
Duplicate ID: 444 For Courses: OOP2 , Physics
BUILD SUCCESSFUL (total time: 0 seconds)
```

Deliverables(s)

You are required to implement and deliver a Java program as described in the previous section.