Data Structures and Algorithms

CSE2001

Lab - 2 - Assignment - 1

Yashwanth Reddy
19BCE7362
Date- 01stJuly2021

Linked List

```
import java.util.*;
class Node{
int data;
Node next;
}
class Linkedlist{
   static Node head=null,tail=null;
   static Scanner sc=new Scanner(System.in);
   static void create()
   {

      System.out.println("Enter total number of nodes");
      int n=sc.nextInt();
      System.out.println("Enter "+n+" data values");
      for(int i=0;i<n;i++)
      {
      Node temp=new Node();

      temp.data=sc.nextInt();
      temp.next=null;
```

```
    if(head==null)
        head=tail=temp;
    else{
        tail.next=temp;
    tail=temp;}
    }
}
static void display()
{
    if(head==null)
        System.out.println("List is Empty");
    else
    {
    Node temp;
    temp=head;
    while(temp!=null)
        {
            System.out.print(temp.data+"  ");
            temp=temp.next;
        }
        System.out.println();
    }
}

static void Ins_Begin()
{
    Node temp=new Node();
    System.out.println("Enter data");
    temp.data=sc.nextInt();
    temp.next=null;
    if(head==null)
```

```
            head=tail=temp;
        else
        {
            temp.next=head;
            head=temp;
        }
    }
    static void Ins_End()
    {
        Node temp=new Node();
        System.out.println("Enter data");
        temp.data=sc.nextInt();
        temp.next=null;
        if(head==null)
            head=tail=temp;
        else
        {
            tail.next=temp;
            tail=temp;
        }
    }
    static void Ins_Pos(int p)
    {
        Node c=head;
        int count=1;
        while(c!=null)
        {
            if(count==p-1)
                break;
            c=c.next;
            count++;
```

```
    }
    if(c!=null)
    {
        Node temp=new Node();
        System.out.println("Enter data");
        temp.data=sc.nextInt();
        temp.next=c.next;
        c.next=temp;
        if(tail==c)
            tail=temp;
    }
    else
    {
        System.out.println("Invalid position");
    }
}
static void Delete_head()
{
    Node temp=head;
    if(head==tail)
        head=tail=null;
    else
    {
        head=head.next;
        temp.next=null;
        temp=null;
}

}
static void Delete_Pos(int p)
{
```

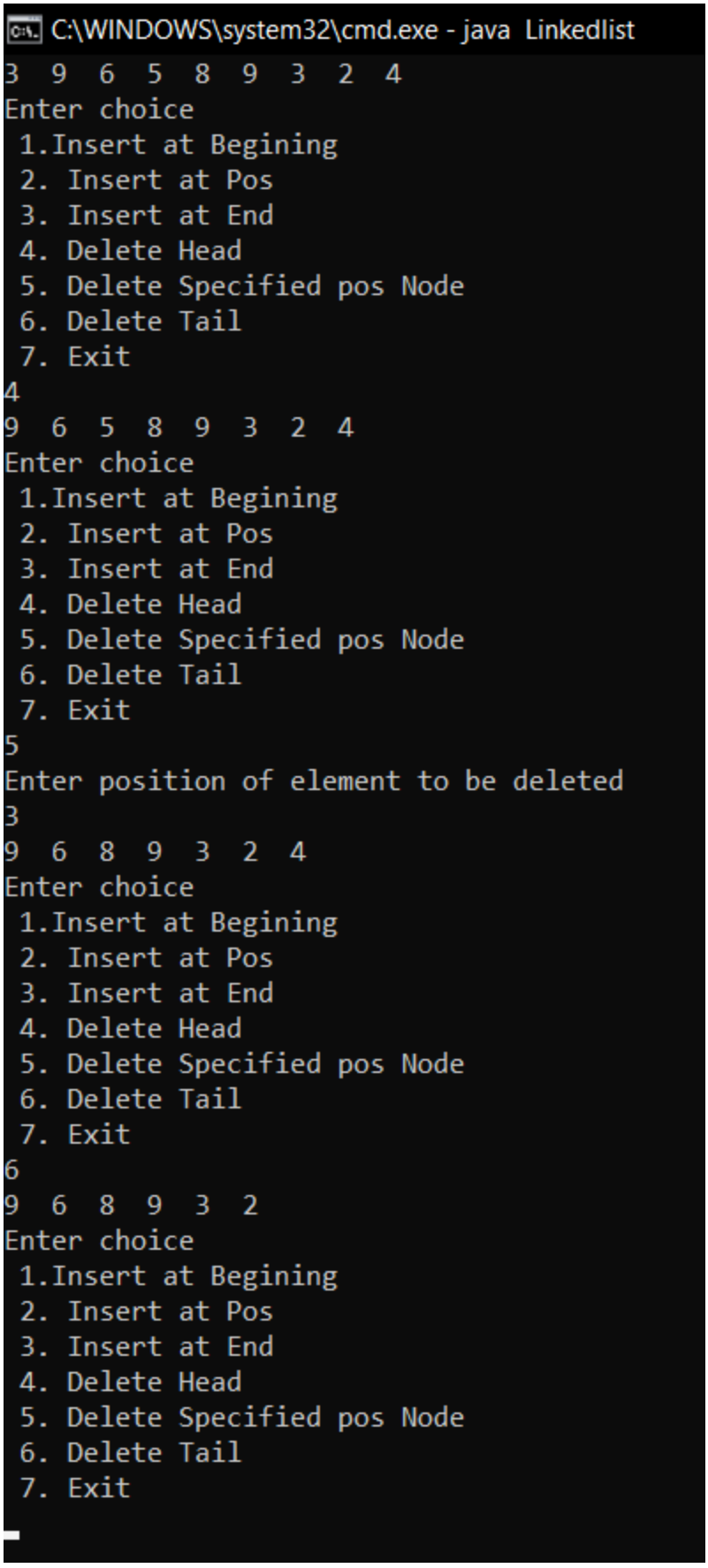
```
    Node c=head,prev=null;
    int count=1;
    while(c!=null)
    {
        if(count==p)
            break;
        prev=c;
        c=c.next;
        count++;
    }
    if(c!=null)
    {
        if(head==tail)
        head=tail=null;
        else
        {
        prev.next=c.next;
            if(c==tail)
                tail=prev;
        c=null;
        }
    }
    else
    {
        System.out.println("Invalid position");
    }
}
static void Delete_tail()
{
    Node temp=head;
    if(head==tail)
```

```
            head=tail=null;
        else
        {
            while(temp.next!=tail)
            {
                temp=temp.next;
            }
            tail=temp;
            tail.next=null;
        }


    }

public static void main(String args[]){

    create();
    display();
    int ch,p;
    while(true)
    {
        System.out.println("Enter choice \n 1.Insert at Begining \n 2. Insert at Pos\n 3. Insert at End \n 4. Delete Head \n 5. Delete Specified pos Node \n 6. Delete Tail  \n 7. Exit");
        ch=sc.nextInt();
        switch(ch)
        {
            case 1: Ins_Begin();
                    display();
                    break;
            case 2: System.out.println("Enter position of new element");
```

```
                p=sc.nextInt();
                Ins_Pos(p);
                display();
                break;
        case 3: Ins_End();
                display();
                break;
        case 4: Delete_head();
                display();
                break;
        case 5: System.out.println("Enter position of element to be deleted");
                p=sc.nextInt();
                Delete_Pos(p);
                display();
                break;
        case 6: Delete_tail();
                display();
                break;

        case 7:System.exit(0);
                break;
        default: System.out.println("Invalid choice");
                break;
      }
   }
}
}
```

Output

```
C:\WINDOWS\system32\cmd.exe - java  Linkedlist

C:\Users\yashw\Desktop\Summer\Labs>javac Linkedlist.java --release 8

C:\Users\yashw\Desktop\Summer\Labs>java Linkedlist
Enter total number of nodes
6
Enter 6 data values
6
5
8
9
3
2
6  5  8  9  3  2
Enter choice
 1.Insert at Begining
 2. Insert at Pos
 3. Insert at End
 4. Delete Head
 5. Delete Specified pos Node
 6. Delete Tail
 7. Exit
1
Enter data
3
3  6  5  8  9  3  2
Enter choice
 1.Insert at Begining
 2. Insert at Pos
 3. Insert at End
 4. Delete Head
 5. Delete Specified pos Node
 6. Delete Tail
 7. Exit
2
Enter position of new element
2
Enter data
9
3  9  6  5  8  9  3  2
Enter choice
 1.Insert at Begining
 2. Insert at Pos
 3. Insert at End
 4. Delete Head
 5. Delete Specified pos Node
 6. Delete Tail
 7. Exit
3
Enter data
4
3  9  6  5  8  9  3  2  4
```


C:\WINDOWS\system32\cmd.exe - java Linkedlist
3 9 6 5 8 9 3 2 4
Enter choice
1.Insert at Begining
2. Insert at Pos
3. Insert at End
4. Delete Head
5. Delete Specified pos Node
6. Delete Tail
7. Exit
4
9 6 5 8 9 3 2 4
Enter choice
1.Insert at Begining
2. Insert at Pos
3. Insert at End
4. Delete Head
5. Delete Specified pos Node
6. Delete Tail
7. Exit
5
Enter position of element to be deleted
3
9 6 8 9 3 2 4
Enter choice
1.Insert at Begining
2. Insert at Pos
3. Insert at End
4. Delete Head
5. Delete Specified pos Node
6. Delete Tail
7. Exit
6
9 6 8 9 3 2
Enter choice
1.Insert at Begining
2. Insert at Pos
3. Insert at End
4. Delete Head
5. Delete Specified pos Node
6. Delete Tail
7. Exit

Double Linked List

```
import java.util.*;


class Node
{
   protected int data;
   protected Node next, prev;


   public Node()
   {
      next = null;
      prev = null;
      data = 0;
   }

   public Node(int d, Node n, Node p)
   {
      data = d;
      next = n;
      prev = p;
   }

   public void setLinkNext(Node n)
   {
      next = n;
   }

   public void setLinkPrev(Node p)
   {
      prev = p;
```

```
    }

    public Node getLinkNext()
    {
        return next;
    }

    public Node getLinkPrev()
    {
        return prev;
    }

    public void setData(int d)
    {
        data = d;
    }

    public int getData()
    {
        return data;
    }
}


class linkedList
{
    protected Node start;
    protected Node end ;
    public int size;


    public linkedList()
    {
```

```
    start = null;
    end = null;
    size = 0;
}

public boolean isEmpty()
{
    return start == null;
}

public int getSize()
{
    return size;
}

public void insertAtStart(int val)
{
    Node nptr = new Node(val, null, null);
    if(start == null)
    {
        start = nptr;
        end = start;
    }
    else
    {
        start.setLinkPrev(nptr);
        nptr.setLinkNext(start);
        start = nptr;
    }
    size++;
}
public void insertAtEnd(int val)
{
```

```
    Node nptr = new Node(val, null, null);
    if(start == null)
    {
        start = nptr;
        end = start;
    }
    else
    {
        nptr.setLinkPrev(end);
        end.setLinkNext(nptr);
        end = nptr;
    }
    size++;
}
public void insertAtPos(int val , int pos)
{
    Node nptr = new Node(val, null, null);
    if (pos == 1)
    {
        insertAtStart(val);
        return;
    }
    Node ptr = start;
    for (int i = 2; i <= size; i++)
    {
        if (i == pos)
        {
            Node tmp = ptr.getLinkNext();
            ptr.setLinkNext(nptr);
            nptr.setLinkPrev(ptr);
            nptr.setLinkNext(tmp);
            tmp.setLinkPrev(nptr);
        }
```

```
            ptr = ptr.getLinkNext();
        }
        size++ ;
    }
    public void deleteAtPos(int pos)
    {
        if (pos == 1)
        {
            if (size == 1)
            {
                start = null;
                end = null;
                size = 0;
                return;
            }
            start = start.getLinkNext();
            start.setLinkPrev(null);
            size--;
            return ;
        }
        if (pos == size)
        {
            end = end.getLinkPrev();
            end.setLinkNext(null);
            size-- ;
        }
        Node ptr = start.getLinkNext();
        for (int i = 2; i <= size; i++)
        {
            if (i == pos)
            {
                Node p = ptr.getLinkPrev();
                Node n = ptr.getLinkNext();
```

```
            p.setLinkNext(n);
            n.setLinkPrev(p);
            size-- ;
            return;
        }
        ptr = ptr.getLinkNext();
    }
}

public void display()
{
    System.out.print("\nDoubly Linked List = ");
    if (size == 0)
    {
        System.out.print("empty\n");
        return;
    }
    if (start.getLinkNext() == null)
    {
        System.out.println(start.getData() );
        return;
    }
    Node ptr = start;
    System.out.print(start.getData()+ " <-> ");
    ptr = start.getLinkNext();
    while (ptr.getLinkNext() != null)
    {
        System.out.print(ptr.getData()+ " <-> ");
        ptr = ptr.getLinkNext();
    }
    System.out.print(ptr.getData()+ "\n");
}
```

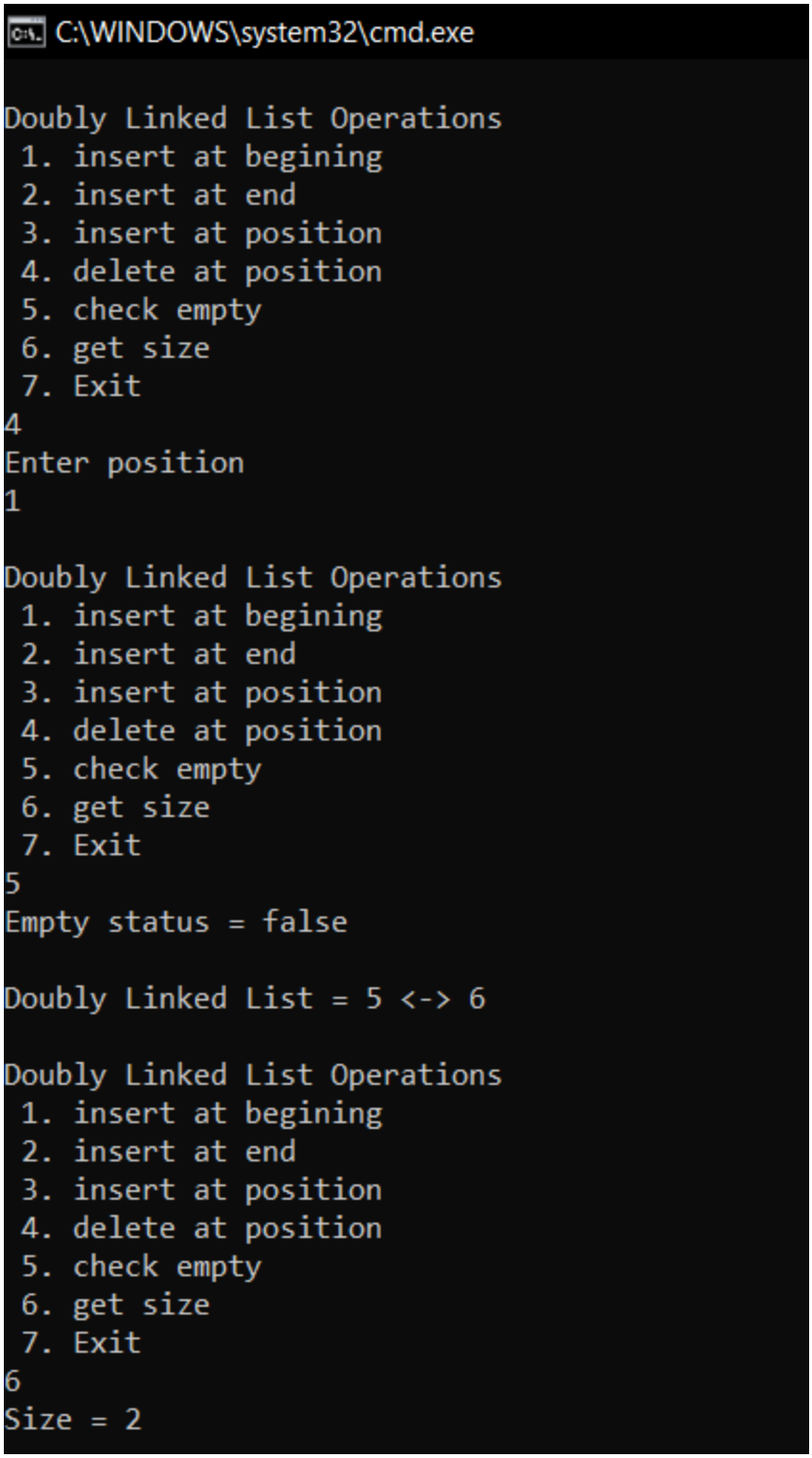
```
}
public class DoublyLinkedList
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);

        linkedList list = new linkedList();
        System.out.println("Doubly Linked List Test\n");
        char ch;
        while(true){

            System.out.println("\nDoubly Linked List Operations\n 1. insert at begining
\n 2. insert at end \n 3. insert at position \n 4. delete at position \n 5. check empty
\n 6. get size \n 7. Exit ");

            int choice = scan.nextInt();
            switch (choice)
            {
            case 1 :
                System.out.println("Enter integer element to insert");
                list.insertAtStart( scan.nextInt() );
                list.display();
                break;
            case 2 :
                System.out.println("Enter integer element to insert");
                list.insertAtEnd( scan.nextInt() );
                list.display();
                break;
            case 3 :
                System.out.println("Enter integer element to insert");
                int num = scan.nextInt() ;
                System.out.println("Enter position");
```

```
            int pos = scan.nextInt() ;
            if (pos < 1 || pos > list.getSize() ){
                System.out.println("Invalid position\n");
                list.display();
            }
            else{
                list.insertAtPos(num, pos);
                list.display();
            }
            break;
        case 4 :
            System.out.println("Enter position");
            int p = scan.nextInt() ;
            if (p < 1 || p > list.getSize() ){
                System.out.println("Invalid position\n");
                list.display();
            }
            else{
                list.deleteAtPos(p);
            }
            break;
        case 5 :
            System.out.println("Empty status = "+ list.isEmpty());
            list.display();
            break;
        case 6 :
            System.out.println("Size = "+ list.getSize() +" \n");
            list.display();
            break;
        case 7:
            list.display();
            System.exit(0);
            break;
```

```
            default :
                System.out.println("Wrong Entry \n ");
                break;
            }
        }
    }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

C:\Users\yashw\Desktop\Summer\Labs>javac DoublyLinkedList.java --release 8

C:\Users\yashw\Desktop\Summer\Labs>java DoublyLinkedList
Doubly Linked List Test


Doubly Linked List Operations
 1. insert at begining
 2. insert at end
 3. insert at position
 4. delete at position
 5. check empty
 6. get size
 7. Exit
1
Enter integer element to insert
2

Doubly Linked List = 2

Doubly Linked List Operations
 1. insert at begining
 2. insert at end
 3. insert at position
 4. delete at position
 5. check empty
 6. get size
 7. Exit
2
Enter integer element to insert
6

Doubly Linked List = 2 <-> 6

Doubly Linked List Operations
 1. insert at begining
 2. insert at end
 3. insert at position
 4. delete at position
 5. check empty
 6. get size
 7. Exit
3
Enter integer element to insert
5
Enter position
2

Doubly Linked List = 2 <-> 5 <-> 6

Doubly Linked List Operations
```


C:\WINDOWS\system32\cmd.exe
Doubly Linked List Operations
1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
7. Exit
4
Enter position
1
Doubly Linked List Operations
1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
7. Exit
5
Empty status = false
Doubly Linked List = 5 <-> 6
Doubly Linked List Operations
1. insert at begining
2. insert at end
3. insert at position
4. delete at position
5. check empty
6. get size
7. Exit
6
Size = 2

```
Doubly Linked List = 5 <-> 6

Doubly Linked List Operations
 1. insert at begining
 2. insert at end
 3. insert at position
 4. delete at position
 5. check empty
 6. get size
 7. Exit
7

Doubly Linked List = 5 <-> 6
```